Protocoles de re-chiffrement pour le stockage de données

Julien Devigne

▶ **To cite this version:**

Julien Devigne. Protocoles de re-chiffrement pour le stockage de données. Cryptographie et sécurité [cs.CR]. Université de Caen, 2013. Français. NNT : . tel-01081377

HAL Id: tel-01081377
https://hal.science/tel-01081377

Submitted on 7 Nov 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Université de Caen Basse-Normandie

École doctorale SIMEM

Thèse de doctorat

présentée et soutenue publiquement le 13 décembre 2013

par

Julien Devigne

pour obtenir le

Doctorat de l'Université de Caen Basse-Normandie

(Spécialité Informatique et Applications)

Protocoles de re-chiffrement pour le stockage de données

devant le jury composé de :

Louis Goubin, Professeur des Universités, Université de Versailles Saint-Quentin-en-Yvelines (rapporteur)
Javier Herranz, Chercheur, Universitat Politècnica de Catalunya - BarcelonaTech (rapporteur)
Thierry Berger, Professeur des Universités, Université de Limoges
Fabien Laguillaumie, Professeur des Universités, Université Claude Bernard Lyon 1
Benoît Libert, Ingénieur de recherche, Technicolor
Pascal Paillier, Ingénieur de recherche, Cryptoexperts
Damien Vergnaud, Maître de conférences, ENS (invité)
Brigitte Vallée, Directrice de recherche CNRS, Université de Caen Basse-Normandie (directrice de thèse)
Sébastien Canard, Ingénieur de recherche, Orange Labs (co-directeur de thèse)

Travaux effectués au sein de l'Applied Crypto Group des Orange Labs à Caen
42, rue des Coutures, BP6243 – 14066 CAEN Cedex 4

Université de Caen Basse-Normandie

École doctorale SIMEM

Thèse de doctorat

présentée et soutenue publiquement le 13 décembre 2013

par

Julien Devigne

pour obtenir le

Doctorat de l'Université de Caen Basse-Normandie

(Spécialité Informatique et Applications)

Protocoles de re-chiffrement pour le stockage de données

devant le jury composé de :

Louis Goubin, Professeur des Universités, Université de Versailles Saint-Quentin-en-Yvelines (rapporteur)
Javier Herranz, Chercheur, Universitat Politècnica de Catalunya - BarcelonaTech (rapporteur)
Thierry Berger, Professeur des Universités, Université de Limoges
Fabien Laguillaumie, Professeur des Universités, Université Claude Bernard Lyon 1
Benoît Libert, Ingénieur de recherche, Technicolor
Pascal Paillier, Ingénieur de recherche, Cryptoexperts
Damien Vergnaud, Maître de conférences, ENS (invité)
Brigitte Vallée, Directrice de recherche CNRS, Université de Caen Basse-Normandie (directrice de thèse)
Sébastien Canard, Ingénieur de recherche, Orange Labs (co-directeur de thèse)

Travaux effectués au sein de l'Applied Crypto Group des Orange Labs à Caen
42, rue des Coutures, BP6243 – 14066 CAEN Cedex 4

Remerciements

Le temps est venu de remercier toutes les personnes qui ont contribué de près ou de loin à cette thèse. Comme de nombreux thésards pourront le confirmer, cette tâche est loin d'être aisée, d'autant plus qu'elle ne traite pas de technique. Pas question d'utiliser des formules mathématiques, ce qui serait pourtant bien pratique. En effet, une solution certainement trop triviale aurait pu consister à énoncer les phrases suivantes :

Soit `Remerciement`*, l'ensemble des personnes ayant leur place dans cette page.*
$\forall x \in$ `Remerciement`*, je remercie grandement* x *pour toute l'aide apportée.*

Il est évident que de tels remerciements ne refléteraient pas la contribution de chacun et ne pourraient qu'être insuffisants.

Avant tout, je souhaite remercier Louis Goubin et Javier Herranz qui ont accepté la lourde tâche de rapporter mon mémoire, ainsi que Thierry Berger, Benoît Libert, Pascal Paillier qui ont bien voulu faire partie de mon jury.

Je remercie vivement les personnes qui sont à l'origine de ma thèse : Brigitte Vallée, ma directrice de thèse, Sébastien Canard, mon co-directeur et encadrant industriel et Fabien Laguillaumie, mon encadrant académique. Toute cette aventure n'aurait pas pu se réaliser sans vous. Je vous suis reconnaissant de m'avoir fait confiance en me choisissant ainsi que de m'avoir accompagné tout au long de ces trois années. Ce fut un réel plaisir de travailler et d'apprendre en votre compagnie.

Je remercie également Marc Joye, qui en me prenant comme stagiaire à la fin de mes études, m'a permis de découvrir le monde de la recherche cryptographique et ainsi de me conforter dans mon envie de réaliser une thèse. Je serai aussi toujours reconnaissant à Alain Ninet, un de mes nombreux professeurs de mathématiques, pour m'avoir fait découvrir la cryptographie lors d'un module appelé "mathématiques *appliquées* à l'informatique". Ce titre avait, pour le jeune étudiant en mathématiques fondamentales que j'étais, une connotation négative et c'est pourtant cet enseignement qui a défini la carrière que j'ai choisie de poursuivre (en faisant des maths "utiles").

Merci aussi aux deux équipes que j'ai intégrées et qui m'ont (si gentiment) accueilli au sein d'Orange Labs, et plus particulièrement à Dominique, Emmanuel, Jacques, Jean-François, Jennifer, Mohammed, Nicolas, Saïd, Stéphane, Thibault, Yvan et tous les autres que je ne peux citer. J'ai une pensée amicale pour Jacques Traore (dit l'Oracle) qui m'a fait une place dans son bureau - je ne t'oublie pas non plus Olivier, ton tour arrive - et qui m'as appris tant de choses (et surtout tant d'anecdotes). Je tiens également à remercier certains thésards d'Orange avec qui je pouvais parler crypto sans avoir recours à un "décryptage" des conversations : Amandine et Roch (que je remercie aussi pour le template LaTeX de ce mémoire) ainsi que Ghada et Olivier (que j'ai accueilli dans MON bureau pour les derniers mois de ma thèse) à qui je souhaite bonne chance pour leur dernière année.

Je n'oublie pas non plus mon équipe académique de l'Université de Caen ainsi que toutes les personnes que j'y cotoyais, en général le jeudi (ce qui leur rappelait que le week-end approchait) et ce durant 3 ans. En particulier, j'ai une pensée pour Aude, Eleonora, Hien, Julien, Laura, Laurie, Loïck, Mariya, Morgan, Patrick, Van et Xavier.

Je voudrais également citer mes camarades belges (ou belges d'adoption) qui sont passés ou sont encore à l'UCL, à savoir Iwen, Thomas et Vincent pour nos nombreuses discussions (pas

forcément cryptographiques) lors de nos périples à l'étranger. J'en profite pour remercier Damien Vergnaud pour les échanges que nous avons eu sur le sujet de cette thèse. Avec lui et Thomas, nous avons découverts des mets chinois excellents (et soi-disant un peu particulier) qui resteront d'excellents souvenirs.

Maintenant, j'en arrive à des remerciements plus personnels, à commencer par ma famille et notamment ma mère pour le soutien qu'elle m'a toujours apporté. Je la remercie aussi, ainsi que Daniel et Johanna, pour avoir relu et corrigé ce mémoire. Merci également à tous mes "hôtels" qui m'ont accueilli lors de mes nombreux déplacements parisiens : Adeline, Christophe, Jérémy, Marion et Max. Je finirai par Anne-Hélène, Bubu, Claire, Etienne, Floto, Marjo, Oliv, Rém et Vanessa, non encore cités, pour notamment nos vacances régulières à Montlivault et à Reims ; ainsi qu'Alex, Banban, Didier, Pauline, Raphaël, Séb et Sonja pour les weekends parisiens et rennais sans oublier ma petite soeur d'adoption, Anna.

à mon père

Table des matières

Introduction **1**

I Préliminaires **3**

1 Réalisation d'un stockage dématérialisé **5**
1.1 Stockage dématérialisé . . . 5
1.1.1 Propriétés souhaitées . . . 5
1.1.2 Revue des solutions de base . . . 7
1.1.3 Solutions cryptographiques envisagées . . . 8
1.2 Solution retenue : le serveur de re-chiffrement . . . 9
1.2.1 Présentation rapide du serveur de re-chiffrement . . . 9
1.2.2 Réalisation détaillée d'un stockage de données . . . 11
1.2.3 Particularités d'une telle solution . . . 12

2 Introduction technique **15**
2.1 Outils mathématiques . . . 15
2.1.1 Groupe . . . 15
2.1.2 Courbes elliptiques et couplages . . . 16
2.1.3 Problèmes difficiles dans les groupes cycliques d'ordre premier . . . 17
2.2 Fonctions cryptographiques . . . 20
2.2.1 Fonction à sens unique . . . 20
2.2.2 Fonction de hachage . . . 20
2.3 Sécurité en cryptographie . . . 21
2.3.1 Méthodologie de la sécurité prouvée . . . 21
2.3.2 Modèles . . . 21
2.4 Une primitive essentielle : le chiffrement . . . 22
2.4.1 Sécurité prouvée pour la primitive de chiffrement . . . 22
2.4.2 Le chiffrement à clé publique . . . 23
2.4.3 Le chiffrement à clé secrète . . . 25
2.5 Outils cryptographiques . . . 27
2.5.1 Signature . . . 27
2.5.2 Signature à usage unique . . . 27
2.5.3 La construction CHK . . . 27
2.5.4 La construction Fujisaki-Okamoto . . . 28

II Modélisation du serveur de re-chiffrement **29**

3 Unification des différents modèles de serveur de re-chiffrement **31**
3.1 Définition d'un serveur de re-chiffrement . . . 31

3.1.1 Modèle unifié . . . 32
3.1.2 Exemple illustratif . . . 34
3.1.3 Production d'un chiffré . . . 36
3.2 Caractéristiques . . . 36
3.2.1 Nombre de re-chiffrements successifs . . . 36
3.2.2 Type de confiance entre le délégant et le délégué . . . 38
3.3 Sécurité d'un serveur de re-chiffrement . . . 39
3.3.1 Notions de sécurité . . . 39
3.3.2 Restrictions naturelles dans les moyens de l'adversaire . . . 42
3.3.3 Restrictions non naturelles récurrentes . . . 43
3.3.4 Spécificités liées aux preuves de sécurité . . . 44
Note sur les sécurités secondaires . . . 46
3.4 Sécurité pour le stockage de données . . . 47
3.4.1 Utilité de l'anonymat de la clé de re-chiffrement ? . . . 47
3.4.2 Indiscernabilité des messages chiffrés . . . 47

4 Modèles classiques de serveur de re-chiffrement 49
4.1 Revue des schémas existants et idées de construction . . . 49
4.1.1 Re-chiffrement infini et bidirectionnel . . . 50
4.1.2 Re-chiffrement fini et bidirectionnel . . . 51
4.1.3 Re-chiffrement unique et unidirectionnel . . . 52
4.1.4 Re-chiffrement multiple de cardinal n supérieur à 1 et unidirectionnel . . . 56
4.1.5 Re-chiffrement infini et unidirectionnel . . . 58
4.2 Relations entre les différents types . . . 60
4.2.1 Le re-chiffrement unidirectionnel implique le re-chiffrement bidirectionnel . 60
4.2.2 Le re-chiffrement bidirectionnel implique le re-chiffrement unidirectionnel . 61
4.2.3 Le re-chiffrement infini implique le re-chiffrement multiple de cardinal m, qui implique lui-même le re-chiffrement multiple de cardinal n avec m supérieur à n . . . 63
4.2.4 Le re-chiffrement multiple de cardinal n implique le re-chiffrement multiple de cardinal m avec m supérieur à n, qui implique lui même le re-chiffrement infini . . . 63
4.2.5 Conclusion sur les différentes relations . . . 64
Récapitulation des différents schémas . . . 64

III Gestion de la sécurité 67

5 Amélioration de la sécurité RCCA 69
5.1 Préliminaires . . . 69
5.1.1 La sécurité RCCA . . . 69
5.1.2 Rafraîchissement public d'un chiffré . . . 70
5.2 Amélioration de la sécurité RCCA pour le chiffrement à clé publique . . . 71
5.2.1 Utilisation d'un chiffrement hybride . . . 71
5.2.2 La construction Canetti-Krawczyk-Nielsen . . . 71
5.3 Amélioration de la sécurité RCCA pour les schémas offrant un re-chiffrement unique et unidirectionnel . . . 72
5.3.1 Une construction générique . . . 72
5.3.2 Sécurité du schéma . . . 75
5.3.3 Un exemple de construction plus pratique sur le schéma Libert-Vergnaud . 77
5.4 À propos des autres types . . . 79

5.4.1 Le type bidirectionnel et unique . . . 79
5.4.2 Les schémas avec re-chiffrement multiple strictement supérieur à 1 ou infini 79

6 Conception de schémas sûrs et fondés sur le chiffrement Hash ElGamal 81
6.1 Revue des schémas existants . . . 81
6.1.1 Boîte à outils . . . 81
6.1.2 Le schéma de Deng *et al.* . . . 82
6.1.3 Le schéma de Chow *et al.* . . . 83
6.1.4 Analyse de la sécurité . . . 84
6.2 Un nouveau schéma . . . 86
6.2.1 Preuve de connaissance avec extracteur . . . 86
6.2.2 Description du nouveau schéma . . . 87
6.2.3 Sécurité du schéma . . . 88

IV Gestion des droits d'accès 97

7 Gestion dynamique des droits d'accès 99
7.1 Un cas d'usage particulier . . . 99
7.1.1 Problématique . . . 99
7.1.2 Un nouveau type de schéma - Le serveur de re-chiffrement combiné . . . 101
7.1.3 Fonctionnalités . . . 102
7.2 Résultats génériques . . . 103
7.2.1 Le re-chiffrement combiné implique le re-chiffrement bidirectionnel infini et le re-chiffrement unidirectionnel unique . . . 103
7.2.2 La réciproque est-elle vraie ? . . . 103
7.2.3 Trois re-chiffrements bidirectionnels et infinis impliquent le re-chiffrement combiné . . . 104
7.3 Une construction pratique . . . 107
7.3.1 Une version combinée du schéma Libert-Vergnaud . . . 107
7.3.2 Propriétés additionnelles . . . 109
Note sur un serveur de re-chiffrement combiné généralisé . . . 111

8 Gestion fine des droits d'accès 115
8.1 Le serveur de re-chiffrement conditionnel . . . 115
8.1.1 Délégation de type "tout ou rien" . . . 115
8.1.2 Définition . . . 116
8.1.3 État de l'art . . . 118
8.1.4 Compatibilité avec la gestion dynamique des droits d'accès . . . 119
8.2 Une gestion des droits d'accès plus fine . . . 119
8.2.1 Limite de la granularité des conditions . . . 120
8.2.2 Le serveur de re-chiffrement sélectif . . . 121
8.2.3 Notre schéma avec re-chiffrement sélectif et unidirectionnel . . . 124

***Note sur les modèles étendus de serveur de re-chiffrement* 129**
Le chiffrement basé sur l'identité . . . 129
Le chiffrement de groupe . . . 131
Le chiffrement basé sur les attributs . . . 132

Conclusion 135

Liste des figures 138

Liste des tableaux **138**

Publications personnelles **139**

Brevets **139**

Bibliographie **140**

Introduction

La *cryptographie* vient des mots en grec ancien *kruptos* (“caché”) et *graphein* (“écrire”) et signifie l’“écriture secrète”. Son utilisation remonte à l’antiquité où son but était de protéger un message secret lors de sa transmission. Elle s’est depuis démocratisée avec l’apparition de l’informatique et est désormais composée de trois caractéristiques fondamentales :

- la *confidentialité* qui garantit le caractère secret du message transmis ;
- l’*intégrité* qui s’assure que le message transmis n’a pas été modifié ;
- l’*authentification* qui permet de vérifier l’identité de l’émetteur.

Dans ce mémoire, nous allons utiliser la cryptographie pour fournir des moyens préservant la vie privée dans une informatique dématérialisée. En particulier, nous nous intéresserons à la protection des données personnelles afin que seules les entités autorisées puissent en obtenir l’accès.

La confidentialité des données dans l’informatique dématérialisée

L’*informatique dématérialisée*, couramment désignée par le terme anglais *cloud*, consiste à ne plus gérer ses données sur son propre dispositif (ordinateur, téléphone, ...) mais à en confier la gestion à une entité tierce accessible par Internet. De nos jours, on entend souvent parler de l’avènement du cloud avec les nouveaux produits technologiques qui sont connectés en permanence à Internet, comme les smartphones/tablettes, et avec des connexions réseaux de plus en plus performantes (3G/4G, ADSL/fibre, ...). Les utilisateurs peuvent ainsi accéder à leurs données de n’importe où et à n’importe quel moment. Cependant, un des risques majeurs de ces technologies est la perte de confidentialité de leurs données puisqu’elles sont confiées à une entité tierce, qui leur est souvent inconnue.

L’informatique dématérialisée est divisée en deux catégories :

- le calcul dématérialisé, de l’anglais *cloud computing*, où le traitement des données est confié à l’entité tierce ;
- le stockage dématérialisé, de l’anglais *cloud storage*, où le stockage des données est assuré par cette même entité.

Ces deux informatiques peuvent être compatibles entre elles, comme le propose, par exemple, le système Amazon Cloud Drive [Ama]. Nous décidons cependant de nous focaliser sur le stockage dématérialisé et de concevoir une réalisation de celui-ci possédant certaines propriétés [1] :

- un utilisateur doit pouvoir partager ses données avec les utilisateurs en qui il a confiance ;
- un utilisateur non autorisé ne doit pas pouvoir accéder aux données, notamment le serveur qui est un utilisateur non autorisé ;
- un utilisateur stockant des données dans un serveur doit pouvoir gérer les droits d’accès comme il le souhaite.

1. Nous reviendrons brièvement, page 132, sur le calcul dématérialisé possédant les mêmes propriétés que le stockage dématérialisé que nous proposons de réaliser.

Plan du mémoire

Dans ce mémoire, nous nous intéresserons à la problématique du stockage dématérialisé assurant la protection des données personnelles. Dans le Chapitre 1, nous commencerons par l'exposer en détail et passerons en revue les différentes solutions, basiques ou avancées, qu'offre la cryptographie. Pour les raisons que nous préciserons, notre choix se portera sur l'emploi de la primitive de *serveur de re-chiffrement*, de l'anglais *proxy re-encryption*. Le Chapitre 2 présentera quant à lui les pré-requis de ce mémoire.

Nous décrirerons formellement la primitive de serveur de re-chiffrement dans le Chapitre 3. Nous présenterons en particulier un modèle de serveur de re-chiffrement qui regroupera tous les modèles que l'on peut rencontrer dans la littérature, afin de ne pas perdre le lecteur dans toutes les descriptions possibles. Dans le Chapitre 4, nous passerons en revue les différentes propriétés liées à cette primitive et en profiterons pour exposer les différents schémas existants. Nous étudierons également les relations possibles entre ces différentes propriétés.

Il sera alors temps de nous concentrer sur la sécurité de certains schémas de serveur de re-chiffrement. Nous présenterons dans le Chapitre 5 les résultats de l'étude, menée conjointement avec Sébastien Canard, consistant à transformer les schémas de serveur de re-chiffrement atteignant la sécurité IND-RCCA en des schémas atteignant la sécurité IND-CCA. Dans le Chapitre 6, nous analyserons les schémas de serveur de re-chiffrement à base de chiffrement Hash ElGamal et de preuves de connaissances non interactives "à la Schnorr" et nous nous concentrerons notamment sur le schéma [CWYD10]. Nous nous apercevrons qu'ils présentent des failles de sécurité non prises en compte par leur modèle de sécurité et proposerons des modifications qui peuvent être apportées pour les combler. Ces travaux ont été réalisés avec Sébastien Canard et Fabien Laguillaumie et ont fait l'objet d'une publication dans le journal JISIS lors de la conférence ProvSec [CDL11].

Enfin, nous reviendrons sur la problématique de la réalisation d'un stockage dématérialisé et nous attacherons à modifier la primitive de serveur de re-chiffrement afin de parvenir au stockage de données souhaité. Nous commencerons par étudier dans le Chapitre 7 la gestion dynamique des droits d'accès et proposerons un nouveau modèle que nous appellerons *serveur de re-chiffrement combiné*. Ces recherches ont été menées avec Sébastien Canard. Dans le Chapitre 8, nous nous concentrerons sur une autre fonctionnalité, à savoir la gestion fine des droits d'accès. Sur le sujet, la littérature propose le modèle de serveur de re-chiffrement conditionnel. Nous mettrons en évidence qu'il est possible de parvenir à une gestion des droits d'accès encore plus précise et élaborerons un nouveau modèle que nous nommerons *serveur de re-chiffrement sélectif*. Cette étude a été réalisée avec Eleonora Guerrini et Fabien Laguillaumie. Pour chacun de ces nouveaux modèles, nous montrerons comment réaliser de tels schémas. Ces deux modèles sont en plus compatibles entre eux.

Première partie

Préliminaires

Chapitre 1

Réalisation d'un stockage dématérialisé

Nous présentons dans ce chapitre la réalisation d'un stockage dématérialisé et exposons les propriétés que nous voulons en obtenir notamment en matière de sécurité. Nous passons en revue les différentes solutions et décrivons celle que nous avons retenue et qui est fondée sur un serveur de re-chiffrement aussi appelé serveur de transchiffrement.

1.1 Stockage dématérialisé

Le stockage dématérialisé, également appelé stockage de données sur serveur, consiste à offrir aux utilisateurs la possibilité de stocker des données sur un serveur. L'intérêt d'un tel système est que le serveur va a priori être connecté en permanence à Internet. De n'importe où et à n'importe quel moment, un utilisateur pourra accéder à ses données, en récupérer et en stocker, en passant par les différents réseaux de connexion (3G/4G, adsl/fibre, satellite, ...). Ceci est d'autant plus intéressant lorsque le dispositif qu'il utilise dispose de peu de mémoire, comme par exemple un téléphone mobile. Bon nombre de systèmes proposent ce type de service. C'est le cas de DropBox [Dro], iCloud [Icl] et MEGA [Meg], connus mondialement, ou encore Cloudwatt [Clo] et Numergy [Num]. Les deux derniers sont issus du projet Andromède [And] dont le but officiel est la réalisation d'un "cloud souverain" sécurisé pour le gouvernement français, son administration et les entreprises françaises.

Cette application peut également être utilisée pour d'autres types de stockage de données, comme par exemple dans des serveurs non dématérialisés. C'est le cas d'un ordinateur familial qui joue le rôle de serveur pour stocker les données et les envoyer aux appareils électroniques du domicile.

1.1.1 Propriétés souhaitées

Le but de cette application n'est pas seulement de stocker des données et de pouvoir y accéder à tout moment, mais aussi d'être en mesure de les partager avec d'autres. Cette notion de partage est très importante de nos jours, car les utilisateurs s'échangent de plus en plus d'informations. Pour autant, ils ne souhaitent pas que leurs données soient accessibles à n'importe qui, mais uniquement aux utilisateurs qu'ils ont autorisés. Ainsi, le site Facebook [Fac], qui permet notamment aux utilisateurs de partager avec leurs amis tout un tas d'informations personnelles, a fait l'objet de nombreuses polémiques [1] ces dernières années pour le non respect de la vie privée de ses internautes.

1. cf les articles du monde [LeMb, LeMa].

Dans un contexte professionnel, nous pouvons citer aussi le cas de collègues qui s'échangent leurs réflexions et contributions sur leur dernier sujet de recherche sans pour autant que des "espions" ne soient en mesure de leur voler leurs idées. Ils utilisent généralement un logiciel de gestion de versions appellé *subversion* ou *svn* [Sub]. Comme son nom l'indique, ce système a l'avantage de pouvoir gérer les différentes versions d'un fichier modifié par plusieurs collègues en même temps et de produire un historique des modifications effectuées par chacun. Le serveur, qui effectue la gestion des versions, stocke également les fichiers correspondants et y a un accès en clair, ce qui peut s'avérer problématique s'il est corrompu. Ce service n'est donc pas la solution si on recherche surtout à préserver la confidentialité des données.

Ainsi, l'utilisateur ne se limite plus uniquement à un usage strictement personnel de ses données, mais il peut en étendre l'accès à d'autres personnes. On parle alors de partage des données. De plus, l'accès aux données est réservé aux seuls utilisateurs autorisés. C'est ce que l'on appelle la *confidentialité des données*. Le système souhaité est résumé dans la Figure 1.1. Bien évidemment, dans cette figure, toutes les entités mentionnées n'auront pas accès à toutes les données stockées, puisque par exemple les professionnels de santé n'auront accès qu'aux données de santé.

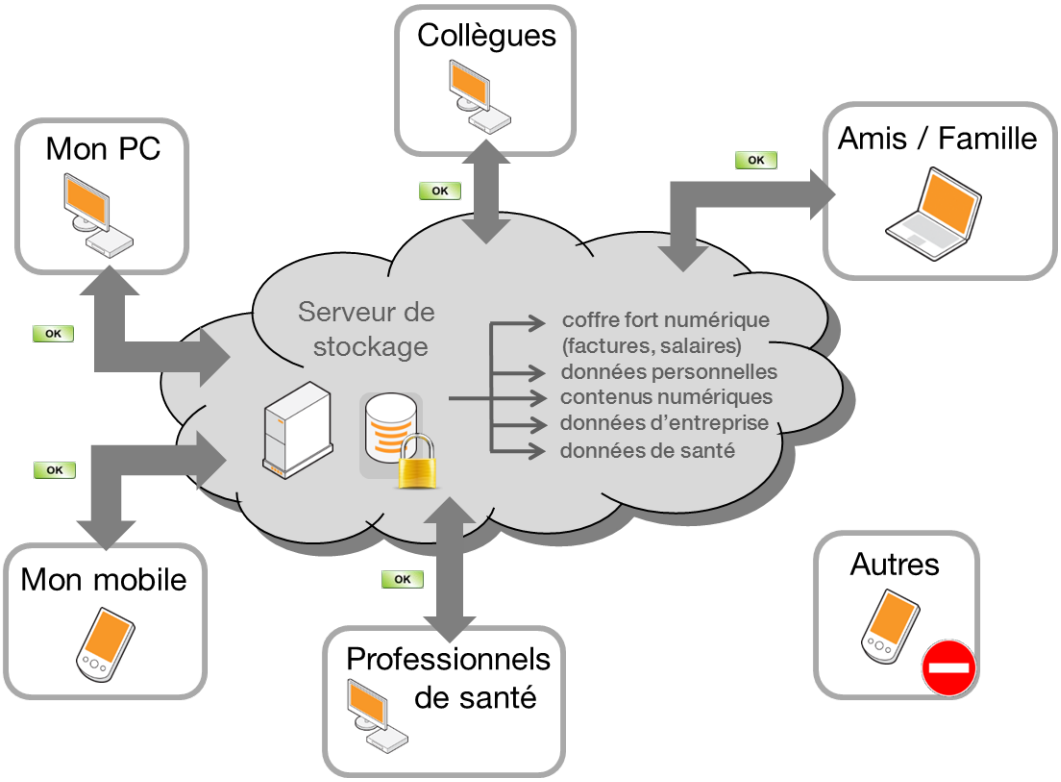

Figure 1.1 – Vue d'ensemble du stockage dématérialisé souhaité

Pour finir, nous voulons également offrir aux utilisateurs le partage le plus fonctionnel et le plus flexible possible. Dans certains des systèmes cités précédemment, tels qu'iCloud et Dropbox, l'utilisateur peut moduler le partage comme il le souhaite. Une fois les données stockées, il peut sélectionner les droits d'accès de son choix en les précisant au serveur qui se charge alors de leur gestion.

1.1.2 Revue des solutions de base

Plusieurs solutions existent et peuvent être mises en place. On en distingue trois catégories. Seules les deux premières sont vraiment utilisées, la troisième n'étant clairement pas efficace.

Sécurité assumée entièrement par le serveur. La première façon, et aussi la plus basique, de réaliser un tel système est de faire totalement confiance au serveur de stockage et de lui confier toute la responsabilité de la sécurité. Le serveur a alors accès aux données en clair et les stocke de manière protégée. Lorsqu'un utilisateur demande à y accéder, le serveur vérifie qu'il dispose bien des droits d'accès avant de lui transmettre les informations.

L'avantage de ce système est que sa mise en place pour l'utilisateur est très simple. Il n'a en effet qu'à fournir au serveur les données à stocker (de façon sécurisée bien sûr, comme par exemple en utilisant le protocole TLS), ainsi que la liste des personnes autorisées à y accéder. Par la suite, il pourra également modifier les droits d'accès très facilement, leur gestion est donc dynamique. En contrepartie, la confidentialité des données n'est pas totalement assurée puisque le serveur, qui n'est a priori pas un utilisateur autorisé, a nécessairement accès aux données. Cela signifie aussi que si un individu malveillant arrivait à prendre le contrôle du serveur ou à en récupérer des données, il aurait alors accès aux informations en clair. Il faut donc que le serveur dispose d'une sécurité accrue pour protéger les données stockées. Par conséquent, l'utilisation d'un tel système nécessite davoir une confiance absolue envers le serveur. Malgré tout, la facilité de mise en place pour l'utilisateur en fait la solution la plus répandue. Nous la retrouvons par exemple au sein des systèmes DropBox et iCloud.

Divulgation des clés de déchiffrement. Une autre façon de faire, ne nécessitant pas une confiance absolue dans le serveur, consiste pour l'utilisateur à chiffrer (de manière asymétrique ou symétrique) ses données et à envoyer la clé de déchiffrement aux personnes avec lesquelles il souhaite partager l'information. De cette manière, le serveur, qui stocke des données chiffrées, ne peut donc plus accéder aux données en clair. Ce n'est donc plus le serveur comme précédemment mais l'utilisateur lui-même qui maîtrise l'accès aux données en clair : il choisit le protocole de chiffrement, il adresse la clé de déchiffrement à qui et quand il le souhaite. Cependant, dès qu'un utilisateur est en possession de la clé, il nest plus possible d'annuler son droit d'accès, à moins de supprimer la donnée en question sur le serveur. Mais dans ce cas, elle est alors inaccessible de tout les utilisateurs autorisés.

La confidentialité des données est certes assurée. En effet, même si le serveur dévoile les données chiffrées à des individus malveillants, ils ne pourront pas obtenir les clairs correspondants puisque ni eux ni le serveur ne possèdent la clé de déchiffrement. En revanche, l'utilisateur procède à une divulgation de sa clé de déchiffrement aux autres utilisateurs, ce qui n'est pas forcément acceptable pour des raisons évidentes de sécurité. Cette solution est notamment utilisée dans l'application MEGA.

Duplication des données chiffrées. Une dernière façon standard de concevoir un tel système consiste à reprendre en partie l'idée précédente. Les données sont toujours chiffrées par l'utilisateur mais cette fois-ci le partage ne va plus se faire via une divulgation des clés de déchiffrement mais par une duplication des données chiffrées. Pour cela, l'utilisateur souhaitant partager ses données - que l'on appellera propriétéaire afin de le distinguer, puisque les données lui appartiennent - produit un chiffré par donnée et par utilisateur en qui il a confiance. Il y a alors deux possibilités décrites ainsi :

- soit les chiffrés sont tous produits lors du stockage des données et le serveur les stocke tous. Lorsqu'un utilisateur souhaite accéder à une donnée, le serveur lui envoie le chiffré correspondant à cette donnée et qui lui correspond. Cette solution nécessite un espace de stockage important qui dépend directement du nombre d'utilisateurs autorisés.
- soit les chiffrés sont produits à la demande. Le propriétaire ne chiffre ses données que pour lui avant de les envoyer au serveur. Lorsqu'un utilisateur autorisé souhaite accéder à une

de ces données, le serveur prévient le propriétaire qui récupère la donnée demandée. Ce dernier la déchiffre, puis la chiffre pour l'utilisateur à qui il la transmet via le serveur ou directement. Dans cette solution, le propriétaire doit être disponible à tout moment pour que le partage soit efficace.

Ces deux solutions respectent bien la confidentialité des données puisque celles-ci ne sont chiffrées que pour des utilisateurs autorisés. Dans ces solutions, les droits d'accès peuvent également être modifier en ajoutant ou supprimant une donnée chiffrée. En outre, le partage ne se fait plus par divulgation des clés privées, ce qui est un avantage supplémentaire. Pour autant, ces solutions ne sont pas déployées en raison des inconvénients qu'elles présentent. En effet, la première requiert beaucoup d'espace de stockage, ce qui n'est pas financièrement rentable. La seconde, quant à elle, nécessite une présence à tout moment du propriétaire, ce qui est très contraignant.

Conclusion sur les solutions de base. D'après les descriptions de ces différentes méthodes de base, nous nous apercevons qu'aucune d'entre elles ne semble idéale, chaque cas présentant au moins un inconvénient majeur.

1.1.3 Solutions cryptographiques envisagées

Il existe cependant des solutions plus adaptées à cette problématique que les solutions de base et autre que le serveur de re-chiffrement, que nous allons étudier par la suite. Nous les décrivons brièvement.

Le chiffrement de groupe. L'idée du chiffrement de groupe, de l'anglais *broadcast encryption* (BE), fut introduite par Fiat et Naor en 1993 dans [FN93]. Le concept est cette fois-ci de ne plus créer un chiffré pour un utilisateur mais pour un ensemble d'utilisateurs. Bien que compatible avec un contexte de chiffrement à clé secrète, nous nous intéressons ici qu'au contexte de chiffrement à clé publique. Ainsi lorsque l'on souhaite envoyer de manière sécurisée de l'information à tout un groupe d'entités, on produit un chiffré pour la clé publique du groupe. La manière la plus simple de concevoir un tel schéma est d'utiliser l'idée de duplication des données chiffrées que nous venons de voir ci-dessus. Un chiffré est composé d'autant de chiffrés que le groupe considéré comporte d'utilisateurs. Il en est de même pour la clé publique du groupe qui est composée des différentes clés publiques de ces utilisateurs. Pour des raisons évidentes, cette solution n'est pas efficace. L'intérêt du chiffrement de groupe est que l'on est capable de concevoir des schémas plus efficaces avec des chiffrés et des clés publiques de groupe dont les tailles ne sont plus linéaires en le nombre d'utilisateurs, comme par exemple [BGW05].

Cette brique cryptographique peut être utilisée pour partager, de manière confidentielle, un chiffré avec d'autres utilisateurs. En effet, il suffit de savoir, au moment où l'on calcule le chiffré, quels seront les utilisateurs qui y auront accès et de destiner le chiffré à tout ce groupe d'utilisateurs. Chaque utilisateur pourra ainsi obtenir l'information lorsqu'il le souhaite et indépendamment des autres. Cependant, comme nous venons de l'expliquer, cette solution nécessite de savoir à l'avance qui aura accès aux données. Il est donc impossible d'ajouter des droits d'accès aux données une fois les chiffrés générés - à moins de récupérer la donnée en clair et de la chiffrer pour de nouveaux utilisateurs. Cette solution ne nous paraît donc pas idéale pour les utilisateurs qui vont stocker leurs données sur un serveur et décider, à leur gré, qui aura accès à leurs données ou non.

Le chiffrement basé sur les attributs. Le chiffrement basé sur les attributs, de l'anglais *attribute-based encryption* (ABE), peut être vu comme une sous catégorie du chiffrement de groupe, les propriétés obtenues sont donc similaires. L'idée consiste toujours à ce qu'un chiffré soit destiné à un groupe d'utilisateurs. Cependant, ce groupe n'est plus choisi comme un ensemble d'utilisateurs quelconques, mais comme un ensemble d'utilisateurs possédant un certain nombre d'attributs, d'où son nom. Il fut introduit par Sahai et Waters dans [SW05]. Les clés

(privées/publiques) des utilisateurs sont liées aux attributs de ceux-ci et sont générées par une autorité de confiance. Nous retrouvons dans ce type de chiffrement les inconvénients du chiffrement de groupe. L'objectif est également d'obtenir des chiffrés et des clés dont la taille est indépendante du nombre d'attributs considérés.

Des travaux ont également été menés pour rendre cette brique plus fonctionnelle, comme le chiffrement basé sur l'identité avec joker présenté par Abdalla *et al.* dans [ABC+11] et sur lequel nous reviendrons au Chapitre 8. Prenons un exemple simple : tous les étudiants de l'Université de Caen Basse-Normandie possèdent une adresse mail se terminant par "@etu.unicaen.fr". Une donnée pourra alors être chiffrée à destination de tous ces étudiants. Au moment du chiffrement, l'utilisateur qui chiffre la donnée n'a alors pas à connaître en détail la liste de ces derniers. Dans notre exemple, rien ne dit pourtant que le propriétaire des données à une confiance absolue envers tous ces utilisateurs, ce qui peut être problématique pour notre application de stockage. Il se peut cependant que cette solution soit intéressante dans des usages spécifiques, comme par exemple un hôpital où tous les salariés de cet hôpital vont être, a priori, de confiance. En effet, si un patient est traité pour une pathologie cancéreuse, il paraît normal que tous les médecins du service de cancérologie puissent accéder à son dossier. Ce chiffrement serait est valable quand il n'y a pas de distinction possible entre les utilisateurs possédant les mêmes attributs. Dans le cas contraire, nous voyons qu'il ne permet pas un partage sélectif et fonctionnel des données.

1.2 Solution retenue : le serveur de re-chiffrement

Nous nous proposons d'utiliser le serveur de re-chiffrement comme alternative à toutes ces solutions. En effet, comme nous allons le voir, son utilisation va permettre de réaliser un système d'informatique dématérialisée de stockage sans aucun des inconvénients majeurs évoqués précédemment et offrant les fonctionnalités souhaitées. Nous commençons par décrire informellement cette brique cryptographique. Ce mémoire étant essentiellement axé sur cette brique, de nombreux détails seront données dans les prochains chapitres.

1.2.1 Présentation rapide du serveur de re-chiffrement

Historiquement, le premier article à traiter de serveur de re-chiffrement, de l'anglais *proxy re-encryption* (PRE), est celui de Blaze, Bleumer et Strauss dans [BBS98]. L'idée originelle de cet article est de concevoir un système de suivi de courriels chiffrés.

Un tel système permet de traiter l'information contenue dans un mail même en l'absence de la personne désignée comme réceptrice dudit mail. Prenons l'exemple de deux utilisateurs : Alice et Bob. Ainsi, lorsqu'Alice n'est pas disponible, le serveur, chargé de lui transmettre ses courriels, les transfère à Bob, désigné par Alice pour traiter cette information. Afin de rendre la lecture impossible pour le serveur ou toute autre entité, les courriels sont protégés par un système de chiffrement. Dans un tel système et compte tenu des propriétés du chiffrement, il est impossible de concevoir un suivi de courriels chiffrés puisque seule Alice peut les déchiffrer, sauf si bien-sûr celle-ci dévoile sa clé de déchiffrement. Le système présenté dans l'article [BBS98] est novateur puisqu'il permet d'envisager ce suivi sans qu'Alice n'ait à dévoiler sa clé de déchiffrement.

Il fonctionne de la manière suivante : Alice crée, en accord avec Bob, une *clé de re-chiffrement* qui va permettre au serveur témoignant de l'absence de cette dernière, de transformer - on parle alors de re-chiffrer - un courriel chiffré pour Alice en un courriel chiffré pour Bob.

L'ensemble des outils qui permettent de réaliser ce suivi de courriels chiffrés est appelé "primitive de serveur de re-chiffrement". Cette primitive est composée de deux parties :

– un schéma de chiffrement à clé publique qui offre les fonctionnalités de chiffrement et de déchiffrement pour garantir la confidentialité des messages (voir Chapitre 2). Ainsi, un

chiffré est produit en utilisant une clé publique. Quant à son déchiffrement, il utilise la clé privée correspondante ;
- un schéma de re-chiffrement qui permet de modifier le destinataire d'un chiffré. Ceci est rendu possible par l'utilisation d'une clé de re-chiffrement. Cette opération se passe sans que les deux entités concernées - destinataire originel et nouveau destinataire - n'aient à être présentes, ce qui implique qu'elle soit effectuée par une entité tierce, appelée Serveur. De plus, pour préserver la confidentialité des messages, introduite par l'opération de chiffrement, elle ne doit dévoiler aucune information sur les messages en clair ni a fortiori sur les clés privées utilisées.

La figure 1.2 résume le fonctionnement général d'une primitive cryptographique de serveur de re-chiffrement et permet de voir quelles sont les entités concernées et ce qu'elles doivent faire, où :

- $(sk_{\mathtt{A}}, pk_{\mathtt{A}})$ (respectivement $(sk_{\mathtt{B}}, pk_{\mathtt{B}})$) représente la paire de clés (privée/publique) d'Alice (respectivement de Bob) utilisée pour le schéma de chiffrement.
- l'Utilisateur et le Serveur représentent respectivement toute entité souhaitant envoyer un message à Alice et toute entité stockant la donnée.
- $rk_{\mathtt{A}\to\mathtt{B}}$ est la clé de re-chiffrement d'Alice vers Bob, qui est calculée au préalable et donnée au Serveur. Nous verrons différentes manières de calculer un telle clé aux Chapitre 3 et 4.

Le protocole de suivi de courriels chiffrés se déduit de cette figure, en remarquant bien qu'Alice n'intervient pas dans le processus qui permet à Bob de récupérer le message en clair.

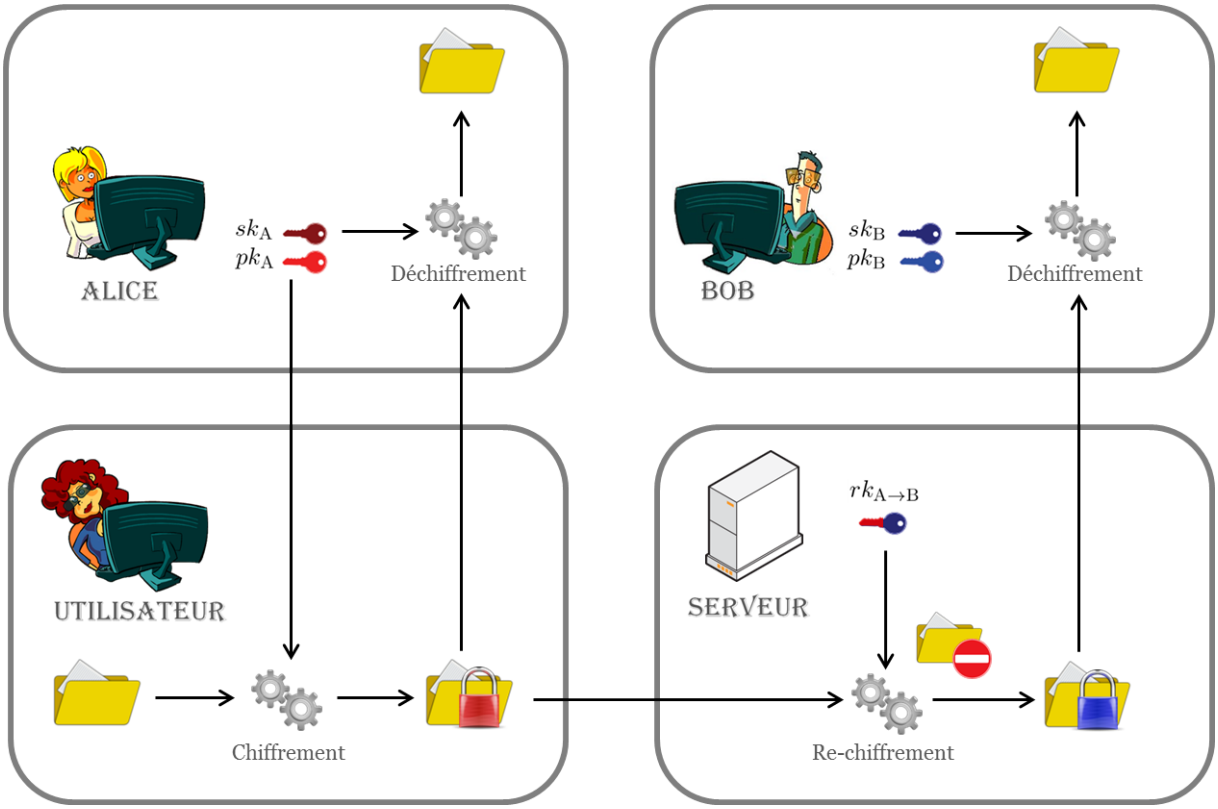

Figure 1.2 – Fonctionnement général d'un PRE

1.2.2 Réalisation détaillée d'un stockage de données

Nous nous intéressons maintenant à la solution que nous avons retenue pour concevoir une application de stockage de données sur serveur fondée sur le serveur de re-chiffrement. Cette utilisation a été décrite par Ateniese *et al.* dans [AFGH06]. Pour faciliter la compréhension du lecteur, nous prenons un exemple simple où seulement trois dispositifs sont autorisés à récupérer les données. La description de la solution est bien sûr compatible avec un nombre plus important de dispositifs. Les Figures 1.3 et 1.4 résument la description de ce système où, pour tout dispositif $\mathtt{D}_1$ et $\mathtt{D}_2$, $(sk_{\mathtt{D}_1}, pk_{\mathtt{D}_1})$ et $(sk_{\mathtt{D}_2}, pk_{\mathtt{D}_2})$ représentent respectivement la paire de clé (privée/publique) de $\mathtt{D}_1$ et de $\mathtt{D}_2$ et $rk_{\mathtt{D}_1 \to \mathtt{D}_2}$ la clé de re-chiffrement de $\mathtt{D}_1$ vers $\mathtt{D}_2$.

Idées principales. Notre solution s'articule en 2 étapes :

- le chiffrement qui va servir à protéger la confidentialité des données en évitant que n'importe qui ne puisse y avoir accès ;
- le re-chiffrement qui va permettre le partage des données et d'en contrôler l'accès par l'utilisation d'une clé de re-chiffrement.

Préliminaires. Le contexte considéré est le suivant : nous disposons d'un premier utilisateur, Alice qui souhaite stocker sur un serveur des données. Elle souhaite pouvoir accéder à ces données depuis son ordinateur et son smartphone, que l'on note respectivement $\mathtt{A}_1$ et $\mathtt{A}_2$. Elle souhaite également que Bob puisse accéder aux données via $\mathtt{B}_1$, son ordinateur portable [2]. Chacun des dispositifs évoqués possède une paire de clés (publique/privée) qu'il a générée.

Stockage des données. La partie de stockage est standard. Nous nous plaçons dans le cas où Alice possède des données sur son dispositif $\mathtt{A}_1$. Elle chiffre alors toutes ses données pour ce dernier en utilisant la clé publique de celui-ci et l'algorithme de chiffrement du schéma de serveur de re-chiffrement considéré. Nous qualifierons $\mathtt{A}_1$ de dispositif "maître", car c'est à partir de lui que seront décidés ceux qui pourront accéder aux données. Ce dispositif envoie ensuite les chiffrés produits au serveur de stockage.

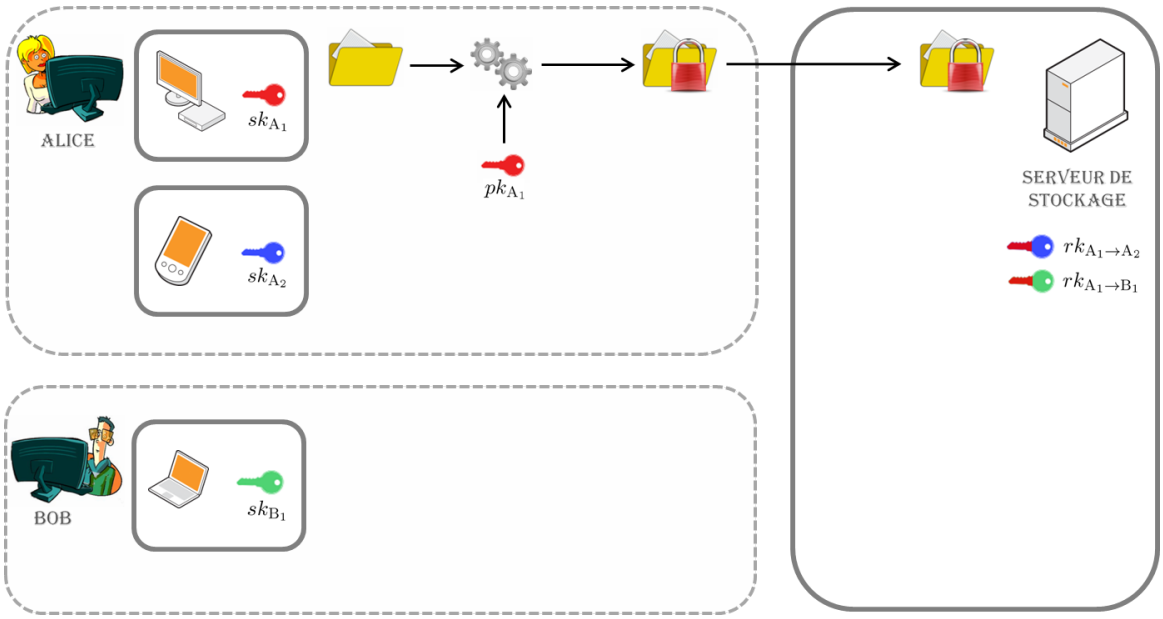

Figure 1.3 – Stockage des données et des droits d'accès

2. Afin de ne pas trop compliquer la description du service, nous n'avons considéré qu'un seul dispositif $\mathtt{B}_1$ pour Bob. Nous verrons au Chapitre 7 que la considération de plusieurs de ses dispositifs pourra donner des fonctionnalités intéressantes.

Attribution des droits d'accès. Pour le moment, seul l'appareil "maître", en l'occurence $\mathtt{A}_1$, a les droits d'accès aux données. Pour ajouter un nouveau dispositif à ces droits d'accès, il suffit de donner au serveur de stockage une clé de re-chiffrement associée à ce nouveau dispositif. Cette opération doit être effectuée pour chaque dispositif autorisé. De ce fait, nous avons deux clés de re-chiffrement à calculer et à donner au serveur : celle pour $\mathtt{A}_2$ et celle pour $\mathtt{B}_1$. Cette opération ne se fait qu'une seule fois pour chaque dispositif, ce n'est donc pas contraignant, surtout si beaucoup de données sont stockées et partagées. On remarque également que cette phase présente l'avantage d'être indépendante de celle du stockage et peut donc être réalisée à l'avance ou bien après cette dernière.

Récupération des données. Une fois les données et les clés de re-chiffrement en possession du serveur, chaque dispositif autorisé peut accéder à sa guise aux données, indépendamment de la présence ou non des autres. Il est évident que le dispositif $\mathtt{A}_1$ peut récupérer les données en clair puisque les données sont chiffrées pour lui. Concernant les autres dispositifs, par exemple $\mathtt{A}_2$, le serveur ne peut tout simplement pas lui envoyer les données chiffrées qu'il stocke puisque $\mathtt{A}_2$ est incapable de les déchiffrer. Il utilise alors la clé de re-chiffrement de $\mathtt{A}_1$ vers $\mathtt{A}_2$ pour modifier les données chiffrées en de nouvelles à destination cette fois-ci de $\mathtt{A}_2$, qui pourra alors les déchiffrer. $\mathtt{A}_2$ est ainsi capable de récupérer les données à sa guise. Ce principe est indépendant de $\mathtt{A}_2$ et peut aussi bien être effectué pour $\mathtt{B}_1$. De cette manière, chaque dispositif autorisé peut récupérer les données lorsqu'il le souhaite.

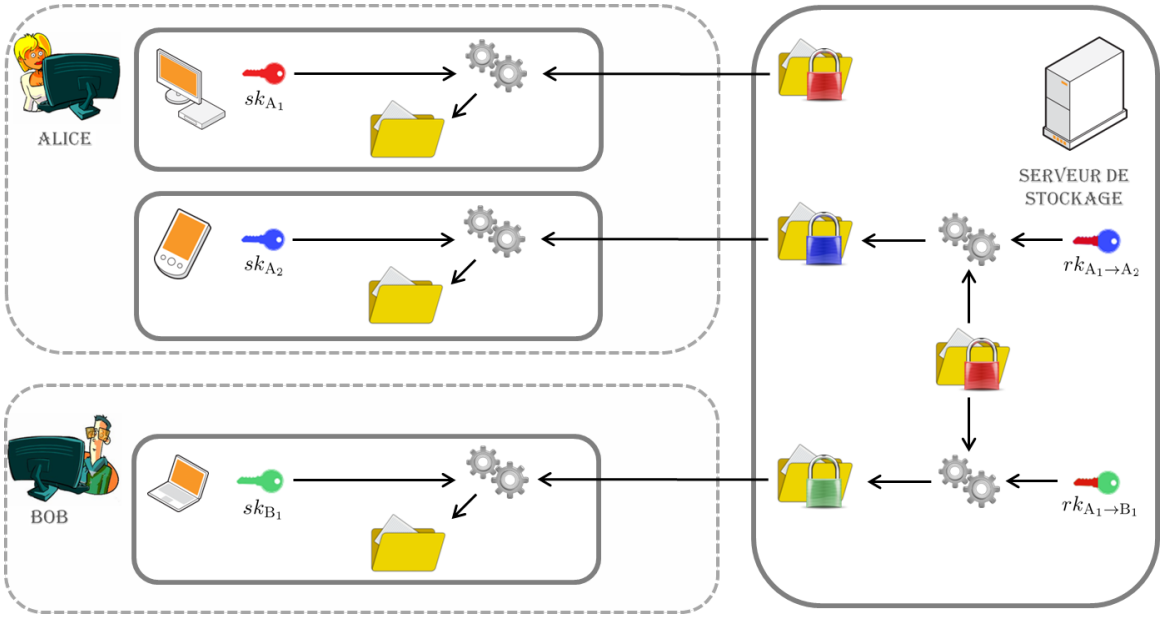

Figure 1.4 – Récupération des données

1.2.3 Particularités d'une telle solution

Nous venons de voir comment l'utilisation du serveur de re-chiffrement permet de concevoir un stockage de données fonctionnel, qui permet à la fois de stocker et de partager ses données avec des utilisateurs de confiance. Nous examinons maintenant les différentes caractéristiques de ce système.

Confidentialité des données. La confidentialité des données est assurée par le fait que les données sont chiffrées seulement pour les utilisateurs autorisés. Et par hypothèse, la partie re-chiffrement ne met pas cette confidentialité en péril.

Non divulgation des clés de déchiffrement. D'après les hypothèses énoncées sur un schéma de serveur de re-chiffrement, les clés de déchiffrement ne sont ici pas divulguées. En effet, nous utilisons un chiffrement à clé publique qui ne les dévoile pas. La partie re-chiffrement, quant à elle, ne révèle pas d'information sur ces clés. Ainsi une telle application ne consiste pas en une divulgation des clés de déchiffrement.

Confiance dans le serveur de stockage. Comme nous venons de le voir, la confiance dans le serveur de stockage n'est pas totale. Certes, nous ne lui donnons pas accès aux données en clair, mais il a tout de même un rôle important. Il doit évidemment stocker les données puis les envoyer aux différents utilisateurs. Mais il est également chargé, lors d'une requête d'un utilisateur, de potentiellement effectuer un re-chiffrement des données chiffrées. Pour que la solution puisse être fonctionnelle, il faut faire confiance au serveur afin qu'il réalise effectivement les opérations souhaitées. Ceci n'est en aucun cas un problème par rapport aux autres solutions d'informatique dématérialisée de stockage. En effet, dans tous ces systèmes, rien n'empêche le serveur de ne pas effectuer ce qu'on lui demande. Il peut toujours décider de ne pas envoyer les données demandées aux utilisateurs. Ce système basé sur le serveur de re-chiffrement ne requiert donc pas une plus grande confiance dans le serveur de stockage que les autres systèmes.

Dans le cadre du stockage de données basé sur le serveur de re-chiffrement, nous pouvons résumer la confiance dans le serveur de stockage ainsi : nous faisons confiance au serveur de stockage pour qu'il fasse ce que les utilisateurs lui demandent mais ne lui permettons pas d'obtenir les données en clair et ne lui laissons pas gérer les utilisateurs autorisés.

Capacité de stockage. Par rapport à un système de stockage habituel, le serveur stocke en plus des clés de re-chiffrement. Ces clés peuvent être nombreuses puisque si l'on considère n dispositifs, il y a alors au maximum $n.(n-1)$ clés de re-chiffrement envisageables si chaque utilisateur souhaite partager ses données avec tous les autres [3]. Même si cela peut faire beaucoup de clés de re-chiffrement possibles, leur nombre est indépendant de la quantité de données à stocker, ce qui est d'autant plus intéressant lorsque beaucoup de données sont stockées.

Concernant le stockage des données, si l'on souhaite que le serveur n'ait pas accès aux données en clair, il n'y a pas d'autre solution que de les chiffrer. Avec ce système, les données sont chiffrées pour une entité et une seule. De plus, il n'y a pas duplication de chiffrés comme c'est par exemple le cas dans certaines solutions standards évoquées dans la Section 1.1.2. Ainsi le stockage total n'est pas si important au regard des données à stocker et des fonctionnalités offertes par le système.

Stockage de gros fichiers - Utilisation du chiffrement hybride. Il est bien connu que le chiffrement à clé publique est moins adapté que celui à clé secrète lorsque l'on souhaite chiffrer de gros fichiers. La solution courante pour chiffrer de tels fichiers est d'utiliser un système de chiffrement hybride, aussi appelé KEM/DEM, de l'anglais *key encapsulation mechanism/data encapsulation mechanism*, formalisé par Cramer et Shoup dans [CS02, CS04]. Pour rappel, ce système consiste à utiliser deux chiffrements, un à clé publique et l'autre à clé secrète. Les fichiers sont chiffrés avec l'algorithme à clé secrète en utilisant une clé temporaire, appelée clé de session. Cette clé de session est, elle-même, chiffrée avec l'algorithme à clé publique. De cette manière, la seule donnée chiffrée avec un algorithme à clé publique est une clé secrète, qui est un élément de petite taille. Cela permet de bénéficier des fonctionnalités du chiffrement à clé publique tout en chiffrant de manière efficace avec un algorithme à clé secrète. Le chiffrement hybride est ainsi idéal pour protéger de gros fichiers.

Cette idée est naturellement compatible avec le serveur de re-chiffrement. Le fonctionnement va rester le même. On aura toujours une clé de session par fichier et seule cette clé sera chiffrée avec le serveur de re-chiffrement pour le dispositif autorisé à récupérer ce fichier. Cela permet d'utiliser le serveur de re-chiffrement sur tout type de données.

3. Nous verrons comment réduire le nombre de ces clés au Chapitre 7.

Non-duplication des données chiffrées. Pour autant, la capacité de stockage disponible du serveur peut rapidement devenir nulle. D'autant plus, si plusieurs utilisateurs souhaitent chacun stocker une même donnée, puisque dans ce cas, le serveur stocke plusieurs fois un chiffré de celle-ci. Le chiffrement hybride ne changerait rien a priori puisque la clé de session est générée aléatoirement et serait donc différente pour chaque utilisateur. La solution serait donc de choisir cette clé de session de manière à ce qu'elle soit la même pour chacun de ces chiffrés. Le chiffré symétrique de la donnée sera alors le même et le serveur n'aura à stocker que celui-ci ainsi que les n chiffrés asymétriques de la clé de session, ce qui permet un gain de stockage non négligeable. Douceur *et al.* ont exploré cette idée dans [DAB+02] puis récemment Bellare *et al.* dans [BKR13] où ils ont notamment proposé des chiffrement symétriques dont la clé secrète est générée à partir du message et dont le chiffré ne dévoile aucune information sur cette clé ou sur le message.

Dynamique des droits d'accès. Comme évoqué dans la description de la solution, la phase d'attribution des droits d'accès est indépendante de la phase de stockage des données. Cela permet de pouvoir ajouter, à n'importe quel moment, des utilisateurs. De cette manière, lorsqu'un propriétaire stocke ses données, il n'est pas obligé de savoir à cet instant précis qui il va autoriser à y accéder. Le système possède donc une certaine dynamique sur les droits d'accès, sur laquelle nous reviendrons dans le Chapitre 7.

Conclusion. Nous voyons, au travers de l'exemple présenté, que la primitive cryptographique de serveur de re-chiffrement permet la réalisation d'une application d'informatique dématérialisée de stockage. Cette solution offre tout un ensemble de propriétés intéressantes. De plus, elle n'est pas concernée par les invonvénients majeurs des solutions de base décrites dans la Section 1.1.2.

En particulier, cette primitive rend possible le partage des données entre différents utilisateurs. Cependant, le partage que nous avons considéré était très précis, nous n'avons partagé qu'une seule donnée entre trois dispositifs fixes. Il ne s'intéressait pas non plus à l'utilisation que les différents usagers vont en faire. Cet exemple permet seulement de comprendre l'idée générale qui sera utilisée dans cette application. Concrètement, nous verrons que, dans un cadre moins précis, la réalisation d'une telle application s'avèrera plus complexe. En effet, en considérant par exemple un nombre de dispositifs variant au cours du temps, les fonctionnalités offertes par les schémas existants de cette primitive ne seront pas suffisants. Nous nous intéresserons à ajouter d'autres fonctionnalités, en les intégrant directement dans la conception de la primitive. Le but sera d'obtenir, au final, une application possédant toutes les propriétés que nous venons de voir, qui soit la plus fonctionnelle possible pour l'utilisateur.

Chapitre 2

Introduction technique

Sommaire

2.1 Outils mathématiques 15
2.1.1 Groupe 15
2.1.2 Courbes elliptiques et couplages 16
2.1.3 Problèmes difficiles dans les groupes cycliques d'ordre premier 17
2.2 Fonctions cryptographiques 20
2.2.1 Fonction à sens unique 20
2.2.2 Fonction de hachage 20
2.3 Sécurité en cryptographie 21
2.3.1 Méthodologie de la sécurité prouvée 21
2.3.2 Modèles 21
2.4 Une primitive essentielle : le chiffrement 22
2.4.1 Sécurité prouvée pour la primitive de chiffrement 22
2.4.2 Le chiffrement à clé publique 23
2.4.3 Le chiffrement à clé secrète 25
2.5 Outils cryptographiques 27
2.5.1 Signature 27
2.5.2 Signature à usage unique 27
2.5.3 La construction CHK 27
2.5.4 La construction Fujisaki-Okamoto 28

Nous présentons dans ce chapitre les pré-requis mathématiques et cryptographiques dont nous aurons besoin tout au long de ce mémoire.

2.1 Outils mathématiques

2.1.1 Groupe

Définition 1 (Groupe). *Un groupe* $(\mathbb{G}, \cdot)$ *est la donnée d'un ensemble* $\mathbb{G}$ *et d'une loi de composition interne* $\cdot \ : \ \mathbb{G} \times \mathbb{G} \to \mathbb{G}$ *satisfaisant les propriétés suivantes :*

Associativité. $\forall\ (x, y, z) \in \mathbb{G}^3,\ (x \cdot y) \cdot z = x \cdot (y \cdot z)$

Elément neutre. $\exists\ e \in \mathbb{G}\ /\ \forall\ x \in \mathbb{G},\ x \cdot e = e \cdot x = x$

Symétrique. $\forall\ x \in \mathbb{G},\ \exists!\ y_x \in \mathbb{G}\ /\ x \cdot y_x = y_x \cdot x = e$

e est appelé *élément neutre* de $\mathbb{G}$ et est noté $1_{\mathbb{G}}$.

y_x est appelé *symétrique* de x. *Il* est noté $-x$ et appelé *opposé* dans le cas d'une loi additive et est noté x^{-1} et appelé *inverse* dans le cas d'une loi multiplicative. La notation multiplicative sera utilisée par défaut.

Remarque 1. S'il n'y a pas d'ambiguïté sur la loi de composition interne utilisée (notée $\cdot$ dans la définition précédente), nous désignerons tout simplement ce groupe par $\mathbb{G}$.

Un groupe $(\mathbb{G}, \cdot)$ est un *groupe commutatif* si $\forall\ (x, y) \in \mathbb{G}^2,\ x \cdot y = y \cdot x$. Il est dit *fini* si le cardinal de $\mathbb{G}$ est fini. Ce cardinal est alors appelé *ordre de* $(\mathbb{G}, \cdot)$. Par la suite, nous ne considérerons que des groupes finis commutatifs.

Notations. $\forall\ (x, n) \in \mathbb{G} \times \mathbb{N}^*,\ x^n := \underbrace{x \cdot x \cdot \ \ldots\ \cdot x}_{n \text{ times}}$, $x^{-n} := (x^n)^{-1}$ et $x^0 := 1_{\mathbb{G}}$.

Soit $x \in \mathbb{G}$, la fonction $n \in \mathbb{N} \mapsto x^n$ est appelée *exponentielle en base* x.

Un élément x de $(\mathbb{G}, \cdot)$ est dit *générateur* si $\forall\ y \in \mathbb{G},\ \exists\ n \in \mathbb{N}\ /\ y = x^n$. Un groupe $(\mathbb{G}, \cdot)$, dont l'ordre est premier, a la particularité que tout élément différent de l'élément neutre est générateur. Sauf mention du contraire, nous utiliserons tout au long de ce mémoire des groupes d'ordre premier.

2.1.2 Courbes elliptiques et couplages

Les courbes eliptiques sont des cubiques lisses, *i.e* possédant une tangente en chaque point de la courbe. Elles sont très utilisées en mathématiques et ont par exemple permis à Wiles [1] de démontrer le dernier théorème de Fermat. Elles le sont également en cryptographie où, bien que souvent représentées comme des courbes algébriques affines, elles sont en réalité projectives. Cette particularité permet de considérer d'éventuels points à l'infini [2] afin que l'ensemble des points d'une telle courbe forme un groupe. Ces courbes présentent l'avantage, par rapport aux groupes multiplicatifs d'un corps fini, de posséder, pour un même niveau de sécurité, des éléments de taille plus petite. Cette particularité vient du fait qu'on ne dispose pas d'algorithmes pour calculer le logartihme discret, qui soient plus efficaces que les algorithmes génériques (fonctionnant sur n'importe quel groupe), contrairement aux groupes multiplicatifs d'un corps fini avec l'algorithme de calcul d'index présenté dans sa forme actuelle par Adleman dans [Adl79].

Elles ont également permis l'exploration d'une nouvelle branche importante de la cryptographie, celle basée sur les couplages qui a été initiée par Joux avec son protocole tripartite d'échange de clés [Jou00]. Ces couplages sont utilisés dans de nombreuses primitives comme par exemple le chiffrement basé sur l'identité dont nous parlerons en fin de mémoire (page 129).

Nous donnons ici la définition d'un couplage défini sur des groupes d'ordre p premier tel que nous l'utiliserons par la suite [3].

Définition 2 (Couplage). *Soit p un nombre premier et $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ trois groupes d'ordre p. Un couplage est une application $e\ :\ \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ satisfaisant les propriétés suivantes :*

Bilinéarité. $\forall\ (g_1, g_2) \in \mathbb{G}_1 \times \mathbb{G}_2,\ \forall\ (a, b) \in \mathbb{N}^2,\ e(g_1^a, g_2^b) = e(g_1^b, g_2^a) = e(g_1, g_2)^{ab}$

Non-dégénérescence. $\forall\ (g_1, g_2) \in \mathbb{G}_1 \times \mathbb{G}_2,\ e(g, h) = 1_{\mathbb{G}_T} \implies g_1 = 1_{\mathbb{G}_1} \vee g_2 = 1_{\mathbb{G}_2}$

1. Voir [Wil95].
2. Contrairement à une idée courante, une courbe elliptique peut posséder plusieurs points à l'infini comme c'est le cas des courbes présentées dans [JTV10] ou de leurs versions binaires que nous avons étudiées avec Marc Joye dans [DJ11], toutes basées sur le modèle de Huff [Huf48].
3. Il existe d'autres couplages utilisés en cryptographie comme ceux établis sur des groupes d'ordre composite $p \cdot q$, cf [BGN05].

Remarque 2. En cryptographie, nous ne considérerons que des couplages qui sont calculables par un algorithme fonctionnant en temps polynomial dont la définition est donnée dans la section suivante[4].

$(p, e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ est appelé *environnement bilinéaire*. Il est dit *asymétrique* si $\mathbb{G}_1 \neq \mathbb{G}_2$ et *symétrique* si $\mathbb{G}_1 = \mathbb{G}_2 := \mathbb{G}$.

2.1.3 Problèmes difficiles dans les groupes cycliques d'ordre premier

Nous présentons dans cette section les problèmes difficiles que nous utiliserons au cours de ce mémoire et sous lesquels les schémas présentés seront prouvés sûrs. Ce sont en réalité des problèmes algorithmiques, qui supposent qu'aucun algorithme fonctionnant en "un temps pas trop important" n'ait "une chance trop importante" de les résoudre. Nous utilisons une probabilité discrète[5] pour qualifier le succès d'un tel algorithme. Les notions de "temps pas trop important" et "chance trop importante" sont, quant à elle, caractérisées par les notions d'algorithmes polynomiaux et de fonctions négligeables ainsi définies :

Définition 3 (**Algorithme polynomial**). *Un algorithme, dont la taille des entrées est paramétrée par une fonction* `param` *définie sur* $\mathbb{N}$*, est dit polynomial s'il existe un polynôme* $t \in \mathbb{R}[X]$*, tel que pour tout* $\lambda \in \mathbb{N}$*, le temps d'éxécution de l'algorithme sur une entrée paramétrée par* `param`(λ) *soit majoré par* $|t|$*. Par la suite,* λ *sera appelé* paramètre de sécurité.

Définition 4 (**Fonction négligeable**). *Une fonction* $\epsilon : \mathbb{N} \to \mathbb{R}$ *est dite négligeable si :*

$$\forall\ P \in \mathbb{R}[X],\ \exists\ N \in \mathbb{N}\ /\ \forall\ \lambda \geq N,\ |\epsilon(\lambda)| \leq \frac{1}{|P(\lambda)|}$$

Nous dirons qu'une fonction f est *à distance négligeable de* x pour $x \in \mathbb{R}$ si la fonction $y \mapsto f(y) - x$ est négligeable.
Nous dirons qu'une opération (ou problème) est *difficile* (en réalité asymptotiquement difficile) si la probabilité de succès de cette opération (ou problème) est majorée par la valeur absolue d'une fonction négligeable.

Définition 5 (**Problème algorithmique difficile**). *Soit* t *un polynôme et* ϵ *une fonction négligeable, un problème algorithmique est dit* (t, ϵ)*-difficile dans un environnement généré par un algorithmique* $\mathcal{G}$*, dont la taille des entrées est paramétrée par une fonction* `param`*, si pour tout* $\lambda \in \mathbb{N}$ *(appelé paramètre de sécurité), la probabilité de résolution du problème, dans un environnement généré par* $\mathcal{G}$ *sur une entrée paramétrée par* λ*, par n'importe quel algorithme s'exécutant en temps au plus* $t(\lambda)$ *est inférieure ou égale à* $|\epsilon(\lambda)|$*.*

Dans les problèmes que nous aborderons, l'environnement sera soit un groupe $\mathbb{G}$ d'ordre p, soit un environnement bilinéaire $(p, e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t)$, où p est calculé à partir du paramètre de sécurité λ. Étant donné que cette notion de problème algorithmique difficile est liée à celle de fonction négligeable, le problème ne sera réellement difficile qu'asymptotiquement et dépendra de la puissance de calcul disponible, ainsi que de l'algorithme $\mathcal{G}$ générant l'environnement. Il ne faut donc pas considérer un paramètre de sécurité trop petit. En pratique, l'ANSSI recommande dans [ANS10], pour une utilisation ne dépassant pas 2020, d'utiliser par exemple des groupes dont l'ordre est un nombre premier d'au moins 200 bits dans le cas du problème du logarithme discret.

Nous présentons maintenant les problèmes algorithmiques difficiles que nous utiliserons tout au long de ce mémoire. Le premier est bien évidemment celui du logarithme discret sur lequel sont basés tous les problèmes algorithmiques sur les groupes cycliques d'ordre premier.

4. Cf [CFA+05] pour une bonne référence sur le sujet.
5. Voir [BL07] pour plus de détails.

Problème du logarithme discret. En considérant la fonction exponentielle en base g, non plus définie sur $\mathbb{N}$ mais sur $\mathbb{Z}_p$ par : $\bar{x} \in \mathbb{Z}_p \mapsto g^x$, où x est un élément dont la classe modulo p est $\bar{x}$. Si p est notamment l'ordre de $\mathbb{G}$, cette fonction est bien définie, car $g^p = 1_{\mathbb{G}}$ d'après le théorème de Lagrange. Si en plus, g est générateur de $\mathbb{G}$, elle est alors bijective de $\mathbb{Z}_p$ dans $\mathbb{G}$. Il est donc possible de considérer sa fonction réciproque que l'on appelle *logarithme discret en base g*. Bien que le calcul de la fonction exponentielle en base g puisse être réalisé par un algorithme polynomial, il existe des groupes pour lesquels sa fonction réciproque ne peut pas l'être. Les groupes multiplicatifs de corps finis ou les courbes elliptiques en sont des exemples typiques, couramment utilisés en cryptographie. Dans de tels cas, nous dirons que le problème du logarithme discret est un problème algorithmique difficile dont la définition formelle est donnée ci-dessous.

Définition 6 (DL)**.** *Le problème du* logarithme discret*, noté* (DL)*, dans un groupe $\mathbb{G}$ d'ordre p pour un élément g générateur de $\mathbb{G}$ consiste à calculer, à partir de g^a, l'élément a qui est généré aléatoirement dans $\mathbb{Z}_p^*$. La probabilité de résolution de ce problème pour un algorithme (ou adversaire) $\mathcal{A}$ est exprimée par :*

$$\mathtt{Pr}\Big[a \xleftarrow{\$} \mathbb{Z}_p^* \ ; \ a' \leftarrow \mathcal{A}(p, \mathbb{G}, g, g^a) \ ; \ a' = a\Big].$$

À partir de ce problème, nous pouvons en définir d'autres dont nous en énonçons deux types. Les premiers sont dits calculatoires où l'on demande de *calculer* une solution à une instance donnée. Dans ce cas, la probabilité de résolution du problème est similaire à celle que nous venons d'exposer. La deuxième catégorie correspond aux problèmes dits décisionnels, dont le but est de *décider* si une solution potentielle d'un problème calculatoire pour une instance donnée en est bien la solution [6]. La probabilité de résolution d'un problème décisionnel est décrite par la différence en valeur absolue des probabilités de succès d'une instantiation du problème calculatoire sur deux solutions potentielles différentes : l'une étant la solution valide du problème calculatoire et l'autre étant générée aléatoirement. Par exemple, pour le problème décisionnel DBDH que nous verrons ci-dessous, la probabilité de résolution est égale à :

$$\Big|\mathtt{Pr}\Big[a,b,c \xleftarrow{\$} \mathbb{Z}_p^* \ ; \ 1 \leftarrow \mathcal{A}\big((p,e,\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_t), g_1, g_2, g_2^a, g_2^b, g_2^c, e(g_1,g_2)^{abc}\big)\Big] - \mathtt{Pr}\Big[a,b,c,d \xleftarrow{\$} \mathbb{Z}_p^* \ ; \ 1 \leftarrow \mathcal{A}\big((p,e,\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_t), g_1, g_2, g_2^a, g_2^b, g_2^c, e(g_1,g_2)^{d}\big)\Big]\Big|.$$

Problèmes calculatoires.

Définition 7 (CDH)**.** *Le problème de* Diffie-Hellman calculatoire*, noté* (CDH)*, dans un groupe $\mathbb{G}$ d'ordre p pour un élément g générateur de $\mathbb{G}$ consiste à calculer, à partir de (g^a, g^b) où a et b sont choisis aléatoirement dans $\mathbb{Z}_p^*$, l'élément g^c où $c = ab$.*

Ce problème fut introduit par Diffie et Hellman pour prouver la sécurité de leur protocole d'échange de clés [DH76]. En considérant une relation différente et éventuellement des valeurs supplémentaires, il est possible d'en obtenir de nombreuses variantes.

Définition 8 (DCDH)**.** *Le problème* Diffie-Hellman calculatoire divisible*, noté* (DCDH)*, dans un groupe $\mathbb{G}$ d'ordre p pour un élément g générateur de $\mathbb{G}$ consiste à calculer, à partir de (g^a, g^b) où a et b sont choisis aléatoirement dans $\mathbb{Z}_p^*$, l'élément g^c où $c = b/a$.*

Dans [BDZ03], cette hypothèse est prouvée être équivalente à l'hypothèse CDH exposée ci-dessus. Nous l'utiliserons au Chapitre 6, plutôt que cette dernière, afin de mieux comprendre la preuve du Théorème 34.

6. Tout problème calculatoire n'a pas forcément une version décisionnelle difficile.

Définition 9 (DHI). *Le problème d'*inverion de Diffie-Hellman calculatoire*, noté* (DHI)*, dans un groupe* $\mathbb{G}$ *d'ordre p pour un élément g générateur de* $\mathbb{G}$ *consiste à calculer, à partir de* g^a *où a est choisi aléatoirement dans* $\mathbb{Z}_p^*$*, l'élément* g^b *où* $b = 1/a$*.*

Dans [BDZ03], il est prouvé être équivalent au problème SCDH introduit par Maurer et Wolf dans [MW96], qui traduit la difficulté de calculer g^c à partir de g^a, où $c = a^2$.

Définition 10 (2-3-CDH). *Le problème* Diffie-Hellman calculatoire divisible d'ordre 2-3*, noté* (2-3-CDH)*, dans un groupe* $\mathbb{G}$ *d'ordre p pour un élément g générateur de* $\mathbb{G}$ *consiste à calculer, à partir de* (g^a, g^b) *où a et b sont choisis aléatoirement dans* $\mathbb{Z}_p^*$*, un couple* (C, D) *où* $C \neq 1_{\mathbb{G}}$ *et* $D = C^{ab}$*.*

Ce problème fut introduit par Kunz-Jacques et Pointcheval dans [KJP06].

Définition 11 (BDH). *Le problème* Diffie-Hellman bilinéaire*, noté* (BDH)*, dans un environnement bilinéaire* $(p, e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ *pour deux éléments* g_1 *et* g_2*, générateurs respectifs de* $\mathbb{G}_1$ *et de* $\mathbb{G}_2$ *consiste à trouver, à partir de* (g_2^a, g_2^b, g_2^c) *où a, b, et c sont choisis aléatoirement dans* $\mathbb{Z}_p^*$*, l'élément* $e(g_1, g_2)^{abc}$*.*

Ce problème fut introduit par Joux dans [Jou04] dans le cas où $\mathbb{G}_1 = \mathbb{G}_2$.

Problèmes décisionnels.

Définition 12 (V-DDH). *Le problème* V-Diffie-Hellman décisionnel*, noté* (V-DDH)*, dans un groupe* $\mathbb{G}$ *d'ordre p pour un élément g générateur de* $\mathbb{G}$ *consiste à décider, à partir de* (g^a, g^{ab}, g^{ac}, T) *où a, b et c sont choisis aléatoirement dans* $\mathbb{Z}_p^*$*, si* $T = g^{bc}$*.*

Ce problème fut introduit par Mao et Ao dans [MA09a].

Définition 13 (DBDH). *Le problème* Diffie-Hellman bilinéaire décisionnel*, noté* (DBDH)*, dans un environnement bilinéaire* $(p, e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ *pour deux éléments* g_1 *et* g_2 *générateurs respectifs de* $\mathbb{G}_1$ *et de* $\mathbb{G}_2$ *consiste à décider, à partir de* (g_2^a, g_2^b, g_2^c, T) *où a, b et c sont choisis aléatoirement dans* $\mathbb{Z}_p^*$*, si* $T = e(g_1, g_2)^{abc}$*.*

Ce problème correspond à la version décisionnelle du problème BDH.

Définition 14 (Aug-DBDH). *Le problème* Diffie-Hellman bilinéaire décisionnel augmenté*, noté* (Aug-DBDH)*, dans un environnement bilinéaire* $(p, e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ *pour deux éléments* g_1 *et* g_2 *générateurs respectifs de* $\mathbb{G}_1$ *et de* $\mathbb{G}_2$ *consiste à décider, à partir de* $(g_2^a, g_2^b, g_2^c, g_2^{a^2 b}, T)$ *où a est choisi aléatoirement dans* $\mathbb{Z}_p^*$*, si* $T = e(g_1, g_2)^{abc}$*.*

Ce problème fut introduit par Libert et Vergnaud dans [LV08a].

Définition 15 (q-DBDHI). *Le problème d'*inversion bilinéaire de Diffie-Hellman décisionnel d'ordre q*, noté* ($q-$DBDHI)*, dans un environnement bilinéaire symétrique* $(p, e, \mathbb{G}, \mathbb{G}_T)$ *pour g générateur de* $\mathbb{G}$ *consiste à décider, à partir de* $(g^a, \ldots, g^{(a^q)}, T)$ *où a est choisi aléatoirement dans* $\mathbb{Z}_p^*$*, si* $T = e(g, g)^{1/a}$*.*

Ce problème fut introduit par Boneh et Boyen dans [BB04].

Définition 16 (3-QDBDH). *Le problème* Diffie-Hellman bilinéaire décisionnel avec quotient d'ordre 3*, noté* (3-QDBDH)*, dans un environnement bilinéaire symétrique* $(p, e, \mathbb{G}, \mathbb{G}_T)$ *pour g générateur de* $\mathbb{G}$ *consiste à décider, à partir de* $(g^a, g^{a^2}, g^{a^3}, g^b, T)$ *où a et b sont choisis aléatoirement dans* $\mathbb{Z}_p^*$*, si* $T = e(g, g)^{b/a}$*.*

Ce problème fut introduit par Libert et Vergnaud dans [LV08b] où ils montrèrent qu'il est plus difficile que le problème 3-DBDHI.

Définition 17 (n-DBDHE)**.** *Le problème d'*exposant Diffie-Hellman bilinéaire décisionnel d'ordre n*, noté* (n-DBDHE)*, dans un environnement bilinéaire symétrique* $(p, e, \mathbb{G}, \mathbb{G}_T)$ *pour* g *et* h *générateurs de* $\mathbb{G}$ *consiste à décider, à partir de* $(g^a, g^{a^2}, \ldots, g^{a^n}, g^{a^{n+2}}, \ldots, g^{a^{2n}}, T)$ *où* a *est choisi aléatoirement dans* $\mathbb{Z}_p^*$*, si* $T = e(g, h)^{a^{n+1}}$.

Ce problème fut introduit par Boneh *et al.* dans [BBG05].

Définition 18 ((P, Q, f)-GBDDHE)**.** *Soient* $p, s \in \mathbb{N}^*$ *et* $P = (p_1, \ldots, p_s)$ *et* $Q = (q_1, \ldots, q_s)$*, deux* s*-upplets de polynômes de* $\mathbb{Z}_p[X_1, X_n]$ *avec* $p_1 = q_1 = 1$*. Soient de plus* $a_{i,j}, b_j \in \mathbb{Z}_p$ *pour* $(i, j) \in [\![1; s]\!]$ *et* $f = \sum_{i,j} a_{i,j} \cdot p_i \cdot p_j + \sum_i b_i \cdot q_i$*. Le problème d'*exposant Diffie-Hellman général (P, Q, f) *dans un environnement bilinéaire symétrique* $(p, e, \mathbb{G}, \mathbb{G}_T)$ *pour* g *générateur de* $\mathbb{G}$ *et* $h = e(g, g)$*, noté* ((P, Q, f)-GBDDHE)*, consiste à décider, à partir de* $(g^{p_1(x)}, \ldots, g^{p_s(x)}, h^{q_1(x)}, \ldots, h^{q_s(x)}, T)$ *où* x *est choisi aléatoirement parmi* $\mathbb{Z}_p^n$*, si* $T = g^{f(x)}$.

Ce problème fut introduit par Delerablée dans [Del07].

2.2 Fonctions cryptographiques

Nous présentons rapidement deux types de fonctions couramment utilisées en cryptographie, à savoir les fonctions à sens unique et les fonctions de hachage [7].

2.2.1 Fonction à sens unique

Une fonction est dite *à sens unique* s'il est facile de calculer l'image d'un élément mais difficile de calculer un antécédent d'une image choisie aléatoirement. La facilité de calcul d'une image est exprimée par l'existence d'un algorithme polynomial pouvant effectuer ce calcul. Quant à la difficulté de calculer un inverse, elle est caractérisée par une probabilité de succès du calcul majorée par la valeur absolue d'une fonction négligeable. L'exemple typique de fonction que nous considérerons comme étant à sens unique et que nous utiliserons est la fonction exponentielle dont nous avons déjà parlé. Cette fonction est à sens unique sous l'hypothèse DL.

Nous dirons également qu'une fonction à sens unique est *à trappe* s'il existe, comme son nom l'indique, une trappe (ou un secret), dont la possession permet de calculer facilement un antécédent d'une image choisie aléatoirement, exprimée aussi par l'existence d'un algorithme polynomial réalisant le calcul. Le chiffrement par exemple est une fonction à trappe où la trappe est la clé de déchiffrement.

2.2.2 Fonction de hachage

Une *fonction de hachage* est une fonction à sens unique qui permet de transformer n'importe quelle chaîne de bits en une empreinte de taille fixe. Une fonction de hachage $\mathcal{H}$ est considérée comme *cryptographiquement sûre* si elle possède les propriétés suivantes :

Résistance à la pré-image. Étant donné y appartenant à l'image de $\mathcal{H}$, il doit être difficile de trouver $x \in \{0, 1\}^*$ tel que $\mathcal{H}(x) = y$.

Résistance à la seconde pré-image. Étant donné $x \in \{0, 1\}^*$, il doit être difficile de trouver $x' \in \{0, 1\}^* \setminus \{x\}$ tel que $\mathcal{H}(x') = \mathcal{H}(x)$.

Pour nos besoins, nous supposerons aussi que les fonctions de hachage possèdent une propriété supplémentaire appelée *résistance aux collisions* et définie comme suit.

Résistance aux collisions. Il doit être difficile de trouver deux éléments distincts x et x' appartenant à $\{0, 1\}^*$ tel que $\mathcal{H}(x) = \mathcal{H}(x')$. Cette propriété implique notamment la résistance à la seconde pré-image.

7. Une description plus détaillée peut être trouvée dans [KL07].

Les fonctions de hachage les plus connues sont les fonctions MD-5 [Riv92] et SHA-1 [NIS95]. Leur sécurité est cependant ébranlée et il est recommandé d'utiliser des fonctions de la famille SHA-2 [NIS02] ou même la fonction Keccak [BDPA11] élue récemment comme la gagnante de la compétition SHA-3 organisée par le NIST (National Institute of Standards and Technology).

2.3 Sécurité en cryptographie

En cryptographie, on cherche à prouver (quand c'est possible) la sécurité des algorithmes que l'on met au point. En cryptographie asymétrique, elle est même systématique. On parle alors de *sécurité prouvée*. Cette démarche ne se fait pas au hasard et suit un déroulement précis.

2.3.1 Méthodologie de la sécurité prouvée

Pour évaluer la sécurité, il nous faut tout d'abord définir une *primitive cryptographique* correspondant à l'ensemble d'algorithmes considérés et décrire les propriétés qu'ils doivent vérifier. Ainsi dans le cadre de ce mémoire, nous étudierons la primitive cryptographique de *serveur de re-chiffrement* dont la définition est donnée au Chapitre 3. Une implémentation spécifique des algorithmes décrits par cette primitive s'appelle un *schéma*. C'est donc la sécurité d'un schéma que l'on souhaite prouver.

On fait ensuite interagir les algorithmes de la primitive considérée avec d'autres algorithmes, que l'on appelle *adversaires*, ce qui permet de définir un modèle de sécurité. Selon les fonctionnalités du schéma et le niveau de sécurité désiré, différents types d'interactions pourront être considérés. Plus un modèle considérera de types d'interactions possibles entre les adversaires et les algorithmes de la primitive, meilleure sera la robustesse de la sécurité, le but étant d'obtenir un modèle où toutes les interactions possibles (dépendant des fonctionnalités souhaitées) soient prises en compte. Ce modèle précise également une condition de *gain* décrivant la situation où les interactions s'arrêtent et des conditions pour que les adversaires remporte le gain. Au cours de ce mémoire, ce gain correspondra au fait que les adversaires aient été capable de discerner quel message a été chiffré.

Une fois que l'on dispose du modèle de sécurité, on décrit les interactions en utilisant le schéma considéré (les implémentations des algorithmes). Il n'est pas nécessaire de connaître le fonctionnement interne des adversaires, mais seulement de savoir quel type d'interaction il peut disposer avec les algorithmes de la primitive. Le schéma sera alors dit sûr s'il n'y a pas des adversaires remportant le gain avec un avantage non négligeable. Le but est donc de prouver, sous certaines conditions, qu'il n'en existe pas. Nous sommes alors amenés à faire une preuve par réduction, dans laquelle on simule la réponse aux interactions possibles, que l'on appellera *oracles*, de manière à ce que les adversaires ne s'aperçoivent pas de la supercherie [8], afin de pouvoir l'utiliser pour résoudre un certain problème algorithmique difficile. On parle alors d'utilisation *en boîte noire* des adversaires. Le problème considéré étant difficile, cela signifie qu'il n'existe pas de tels adversaires et donc que le schéma est sûr pour le modèle de sécurité considéré.

2.3.2 Modèles

Nous utiliserons dans ce mémoire les deux modèles les plus courants pour prouver la sécurité de nos schémas : le modèle de l'oracle aléatoire et le modèle standard. Nous ne mentionnerons le modèle utilisé que dans les théorèmes et les preuves que nous exposerons. Nous invitons ainsi le lecteur curieux à se référer aux articles que nous citerons pour en découvrir le modèle utilisé.

8. Comme nous le verrons au Chapitre 6, cette particularité est très importante et sans elle, on ne peut conclure que le schéma est sûr ou pas.

Modèle de l'oracle aléatoire. Ce modèle fut introduit par Fiat et Shamir dans [FS86] et formalisé dans [BR93] par Bellare et Rogaway. L'idée de ce modèle est de considérer qu'une fonction retourne des valeurs parfaitement aléatoires. Dans les preuves de sécurité, elle sera remplacée par un oracle aléatoire retournant un aléa que l'on maîtrisera. Bien qu'idéalisé, cet oracle reste une fonction. Il doit donc être déterministe, *i.e* retourner la même valeur sur deux entrées identiques. Dans le cas de la cryptographie à clé publique, on idéalise une fonction de hachage. Dans le cas de la cryptographie à clé secrète, on peut être amené à considérer d'autres fonctions comme fonctions idéales. Dans [LPS12], c'est par exemple une permutation qui est modélisée comme oracle aléatoire.

Des travaux ([CGH98, CGH04, BBP04]) ont étudié la fragilité de ce modèle de l'oracle aléatoire. Ils ont prouvé dans le cas de la cryptographie à clé publique, qu'il était possible de concevoir un schéma qui soit sûr dans ce modèle mais qui ne le serait plus si on considérant n'importe quelle instantiation de fonction de hachage à la place de l'oracle aléatoire. Pour autant, ce modèle permet de prouver facilement la sécurité de nombreux schémas efficaces et reste donc très utilisé.

Modèle standard. Le modèle standard ne fait aucune idéalisation spécifique, contrairement au modèle de l'oracle aléatoire. Il ne considère que les propriétés induites par les éléments pris en compte, comme par exemple le fait qu'une fonction de hachage puisse être résistante aux collisions, sa sortie ne pouvant plus être choisie aléatoirement. Un schéma sûr dans ce modèle possède donc une sécurité plus forte, puisque celle-ci ne reposera que sur des hypothèses calculatoires liées à des problèmes algorithmiques difficiles. En particulier, les implémentations resteront sûres tant que les fonctions considérées possèdent les propriétés annoncées, comme par exemple celle d'être sans collisions pour une fonction de hachage.

2.4 Une primitive essentielle : le chiffrement

Le chiffrement est sans doute la primitive cryptographique la plus connue. Elle permet d'assurer la confidentialité des données dont nous avons déjà parlée au Chapitre 1. Elle sera donc au cœur de notre étude. Il existe deux grandes catégories ainsi décrites comme suit.

- Le chiffrement asymétrique, qui permet d'envoyer des données à un utilisateur sans disposer de secret en commun avec lui. Le chiffrement des données s'effectue avec la clé publique de l'utilisateur à qui l'on souhaite les envoyer. Ce dernier peut alors les récupérer avec sa clé privée.
- Le chiffrement symétrique, qui permet d'envoyer des données à un utilisateur avec qui l'on dispose d'un secret en commun. Ce secret en commun est appelé clé secrète et est utilisé lors du chiffrement et du déchiffrement.

Comme expliqué dans [MVO96], le chiffrement est une fonction à sens unique avec trappe où la trappe sera désormais appelée clé. Selon le cas considéré, ce sera une clé secrète ou un clé privée.

2.4.1 Sécurité prouvée pour la primitive de chiffrement

Une grande partie de notre étude utilisera des primitives de chiffrement. Nous nous intéresserons donc à la sécurité prouvée sur ce type de primitive. L'adversaire peut vouloir obtenir différents objectifs, ce qui donne lieu à des sécurités différentes décrites de la plus faible à la plus forte :

1. Il peut chercher à recouvrer la trappe (la clé) qui permettra alors de récupérer toutes les données chiffrées. C'est le but le plus difficile, puisque s'il y arrive, l'adversaire aura alors entièrement cassé le schéma. Il est le but historique sur lequel se focalisait les premières cryptanalyses.

2. Il peut chercher à inverser la fonction à sens unique sans avoir nécessairement retrouvé la trappe. S'il y arrive, il peut alors retrouver aussi la donnée chiffrée.
3. Il peut chercher à obtenir de l'information sur la donnée chiffrée. Contrairement aux deux précédents objectifs, ce dernier n'a pas comme but de retrouver la donnée chiffrée mais seulement d'en obtenir une certaine information. Cette sécurité fut introduite par Goldwasser et Micali dans [GM82] et est appelée *sécurité sémantique*. Elle fut ensuite formalisée par les mêmes auteurs dans [GM84] où le but décrit de l'adversaire est de pouvoir distinguer, à partir de deux messages m_0 et m_1, celui qui a été chiffré par le schéma considéré.
4. Il peut chercher à modifier un chiffré C, afin de produire un chiffré C' qui vérifie une certaine relation avec C. Cette sécurité, appelée *non-malléabilité*, fut exposée dans [DDN91] par Dolev *et al.* puis reformulée dans [BDPR98] par Bellare *et al.*.

L'adversaire peut également disposer de diverses attaques. Les différentes attaques que nous considérerons se regroupent en deux catégories. Dans une *attaque à clair choisi*, de l'anglais *chosen plaintext attack* et notée CPA, l'adversaire dispose de certaines données fixes et d'un moyen d'obtenir le chiffré des messages de son choix. Ces données fixes peuvent être les paramètres publics du schéma et, dans le cas du chiffrement asymétrique, des clés publiques. Dans le cas du serveur de re-chiffrement, nous considérerons également des clés de re-chiffrement qui ne sont pas a priori des données publiques. Une *attaque à chiffré choisi*, de l'anglais *chosen ciphertext attack* et notée CCA, offre des moyens supplémentaires à l'adversaire. En effet, il va avoir à sa disposition ce que l'on appelle des *oracles* qui pourront l'aider à atteindre son objectif. Habituellement, ces oracles sont dits *de déchiffrement* et lui permettent d'obtenir le clair correspondant à un chiffré de son choix, sous certaines conditions que nous détaillerons par la suite. Dans le cas du serveur de re-chiffrement, il disposera d'oracles supplémentaires dits *de re-chiffrement* qui lui permetteront d'obtenir le chiffré résultant d'un re-chiffrement de son choix, également sous certaines conditions.

Il a été démontré, dans [DDN91, BDPR98], que la sécurité sémantique est équivalente dans le cas de l'attaque CCA à la non-malléabilité. Cette première étant la plus facile à caractériser, nous nous focaliserons sur elle dans toute la suite du mémoire. Nous utiliserons sa notation habituelle, à savoir IND, pour *indiscernabilité de messages chiffrés*, ce qui nous donnera une sécurité notée IND-CCA.

Nous décrivons maintenant les deux grandes catégories de chiffrement que nous venons d'évoqués. Pour chacun, nous parlerons également de leur consistance qui caractérise le bon fonctionnement d'un schéma, *i.e* que tout message correctement chiffré doit être correctement déchiffré. Nous étudierons aussi les modèles de sécurité respectifs pour la sécurité IND-CCA.

2.4.2 Le chiffrement à clé publique

La particularité de cette primitive est que la clé qui sert à rendre l'information confidentielle n'est pas la même que celle qui sert à la récupérer.

Définition 19 (**Chiffrement asymétrique ou (à clé publique)**)**.** *Un* schéma de chiffrement à clé publique PUB *est la donnée d'un espace des messages $\mathcal{M}$, et d'un ensemble de quatre algorithmes* {Initialiser, Clé, Chiffrer, Déchiffrer} *décrits comme suit.*

- Initialiser$(\lambda) \to \mathcal{P}$ *: prend en entrée un paramètre de sécurité $\lambda \in \mathbb{N}$ et retourne les paramètres publics $\mathcal{P}$ du schéma qui sont donnés en entrée de tous les autres algorithmes.*
- Clé$(\mathcal{P})$ *: prend en entrée les paramètres publics $\mathcal{P}$ et retourne une paire de clés (privée/publique) (sk, pk), telle que pk permette de chiffrer et sk de déchiffrer.*
- Chiffrer$(\mathcal{P}, pk, m)$ *: prend en entrée les paramètres publics $\mathcal{P}$, une clé publique pk et un message m appartenant à $\mathcal{M}$ et retourne un chiffré C_{pk} pour la clé publique pk.*

$\mathsf{Déchiffrer}(\mathcal{P}, sk, C_{pk})$ *: prend en entrée les paramètres publics* $\mathcal{P}$*, une clé privée* sk *et un chiffré* C_{pk} *pour la clé publique* pk *associée à la clé privée* sk *et retourne un message* m *ou un symbole* $\perp$ *signifiant que le chiffré n'est pas valide.*

Consistance. Dans un schéma de chiffrement à clé publique, un message chiffré avec une clé publique se déchiffre avec la clé privée correspondante. Ainsi, pour tous les paramètres publics générés par l'algorithme Initialiser, pour toute paire de clés (privée/publique) (sk, pk) générée avec l'algorithme Clé et pour tout message $m \in \mathcal{M}$, la propriété suivante doit être vérifiée :

$$\mathsf{Déchiffrer}(\mathcal{P}, sk, \mathsf{Chiffrer}(\mathcal{P}, pk, m)) \rightarrow m.$$

Scénario d'attaque. Le scénario d'attaque contre une indiscernabilité de chiffrés entre un adversaire et un challengeur pour un schéma de chiffrement à clé publique est décrit ci-dessous.

Phase d'initialisation

- Le challengeur initialise les paramètres publics $\mathcal{P}$ du système en fonction d'un paramètre de sécurité λ via Initialiser et les donne à l'adversaire.
- A la demande de l'adversaire portant sur la création d'une entité $\mathtt{A}$, il génère une paire de clés $(sk_{\mathtt{A}}, pk_{\mathtt{A}})$ et répond de la manière suivante :
  - si la requête porte sur la création d'une entité $\mathtt{A}$ honnête, le challengeur donne $(pk_{\mathtt{A}})$ à l'adversaire ;
  - si la requête porte sur la création d'une entité $\mathtt{A}$ corrompue, le challengeur donne $(sk_{\mathtt{A}}, pk_{\mathtt{A}})$ à l'adversaire.

Phase 1

- L'adversaire effectue différentes requêtes auprès du challengeur afin que celui lui retourne le déchiffrement d'un chiffré. S'il souhaite obtenir le déchiffrement d'un chiffré C destiné à $\mathtt{A}$, il formule sa requête sur l'entrée $(C, \mathtt{A})$. Le challengeur utilise la clé privée $sk_{\mathtt{A}}$ pour obtenir le message correspondant et le lui transmet. Dans le cas où le chiffré C est mal formé, il lui fournit $\perp$. Lors d'une telle opération, nous dirons que l'adversaire fait appel à un *oracle de déchiffrement.*

Challenge

- L'adversaire fournit au challengeur deux messages m_0 et m_1 appartenant tous les deux à $\mathcal{M}$. Il choisit également une entité honnête $\mathtt{A}^*$ dont la paire de clés sera désormais notée (sk^*, pk^*). Le challengeur lui retourne un chiffré $C^* = \mathsf{Chiffrer}(\mathcal{P}, pk^*, m_b)$ où b est tiré aléatoirement parmi $\{0, 1\}$.

Phase 2

- L'adversaire peut effectuer un certain nombre de requêtes correspondantes à celles autorisées dans la *Phase 1* avec la restriction que si la requête porte sur l'entrée $(C^*, \mathtt{A}^*)$, le challengeur lui retourne alors $\perp$.

Réponse

- L'adversaire renvoie finalement une réponse b' appartenant à $\{0, 1\}$ et est déclaré vainqueur si $b' = b$.

Définition 20. *Désignons par* $\mathcal{A}$ *l'adversaire contre le challenge et par* $\mathsf{Adv}_{\mathsf{PUB},\mathcal{A},\lambda}$ *l'avantage qu'il a de gagner le scénario par rapport à* $1/2$ *- probabilité de gagner le challenge en répondant aléatoirement. Soit alors* $\mathsf{Adv}_{\mathsf{PUB},\lambda} = \max_{\mathcal{A}}(\mathsf{Adv}_{\mathsf{PUB},\mathcal{A},\lambda})$*, le maximum des avantages pris sur tous les adversaires* $\mathcal{A}$ *fonctionnant en temps polynomial du paramètre de sécurité* λ*. Le schéma de chiffrement à clé publique est dit sûr contre l'indiscernabilité de messages chiffrés si* $\mathsf{Adv}_{\mathsf{PUB},\lambda}$ *est négligeable en le paramètre de sécurité* λ*.*

Corruption adaptative ou non ? Nous avons considéré un scénario de corruption non adaptatif où les entités corrompues et honnêtes sont choisies par l'adversaire lors de la génération des clés publiques et privées par le challengeur. Il existe cependant un scénario a priori plus fort : le scénario de corruption adaptatif. Dans ce scénario, les entités sont toutes considérées au départ comme étant honnêtes et au fil du scénario, l'adversaire peut décider de les corrompre à sa guise.

Nous avons bien dit a priori, car ce scénario est en fait équivalent au scénario non adaptatif. Pour prouver cette équivalence, il suffit simplement d'effectuer une réduction du scénario non adaptatif au scénario adaptatif, la réciproque étant évidente. L'idée principale de cette réduction, qui rejoint l'idée du jeton de Coron [Cor00], se résume à prédire si une entité sera corrompue ou non par l'adversaire au fil du scénario avec une certaine probabilité. Le but est de jouer ensuite sur la valeur de cette probabilité en fonction du nombre d'utilisateurs considérés pour que cela corresponde le mieux avec les requêtes effectuées par l'adversaire. La réduction que l'on obtient est bien polynomiale [9] et permet d'obtenir l'équivalence de ces deux scénarios. Cette idée de réduction a d'abord été utilisée pour le serveur de re-chiffrement par Weng *et al.* dans [WCY+10] et est clairement applicable à tous les schémas de serveur de re-chiffrement.

Ainsi, nous n'avons pas à nous soucier du modèle de corruption adaptatif, ce qui permet de simplifier les preuves de sécurité et leur compréhension [10].

Considération de plusieurs entités. Contrairement au scénario que nous venons de présenter et afin de simplifier la description des preuves de sécurité, on ne considère habituellement qu'une seule entité (honnête) qui sera l'entité cible du challenge. Pour un chiffrement à clé publique, les paires de clés de deux entités différentes sont indépendantes et ne sont reliées entre elles par aucun élément que ce soit, l'adversaire ne pourra donc obtenir d'information sur l'entité cible du challenge à partir d'autres entités. Dans le cas du serveur de re-chiffrement, ce n'est pas le cas puisque deux entités peuvent être reliées par une clé de re-chiffrement, nous ne pouvons donc considérer cette simplification. Étant donné que, dans nos constructions, nous utiliserons des chiffrements à clé publiques, nous serons donc amenés à considérer plusieurs entités pour les scénarios d'attaque de ces chiffrements, ce qui justifie le non-emploi de cette simplification.

Le chiffrement Elgamal. Le *chiffrement ElGamal* est l'un des chiffrements à clé publique les plus connus. Il est à la base de tous les schémas présentés dans ce mémoire. Il a été exposé par ElGamal dans [Gam84]. La sécurité prouvée sur ce schéma repose sur la difficulté d'inverser la fonction exponentielle dans un groupe d'ordre premier. Même s'il est possible d'utiliser le schéma originel, nous utiliserons en général une variante qui permettra de comprendre plus facilement le fonctionnement de nos schémas. Celle-ci consiste à chiffrer un message m en $(C_1, C_2) = (X^r, g^r \cdot m)$ où g est un générateur du groupe, $X = g^x$ est la clé publique de l'utilisateur avec $x \in \mathbb{Z}_p^*$ sa clé privée. Le déchiffrement de ce chiffré se fait en calculant $m = C_2/C_1^{1/x}$. La version originelle consiste à considérer le chiffré $(C_1, C_2) = (g^r, X^r \cdot m)$ où le déchiffrement s'effectue en calculant $m = C_2/C_1^x$. Ce schéma, dans ces deux versions, est prouvé IND-CPA sûr sous l'hypothèse DDH, dans le modèle standard.

2.4.3 Le chiffrement à clé secrète

La particularité de cette primitive est que la clé qui sert à rendre l'information confidentielle et à la récupérer est la même.

Définition 21 (Chiffrement symétrique (ou à clé secrète)). *Un* schéma de chiffrement à clé secrète SEC *est la donnée d'un espace des messages $\mathcal{M}$, ainsi que d'un ensemble de quatre*

9. Elle laisse apparaître un facteur $1/n$ avec n dépendant du nombre d'utilisateurs considérés. En prenant directement en compte un scénario de corruption adaptatif, on pourrait espérer se passer de ce facteur polynomial.

10. Le lecteur soucieux de voir une telle simplification est invité à comparer la preuve donnée dans [CDL11] et sa version dans le modèle de corruption non adaptatif que nous présenterons au Chapitre 6.

algorithmes {Initialiser, Clé, Chiffrer, Déchiffrer} *décrits comme suit.*

Initialiser(λ) $\rightarrow \mathcal{P}$ *: prend en entrée un paramètre de sécurité* $\lambda \in \mathbb{N}$ *et retourne les paramètres publics* $\mathcal{P}$ *du schéma qui sont donnés en entrée de tous les autres algorithmes.*

Clé($\mathcal{P}$) *: prend en entrée les paramètres publics* $\mathcal{P}$ *et retourne une clé secrète* k*, qui permet de chiffrer et également de déchiffrer.*

Chiffrer($\mathcal{P}, k, m$) *: prend en entrée les paramètres publics* $\mathcal{P}$*, une clé secrète* k *et un message* m *appartenant à* $\mathcal{M}$ *et retourne un chiffré* C_k *pour la clé secrète* k*.*

Déchiffrer($\mathcal{P}, k, C_k$) *: prend en entrée les paramètres publics* $\mathcal{P}$*, une clé secrète* k *et un chiffré* C_k *pour la clé secrète* k *et retourne un message* m *ou un symbole* $\perp$ *signifiant que le chiffré n'est pas valide.*

Consistance. Dans un schéma de chiffrement à clé secrète, un message chiffré avec une clé secrète doit bien être déchiffré avec la même clé. Ainsi, pour tous les paramètres publics générés par l'algorithme Initialiser, pour toute clé secrète k générée avec l'algorithme Clé et pour tout message $m \in \mathcal{M}$, la propriété suivante doit être vérifiée :

$$\text{Déchiffrer}(\mathcal{P}, k, \text{Chiffrer}(\mathcal{P}, k, m)) \rightarrow m.$$

Nous présentons également la sécurité IND-CCA pour les chiffrements à clé secrète que nous utiliserons au Chapitre 5.

Scénario d'attaque. Le scénario d'attaque contre une indiscernabilité de chiffrés entre un adversaire et un challengeur pour un schéma de chiffrement à clé secrète est décrit ci-dessous.

Phase d'initialisation

– Le challengeur initialise les paramètres publics $\mathcal{P}$ du système en fonction d'un paramètre de sécurité λ via Initialiser et les donne à l'adversaire.

Challenge

– L'adversaire fournit au challengeur deux messages m_0 et m_1 appartenant tous les deux à $\mathcal{M}$. Ce dernier génère une clé secrète K^* en utilisant l'algorithme Clé et lui retourne le chiffré $C^* = \text{Chiffrer}(\mathcal{P}, K^*, m_b))$ où b est tiré aléatoirement parmi $\{0, 1\}$.

Phase de requêtes

– L'adversaire effectue différentes requêtes auprès du challengeur afin que celui-ci lui retourne le déchiffrement d'un chiffré. S'il souhaite obtenir le déchiffrement d'un chiffré C, il formule sa requête sur l'entrée C. Le challengeur utilise la clé secrète K^* pour obtenir le message correspondant et le lui transmet, à condition que $C \neq C^*$. Dans le cas où le chiffré C est mal formé, il lui fournit $\perp$. Lors d'une telle opération, nous dirons que l'adversaire fait appel à un *oracle de déchiffrement.*

Réponse

– L'adversaire renvoie finalement une réponse b' appartenant à $\{0, 1\}$ et est déclaré vainqueur si $b' = b$.

Définition 22. *Désignons par* $\mathcal{A}$ *l'adversaire contre ce challenge et par* $\text{Adv}_{\text{SEC},\mathcal{A},\lambda}$*, l'avantage qu'il a de gagner ce scénario par rapport à* 1/2 *- probabilité de gagner le challenge en répondant aléatoirement. Soit alors* $\text{Adv}_{\text{SEC},\lambda} = \max_{\mathcal{A}}(\text{Adv}_{\text{SEC},\mathcal{A},\lambda})$*, le maximum des avantages pris sur tous les adversaires* $\mathcal{A}$ *fonctionnant en temps polynomial du paramètre de sécurité* λ*. Le schéma de chiffrement à clé secrète est dit sûr contre l'indiscernabilité de messages chiffrés si* $\text{Adv}_{\text{SEC},\lambda}$ *est négligeable en le paramètre de sécurité* λ*.*

2.5 Outils cryptographiques

Enfin, nous présentons certains outils et constructions souvent utilisés dans des schémas de serveur de re-chiffrement et dont nous nous servirons également. Pour des besoins spécifiques, nous serons amenés à utiliser d'autres outils cryptographiques non présentés ici que nous décrirons lors de leur utilisation.

2.5.1 Signature

Une signature est une primitive cryptographique introduite par Diffie et Hellman dans [DH76] et qui permet d'authentifier une donnée.

Définition 23 (**Signature**). *Un* schéma de signature SIG *est la donnée d'un espace des messages* $\mathcal{M}$*, et d'un ensemble de quatre algorithmes* $\{\mathsf{Initialiser}, \mathsf{Clé}, \mathsf{Signer}, \mathsf{Vérifier}\}$ *décrits comme suit.*

- $\mathsf{Initialiser}(\lambda) \to \mathcal{P}$ *: prend en entrée un paramètre de sécurité* $\lambda \in \mathbb{N}$ *et retourne les paramètres publics* $\mathcal{P}$ *du schéma qui sont donnés en entrée de tous les autres algorithmes.*
- $\mathsf{Clé}(\mathcal{P})$ *: prend en entrée les paramètres publics* $\mathcal{P}$ *et retourne une paire de clés (signature/vérification)* (sk, vk)*, telle que* sk *permette de signer et* sk *de vérifier.*
- $\mathsf{Signer}(\mathcal{P}, sk, m)$ *: prend en entrée les paramètres publics* $\mathcal{P}$*, une clé de signature* sk *et un message* m *appartenant à* $\mathcal{M}$ *et retourne une signature* σ*.*
- $\mathsf{Vérifier}(\mathcal{P}, pk, m, sigma)$ *: prend en entrée les paramètres publics* $\mathcal{P}$*, une clé de vérification* vk *et une signature* σ *et retourne un bit valant 1 si la signature est valide pour le message* m *et la clé de vérification* vk *et 0 sinon.*

Consistance. Il faut évidemment que toute signature d'un message soit valide. Ainsi, pour tous les paramètres publics générés par l'algorithme $\mathsf{Initialiser}$, pour toute paire de clés (signature/vérification) (sk, vk) générée avec l'algorithme $\mathsf{Clé}$ et pour tout message $m \in \mathcal{M}$, la propriété suivante doit être vérifiée

$$\mathsf{Vérifier}(\mathcal{P}, vk, \mathsf{Signer}(\mathcal{P}, sk, m)) \to 1.$$

2.5.2 Signature à usage unique

Comme son nom l'indique, un schéma de *signature à usage unique* est un schéma de signature, donc composé des algorithmes précédents et dont la clé de signature ne permet de signer qu'un seul message, auquel elle est liée. Cette idée est contraire aux schémas de signature "usuels" correspondant à la définition précédente, où cette clé peut être utilisée autant de fois que l'on veut et pour n'importe quel message. Une définition plus formelle de ce type de signatures peut être trouvée dans l'ouvrage [Gol04]. Nous l'utiliserons en boîte noire dans la construction CHK décrite ci-dessous pour obtenir des schémas de chiffrement IND-CCA sûr. Nous utiliserons les notations habituelles, à savoir OTS pour désigner cette primitive et (ssk, svk) pour une paire de clés (sigature/vérification).

2.5.3 La construction CHK

La construction CHK (ou paradigme CHK), due à Canetti *et al.* et présentée dans [CHK04], fournit un moyen d'obtenir des schémas de chiffrement à clé publique prouvés IND-CCA sûrs. La construction utilise notamment une signature à usage unique dont nous venons de parler et un chiffrement basé sur l'identité prouvé IND-CPA sûr et dont nous reparlerons page 129. L'idée de cette construction consiste à chiffrer un message avec l'algorithme de chiffrement basé

sur l'identité pour une identité correspondant à une clé de vérification de la signature à usage unique et de signer le chiffré obtenu avec la clé de signature correspondante. Nous pouvons en conclure que si la signature à usage unique est valide, c'est qu'elle a bien été produite pour le chiffré en question et donc qu'il n'a pas été modifié. Nous obtenons de cette manière la propriété de non-malléabilité, même contre une attaque active, d'où la possibilité d'obtenir une sécurité IND-CCA dans le modèle standard si le chiffrement basé sur l'identité et la signature à usage unique sont prouvés sûrs dans le modèle standard.

Elle est très utilisée pour la réalisation de schémas de serveur de re-chiffrement, notamment pour obtenir des schémas prouvés sûrs dans le modèle standard. Pour autant, elle n'est pas suffisante à elle seule pour obtenir à coup sûr une sécurité IND-CCA puisque le schéma [LV08b] n'atteint par exemple que la sécurité IND-RCCA dont nous parlerons au Chapitre 5.

2.5.4 La construction Fujisaki-Okamoto

La construction Fujisaki-Okamoto exposée dans [FO99], fournit un moyen d'obtenir des schémas de chiffrement à clé publique prouvée IND-CCA sûrs permettant de chiffrer n'importe quelle chaîne de bits. Elle utilise notamment un chiffrement à clé publique prouvé IND-CPA sûr et un chiffrement à clé secrète prouvé OW-CPA, *i.e* résistant à une attaque CPA où le but de l'adversaire est de retrouver le message chiffré. Pour chiffrer un message m, il faut d'abord tirer un aléa σ, qui correspondra en quelque sorte à une clé de session d'un chiffrement hybride. Cette construction se décrit ainsi :

- σ est chiffré avec l'algorithme de chiffrement à clé publique pour la clé publique du destinataire en utilisant comme aléa un haché de $\sigma \| m$;
- le message est chiffré avec l'algorithme de chiffrement à clé secrète en utilisant le haché [11] de σ comme clé.

Le déchiffrement consiste à retrouver d'abord σ, puis m. Ceci permet, lors du déchiffrement, de tester si le chiffré est bien formé. Pour les mêmes raisons que précédemment, on obtient une sécurité IND-CCA dans le modèle de l'oracle aléatoire.

Cette construction est utilisée dans de nombreux schémas de serveur de re-chiffrement, comme par exemple [DWLC08, CWYD10]. Nous verrons cependant, au Chapitre 6 qu'elle n'est pas suffisante pour obtenir une sécurité IND-CCA pour les schémas de serveur de re-chiffrement. L'utilisation de cette construction sur le chiffrement ElGamal donne le *chiffrement Hash ElGamal* que nous utiliserons dans ce même chapitre.

11. On utilise ici une autre fonction de hachage que celle utilisée au-dessus.

Deuxième partie

Modélisation du serveur de re-chiffrement

Chapitre 3

Unification des différents modèles de serveur de re-chiffrement

Sommaire

3.1 Définition d'un serveur de re-chiffrement 31
3.1.1 Modèle unifié 32
3.1.2 Exemple illustratif 34
3.1.3 Production d'un chiffré 36
3.2 Caractéristiques 36
3.2.1 Nombre de re-chiffrements successifs 36
3.2.2 Type de confiance entre le délégant et le délégué 38
3.3 Sécurité d'un serveur de re-chiffrement 39
3.3.1 Notions de sécurité 39
3.3.2 Restrictions naturelles dans les moyens de l'adversaire 42
3.3.3 Restrictions non naturelles récurrentes 43
3.3.4 Spécificités liées aux preuves de sécurité 44
***Note sur les sécurités secondaires* 46**
3.4 Sécurité pour le stockage de données 47
3.4.1 Utilité de l'anonymat de la clé de re-chiffrement ? 47
3.4.2 Indiscernabilité des messages chiffrés 47

Nous nous intéressons dans ce chapitre à la modélisation de la primitive de serveur de re-chiffrement. Nous en donnons ainsi une définition générale et nous nous attardons ensuite sur les notions de sécurité.

3.1 Définition d'un serveur de re-chiffrement

Nous avons vu dans le chapitre précédent une présentation informelle d'un serveur de re-chiffrement. Nous en donnons maintenant une définition syntaxique rigoureuse. Comme nous le verrons par la suite, il existe bon nombre de modèles de serveur de re-chiffrement offrant chacun des fonctionnalités différentes. Par exemple, Canetti *et al.* formalise dans [CH07] le modèle présenté dans [BBS98] où un chiffré peut être re-chiffré autant de fois que souhaité et où la clé de re-chiffrement de Alice vers Bob permet également de re-chiffrer de Bob vers Alice. À l'opposé, Ateniese *et al.* ont eux formalisé dans [AFGH06] un modèle qui considère qu'un chiffré ne peut être re-chiffré qu'une seule fois et où la clé de re-chiffrement dont on vient de parler ne permet pas d'effectuer la modification que nous venons d'énoncer ci-dessus. La littérature se compose ainsi de différents modèles. Il est également possible d'en concevoir de nouveaux, non pris en compte par ces derniers, par exemple en en combinant certains. C'est par exemple le cas des travaux

présentés au Chapitre 7. Chacun de ces modèles considère un modèle de sécurité établi sur une base commune, à savoir un chiffrement à clé publique, mais les propriétés qui en découlent vont varier d'une fonctionnalité à l'autre. Afin de simplifier l'étude de ces différents cas possibles, qui présentent tous des fonctionnalités différentes, nous présentons ici un seul et unique modèle qui englobe tous ces différents cas ainsi que tous ceux que l'on puisse imaginer.

3.1.1 Modèle unifié

Environnement d'un modèle. En observant les schémas existants, nous nous apercevons que ces différents modèles de serveur de re-chiffrement reposent sur une même base. En effet, ils commencent tous par décrire ce que nous décidons d'appeler un *environnement*, noté $\mathsf{Environnement}$, et qui va permettre de les distinguer. Cet environnement est composé de trois points essentiels décrits ainsi :

- la donnée d'un ensemble de types de chiffrés, que l'on note Type. Nous allons être amenés à manipuler différents types de chiffrés possibles. En effet, d'un point de vue théorique, il n'y a aucune raison pour qu'un chiffré qui a été re-chiffré ait la même forme que le chiffré originel. Il n'y a également aucune raison pour que l'on puisse effectuer les mêmes opérations de re-chiffrement sur ces deux chiffrés. L'ensemble Type permet donc de distinguer les différents chiffrés que l'on pourra produire [1]. Cet ensemble peut être composé de n'importe quels symboles ou mots, mais il est plus courant d'utiliser des nombres entiers naturels. La plupart des schémas existants considèrent d'ailleurs les ensembles $\{1\}$ ou $\{1, 2\}$;
- la donnée d'un ensemble $\mathsf{Liaison} \subset \mathsf{Type} \times \mathsf{Type}$. Cet ensemble décrit les re-chiffrements possibles entre les différents types de chiffrés. Dans le cas d'un ensemble $\mathsf{Type} = \{1, 2, 3\}$, nous pouvons avoir, par exemple, l'ensemble $\mathsf{Liaison} = \{(1, 1), (2, 1), (2, 3)\}$. Cela signifie que l'on peut modifier un chiffré de type 1 en un chiffré de type 1, alors qu'un chiffré de type 2 peut être modifié en un type 1 ou en un type 3 ;
- la donnée, pour chaque liaison $(\eta, \eta') \in \mathsf{Liaison}$, de deux ensembles formels $\mathsf{D}_{(\eta,\eta'),de}$ et $\mathsf{D}_{(\eta,\eta'),vers}$ sélectionnés parmi les ensembles

$$\{\texttt{clé privée}, \texttt{clé publique}, (\texttt{clé privée, clé publique})\}.$$

  Dans le cas par exemple où $\mathsf{D}_{(\eta,\eta'),de} = \{(\texttt{clé privée}, \texttt{clé publique})\}$ et $\mathsf{D}_{(\eta,\eta'),vers} = \{\texttt{clé publique}\}$, nous considérerons alors que pour calculer une clé de re-chiffrement sur la liaison (η, η'), il faut utiliser la paire de clés (privée,publique) du délégant et la clé publique du délégué. Pour le calcul d'une clé de re-chiffrement d'un dispositif $\mathtt{A}$ vers un dispositif $\mathtt{B}$ sur la liaison (η, η'), nous noterons respectivement $\mathsf{D}^{\mathtt{A}}_{(\eta,\eta'),de}$ et $\mathsf{D}^{\mathtt{B}}_{(\eta,\eta'),vers}$, les ensembles $\mathsf{D}_{(\eta,\eta'),de}$ et $\mathsf{D}_{(\eta,\eta'),vers}$ appliqués aux données de $\mathtt{A}$ et de $\mathtt{B}$; ce qui nous donne sur l'exemple considéré : $\mathsf{D}^{\mathtt{A}}_{(\eta,\eta'),de} = (sk_{\mathtt{A}}, pk_{\mathtt{A}})$ et $\mathsf{D}^{\mathtt{B}}_{(\eta,\eta'),vers} = pk_{\mathtt{B}}$.

Les environnements des deux schémas évoqués ci-dessus sont décrits ainsi :

- [CH07] considère les ensembles $\mathsf{Type} = \{(1)\}$, $\mathsf{Liaison} = \{(1, 1)\}$ et les ensembles formels $\mathsf{D}_{(1,1),de} = \mathsf{D}_{(1,1),vers} = \{\texttt{clé privée}\}$.
- [LV08b] considère les ensembles $\mathsf{Type} = \{(1, 2)\}$, $\mathsf{Liaison} = \{(1, 2)\}$ et les ensembles formels $\mathsf{D}_{(1,2),de} = \{\texttt{clé privée}\}$ et $\mathsf{D}_{(1,2),vers} = \{\texttt{clé publique}\}$.

Chaque modèle consiste ensuite à décrire les algorithmes qui utiliseront cet environnement. Par exemple, chaque modèle considère un algorithme de chiffrement qui décrit, pour chaque type $\eta \in \mathsf{Type}$, comment masquer un message m en un chiffré de type η. La spécification d'un modèle dépend donc de son environnement. Étant donné que nous avons considéré un environnement général, nous pouvons donner une définition générale d'un schéma de serveur de re-chiffrement.

1. Des chiffrés de type différent n'auront pas forcément des structures différentes comme nous le verrons avec les serveurs de re-chiffrement conditionnel au Chapitre 8.

Définition 24 (Serveur de re-chiffrement)**.** *Un* schéma de serveur de re-chiffrement (PRE) *est la donnée d'un espace des messages $\mathcal{M}$, d'un environnement* Environnement *tel que nous venons de le décrire, ainsi que d'un ensemble de six algorithmes* {Initialiser, Clé, Délégation, Chiffrer, Re-chiffrer, Déchiffrer}[2] *décrits comme suit.*

- Initialiser$(\lambda) \to \mathcal{P}$ *: prend en entrée un paramètre de sécurité $\lambda \in \mathbb{N}$ et retourne les paramètres publics $\mathcal{P}$ du schéma qui sont donnés en entrée de tous les autres algorithmes.*
- Clé$(\mathcal{P})$ *: prend en entrée les paramètres publics $\mathcal{P}$ et retourne une paire de clés (privée/publique) (sk, pk), telle que pk permette de chiffrer et sk de déchiffrer.*
- Délégation$(\mathcal{P}, (\eta, \eta'), \mathsf{D}^{\mathtt{A}}_{(\eta,\eta'),de}, \mathsf{D}^{\mathtt{B}}_{(\eta,\eta'),vers})$ *: prend en entrée les paramètres publics $\mathcal{P}$, une liaison (η, η') de* Liaison *et les deux ensembles $\mathsf{D}^{\mathtt{A}}_{(\eta,\eta'),de}$ et $\mathsf{D}^{\mathtt{B}}_{(\eta,\eta'),vers}$ et retourne une clé de re-chiffrement $R_{\mathtt{A},\eta\to\mathtt{B},\eta'}$ du type η de $\mathtt{A}$ vers le type η' de $\mathtt{B}$.*
- Chiffrer$(\mathcal{P}, \eta, pk, m)$ *: prend en entrée les paramètres publics $\mathcal{P}$, un type η appartenant à* Type*, une clé publique pk et un message m appartenant à $\mathcal{M}$ et retourne un chiffré $C_{pk,\eta}$ de type η pour la clé publique pk.*
- Re-chiffrer$(\mathcal{P}, (\eta, \eta'), R_{\mathtt{A},\eta\to\mathtt{B},\eta'}, C_{pk_\mathtt{A},\eta})$ *: prend en entrée les paramètres publics $\mathcal{P}$, une paire de types (η, η') appartenant à* Liaison*, une clé de re-chiffrement $R_{\mathtt{A},\eta\to\mathtt{B},\eta'}$ du type η pour $\mathtt{A}$ vers le type η' pour $\mathtt{B}$ et un chiffré $C_{pk_\mathtt{A},\eta}$ de type η pour l'entité $\mathtt{A}$ et retourne un chiffré $C_{pk_\mathtt{B},\eta'}$ du même message en clair - et ce, sans en obtenir d'information - de type η' pour l'entité $\mathtt{B}$ ou un symbole $\perp$ signifiant que le chiffré n'est pas valide (pour la clé de re-chiffrement considérée).*
- Déchiffrer$(\mathcal{P}, \eta, sk, C_{pk,\eta})$ *: prend en entrée les paramètres publics $\mathcal{P}$, un type η appartenant à* Type*, une clé privée sk et un chiffré $C_{pk,\eta}$ de type η pour la clé publique pk associée à la clé privée sk et retourne un message m ou un symbole $\perp$ signifiant que le chiffré n'est pas valide.*

Consistance. Comme dans tout schéma de chiffrement, on désire que tout chiffré correctement généré puisse être déchiffré par l'algorithme Déchiffrer. Il faut vérifier cette propriété pour tous les algorithmes.

Si l'on ne s'intéresse, pour commencer, qu'à la partie chiffrement, on retrouve la consistance habituelle d'un schéma de chiffrement. En particulier, pour tous les paramètres publics $\mathcal{P}$ générés par l'algorithme Initialiser, pour tout type $\eta \in$ Type, pour toute paire de clés (sk, pk) générée avec l'algorithme Clé et pour tout message $m \in \mathcal{M}$, la propriété suivante doit être vérifiée :

$$\text{Déchiffrer}(\mathcal{P}, \eta, sk, \text{Chiffrer}(\mathcal{P}, \eta, pk, m)) \to m.$$

Si l'on considère ensuite les re-chiffrements possibles, il faut que chaque re-chiffré puisse correctement être déchiffré par l'entité à qui il est destiné. En particulier, pour tous les paramètres publics $\mathcal{P}$ générés par l'algorithme Initialiser, pour tout entier naturel $n > 1$, pour tout n-uplet $(\eta_1, \cdots, \eta_n) \in \text{Type}^n$ tel que $(\eta_1, \eta_2), \cdots, (\eta_{n-1}, \eta_n) \in$ Liaison, pour tout n-uplet d'entités $(\mathtt{A}_1, \cdots, \mathtt{A}_n)$ avec paires de clés respectives $(sk_1, pk_1), \cdots (sk_n, pk_n)$ générées avec l'algorithme Clé, pour toutes clés de re-chiffrements $rk_{\mathtt{A}_1,\eta_1\to\mathtt{A}_2,\eta_2}, \cdots, rk_{\mathtt{A}_{n-1},\eta_{n-1}\to\mathtt{A}_n,\eta_n}$ générées avec l'algorithme Délégation et pour tout message $m \in \mathcal{M}$, la propriété suivante doit être vérifiée :

$$\begin{aligned}\text{Déchiffrer}(\mathcal{P}, \eta_n, sk_n, \text{Re-chiffrer}(\mathcal{P}, (\eta_{n-1}, \eta_n), rk_{\mathtt{A}_{n-1},\eta_{n-1}\to\mathtt{A}_n,\eta_n}, \ldots,\\ \text{Re-chiffrer}(\mathcal{P}, (\eta_1, \eta_2), rk_{\mathtt{A}_1,\eta_1\to A_2,\eta_2}, \text{Chiffrer}(\mathcal{P}, \eta_1, pk_1, m))\ldots)) \to m.\end{aligned}$$

Remarque 3. Une clé de re-chiffrement d'une entité A vers une entité B est un moyen pour B de parvenir à obtenir l'information destinée à A. Cela ne doit être possible que dans le cas où A est

2. Le lecteur avisé reconnaîtra un schéma de chiffrement à clé publique en ne considérant que les algorithmes {Initialiser, Clé, Chiffrer, Déchiffrer}.

d'accord. Le calcul de cette clé de re-chiffrement doit donc comporter certaines opérations qui ne peuvent être faites que par `A`. Dans le cas contraire, il serait facile de casser le système de chiffrement du serveur de re-chiffrement en calculant des clés de re-chiffrement sans l'accord de `A`. Pour cette raison, le calcul de la clé de re-chiffrement de `A` vers `B` doit nécessiter un élément secret connu seulement de `A`. Cela implique notamment que tous les ensembles formels $\mathsf{D}_{(\eta,\eta'),de}$ sont choisis parmi $\{\texttt{clé privée}, (\texttt{clé privée,clé publique})\}$.

Vocabulaire.

1. Nous dirons qu'un chiffré est *re-chiffré* s'il existe une liaison $(\eta, \eta') \in \mathsf{Liaison}$ telle que ce chiffré ait été obtenu par un re-chiffrement de η vers η'. S'il n'en existe pas, nous dirons qu'il est *non re-chiffré*. L'existence d'une liaison (η, η') permet, en particulier, de savoir si un chiffré est potentiellement re-chiffré. Elle ne permet pas, par contre, de conclure s'il a bien été re-chiffré, puisqu'il peut avoir été produit par l'algorithme Chiffrer sur le type η'.
2. Nous dirons qu'un chiffré est *re-chiffrable* s'il existe une liaison (η, η') appartenant à Liaison telle que η soit le type du chiffré considéré. Dans le cas contraire, nous dirons que le chiffré est *non re-chiffrable*. L'ensemble Liaison permet de distinguer si un chiffré est re-chiffrable ou non.
3. Nous dirons qu'une entité `A` est un *délégant* et qu'une entité `B` est son *délégué* s'il existe une clé de re-chiffrement de `A` vers `B`. Nous pourrons au besoin préciser les types de chiffrés concernés, comme par exemple `A` est un délégant pour le type 1 et `B` est son délégué pour le type 2.

Notations. Nous serons souvent amenés à décrire les différents algorithmes d'un schéma de serveur de re-chiffrement étape par étape, en les définissant à tour de rôle sur des entrées différentes. Afin de faciliter toutes ces descriptions, nous noterons $\mathsf{Chiffrer}_\eta$ et $\mathsf{Déchiffrer}_\eta$ les algorithmes respectifs Chiffrer et Déchiffrer sur l'entrée de type η. De même, les algorithmes Délégation et Re-chiffrer sur l'entrée de liaison (η, η') seront notés respectivement $\mathsf{Délégation}_{(\eta,\eta')}$ et $\mathsf{Re\text{-}chiffrer}_{(\eta,\eta')}$.

Remarque 4. Afin de proposer une définition la plus fonctionnelle possible, nous avons considéré que le schéma possède un algorithme Chiffrer décrit sur le type η pour chaque $\eta \in \mathsf{Type}$. En pratique, cette caractéristique n'est pas obligatoire. Si l'on souhaite disposer d'un serveur de re-chiffrement sur des ensembles $\mathsf{Type} = \{1, 2\}$ et $\mathsf{Liaison} = \{(1, 2)\}$ où l'on souhaite que chaque message que l'on chiffre puisse être re-chiffrer une fois, alors seule la description de Chiffrer sur le type 1 est nécessaire.

3.1.2 Exemple illustratif

Au premier abord, cette définition peut paraître complexe, mais elle permet une fois assimilée d'envisager beaucoup de situations. Afin de se familiariser un peu avec celle-ci et également de découvrir une construction naïve de schémas de serveur de re-chiffrement, nous en présentons ici un exemple simple. L'idée est de transmettre une clé de déchiffrement au délégué via la clé de re-chiffrement. Elle est notamment utilisée dans le schéma [CWYD10] dont nous étudierons la sécurité au Section 6.1.3. Nous l'utiliserons également à la Section 4.2 afin d'augmenter, dans certains cas, le nombre de re-chiffrements successifs offerts par un schéma.

Plaçons-nous dans le cas le plus simple où la clé à transmettre est celle du délégant [3]. En fait, cela revient à ce qu'un délégant divulgue sa clé privée à ses délégués. Nous avons évoqué, dans le chapitre précédent, le fait que cette divulgation n'était pas souhaitable pour réaliser un stockage

3. Nous verrons, dans les sections exposées ci-dessus, que la clé de déchiffrement transmise ne sera pas celle du délégant afin de ne pas aboutir à une divulgation de la clé de déchiffrement du délégant.

de données basé sur le serveur de re-chiffrement. L'idée est cependant intéressante et sert à la construction de nombreux schémas.

Dans notre exemple, nous omettons volontairement tous les paramètres publics des différents schémas considérés ainsi que leur génération afin de le simplifier au maximum. De plus, nous présentons le schéma de manière progressive, où les différents algorithmes, qui le composent, sont définis au-fur-et-à-mesure de sa conception.

1ère partie : production d'un chiffré. Nous commençons par la partie chiffrement et, plus particulièrement, nous précisons une manière de produire un chiffré (pour un seul type). Pour cela, nous considérons un algorithme de chiffrement à clé publique ENC_1 composé des algorithmes $\{\mathsf{Clé}, \mathsf{Chiffrer}, \mathsf{Déchiffrer}\}$ tels que définis dans le Chapitre 2.

- la génération des clés : nous définissons l'algorithme $\mathsf{PRE.Clé}$ comme étant l'algorithme $\mathsf{ENC}_1.\mathsf{Clé}$. Notons (sk_1, pk_1), la paire de clés (privée/publique) générée par $\mathsf{ENC}_1.\mathsf{Clé}$. Nous considérons alors $(pk, sk) = (pk_1, sk_1)$, la paire de clés (publique privée) retournée par $\mathsf{PRE.Clé}$. Nous faisons la distinction entre les couples (pk, sk) et (pk_1, sk_1), car cette description de $\mathsf{PRE.Clé}$ est provisoire et sera modifiée par la suite.
- le chiffrement/déchiffrement : nous considérons tout simplement ces opérations comme étant celles du schéma ENC_1. Ceci nous permet de produire des chiffrés, dont le type est noté 1. En détail et en notant $C_{pk_1} = \mathsf{ENC}_1.\mathsf{Chiffrer}(pk_1, m)$, nous définissons $\mathsf{PRE.Chiffrer}_1(pk, m) = C_{pk_1}$ et $\mathsf{PRE.Déchiffrer}_1(sk, C_{pk_1}) = \mathsf{ENC}_1.\mathsf{Déchiffrer}(sk_1, C_{pk_1})$.

2ème partie : réalisation du re-chiffrement. Maintenant que nous sommes capables de chiffrer et de déchiffrer pour le type 1, nous nous focalisons sur la manière de transformer ces chiffrés. Pour rappel, nous basons notre re-chiffrement sur la transmission de la clé de déchiffrement aux délégués.

Pour cela, nous nous intéressons à deux utilisateurs, $\mathtt{A}$ et $\mathtt{B}$, qui possèdent chacun une paire de clés (privée/publique), que l'on note respectivement $(sk_{\mathtt{A}}, pk_{\mathtt{A}}) = (sk_{\mathtt{A},1}, pk_{\mathtt{A},1})$ et $(sk_{\mathtt{B}}, pk_{\mathtt{B}}) = (pk_{\mathtt{B},1}, sk_{\mathtt{B},1})$. Nous considérons aussi un chiffré $C_{pk_{\mathtt{A}},1} = \mathsf{PRE.Chiffrer}_1(pk_{\mathtt{A},1}, m)$ de type 1 pour $\mathtt{A}$. Notre idée est donc qu'un délégué $\mathtt{B}$ obtienne un chiffré composé du chiffré $C_{pk_{\mathtt{A}},1}$ ainsi que de la clé privée $sk_{\mathtt{A},1}$. Nous remarquons qu'un re-chiffré destiné à $\mathtt{B}$ n'a pas la même composition qu'un chiffré destiné à $\mathtt{A}$. Nous obtenons ainsi un second type de chiffré, noté 2. Nous pouvons alors mettre à jour certains ensembles : $\mathsf{Type} = \{1, 2\}$ et $\mathsf{Liaison} = \{(1, 2)\}$.

Revenons maintenant sur l'opération de re-chiffrement. Elle consiste à réunir les deux objets évoqués ci-dessus. Ainsi, la clé de re-chiffrement de $\mathtt{A}$ du type 1 vers $\mathtt{B}$ du type 2 doit être composée de la clé privée $sk_{\mathtt{A},1}$. Nous pouvons alors définir $\mathsf{D}_{(1,2),de} = \{\texttt{clé privée}\}$. En revanche, il est évident que cette clé de re-chiffrement offre, à l'entité qui la possède, la possibilité d'obtenir l'information en clair. Pour éviter cela mais permettre à $\mathtt{B}$ d'obtenir quand même cette clé, nous la chiffrons pour lui. Le plus simple serait de la chiffrer pour la clé publique $pk_{\mathtt{B},1}$.

Cependant, pour un paramètre de sécurité donné, rien ne dit qu'il est possible de chiffrer une clé privée du schéma ENC_1 avec l'algorithme de chiffrement de ce schéma. Nous considérons donc un second schéma de chiffrement à clé publique ENC_2 composé des algorithmes $\{\mathsf{Clé}, \mathsf{Chiffrer}, \mathsf{Déchiffrer}\}$ tels que l'espace des messages contienne celui des clés privées du schéma ENC_1. Nous nous retrouvons à considérer des clés supplémentaires. Ainsi, la paire de clés (publique/privée) d'un utilisateur $\mathtt{U}$ est désormais un couple $(pk_{\mathtt{U}}, sk_{\mathtt{U}})$ où $pk_{\mathtt{U}} = (pk_{\mathtt{U},1}, pk_{\mathtt{U},2})$ et $sk_{\mathtt{U}} = (sk_{\mathtt{U},1}, sk_{\mathtt{U},2})$ avec $(pk_{\mathtt{U},2}, sk_{\mathtt{U},2})$ une paire de clés générée par $\mathsf{ENC}_2.\mathsf{Clé}$. Il suffit alors de définir la clé de re-chiffrement évoquée comme $\mathsf{ENC}_2(pk_{\mathtt{B},2}, sk_{\mathtt{A},1})$ pour réaliser le re-chiffrement souhaité. Ceci nous donne également $\mathsf{D}_{(1,2),vers} = \{\texttt{clé publique}\}$.

Pour le déchiffrement d'un chiffré de type 2, $\mathtt{B}$ récupère la clé privée $sk_{\mathtt{A},1}$ en utilisant sa clé privée $sk_{\mathtt{B},2}$ et l'algorithme $\mathsf{ENC}_2.\mathsf{Déchiffrer}$. Il se sert ensuite de cette première clé pour récupérer le message en utilisant l'algorithme $\mathsf{ENC}_1.\mathsf{Déchiffrer}$.

Si l'on souhaite obtenir un schéma correspondant à la Définition 24, il nous reste à décrire

l'algorithme Chiffrer sur le type 2. Nous verrons à la Section 3.2.2 comment il est possible de le décrire en passant par le type 1 et la liaison $(1, 2)$.

Nous obtenons ainsi un schéma de serveur de re-chiffrement. Sa consistance est évidente, à condition que les deux schémas de chiffrement à clé publique ENC_1 et ENC_2 soient consistants eux aussi.

3.1.3 Production d'un chiffré

Selon la Définition 24, illustrée par l'exemple de la section précédente, nous voyons qu'il y a plusieurs manières de produire un chiffré. Il peut être re-chiffré ou non et de différents types. Il est évident que tous les chiffrés n'offriront pas les mêmes fonctionnalités. Nous le constatons simplement en observant les chiffrés re-chiffrables et ceux qui ne le sont pas. Nous sommes ainsi amenés à distinguer ces différentes manières afin d'en étudier leurs fonctionnalités. Pour cela, nous introduisons la notion de chaîne de liaisons. Ceci permet de préciser la manière dont un chiffré va être produit, en listant en quelque sorte les algorithmes qui seront utilisés pour sa réalisation.

Plus précisément, nous appelons *chaîne de liaisons*, tout uplet $\eta = (\eta_0, \cdots, \eta_n) \subset \mathsf{Type}^{n+1}$, tel que, pour tout i compris entre 0 et $n-1$, $(\eta_i, \eta_{i+1}) \in \mathsf{Liaison}$, où n est n'importe quel entier naturel. Cela signifie qu'un chiffré produit en suivant une telle chaîne est issu d'un chiffré non re-chiffré de type η_0, transformé en un chiffré de type η_1, puis en un chiffré de type η_2, et ainsi de suite jusqu'à obtenir un chiffré de type η_n. En particulier, dans le cas où $n = 0$, cela signifie que le chiffré est un chiffré de type η_0 non re-chiffré. Nous appelons n la longueur d'une telle chaîne. Celle-ci permet de caractériser le nombre de re-chiffrements utilisés pour obtenir un chiffré de type η_n construit en suivant cette chaîne de liaisons - ce qui justifie que la chaîne de liaisons comporte $n+1$ éléments bien que sa longueur soit n. Nous notons également Chaîne l'ensemble des chaînes de liaisons possibles. Cet ensemble s'obtient évidemment à partir de l'ensemble Liaison.

3.2 Caractéristiques

Nous avons déjà évoqué le fait qu'il y avait dans la littérature bon nombre de modèles de serveurs de re-chiffrement, offrant des fonctionnalités différentes. Nous présentons ici deux caractéristiques retrouvées dans tous les schémas. La première s'intéresse au nombre de fois que l'on peut re-chiffrer successivement un chiffré. La deuxième concerne le type confiance entre un délégant et un délégué.

3.2.1 Nombre de re-chiffrements successifs

La première fonctionnalité à laquelle nous nous intéressons est le nombre de fois qu'un chiffré peut être successivement re-chiffré. Cela permet par exemple de transformer un chiffré pour une entité `A` en un chiffré pour une entité `B`, puis de transformer ce dernier en un chiffré pour une entité `C` et ainsi de suite. Nous pouvons en déduire une pseudo-transitivité de la confiance. En effet, nous ne sommes pas ici en présence d'une réelle notion de transitivité au sens mathématique. Même si la confiance donnée par `A` à `B` peut être déléguée à son tour à `C`, ce n'est en aucun cas une confiance que `A` donne à `C`. Elle ne peut exister sans passer par `B`. Nous la qualifions donc de pseudo-transitivé.

Habituellement, la littérature aborde deux cas :

- soit il est possible de re-chiffrer plusieurs fois un chiffré ;
- soit il n'est possible de le re-chiffrer qu'une seule fois.

Le premier cas est couramment nommé, en anglais, *multi-hop* et considère un ensemble Liaison de la forme $\{(1, 1)\}$. Cela signifie qu'un chiffré peut être transformé autant de fois qu'on le souhaite. Le second considère un ensemble Liaison de la forme $\{(1, 2)\}$, appelé en anglais *single-hop*. Dans

ce second cas, un chiffré ne peut être transformé qu'une seule fois, à condition toutefois qu'il soit de type 1, sinon il est non re-chiffrable. Nous voyons qu'il y a une autre solution possible : un nombre fini et supérieur strictement à 1 de re-chiffrements successifs, comme c'est le cas avec l'exemple $\mathsf{Liaison} = \{(1,2),(2,3)\}$. Nous considérons ici une approche légèrement différente et plus complète qui va consister à distinguer un nombre fini ou infini de re-chiffrements successifs, afin de prendre en considération toutes les solutions possibles.

Re-chiffrement infini. Dans cette catégorie, nous regroupons tout schéma de serveur de re-chiffrement qui permet de re-chiffrer successivement un chiffré une infinité de fois. Ceci implique notamment que si l'on souhaite donner à un délégué l'autorisation d'accéder à ces données chiffrées, il faut alors avoir conscience que tous les délégués de ce délégué y auront également accès, de même que les délégués des délégués des délégués... Il est possible de distinguer ces schémas des autres en observant l'ensemble $\mathsf{Chaîne}$. En effet, pour que tout chiffré soit re-chiffrable une infinité de fois, il faut et il suffit que, peu importe le nombre de re-chiffrements déjà effectués pour le produire, nous soyons toujours capable de le re-chiffrer. Ceci se traduit par l'existence, pour tout type $\eta \in \mathsf{Type}$, d'une suite $(\mathsf{Ch}_l)_{l\in\mathbb{N}} \subset \mathsf{Chaîne}^{\mathbb{N}}$, telle que, pour tout $l \in \mathbb{N}$, Ch_l soit de longueur non nulle [4], que le premier élément de Ch_0 soit η et que, pour tout $l \in \mathbb{N}$, le dernier élément de Ch_l soit le premier de Ch_{l+1}. L'existence d'une telle suite implique, en quelque sorte, celle d'une chaîne de liaisons de longueur infinie démarrant par η et qui consisterait à mettre bout à bout les différentes chaînes Ch_l. Nous pouvons rencontrer plusieurs types de situation. Par exemple, nous pouvons avoir une chaîne de liaisons de longueur infinie, composée d'un nombre fini de types de la forme $(1,1,\cdots,1,\cdots)$ ou encore $(1,2,3,1,2,3,\cdots,1,2,3,\cdots)$. Nous pouvons également avoir une chaîne de liaisons de longueur infinie composée d'une infinité de types. Il se peut que nous obtenions alors des types différents après chaque re-chiffrement, ce qui nous donne par exemple $(1,2,3,\cdots,l,\cdots)$. Nous pouvons aussi retrouver, après certains re-chiffrements, des types de chiffrés déjà rencontrés, comme par exemple $(1,2,1,3,1,4,\cdots,1,l,\cdots)$.

Re-chiffrement multiple de cardinal n. Cette catégorie considère qu'un chiffré ne peut être re-chiffré qu'un certain nombre de fois fixé à l'avance. Cela permet à un délégant de maîtriser en quelque sorte le nombre de "niveaux" de délégués qui seront autorisés à accéder aux données. Un tel système se caractérise par le fait qu'il ne doit pas exister de chaîne de liaisons de longueur infinie comme exposée ci-dessus. Nous dirons que le schéma possède un re-chiffrement multiple de cardinal n si la plus grande chaîne finie de liaisons appartenant à $\mathsf{Chaîne}$ est de longueur n. Par exemple, un re-chiffrement multiple de cardinal 2 implique que l'on peut re-chiffrer au maximum 2 fois un chiffré. C'est le cas lorsque l'on considère l'ensemble $\mathsf{Liaison} = \{(1,2),(2,3),(1,4)\}$, la plus grande chaîne de liaison appartenant à $\mathsf{Chaîne}$ est $(1,2,3)$, de longueur 2. Plus particulièrement, seul un chiffré de type 1 est re-chiffrable à deux reprises successives, alors qu'un chiffré de type 3 est non re-chiffrable. Ce type de schéma nous permet également d'obtenir un re-chiffrement multiple de cardinal i avec $i < n$. En effet, sur l'exemple considéré et en se limitant aux types 2 et 3, nous obtenons un re-chiffrement multiple de cardinal 1. En particulier, un chiffré de type 2 n'est re-chiffrable qu'une seule fois.

Re-chiffrement unique. Cette catégorie est en réalité une sous-catégorie de la précédente en prenant $n = 1$, ce qui implique qu'un chiffré ne peut être au maximum re-chiffré qu'une seule fois. Nous la distinguons car elle est souvent observée et correspond exactement au cas *single-hop* de la littérature. De cette manière, un délégué ne pourra déléguer les droits d'accès aux fichiers qu'il a obtenu d'un délégant. Dans ce cas, l'ensemble sera typiquement de la forme $\{(1,2)\}$. Mais un ensemble de la forme $\{(1,2),(3,4),(5,6),\cdots\}$ pourra aussi convenir. Nous l'utiliserons notamment dans le Chapitre 8.

4. Cette condition est importante pour ne pas prendre en considération les chaînes de longueur nulle qui n'impliquent aucun re-chiffrement.

3.2.2 Type de confiance entre le délégant et le délégué

Nous nous intéressons ici à la relation entre un délégant et un délégué. Plus particulièrement, nous nous attardons sur le fait qu'une clé de re-chiffrement ne permette que de re-chiffrer du délégant vers le délégué ou si en plus elle autorise le re-chiffrement dans l'autre sens, c'est-à-dire du délégué vers le délégant. Cela revient à un échange possible des rôles entre le délégant et le délégué. Nous présentons en même temps ces deux hypothèses, puisqu'elles sont opposées.

Re-chiffrement bidirectionnel ou unidirectionnel.

Considérons (η, η') une liaison, $\mathtt{A}$ et $\mathtt{B}$ deux entités dont les paires de clés (privée/publique) respectives sont générées par l'algorithme $\mathsf{Clé}$ ainsi que $E_{\mathtt{A},\mathtt{B},\eta,\eta'}$ l'événement : "la clé $R_{\mathtt{B},\eta\to\mathtt{A},\eta'}$ est calculable en temps polynomial (du paramètre de sécurité) à partir de la clé $R_{\mathtt{A},\eta\to\mathtt{B},\eta'}$". Nous dirons alors que le re-chiffrement (respectivement la clé de re-chiffrement) sur la liaison (η, η') est :

- *bidirectionnel* (respectivement *bidirectionnelle*) si la probabilité de $E_{\mathtt{A},\mathtt{B},\eta,\eta'}$, pour $\mathtt{A}$ et $\mathtt{B}$ deux entités quelconques, est à distance négligeable de 1.
- *unidirectionnel* (respectivement *unidirectionnelle*) si la probabilité de $E_{\mathtt{A},\mathtt{B},\eta,\eta'}$, pour $\mathtt{A}$ et $\mathtt{B}$ deux entités quelconques, est à distance négligeable de 0.

Dans toute la suite, nous ne considérerons pas les autres cas, comme par exemple une probabilité à distance négligeable de 0.5. Ainsi, nous n'aurons affaire qu'à deux types de re-chiffrements, ce qui nous permettra de simplifier notre étude. Cette restriction n'est pas contraignante pour notre étude puisqu'aucune occurence des autres cas n'existe dans la littérature.

Caractérisation. Tout comme nous l'avons fait pour la fonctionnalité précédente, il serait intéressant d'obtenir une caractérisation de ces deux particularités en fonction de l'environnement du schéma. Considérons pour cela un re-chiffrement du type η vers le type η'. Il est évident que les ensembles Type et $\mathsf{Liaison}$ ne vont être d'aucune utilité pour déterminer si ce re-chiffrement est bidirectionnel ou unidirectionnel. Les seuls éléments de l'environnement du schéma qui peuvent le déterminer sont donc les ensembles $\mathsf{D}_{(\eta,\eta'),de}$ et $\mathsf{D}_{(\eta,\eta'),vers}$.

Revenons sur la Remarque 3. Nous y avions exposé le fait que $\mathsf{D}_{(\eta,\eta'),de}$ était au moins composé de l'élément `clé privée`. Si le re-chiffrement est bidirectionnel, alors, pour des raisons de symétrie entre délégant et délégué, $\mathsf{D}_{(\eta,\eta'),vers}$ doit également être composé de cet élément. Par contraposée et avec la restriction évoquée précédemment, nous aurions affaire dans le cas contraire à un re-chiffrement unidirectionnel [5]. Cependant, la réciproque n'est pas vraie. En effet, l'élément $g^{sk_\mathtt{B}/sk_\mathtt{A}}$ calculé à partir de $sk_\mathtt{A}$ et $sk_\mathtt{B}$ est une clé de re-chiffrement unidirectionnelle de $\mathtt{A}$ vers $\mathtt{B}$ sous l'hypothèse DHI, utilisée notamment dans [LV08b]. Ceci nous donne, en tout cas, une caractérisation partielle du re-chiffrement unidirectionnel, puisque si $\mathsf{D}_{(\eta,\eta'),vers} = \{\mathtt{clé\ publique}\}$, alors le re-chiffrement est unidirectionnel.

Remarque 5. Comme nous l'avons mentionné à la Section 3.1.2, il est possible de décrire dans certains cas l'algorithme $\mathsf{Déchiffrer}_{\eta'}$ à partir des algorithmes $\mathsf{Déchiffrer}_{\eta}$ et $\mathsf{Re\text{-}chiffrer}_{(\eta,ta')}$. Cette possibilité requiert de disposer d'un re-chiffrement unidirectionnel sur la liaison (η, η') tel que l'ensemble $\mathsf{D}_{(1,2),vers}$ ne soit composé que d'éléments publics. De cette manière, il est possible de calculer une clé de re-chiffrement sur cette liaison sans que le délégué correspondant n'ait à intervenir dans son calcul. Cette caractéristique permet alors d'automatiser la description de $\mathsf{Déchiffrer}_{\eta'}$ pour une entité $\mathtt{B}$ de la manière suivante :

1. Générer une paire de clés $(sk_\mathtt{T}, pk_\mathtt{T})$ pour une entité temporaire $\mathtt{T}$ en utilisant l'algorithme $\mathsf{Initialiser}$ et chiffrer m pour cette clé publique et le type η afin d'obtenir $C_{\mathtt{T},\eta} = \mathsf{Chiffrer}(\mathcal{P}, \eta, pk_\mathtt{T}, m)$.
2. Calculer une clé de re-chiffrement de $\mathtt{T}$ du type η vers $\mathtt{A}$ du type η' afin d'obtenir $R_{\mathtt{T},\eta\to\mathtt{A},\eta'} = \mathsf{Délégation}(\mathcal{P}, (\eta, \eta'), \mathsf{D}^{\mathtt{T}}_{(\eta,\eta'),de}, \mathsf{D}^{\mathtt{A}}_{(\eta,\eta'),vers})$.

5. Sans la restriction considérée, nous ne pourrions pas conclure que le re-chiffrement est unidirectionnel.

3. Re-chiffrer $C_{\mathtt{T},\eta}$ en $C_{\mathtt{A},\eta'} = \mathsf{Re\text{-}chiffrer}(\mathcal{P}, (\eta, \eta'), R_{\mathtt{T},\eta\to\mathtt{A},\eta'}, C_{\mathtt{T},\eta})$.

Il est facile de voir que si le re-chiffrement produit bien un chiffré de type η', alors il en sera de même pour le chiffrement que nous venons de définir. Cette méthode n'est bien sûr pas applicable à tout autre re-chiffrement, en particulier à un re-chiffrement bidirectionnel, puisqu'il nécessite la clé privée du délégué.

Cette technique a un intérêt théorique et permet de simplifier la conception des schémas en automatisant la description de certains algorithmes. Nous ne l'utiliserons pas pour autant en pratique, car comme nous le verrons dans la description du schéma de la Section 6.2, l'utilisation de cette technique implique bien souvent plus de calculs qu'il n'en est réellement nécessaire pour arriver à produire un chiffré de type η'. En effet, il paraît naturellement plus rapide de produire un chiffré pour `B` plutôt que de produire un chiffré pour `A` et de le re-chiffrer ensuite pour `B` et c'est pourquoi elle n'est jamais utilisée dans la description des schémas de la littérature.

3.3 Sécurité d'un serveur de re-chiffrement

Nous présentons maintenant les notions de sécurité que l'on peut rechercher pour un schéma de serveur de re-chiffrement.

3.3.1 Notions de sécurité

Les notions de sécurité, relatives au serveur de re-chiffrement, rejoignent les notions de sécurité vues au Chapitre 2 sur le chiffrement à clé publique. Ainsi, le but idéal pour cette primitive cryptographique correspond également à la sécurité IND-CCA.

Sécurité sémantique. Nous ne présentons que le scénario correspondant à la sécurité IND-CCA, les autres scénarios consistant à restreindre les moyens mis à disposition de l'adversaire. Ce scénario ressemble globalement à celui présenté au Chapitre 2.

Il existe cependant des différences propres au serveur de re-chiffrement. En effet, dans un schéma de serveur de re-chiffrement, nous disposons de fonctionnalités supplémentaires, comme par exemple des liens éventuels entre les différentes entités via les clés de re-chiffrement. Ainsi le scénario doit être adapté pour en tenir compte. Le scénario que nous présentons ici est inspiré de celui présenté par Canetti et Hohenberger dans [CH07] pour les schémas avec re-chiffrement bidirectionnel et infini. Il fut ensuite transposé aux schémas avec re-chiffrement unidirectionnel et unique par Libert et Vergnaud dans [LV08b]. En particulier, il consiste à fournir à l'adversaire des moyens portant sur la partie re-chiffrement et pas seulement sur la partie chiffrement.

Ainsi, en plus de parvenir à corrompre certaines entités et d'obtenir leurs clés privées, nous autorisons l'adversaire à récupérer des clés de re-chiffrement entre les différentes entités. Il est évident que l'adversaire peut obtenir les clés de re-chiffrement entre des entités corrompues puisqu'il possède les clés privées qui permettent de les calculer. Nous modélisons également le fait qu'il puisse corrompre un serveur et donc récupérer des clés de re-chiffrement entre des entités honnêtes, et même entre des entités honnêtes et des entités corrompues.

De plus, comme nous l'avons vu, nous n'avons pas affaire à une seule manière de produire un chiffré. Un chiffré peut, par exemple, être obtenu en utilisant seulement l'algorithme Chiffrer ou par l'exécution successive de ce dernier et d'au moins une fois l'algorithme Re-chiffrer. Il faut alors distinguer toutes les manières de produire un chiffré en se servant de l'ensemble Chaîne. Nous nous retrouvons, ainsi, à devoir considérer de nombreux scénarios contre l'indiscernabilité de chiffrés, correspondant chacun à un élément de cet ensemble. En conséquence, pour prouver qu'un schéma est sécurisé, il faudra prouver qu'il l'est pour chacun de ces scénarios.

Remarque 6. Nous verrons, qu'il est possible, dans certains cas, de réduire le nombre de ces scénarios à considérer. Ce sera notamment le cas lorsqu'il sera possible de décrire l'algorithme Chiffrer sur certains types à partir d'autres types comme expliqué à la Remarque 5.

Nous présentons maintenant le scénario d'attaque que nous allons envisager dans le cadre d'une sécurité IND-CCA, pour une manière de produire un chiffré en suivant la chaîne de liaisons $\mathsf{Ch} \in \mathsf{Chaîne}$ et expliquons ensuite certains de nos choix.

Scénario d'attaque. Le scénario d'attaque contre une indiscernabilité de chiffrés entre un adversaire et un challengeur pour un schéma de serveur de re-chiffrement et pour une manière de produire un chiffré $\mathsf{Ch} \in \mathsf{Chaîne}$, peut être découpé et décrit de la manière suivante :

Phase d'initialisation
- Le challengeur initialise les paramètres publics $\mathcal{P}$ du système en fonction d'un paramètre de sécurité λ en utilisant l'algorithme $\mathsf{Initialiser}$ et les donne à l'adversaire.
- Il génère également une paire de clés (sk^*, pk^*) dont l'entité correspondante sera appelée entité cible et notée $\mathtt{A}^*$. Il donne ensuite pk^* à l'adversaire.
- L'adversaire choisit également, dans le cas où la longueur de la chaîne Ch est de longueur n strictement positive, n entités corrompues $\mathtt{A}_1, \cdots, \mathtt{A}_n$ - pas forcément toutes distinctes deux à deux - et fournit ce choix au challengeur qui s'en servira lors du challenge. Celui-ci génère ensuite, pour chacune de ces entités, une paire de clés (publique/privée) et fournit toutes ces paires de clés, notées respectivement $(sk_1, pk_1), \cdots, (sk_n, pk_n)$ à l'adversaire. Ces entités vont servir au challengeur à produire un chiffré challenge en suivant la chaîne Ch et en passant par les entités successives $\mathtt{A}_1, \cdots, \mathtt{A}_n, \mathtt{A}^*$ afin d'obtenir un chiffré pour l'entité cible $\mathtt{A}^*$.
- À la demande de l'adversaire portant sur la création d'une nouvelle entité $\mathtt{A}$, il génère une paire de clés $(sk_{\mathtt{A}}, pk_{\mathtt{A}})$ et répond de la manière suivante :
  - si la requête porte sur la création d'une entité $\mathtt{A}$ honnête, le challengeur donne $(pk_{\mathtt{A}})$ à l'adversaire.
  - si la requête porte sur la création d'une entité $\mathtt{A}$ corrompue, le challengeur donne $(sk_{\mathtt{A}}, pk_{\mathtt{A}})$ à l'adversaire.

Phase 1 L'adversaire effectue différentes requêtes auprès du challengeur qui lui répond de la manière suivante, avec certaines restrictions dites naturelles que nous exposerons par la suite :
- Si l'adversaire souhaite obtenir la clé de re-chiffrement de $\mathtt{A}$ pour le type η vers $\mathtt{B}$ pour le type η', il formule sa requête sur l'entrée $(\mathtt{A}, \eta, \mathtt{B}, \eta')$. Le challengeur génère alors, grâce à $\mathsf{Délégation}$, cette clé et la fournit à l'adversaire.
- Si l'adversaire souhaite obtenir un re-chiffrement d'un chiffré C de type η destiné à $\mathtt{A}$ en un chiffré de type η' destiné à $\mathtt{B}$, il formule sa requête sur l'entrée $(C, \mathtt{A}, \eta, \mathtt{B}, \eta')$. Le challengeur utilise alors une clé de re-chiffrement de $\mathtt{A}$ pour le type η vers $\mathtt{B}$ pour le type η' - ou la définit comme dans la requête précédente s'il n'en existe pas encore - pour effectuer la transformation du chiffré, afin d'obtenir un chiffré C' qui est alors transmis à l'adversaire. Dans le cas où le chiffré C est mal formé, le challengeur lui fournit $\perp$. Lors d'une telle opération, nous dirons que l'adversaire fait appel à un **oracle de re-chiffrement**.
- Si l'adversaire souhaite obtenir un déchiffrement d'un chiffré C de type η destiné à $\mathtt{A}$, il formule sa requête sur l'entrée $(C, \mathtt{A}, \eta)$. Le challengeur utilise la clé privée $sk_{\mathtt{A}}$ pour obtenir le message correspondant et le transmet à l'adversaire. Dans le cas où le chiffré C est mal formé, le challengeur lui fournit $\perp$. Lors d'une telle opération, nous dirons que l'adversaire fait appel à un **oracle de déchiffrement**.

Challenge Le challengeur fournit à l'adversaire un challenge produit de la manière suivante, selon deux cas possibles [6] :
- la longueur de la chaîne Ch est nulle, Ch est donc de la forme (η^*). L'adversaire fournit au challengeur deux messages m_0 et m_1 appartenant tous les deux à $\mathcal{M}$. Ce dernier lui

6. Nous omettons volontairement, une fois de plus, les paramètres publics $\mathcal{P}$ dans les différents algorithmes considérés.

retourne un chiffré $C^* = \mathsf{Chiffrer}_{\eta^*}(pk^*, m_b)$, qui est bien un chiffré produit en suivant la chaîne Ch, où b est tiré aléatoirement parmi $\{0,1\}$.
– la longueur de la chaîne Ch est strictement positive, Ch est donc de la forme $(\eta_1, \cdots, \eta_n, \eta^*)$ avec $n > 0$. L'adversaire fournit au challengeur deux messages m_0 et m_1 appartenant tous les deux à $\mathcal{M}$. Ce dernier lui retourne un chiffré C^*, décrit ci-dessous, produit en suivant la chaîne Ch et où b est tiré aléatoirement parmi $\{0,1\}$.

$$C^* = \mathsf{Re\text{-}chiffrer}_{(\eta_n,\eta^*)}(rk_{\mathtt{A}_n,\eta_n\to\mathtt{A}^*,\eta^*}, \mathsf{Re\text{-}chiffrer}_{(\eta_{n-1},\eta_n)}(rk_{\mathtt{A}_{n-1},\eta_{n-1}\to\mathtt{A}_n,\eta_n}, \ldots, \mathsf{Re\text{-}chiffrer}_{(\eta_1,\eta_2)}(rk_{\mathtt{A}_1,\eta_1\to\mathtt{A}_2,\eta_2}, \mathsf{Chiffrer}(\eta_1, pk_1, m_b))\ldots)),$$

Phase 2
– L'adversaire peut effectuer un certain nombre de requêtes correspondantes à celles autorisées dans la *Phase 1* avec cependant certaines restrictions supplémentaires liées au challenge que nous détaillons par la suite.

Réponse
– L'adversaire renvoie finalement une réponse b' appartenant à $\{0,1\}$ et est déclaré vainqueur si $b' = b$.

Définition 25. *Dénotons par $\mathcal{A}$ l'adversaire contre ce challenge et par $\mathsf{Adv}_{\mathsf{PRE},\mathcal{A},\mathsf{Ch}}(\lambda)$, l'avantage qu'il a de gagner ce scénario par rapport à 1/2. Soit alors $\mathsf{Adv}_{\mathsf{PRE},\mathsf{Ch}}(\lambda) = \max_{\mathcal{A}}[(\mathsf{Adv}_{\mathsf{PRE},\mathcal{A},\mathsf{Ch}}(\lambda)]$, le maximum des avantages pris sur tous les adversaires $\mathcal{A}$ fonctionnant en temps polynomial en le paramètre de sécurité λ. Le schéma de serveur de re-chiffrement est dit sûr contre l'indiscernabilité de messages chiffrés pour cette manière d'obtention d'un chiffré Ch si $\mathsf{Adv}_{\mathsf{PRE},\mathsf{Ch}}(\lambda)$ est négligeable en le paramètre de sécurité λ.*

Définition 26. *Un schéma de serveur de re-chiffrement est dit sûr contre l'indiscernabilité de chiffrés s'il est sûr contre l'indiscernabilité de messages chiffrés pour chaque chaîne $\mathsf{Ch} \in \mathsf{Chaîne}$.*

Revenons maintenant sur certains choix que nous avons effectués dans le déroulement du scénario d'attaque. Tous ces choix permettent de simplifier la réalisation et la compréhension des preuves de sécurité.

Corruption non adaptative. Nous avons déjà expliqué au Chapitre 2 les raisons du choix du scénario avec corruption non adaptative.

Choix des différentes entités $\mathtt{A}^*, \mathtt{A}_1, \cdots, \mathtt{A}_n$ pendant la *Phase d'initialisation*. Nous avons considéré également, pour des raisons de simplicité, que ces différentes entités sont choisies par l'adversaire au début du scénario dans la phase d'initialisation. Ce scénario est équivalent à celui où ces différentes entités ne sont choisies qu'au moment de la phase *Challenge*, en utilisant le même raisonnement que celui évoqué ci-dessus.

Par exemple, pour l'entité cible, cela revient à choisir à l'avance parmi toutes les entités honnêtes, laquelle sera l'entité cible. La probabilité de faire le bon choix n'est pas négligeable et, plus précisément, est polynomial en le nombre d'utilisateurs. De plus, nous générons cette entité indépendamment de celles qui sont honnêtes afin de simplifier la description des preuves de sécurité. En effet, nous verrons au cours de ce mémoire, que la clé publique de l'entité cible est, en général, générée différemment des clés publiques des entités honnêtes. Ceci évite d'utiliser une technique dite, *du jeton de Coron* [Cor00], consistant à introduire le problème difficile considéré dans les preuves de sécurité dans différentes entités afin de maximiser la probabilité de résoudre ce problème. Cette considération permet de simplifier grandement la réalisation des preuves de sécurité et aurait, par exemple, permit d'éviter certaines erreurs dans la preuve proposée par Chow *et al.* dans leur papier [CWYD10] que nous préciserons au Chapitre 6.

Concernant les entités $\mathtt{A}_1, \cdots, \mathtt{A}_n$, elles sont choisies corrompues puisque le challenge est chiffré pour $\mathtt{A}^*$ et seulement lui. L'adversaire pourrait tout simplement les sélectionner parmi les entités

corrompues générées. Ce scénario est évidemment équivalent au nôtre. Nous avons cependant fait le choix de les générer indépendamment, car il est possible, dans certains schémas, que des restrictions sur ces entités interviennent. C'est notamment le cas du schéma [CWYD10] où $n = 1$ et $\mathtt{A}_1$ doit être honnête si l'adversaire a à sa disposition la clé de re-chiffrement de $\mathtt{A}_1$ vers $\mathtt{A}^*$. Il est ainsi possible, dans de tels exemples, qu'elles soient choisies comme honnêtes. En rejoignant notre remarque sur la génération indépendante de l'entité cible, il est alors également possible qu'elles soient générées, lors des preuves de sécurité, d'une manière particulière différente des autres entités. Afin que notre scénario soit valable pour les schémas ayant de telles restrictions et propriétés, nous avons considéré que les clés correspondantes étaient générées indépendamment des autres entités.

Le scénario IND-CPA. Le scénario que nous avons décrit correspond à une attaque CCA. Celui correspondant à une attaque CPA est très facile à obtenir à partir de ce premier. En fait, il consiste tout simplement à ne pas donner à l'adversaire l'accès aux oracles de re-chiffrement et de déchiffrement que nous avons exposés.

3.3.2 Restrictions naturelles dans les moyens de l'adversaire

Nous venons de voir que dans le cas du scénario IND-CCA, l'adversaire a accès à différents oracles et à différentes clés. Implicitement, l'utilisation de ces différents oracles suppose que l'adversaire ne puisse obtenir toutes les clés privées ni toutes les clés de re-chiffrement qu'il souhaite. C'est en effet le cas et pas seulement pour celui-ci mais également pour celui d'IND-CPA [7]. Ces restrictions interviennent si l'on souhaite que l'adversaire ait un "vrai" challenge. On parle alors de restrictions naturelles. Par exemple, il est évident que l'adversaire ne doit en aucun cas obtenir la clé privée de l'entité cible $\mathtt{A}^*$, auquel cas il pourrait déchiffrer le challenge et gagner à tous les coups. Cette restriction est déjà prise en compte dans notre scénario puisque l'entité cible est honnête et que nous avons considéré un modèle de corruption non adaptatif. Les schémas de chiffrement à clé publique ne considèrent que cette restriction. Étant donné que dans un schéma de serveur de re-chiffrement, nous disposons de liens entre les différentes entités, des restrictions supplémentaires doivent être prises en compte.

L'idée pour trouver toutes ces restrictions est en fait très simple. Il suffit juste de faire attention à ce que l'adversaire n'ait aucun moyen trivial de gagner le challenge. Il faut ainsi l'empêcher de pouvoir déchiffrer le challenge ou un re-chiffré de celui-ci, via l'utilisation d'une clé privée ou de l'oracle de déchiffrement. Nous nous intéressons maintenant à préciser ces restrictions pour les différents challenges possibles.

Cas d'un challenge non re-chiffrable. Dans un tel scénario, le chiffré challenge ne peut être re-chiffré, ainsi les clés de re-chiffrement ne vont pas, a priori, être utiles à l'adversaire. Celui-ci peut ainsi obtenir la clé de re-chiffrement sur une liaison (η, η') d'une entité $\mathtt{A}$ vers une entité $\mathtt{B}$, pour toute liaison (η, η') et toutes entités $\mathtt{A}$ et $\mathtt{B}$. Il n'y a donc pas de restrictions non naturelles sur les clés de re-chiffrement. Pour des raisons similaires, il n'y a aucune restriction sur l'oracle de re-chiffrement, puisque le challenge ne peut être re-chiffré. En revanche, l'adversaire ne doit pas évidemment obtenir le déchiffrement du challenge, qui est donc la seule restriction non naturelle à considérer dans ce cas.

Chiffré challenge re-chiffrable. Contrairement au cas précédent, il n'y a pas qu'une seule restriction à considérer. Soit η^* le type de chiffré du challenge. Il est facile de voir que, pour chaque liaison $(\eta^*, \eta) \in \mathit{Liaison}$, l'adversaire ne doit pas obtenir une clé de re-chiffrement de $\mathtt{A}^*$ pour le type η^* vers n'importe quelle entité corrompue pour le type η. En effet, cette clé permettrait à l'adversaire de re-chiffrer le challenge vers une entité corrompue, entité pour laquelle il possède

7. Bien évidemment, les restrictions concernant les oracles de re-chiffrement ou de déchiffrement ne concernent que les attaques à chiffrés choisis.

la clé privée. L'adversaire peut, en revanche, demander à obtenir n'importe quelle autre clé de re-chiffrement, puisqu'elle ne lui servira a priori à rien. Pour des raisons similaires, ce dernier ne peut pas demander à l'oracle de re-chiffrement, qu'il lui fournisse un re-chiffrement du challenge pour une entité corrompue. De plus, il ne peut toujours pas, bien évidemment, obtenir de l'oracle de déchiffrement un déchiffrement du challenge.

Pour autant, la liste des restrictions n'est pas complète. Nous avons considéré que le challenge est re-chiffrable. Ainsi, l'adversaire peut obtenir, soit en utilisant une clé de re-chiffrement qui lui a été donnée soit par l'oracle de re-chiffrement, le re-chiffrement du challenge pour une entité honnête. En effet, l'adversaire ne pourra pas obtenir directement d'information sur ce re-chiffré ; il est donc autorisé à en disposer. Cependant, il pourrait utiliser d'autres moyens pour en obtenir. Par exemple, il pourrait utiliser des clés de re-chiffrement, si le re-chiffré du challenge est re-chiffrable à son tour, pour le modifier en un chiffré destiné à une entité corrompue. Nous avons alors affaire à des restrictions supplémentaires. C'est en quelque sorte, comme si tout re-chiffré du challenge devenait également un chiffré challenge, sur lequel il faut considérer les différentes restrictions que nous venons de voir. Un chiffré challenge peut ainsi produire d'autres chiffrés challenge par re-chiffrements successifs jusqu'à parvenir à des chiffrés challenges qui soient non rechiffrables et, auquel cas, les dernières restrictions à considérer sont celles évoquées dans le cas précédent.

Il serait compliqué de lister en détail, et formellement, l'ensemble des restrictions possibles. Regardons, par exemple, le cas de l'ensemble $\mathsf{Liaison} = \{1, 2, 3\}$ ainsi que d'un chiffré challenge de type 1 pour une entité $\mathtt{A}^*$. Nous avons vu que l'adversaire ne pouvait obtenir de clés de re-chiffrement de $\mathtt{A}^*$ pour le type 1 vers une entité corrompue pour le type 2. En revanche, il peut obtenir un re-chiffrement du challenge vers le type 2 pour une entité honnête, notée $\mathtt{A}_2$. Il serait alors justifié d'empêcher l'adversaire d'obtenir une clé de re-chiffrement de $\mathtt{A}_2$ pour le type 2 vers une entité corrompue pour le type 3. Cependant, dans le cas général, cette restriction n'a pas de sens, mais dépend des différentes requêtes initiées par l'adversaire. En effet, si celui-ci n'effectue aucune requête de re-chiffrement du challenge ou n'obtient aucune clé de re-chiffrement lui permettant de le re-chiffrer, il n'y a alors aucune raison de ne pas lui donner la clé de re-chiffrement de $\mathtt{A}_2$ pour le type 2 vers une entité corrompue pour le type 3. Cette restriction dépend donc des moyens que l'adversaire a utilisés. En étendant cette idée à un ensemble plus général de $\mathsf{Liaison}$, il deviendrait vite compliqué de lister l'ensemble des restrictions à considérer. En particulier, nous aurions une liste avec beaucoup de "si" dépendant des clés de re-chiffrement obtenues par l'adversaire et des requêtes qu'il a effectuées. L'essentiel pour savoir si une requête de l'adversaire est légitime ou pas est de se souvenir des deux points suivants :

- L'adversaire ne doit pas obtenir un moyen qui lui permettrait de re-chiffrer le challenge pour une entité corrompue.
- L'adversaire ne doit pas obtenir un moyen de déchiffrer le challenge et ses éventuels re-chiffrés successifs.

3.3.3 Restrictions non naturelles récurrentes

Nous avons vu qu'il pouvait y avoir beaucoup de restrictions naturelles dépendant de l'ensemble $\mathsf{Liaison}$ ainsi que des requêtes effectuées par l'adversaire. Pour autant, beaucoup de schémas admettent des restrictions supplémentaires, qui dépendent des différents algorithmes les composant. Par exemple, c'est le cas du schéma [CWYD10], dont l'ensemble $\mathsf{Liaison}$ est $\{(1, 2)\}$. Dans le scénario d'un challenge re-chiffré de $\mathtt{A}_1$ pour le type 1 vers $\mathtt{A}^*$ pour le type 2 de ce schéma, $\mathtt{A}_1$ ne peut être corrompue si l'adversaire obtient la clé de re-chiffrement de $\mathtt{A}_1$ pour le type 1 vers $\mathtt{A}^*$ pour le type 2. La raison annoncée est que cela permettrait à l'adversaire d'obtenir la clé de déchiffrement du challenge. Cette restriction est propre à ce schéma et il n'y a aucune raison pour qu'elle soit prise en compte dans les autres. Nous parlerons alors de restrictions non naturelles. Nous nous attardons à en présenter certaines récurrentes.

La restriction RCCA. L'attaque RCCA, pour attaque à messages chiffrés choisis rejouables - de l'anglais *replayable chosen ciphertext attacks* -, est une sécurité restrictive par rapport à l'attaque CCA. Bien qu'introduite pour les schémas de chiffrement à clé publique par Canetti *et al.* dans l'article [CKN03], elle a également un sens pour les schémas de serveur de re-chiffrement. Dans un tel scénario, l'adversaire n'a plus le droit d'effectuer des requêtes de déchiffrement portant sur l'un des deux messages m_0 et m_1 qu'il a choisi lors de la phase *Challenge*. Nous reviendrons sur l'étude de cette sécurité, dans le cas d'un schéma de serveur de re-chiffrement au Chapitre 5. En particulier, nous étudierons la relation entre les sécurités RCCA et CCA.

Le re-chiffrement bidirectionnel. Les schémas sécurisés de serveur de re-chiffrement offrant un re-chiffrement bidirectionnel dans la littérature ont tous la même caractéristique. En effet, en les observant, on s'aperçoit que les clés de re-chiffrement sont généralement de la forme y/x avec y la clé privée du délégué et x la clé privée du délégant ou sous une forme équivalente comme $y - x$. Il est évident que si l'on possède une telle clé de re-chiffrement et l'une des deux clés privées, on en retrouve facilement la deuxième. Pour cette raison, dans les sécurités étudiées, l'adversaire ne peut obtenir de clés de re-chiffrement entre une entité honnête et une entité corrompue, puisqu'il obtiendrait directement la clé privée de l'entité honnête. Cette spécificité donne donc tout un ensemble de restrictions supplémentaires. Cela peut apparaître très restrictif, mais ces schémas de la littérature sont plus efficaces que ceux offrant un re-chiffrement unidirectionnel, ce qui justifie leur utilisation malgré tout. Ces restrictions seront souvent considérées et nous noterons alors $\mathrm{ATT_B}$, l'attaque ATT à laquelle on les ajoute, pour $\mathrm{ATT} \in \{\mathrm{CPA}, \mathrm{CCA}, \mathrm{RCCA}\}$.

Il faut cependant savoir que l'on est capable de concevoir des schémas avec un re-chiffrement bidirectionnel qui ne nécessitent pas de considérer ces restrictions non naturelles. C'est le cas, par exemple, lorsque nous considérons comme clé de re-chiffrement la concaténation de deux clés correspondant chacune à une clé de re-chiffrement unidirectionnelle - typiquement $(g^{y/x}, g^{x/y})$. Nous reparlerons de cette idée à la Section 4.2.1. Cela donne bien un schéma de serveur de re-chiffrement bidirectionnel. Ainsi, même si l'une des deux entités impliquées dans cette clé est corrompue, la clé privée de la deuxième reste confidentielle. En utilisant ce type de clé de re-chiffrement, on est capable de concevoir un schéma bidirectionnel sans restriction non naturelle en s'appuyant sur un schéma unidirectionnel. Cependant, seule une partie de la clé de re-chiffrement est utilisée lors du re-chiffrement. L'efficacité du schéma sera alors celle du schéma unidirectionnel sous-jacent et donc moins intéressante que celle que l'on peut attendre d'un "vrai" schéma bidirectionnel. Ce type de clé bidirectionnelle peut par exemple être utilisé dans des applications requiérant des confiances mutuelles entre les entités mais où l'on souhaite empêcher qu'une coalition du serveur et d'utilisateurs puisse retrouver la clé privée d'un autre utilisateur.

3.3.4 Spécificités liées aux preuves de sécurité

Vérifiabilité d'un chiffré. Afin d'atteindre une sécurité contre des attaques actives telles que IND-CCA, IND-CCA_B et IND-RCCA, il faut que les différents oracles s'assurent de la validité des chiffrés qu'ils manipulent. Dans le cas contraire, l'adversaire pourrait utiliser ces oracles pour obtenir de l'information qu'il ne devrait pas avoir. S'il y arrive, il peut par exemple casser la non-malléabilité d'un chiffré qui est prouvée, dans [BN08], être équivalente à la sécurité sémantique dans le cas d'une attaque CCA sans restrictions non naturelles.

Cette vérifiabilité se fait habituellement en déchiffrant un chiffré. En effet, lors de cette opération, on retrouve fréquemment toutes les données temporaires qui ont servi à sa production, ce qui permet de voir comment il l'a été. En revanche, le re-chiffrement ne permet pas de voir toutes ses valeurs et donc de s'assurer de la vérifiabilité de cette manière. Il faut dans ce cas que la vérifiabilité du chiffré puisse être observée à partir de données publiques (puisque réalisée par le serveur) - on parle alors de *vérifiabilité publique d'un chiffré* -, comme par exemple la clé publique du destinataire du chiffré. Cependant, tous les chiffrés n'ont pas à disposer de cette vérifiabilité

publique. En effet, les chiffrés non re-chiffrables ne sont pas tenus d'être publiquement vérifiables, même s'il arrive que ce soit le cas, comme par exemple dans le schéma [LV08b].

Cette notion est très importante, bien que souvent négligée dans de nombreux schémas comme dans [MNT10] ou partiellement vérifiée comme dans [CWYD10] surlequel nous reviendrons au Chapitre 6.

Sur la nécessité d'un oracle de déchiffrement. Il est possible de simplifier la réalisation de preuves de sécurité en ne décrivant l'oracle de déchiffrement que sur certains types de chiffrés. Cela consiste à ne pas en donner l'accès à l'adversaire pour certains chiffrés re-chiffrables. Considérons, par exemple, le cas d'une liaison $(1, 2)$. Etant donné qu'un chiffré de type 1 peut toujours être re-chiffré en un type 2, il n'est alors pas nécessaire de définir l'oracle de déchiffrement sur les chiffrés de type 1. En effet, celui-ci peut toujours être vu comme une succession de l'oracle de re-chiffrement puis de l'oracle de déchiffrement sur le type 2. L'adversaire peut donc initier ces deux requêtes s'il souhaite obtenir le message correspondant à un chiffré de type 1 (sauf dans le cas de restrictions naturelles comme vu précédemment).

Cette réflexion a été exposée par Libert et Vergnaud pour leur schéma [LV08b]. Elle a pourtant été réfutée par Hanaoka *et al.* dans [HKK+12] qui exhibèrent un exemple de construction générique de schéma de serveur de re-chiffrement avec la particularité que le schéma obtenu soit :

- sécurisé si l'oracle de déchiffrement sur le type 1 était décrit ;
- non sécurisé si l'oracle de déchiffrement sur le type 1 ne l'était pas.

Cette construction ne fournit en réalité pas de schéma de serveur de re-chiffrement. En effet, il est possible de créer avec l'algorithme $\mathsf{Chiffrer}_1$ produit par cette construction générique, un chiffré de type 1 qui peut être déchiffré mais qui ne peut pas être re-chiffré. Ceci contredit l'existence de la liaison $(1, 2)$. En détail, un chiffré de cette forme est, avec leurs notations, de la forme $(c\|1)$ où c est un chiffré de n'importe quel schéma de serveur de re-chiffrement atteignant la sécurité IND-CCA. L'algorithme $\mathsf{Déchiffrer}_1$ accepte les chiffrés de la forme $(c\|0)$ ou $(c\|1)$, alors que l'algorithme $\mathsf{Re\text{-}chiffrer}_{(1,2)}$ n'accepte que ceux de la forme $(c\|0)$. Leur schéma n'est donc pas consistant et il est facile de voir que la réflexion de Libert et Vergnaud est correcte.

Nous venons d'étudié la non-nécessité de l'oracle de déchiffrement dans le cas d'un chiffré de type 1 re-chiffrable en un type 2 mais elle est généralisable à tout chiffré re-chiffrable. Il faut cependant faire attention et éviter de "tourner en rond", ce qui peut arriver si l'on considère par exemple l'ensemble $\mathsf{Liaison} = \{(1, 2), (2, 1)\}$. Il convient alors de faire un choix et de définir l'oracle de déchiffrement sur un type particulier, sa description sur l'autre type se faisant via re-chiffrement.

Sur la nécessité d'un oracle de re-chiffrement. En restant sur la même idée de simplification des preuves de sécurité, il est également possible de ne pas définir l'oracle de re-chiffrement sur certaines liaisons. Cette réflexion a également été exposée par Libert et Vergnaud dans [LV08b] dans le cadre de leur schéma. Il est très facile de la généraliser pour les scénarios dont le challenge porte sur un chiffré non re-chiffrable. Nous nous intéressons à approfondir ce raisonnement.

L'idée est que s'il n'y a aucune restriction sur les clés de re-chiffrement données à l'adversaire pour une certaine liaison (η, η'), alors il n'est pas nécessaire de définir l'oracle de re-chiffrement sur cette même liaison. En effet, l'adversaire peut disposer de la clé de re-chiffrement concernée par ce re-chiffrement et donc effectuer la requête lui-même. Cette idée s'applique naturellement à tout challenge non re-chiffrable, mais pas uniquement. Considérons, par exemple, l'ensemble $\mathsf{Liaison} = \{(1, 2), (2, 3), (3, 4)\}$ et un challenge de type 3. Il n'y a aucune restriction naturelle sur les clés de re-chiffrement concernées par les liaisons $(1, 2)$ et $(2, 3)$. Nous pouvons ainsi appliquer le raisonnement que nous venons de voir et ne pas définir l'oracle de re-chiffrement sur ces deux liaisons - à moins qu'elles n'impliquent des restrictions non naturelles -, bien que le chiffré de type 3 soit re-chiffrable.

Généralisation : considérons un chiffré challenge de type η^* ainsi qu'une liaison (η, η'). Dans

le cas où il n'existe pas de chaîne de liaisons $\mathsf{Ch} = (\eta_0, \cdots, \eta_n) \in \mathsf{Chaîne}$ tel que $\eta_0 = \eta^*$ et $\eta_n = \eta$ pour $n \in \mathbb{N}$, alors nous pouvons ne pas définir l'oracle de déchiffrement sur la liaison (η, η') - à condition, une fois de plus, qu'il n'y ait pas de restriction non naturelle sur les clés de re-chiffrement concernées par cette liaison (η, η'). La raison évidente est que, s'il n'existe pas une telle chaîne de liaisons, alors il n'y a pas de restriction naturelle sur les clés de re-chiffrement concernées par cette liaison.

Note sur les sécurités secondaires

Jusqu'à présent, nous ne nous sommes intéressés qu'à la sécurité principale du serveur de re-chiffrement. Il existe cependant d'autres sécurités pour cette brique cryptographique que l'on peut considérer comme secondaires ou complémentaires. Elles sont intéressantes dans certains cas précis.

Protection de la clé maître. Dans une optique de protéger la clé privée d'un délégant, Ateniese *et al.* définirent dans leur article [AFGH06], la notion de *protection de la clé privée maître*, de l'anglais *master secret key security*. Le terme "maître" fait référence au délégant puisque c'est lui qui décide qui pourra obtenir l'information après re-chiffrement. L'idée derrière cette notion de sécurité est qu'aucune coalition, du serveur et de plusieurs délégués, ne puisse retrouver cette clé privée maître.

Si l'on ne s'intéresse pas aux différentes restrictions dans la sécurité sur l'indiscernabilité de messages chiffrés, on s'aperçoit que celle-ci implique la protection de la clé maître. En effet, si un adversaire est capable de récupérer la clé privée du délégant, il est alors capable de déchiffrer tout chiffré à destination de celui-ci, et donc de casser l'indiscernabilité.

En particulier, pour tout schéma sûr dans un scénario portant sur un chiffré challenge non re-chiffrable et ne considérant pas de restrictions non naturelles sur les clés de re-chiffrement, cette notion de protection de la clé maître sera vérifiée. En effet, dans ce cas, comme le challenge est non re-chiffrable et comme il n'y a pas de restriction non naturelle, l'adversaire aura accès à toutes les clés de re-chiffrement, ce qui correspond bien au cadre de la protection de la clé maître.

Ce résultat fut énoncé par Libert et Vergnaud pour les schémas avec re-chiffrement unidirectionnel dans [LV08b]. Concernant le cas des schémas avec re-chiffrement bidirectionnel, cette notion ne sera que très rarement vérifiée et ne sera donc pas prise en compte. En effet, dans tous les schémas de la littérature, il est possible, à partir d'une clé de re-chiffrement entre un délégant et un délégué et de la clé privée de ce délégué, de retrouver la clé privée du délégant.

C'est pourquoi cette sécurité est mise au rang de sécurité secondaire et n'est donc pas étudiée dans ce mémoire, puisqu'elle est soit non vérifiée, soit immédiate d'après l'indiscernabilité.

Anonymat de la clé de re-chiffrement. Une deuxième sécurité, apportant cette fois des propriétés supplémentaires, fut introduite pour le serveur de re-chiffrement dans l'article [ABH09] par Ateniese *et al.*. Ils la nommèrent en anglais *key private*, ce que l'on traduira par *anonymat de la clé de re-chiffrement*. L'idée derrière cette notion est inspirée de la sécurité *key privacy* pour les schémas de chiffrement, que l'on peut traduire par anonymat de la clé et qui a été formalisée par Bellare *et al.* dans [BBDP01]. Le but de cet anonymat de la clé de re-chiffrement est que l'on ne puisse distinguer les entités concernées (délégant/délégué) par une clé de re-chiffrement. Cette notion se traduit par le fait que, du point de vue de l'adversaire, cette clé de re-chiffrement doit être indistinguable d'un aléa.

Dans le même article, les auteurs ont conçu le premier schéma atteignant cette sécurité. Finalement très peu de schémas y parviennent, puisque depuis la parution de cet article, seule la construction [SLZ12] semble y arriver. Dans la section suivante, nous nous intéressons à étudier son éventuel intérêt pour le stockage de données.

3.4 Sécurité pour le stockage de données

Nous essayons de voir dans cette section les influences que peuvent avoir les différentes sécurités d'un schéma de serveur de re-chiffrement sur un système de stockage de données tel que présenté dans le Chapitre 1.

3.4.1 Utilité de l'anonymat de la clé de re-chiffrement ?

Revenons tout d'abord sur cette sécurité d'anonymat de la clé de re-chiffrement. Nous avons vu qu'elle se concentrait uniquement sur les clés de re-chiffrement. Pour cette raison, elle n'aura pas d'impact sur la confidentialité des données. De plus, il semble difficile de pouvoir l'atteindre pour notre cas d'usage. En effet, les utilisateurs ne savent pas a priori quels vont être les serveurs qui vont stocker les informations chiffrées, car ils sont à la charge de l'entité qui propose ses services de stockage. Ces serveurs peuvent donc être potentiellement corrompus par un individu malveillant. Les utilisateurs vont pourtant y stocker leurs données. Afin que les serveurs puissent correctement effectuer les opérations de re-chiffrement, ils doivent savoir, lorsqu'une entité demande l'envoi de données, quelle clé de re-chiffrement utiliser pour que les données soient re-chiffrées pour la bonne entité. Ainsi si le serveur est piraté par un individu malveillant, celui-ci pourra savoir à quel couple de (délégant/délégué) correspond la clé de re-chiffrement dérobée. Cet individu sera capable d'outrepasser l'anonymat de la clé de re-chiffrement. Cette opération n'est pas forcément aisée si les serveurs sont suffisamment protégés, mais elle est indépendante du schéma de serveur de re-chiffrement et donc non maîtrisable par l'utilisateur. Pour cette raison, nous ne prenons pas en compte cette sécurité pour le stockage de données à base de serveur de re-chiffrement.

3.4.2 Indiscernabilité des messages chiffrés

Concernant la sécurité du serveur de re-chiffrement, au final seule la sécurité d'indiscernabilité des messages chiffrés nous intéresse, puisque celle-ci va, comme nous l'avons expliqué, impliquer la protection de la clé maître. Cette sécurité est primordiale, puisque c'est elle qui assure la confidentialité des données, afin que l'adversaire, mais également le serveur, ne puisse apprendre d'information sur les messages en clair. Nous avons vu également qu'il y avait plusieurs types d'attaques possibles sur cette sécurité impliquant différents moyens mis à disposition de l'adversaire. Ceux-ci vont impacter des différences au niveau de la sécurité du système de stockage de données.

La sécurité IND-CPA est le minimum requis pour le stockage de données. En effet, le serveur de stockage a à sa disposition toutes les données chiffrées ainsi que les clés de re-chiffrement. Ainsi, si un adversaire arrive à récupérer les données que stocke le serveur, il ne faut pas qu'il puisse en obtenir d'information, ce qui correspond bien à l'attaque de type CPA.

L'attaque CCA et ses variantes possibles (par exemple RCCA), avec plus ou moins de restrictions, permettent d'assurer la confidentialité des données dans des situations particulières. En effet, un schéma résistant à une telle sécurité permet aux données stockées d'être protégées même si un adversaire arrive à prendre, de manière temporaire ou non, le contrôle de certains dispositifs des utilisateurs. Pendant cette période de corruption, l'adversaire peut alors se faire passer pour l'utilisateur et interagir avec le serveur de re-chiffrement et les autres utilisateurs, et ainsi re-chiffrer des messages en utilisant le serveur de stockage mais également les déchiffrer en utilisant le dispositif corrompu. Ceci correspond aux pouvoirs donnés à l'adversaire par les oracles de re-chiffrement et de déchiffrement. Ces situations, bien que plus rares, puisque l'adversaire doit arriver à corrompre les dispositifs, n'en restent pas moins possibles. Dans ce cas, la résistance aux attaques de type CCA est un plus pour l'application de stockage et lui offre une très grande sécurité.

Pour autant, il est utile de se demander si ces sécurités sont vraiment nécessaires pour toutes les applications possibles de stockage de données. Par exemple, si l'on considère un stockage

portant sur des données fixes, comme des photos de vacances, les données seront disponibles en permanence dans le serveur. L'adversaire qui arriverait à corrompre le dispositif d'un utilisateur pourrait récupérer les données qu'il souhaite puisqu'elles seraient disponibles au moment de la corruption. Dans ce cas, la sécurité IND-CCA n'est donc pas requise. En revanche, elle peut s'avérer intéressante si les données stockées sont amenées à évoluer fréquemment ou si elles ne sont stockées dans le serveur que durant un certain laps de temps. Le choix de la sécurité à adopter dépend donc des fonctionnalités désirées pour le stockage.

Il faut également tenir compte des différentes restrictions considérées. En effet, nous avons vu qu'il pouvait y avoir des restrictions non naturelles à prendre en compte pour la sécurité d'un schéma. Ainsi, si le schéma de serveur de re-chiffrement est sûr pour un certain scénario, les données ne seront sécurisées que dans ce scénario avec ces propres restrictions non naturelles. Examinons par exemple la sécurité IND-RCCA. Nous nous plaçons dans le contexte particulier où les données stockées ne peuvent avoir que deux valeurs - typiquement un "oui" et un "non" - et sont amenées à évoluer fréquemment, afin de justifier la considération de la partie CCA. Dans ce cas, dès que l'adversaire arrivera à corrompre temporairement le dispositif d'un utilisateur, il pourra effectuer une requête de déchiffrement sur ces deux données possibles, ce qui est interdit par la sécurité restrictive IND-RCCA, et donc, les données risquent de ne pas être protégées. La solution serait alors d'utiliser un sytème de chiffrement hybride (voir Chapitre 5 pour plus de détails), bien que les données à stocker soient de petites tailles, afin que chaque donnée soit chiffrée pour une clé de session aléatoire.

Au travers de cet exemple, nous voyons que chaque application, avec ses types de données à stocker et leurs utilisations, nécessitera d'étudier quelle sera la sécurité requise. Si l'on souhaite être le plus général possible, sans avoir à rechercher la sécurité la plus adaptée, il faut alors considérer des schémas ayant une sécurité sans aucune restriction non naturelle, ce qui limite le choix.

Chapitre 4

Modèles classiques de serveur de re-chiffrement

Sommaire

4.1 Revue des schémas existants et idées de construction 49
4.1.1 Re-chiffrement infini et bidirectionnel 50
4.1.2 Re-chiffrement fini et bidirectionnel 51
4.1.3 Re-chiffrement unique et unidirectionnel 52
4.1.4 Re-chiffrement multiple de cardinal n supérieur à 1 et unidirectionnel . . 56
4.1.5 Re-chiffrement infini et unidirectionnel 58
4.2 Relations entre les différents types 60
4.2.1 Le re-chiffrement unidirectionnel implique le re-chiffrement bidirectionnel 60
4.2.2 Le re-chiffrement bidirectionnel implique le re-chiffrement unidirectionnel 61
4.2.3 Le re-chiffrement infini implique le re-chiffrement multiple de cardinal m, qui implique lui-même le re-chiffrement multiple de cardinal n avec m supérieur à n 63
4.2.4 Le re-chiffrement multiple de cardinal n implique le re-chiffrement multiple de cardinal m avec m supérieur à n, qui implique lui même le re-chiffrement infini 63
4.2.5 Conclusion sur les différentes relations 64
***Récapitulation des différents schémas* 64**

Dans ce chapitre, nous présentons les modèles dits classiques de schémas de serveur de re-chiffrement et exposons les idées de leurs constructions. Nous détaillons également les schémas que nous utiliserons dans la suite de ce mémoire, à savoir les schémas [LV08b] et [CWYD10]. Nous étudions ensuite les relations entre les différents types de schémas. Pour finir, nous récapitulons l'ensemble de ces schémas ainsi que nos contributions.

4.1 Revue des schémas existants et idées de construction

Nous avons présenté dans le chapitre précédent deux fonctionnalités particulières composées chacune de différentes catégories : re-chiffrement *infini*, *multiple de cardinal* > 1 ou *unique*, et confiance de type *bidirectionnelle* ou *unidirectionnelle*. De par leurs descriptions respectives, ces deux fonctionnalités sont, au moins d'un point de vue théorique, compatibles entre elles. Il est alors possible de concevoir différents types de re-chiffrement, que l'on peut appeler modèles classiques (ou récurrents), considérant une catégorie de chaque fonctionnalité, comme par exemple un re-chiffrement infini et bidirectionnel. Chaque schéma de re-chiffrement existant correspond à un modèle classique, ce qui signifie qu'on ne rencontre pas, par exemple, de schéma offrant un

re-chiffrement bidirectionnel et un re-chiffrement unidirectionnel (nous en concevrons au Chapitre 7). Nous détaillons ici chacun de ces modèles classiques et examinons les schémas correspondants (exception faite des schémas avec re-chiffrement conditionnel que nous réservons pour le Chapitre 8).

4.1.1 Re-chiffrement infini et bidirectionnel

La fonctionnalité d'un tel schéma est de mélanger la notion de confiance mutuelle entre un délégant et un délégué avec la notion de délégation successive infinie.

Tous les schémas de ce type sont basés sur un même principe. Ils considèrent tous le même environnement. L'ensemble Type est un singleton, que l'on note $\{1\}$, l'ensemble Liaison est également un singleton égal à $\{(1,1)\}$. De cette manière, dans tous les algorithmes, il n'est pas nécessaire de préciser les types de chiffrés considérés, étant donné qu'il n'y en a qu'un seul. Quant aux structures de données, on a $\mathsf{D}_{(1,1),de} = \mathsf{D}_{(1,1),vers} = \{\texttt{clé privée}\}$. Ces schémas ont également en commun le type de calcul effectué par l'algorithme Délégation pour obtenir la clé de re-chiffrement. Ce calcul correspond à une loi de composition interne de groupe sur une des deux clés privées et sur le symétrique de la deuxième. Typiquement, selon la notation de la loi de groupe considérée, multiplicative ou additive, la clé de re-chiffrement est de la forme $sk_{\mathtt{B}} - sk_{\mathtt{A}}$ ou $sk_{\mathtt{B}}/sk_{\mathtt{A}}$ où $sk_{\mathtt{A}}$ et $sk_{\mathtt{B}}$ sont respectivement les clés privées du délégant `A` et du délégué `B`. Il est alors facile de voir que ces schémas offrent bien un re-chiffrement bidirectionnel grâce au calcul d'un symétrique pour la loi de groupe. Par exemple, $sk_{\mathtt{A}} - s_{\mathtt{B}}$ s'obtient en calculant $-(sk_{\mathtt{B}} - sk_{\mathtt{A}})$.

Ce type de serveur de re-chiffrement est le premier à avoir vu le jour, puisque c'est celui utilisé dans l'article [BBS98]. Le schéma, alors proposé, repose sur le schéma de chiffrement ElGamal, aulequel on ajoute un algorithme de rechiffrement. L'idée sous-jacente est la base de tous les autres schémas correspondants à ce modèle. Comme nous l'avons vu au Chapitre 2, un chiffré de la variante d'ElGamal d'un message m pour `A` est de la forme $(g^{r\cdot sk_{\mathtt{A}}}, m \cdot g^r)$, calculé à partir de la clé publique $g^{sk_{\mathtt{A}}}$. Le re-chiffrement consiste à calculer $g^{r\cdot sk_{\mathtt{B}}} = (g^{r\cdot sk_{\mathtt{A}}})^{sk_{\mathtt{B}}/sk_{\mathtt{A}}}$, ce qui permet d'obtenir un chiffré de m pour `B` de la forme $(g^{r\cdot sk_{\mathtt{B}}}, m \cdot g^r)$. Ivan et Dodis [ID03] ont proposé un schéma similaire avec comme clé de re-chiffrement, l'élément $sk_{\mathtt{B}} - sk_{\mathtt{A}}$, toujours sur la base d'un chiffré ElGamal, mais dans sa version originale uniquement. Le re-chiffrement de ce schéma consiste à modifier un chiffré de la forme $(g^r, g^{r.sk_{\mathtt{A}}}.m)$ en un chiffré de la forme $(g^r, g^{r\cdot sk_{\mathtt{B}}} \cdot m)$ via l'égalité $g^{r\cdot sk_{\mathtt{B}}} \cdot m = (g^r)^{sk_{\mathtt{B}} - sk_{\mathtt{A}}} \cdot g^{r\cdot sk_{\mathtt{A}}} \cdot m$.

Ces deux constructions de serveur de re-chiffrement sont le fondement de nombreux schémas et pas seulement ceux de ce modèle comme nous le verrons avec le schéma [CWYD10]. Elles reposent en grande partie sur le chiffrement ElGamal et sont, pour les mêmes raisons, prouvées sûres dans un scénario seulement IND-CPA$_{\mathrm{B}}$. Il n'est pas possible d'utiliser la construction de Noar et Yung [NY90] pour rendre ces schémas IND-CCA$_{\mathrm{B}}$. En effet, la preuve de connaissance utilisée dans cette construction pour prouver que deux chiffrés portent bien sur le même message ne serait plus valable après re-chiffrement.

C'est pourquoi, Canetti et Hohenberger ont conçu un schéma [CH07] atteignant une sécurité IND-CCA$_{\mathrm{B}}$. La base du schéma est celle de [BBS98]. Ainsi un chiffré est toujours composé d'une variante d'un chiffré ElGamal et une clé de re-chiffrement est égale aux quotients de deux clés privées. La différence avec cet article est que l'environnement, dans lequel vivent les clés publiques et les messages en clair, dispose de couplages bilinéaires. Pour obtenir une sécurité CCA$_{\mathrm{B}}$, ils ont notamment utilisé la construction CHK décrite au Chapitre 2. De cette manière, Canetti et Hohenberger ont obtenu le premier schéma de serveur de re-chiffrement bidirectionnel, infini et sûr contre une attaque CCA$_{\mathrm{B}}$. Sa sécurité repose sur l'hypothèse DBDH.

Matsuda *et al.* ont proposé, dans [MNT10], une approche différente pour concevoir un tel schéma. La base de leur chiffrement est l'utilisation de fonctions à trappe avec perte d'informations [1], de l'anglais *lossy trapdoor functions* (LTDF), afin de proposer un schéma prouvé

1. Voir [PW08] pour plus de détails.

IND-CCA$_{\mathrm{B}}$ et sans utiliser de couplages bilinéaires, comme c'est le cas du schéma de Canetti-Hohenberger. Pour cela, ils ont conçu un nouveau type de fonctions LTDF adaptées au re-chiffrement, des fonctions LTDF réapplicables. Leur spécificité est de disposer d'un algorithme qui peut transformer l'évaluation d'une fonction LTDF sur des données d'une entité `A` en une évaluation de la même fonction mais sur des données d'une entité `B`, ce qui revient à l'idée du re-chiffrement mais appliqué aux fonctions LTDF. Ces auteurs ont notamment proposé une réalisation de telles fonctions réapplicables en se basant sur des chiffrés ElGamal, où la transformation d'une évaluation pour `A` en une évaluation pour `B` reprend l'idée du re-chiffrement de Ivan et Dodis, et utilisent donc des éléments de transformation de la forme $sk_{\mathtt{B}} - sk_{\mathtt{A}}$. Les auteurs ont annoncé leur schéma comme IND-CCA$_{\mathrm{B}}$. Leurs chiffrés ne disposent pas de vérifiabilité publique. Ce schéma ne peut donc atteindre cette sécurité comme nous l'avons mentionné au Chapitre 3. Zeng *et al.* ont notamment expliqué les erreurs de la preuve de sécurité dans [WZH11] et exhibé une attaque contre ce schéma.

Association naturelle. Nous venons de voir, dans tous ces schémas, que toutes les clés de re-chiffrement liaient "directement" la clé privée du délégué et le symétrique de la clé privée du délégant, ou inversement, pour différentes lois de composition interne. Cette liaison directe se fait donc entre deux objets de même nature. Elle rend ainsi possible, dans le re-chiffrement, le remplacement dans un chiffré de la partie qui correspond à la clé privée de `A` avec celle de `B`. Typiquement, le terme $sk_{\mathtt{B}}/sk_{\mathtt{A}}$ permet de remplacer $g^{sk_{\mathtt{A}}}$ par $g^{sk_{\mathtt{B}}}$.

Il n'est alors pas étonnant d'obtenir par ce principe un re-chiffrement infini. En effet, en remplaçant, par exemple, $g^{sk_{\mathtt{A}}}$ en $g^{sk_{\mathtt{B}}}$, on ne change pas la forme de ce terme, mais juste l'entité concernée. En conséquence, quand on va re-chiffrer un chiffré, on ne va pas modifier sa forme. En utilisant de telles clés de re-chiffrement bidirectionnelles et ce principe de "changement d'identité", on obtient un re-chiffrement qui va fournir une liaison (η, η) dans l'ensemble $\mathsf{Liaison}$, ce qui implique un re-chiffrement infini. En revanche, ces clés de re-chiffrement impliquent toutes les restrictions non naturelles récurrentes pour un re-chiffrement bidirectionnel et que nous avons vues à la Section 3.3.3.

Calcul d'une clé de re-chiffrement $sk_{\mathtt{B}}/sk_{\mathtt{A}}$ avec $sk_{\mathtt{A}}, sk_{\mathtt{B}} \in \mathbb{Z}_p^*$.

Le moyen le plus simple pour Alice et Bob de calculer cette clé est que l'un des deux révèle temporairement sa clé privée à l'autre qui pourra alors calculer la clé de re-chiffrement bidirectionnelle. Cette situation peut être acceptable puisqu'une telle clé de re-chiffrement implique une confiance mutuelle entre les deux protagonistes. Il est cependant possible d'éviter cette situation en faisant intervenir une entité tierce de confiance, typiquement le serveur qui s'occupera du stockage, en suivant le protocole décrit dans la Figure 4.1. De cette manière, seule l'entité tierce obtient la clé de re-chiffrement. Bien évidemment, pour ne pas que les calculs puissent être erronés durant le protocole, il faut que chaque échange de données se fasse de manière chiffrée. Une méthode similaire peut être utilisée pour calculer une clé de re-chiffrement $sk_{\mathtt{B}} - sk_{\mathtt{A}}$.

4.1.2 Re-chiffrement fini et bidirectionnel

Nous réunissons dans la catégorie de re-chiffrement fini, les schémas offrant un re-chiffrement multiple de cardinal strictement supérieur à 1 ou un re-chiffrement unique. Ce type de re-chiffrement est très peu répandu dans la littérature. En effet, le seul qui correspond à cette catégorie est celui présenté dans l'article [DWLC08] et qui offre un re-chiffrement unique. Ce schéma atteint la sécurité IND-CCA$_{\mathrm{B}}$ sous l'hypothèse CDH. L'ensemble Type de ce schéma est $\{1, 2\}$ et son ensemble $\mathsf{Liaison}$ est $\{(1, 2)\}$. Il considère $\mathsf{D}_{(1,2),de} = \mathsf{D}_{(1,2),vers} = \{\texttt{clé privée}\}$.

La clé de re-chiffrement de ce schéma est toujours de la forme $sk_{\mathtt{B}}/sk_{\mathtt{A}}$ où $sk_{\mathtt{A}}$ et $sk_{\mathtt{B}}$ sont respectivement la clé privée du délégant `A` et la clé privée du délégué `B`. Quant au chiffrement, il est basé sur le chiffrement "Hash ElGamal" présenté au Chapitre 2. D'après la forme de la

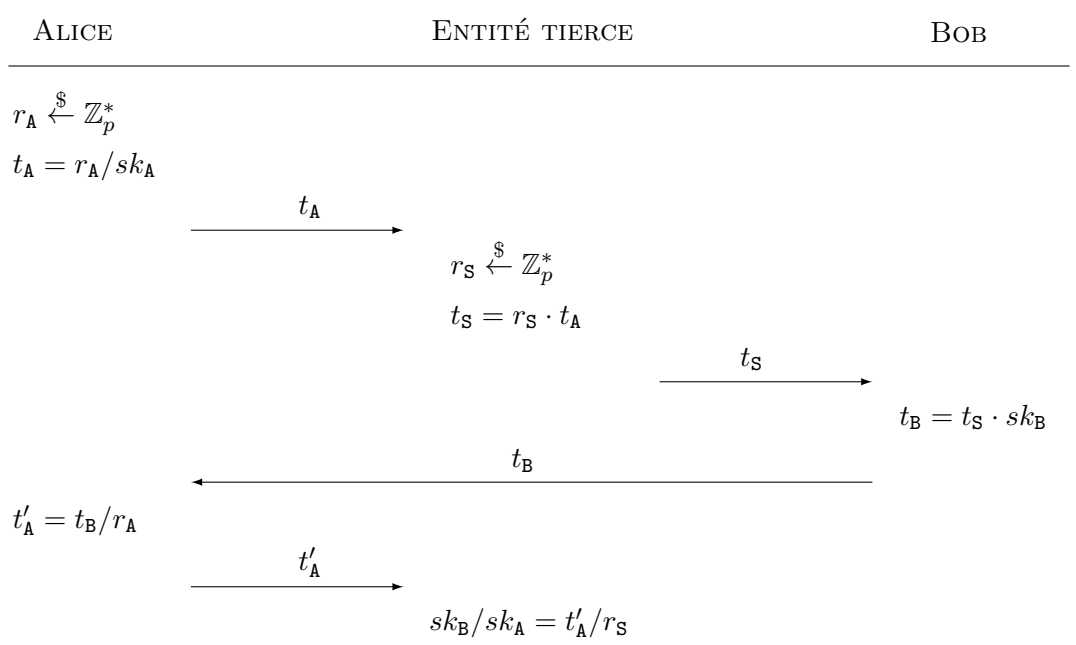

Figure 4.1 – Protocole pour calculer une clé de re-chiffrement bidirectionnelle

clé de re-chiffrement, on pourrait penser au premier abord que ce schéma est un schéma avec re-chiffrement infini, selon les raisons invoquées dans la section précédente. Cependant, contrairement au schéma [CH07], la vérifiabilité publique d'un chiffré est assurée ici par une preuve de connaissance "à la Schnorr" [Sch89]. Certains termes composant cette preuve sont liés à l'entité destinataire du chiffré et ne peuvent être modifiés. En détail et avec leurs notations, cette preuve repose sur une signature (D, s) où $D = pk_{\mathtt{A}}^u$ et $s = u + r \cdot \mathcal{H}_3(D, E, F)$. Avec la clé de re-chiffrement $sk_{\mathtt{B}}/sk_{\mathtt{A}}$, on pourrait alors modifier D en $D' = pk_{\mathtt{B}}^u$. Pour obtenir une signature valide, il faudrait alors pouvoir calculer $s' = u + r \cdot \mathcal{H}_3(D', E, F)$ à partir de s, D, E et F sans connaître r et u, ce qui est impossible. C'est pourquoi, en re-chiffrant, on modifie le type du chiffré. En particulier, le re-chiffré obtenu ne dispose plus de vérifiabilité publique et on a ainsi un re-chiffrement unique.

Nous nous sommes cependant aperçus dans [CDL11] que la sécurité n'était pas celle annoncée, et avons modifié le schéma afin qu'il atteigne bien une sécurité $\mathrm{CCA}_{\mathrm{B}}$ sans aucune autre restriction non naturelle supplémentaire. Les détails de cette étude sont donnés dans le Chapitre 6.

4.1.3 Re-chiffrement unique et unidirectionnel

Ce type de re-chiffrement est le plus courant dans la littérature. Pour cette raison, nous n'étudions pas en détail tous les schémas correspondant et nous nous contentons d'expliquer les différents principes et fonctionnements possibles du re-chiffrement. Ce re-chiffrement se focalise sur une délégation des droits d'accès choisie uniquement par le délégant - un délégué ne peut re-déléguer ses droits d'accès - et sur une confiance à sens unique entre le délégant et ses délégués. Ces schémas possèdent donc des ensembles du type : $\mathsf{Type} = \{1, 2\}$ et $\mathsf{Liaison} = \{(1, 2)\}$.

Pour obtenir une délégation à sens unique du délégant vers le délégué, le moyen le plus simple est d'utiliser comme ensembles de données $\mathsf{D}_{(1,2),de} = \{\texttt{clé privée}\}$ et $\mathsf{D}_{(1,2),vers} = \{\texttt{clé publique}\}$. On voit bien qu'en prenant de telles structures, on obtiendra une clé de re-chiffrement unidirectionnelle puisque l'ensemble $\mathsf{D}_{(1,2),vers}$ n'est pas composé d'éléments privés du délégué. C'est le cas pour tous les schémas de la littérature. Il faut cependant trouver un calcul nécessitant ces deux éléments. On est alors amené à distinguer les différentes idées possibles pour effectuer un tel calcul en trois catégories.

Première catégorie. En général, les schémas se placent dans un groupe $\mathbb{G}$ d'ordre p premier

de générateur g et considèrent comme clé privée sk un élément tiré aléatoirement dans $\mathbb{Z}_p^*$ et comme clé publique $pk = g^{sk}$. À partir de la clé privée $sk_{\mathtt{A}}$ du délégant et de la clé publique $pk_{\mathtt{B}}$ du délégué, le délégant est alors capable de calculer $rk_{(\mathtt{A},1\to\mathtt{B},2)} = pk_{\mathtt{B}}^{1/sk_{\mathtt{A}}} = g^{sk_{\mathtt{B}}/sk_{\mathtt{A}}}$. Dans un tel groupe, l'unidirectionnalité de la clé est assurée sous l'hypothèse DHI. On voit cependant qu'elle ne permet pas de remplacer, contrairement au cas bidirectionnel, un élément de la forme $g^{sk_{\mathtt{A}}}$ en un élément de la forme $g^{sk_{\mathtt{B}}}$. La solution idéale, pour effectuer un re-chiffrement, est de se placer dans un groupe appartenant à un environnement bilinéaire. En effet, en utilisant le couplage bilinéaire, on est capable d'effectuer la transformation voulue puisque $e(g^{sk_{\mathtt{A}}}, rk_{(\mathtt{A},1\to\mathtt{B},2)}) = e(g,g)^{sk_{\mathtt{B}}}$, ce qui fournit ainsi un terme dépendant de $sk_{\mathtt{B}}$ à partir de $g^{sk_{\mathtt{A}}}$.

Plusieurs schémas utilisent ce principe de re-chiffrement en se basant sur un chiffré ElGamal. Ateniese *et al.*, dans [AFGH06], ont proposé une adaptation du chiffrement ElGamal en un schéma unidirectionnel, ils ont ainsi obtenu un schéma IND-CPA sous l'hypothèse q-DBDHI.

Libert et Vergnaud [LV08b] ont mis au point un schéma IND-RCCA sûr sous l'hypothèse 3-QDBDH, en utilisant notamment la construction CHK. Leur re-chiffrement est certes plus complexe que de juste utiliser le couplage bilinéaire comme nous venons de le décrire, et ce afin de garantir cette sécurité RCCA. Ce schéma est détaillé dans la Figure 4.2, où nous avons légèrement modifié leurs notations afin qu'elles correspondent aux schémas que nous présenterons par la suite et où la signature à usage unique (voir Chapitre 2) et les algorithmes la composant sont respectivement notés OTS et $\mathsf{Clé}$, Signer et $\mathsf{Vérifier}$ et les éléments u et v utilisés sont des paramètres publics. Nous avons considéré de plus, dans cette figure, une modification sur le chiffré produit par l'algorithme $\mathsf{Chiffrer}_2$. Le schéma original considère, avec les notations que nous avons utilisées, que $C_{2,\mathtt{B}}$, $D_{\mathtt{B}}$ et $E_{\mathtt{B}}$ sont respectivement de la forme $pk_{\mathtt{B}}^{r\tilde{t}}$, $pk_{\mathtt{B}}^{\tilde{t}}$ et $g^{1/\tilde{t}}$. Nous les avons remplacés par $g^{r\tilde{t}}$, $g^{\tilde{t}}$ et $pk_{\mathtt{B}}^{1/\tilde{t}}$. Cette modification permet de constater facilement, en prenant $\tilde{t} = sk_{\mathtt{A}} \cdot t$, que l'algorithme $\mathsf{Chiffrer}_2$ et l'algorithme $\mathsf{Re\text{-}chiffrer}_{(1,2)}$ produisent bien un chiffré de même type. Elle ne modifie en rien le schéma, puisque les auteurs ont évoqué, dans leur article, qu'un chiffré de type 2 pouvait être rafraîchi publiquement et obtenir la modification que nous considérons. En outre, ce rafraîchissement possible d'un chiffré de type 2 est la raison de la restriction RCCA.

Weng *et al.* ont repris ces idées afin de proposer un schéma dans [WCY+10] atteignant la sécurité IND-CCA sans restrictions non naturelles toujours sous l'hypothèse 3-QDBDH. En observant les preuves de sécurité données dans ce papier, on s'aperçoit vite que cette sécurité n'est pas atteinte. En effet, l'adversaire, qui aurait obtenu une clé de re-chiffrement de l'entité cible du challenge vers une entité honnête, ne peut en obtenir une de cette dernière vers une entité corrompue. Le re-chiffrement étant unique, cette clé de re-chiffrement peut être donnée à l'adversaire dans la sécurité IND-CCA, ce qui est n'est pas le cas ici.

Shao, Cao et Liu ont utilisé des idées similaires, en considérant en plus un schéma de chiffrement à clé secrète IND-CCA, afin de proposer un schéma [SCL10] atteignant la sécurité IND-CCA sans restrictions non naturelles et également sous l'hypothèse 3-QDBDH. Ce schéma a été amélioré par la suite par Wang, Wei et Zhang dans [WWZ10].

Deuxième catégorie. Dans la catégorie précédente, nous avons considéré comme clé de re-chiffrement l'élément $pk_{\mathtt{B}}^{1/sk_{\mathtt{A}}}$. Il y a cependant un autre lien possible entre la clé privée du délégant et la clé publique du délégué. Ce lien consiste à considérer comme clé de re-chiffrement $pk_{\mathtt{B}}^{sk_{\mathtt{A}}} = g^{sk_{\mathtt{B}}.sk_{\mathtt{A}}}$.

Cependant, on voit bien que cette clé est bidirectionnelle. Pour pallier ce problème de bidirectionnalité, l'idée est alors de considérer, non plus une mais deux clés privées indépendantes par utilisateur, de façon à ce que chaque clé privée soit utilisée pour un type de chiffré différent. La clé privée pour un utilisateur $\mathtt{U}$ est donc de la forme $(sk_{\mathtt{U},1}, sk_{\mathtt{U},2})$ et sa clé publique $(pk_{\mathtt{U},1}, pk_{\mathtt{U},2}) = (g^{sk_{\mathtt{U},1}}, g^{sk_{\mathtt{U},2}})$. Ainsi pour re-chiffrer depuis le type 1 vers le type 2, on utilise la clé publique numéro 2 du délégué et la clé privée numéro 1 du délégant. De cette manière, la clé de re-chiffrement

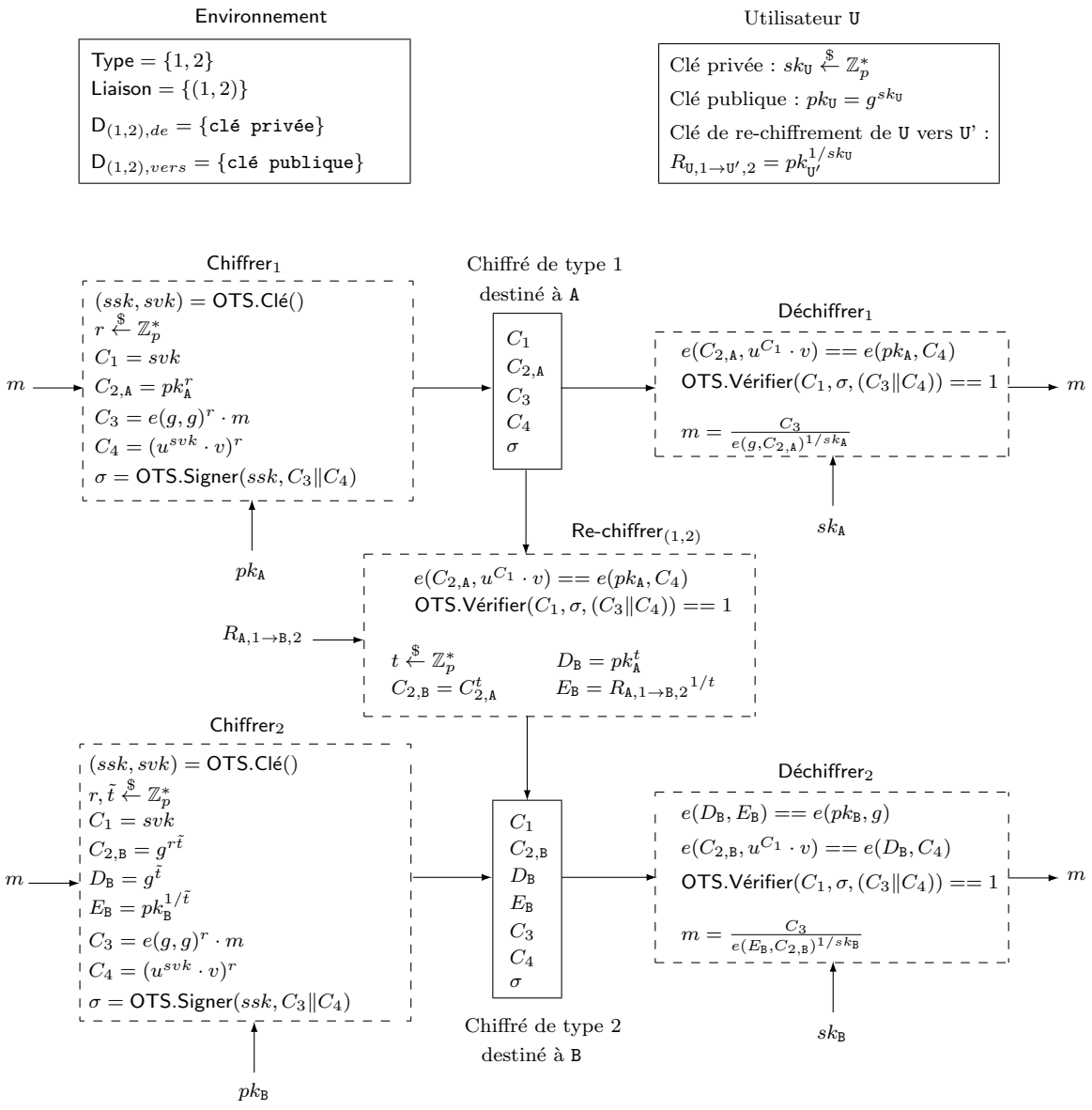

Figure 4.2 – Le schéma [LV08b]

$g^{sk_{\mathtt{B},2} \cdot sk_{\mathtt{A},1}}$ est bien unidirectionnelle puisqu'elle ne permet plus de calculer la clé de re-chiffrement $g^{sk_{\mathtt{A},2} \cdot sk_{\mathtt{B},1}}$.

Pour autant, cela n'empêche pas que B puisse calculer cette clé à la place de A, via $pk_{\mathtt{A},1}^{sk_{\mathtt{B},2}}$. On voit bien que le fait de considérer $pk_{\mathtt{U},1}$ et $pk_{\mathtt{U},2}$ comme des exponentiations sur une même base g pose problème. La solution est de les considérer sur des bases distinctes appartenant éventuellement à des groupes distincts. En prenant un deuxième élément h également générateur d'un groupe d'ordre p, telle que l'on ne dispose pas d'un morphisme de groupe facilement calculable du groupe $\langle h \rangle$ engendré par h, dans le groupe $\langle g \rangle$ engendré par g, on peut définir la clé publique de l'utilisateur U comme $(h^{sk_{\mathtt{U},1}}, g^{sk_{\mathtt{U},2}})$. Ceci permet d'obtenir comme clé de re-chiffrement de A vers B, l'élément $g^{sk_{\mathtt{B},2} \cdot sk_{\mathtt{A},1}}$ via $pk_{\mathtt{B},2}^{sk_{\mathtt{A},1}}$ sans que B ne puisse la calculer à partir de la clé publique $h^{sk_{\mathtt{A},1}}$ et de sa clé secrète $sk_{\mathtt{B},2}$, puisque par hypothèse sur $\langle h \rangle$ et $\langle g \rangle$, on ne peut obtenir $g^{sk_{\mathtt{A},1}}$ à partir de $h^{sk_{\mathtt{A},1}}$. On a ainsi une clé de re-chiffrement unidirectionnelle calculable seulement par le délégant. La difficulté repose donc sur le fait de trouver un tel élément h de sorte que l'on puisse utiliser la clé de re-chiffrement considérée pour passer d'un chiffré de A à un chiffré

de B. Pour cela, tous les schémas actuels utilisent un environnement bilinéaire où typiquement $h = e(g, g)$.

L'idée de ce re-chiffrement fut introduite par Ateniese *et al.* pour leur schéma [AFGH06] qui obtinrent ainsi un schéma IND-CPA. Elle fut ensuite réutilisée par Ateniese *et al.* [ABH09] pour parvenir à un schéma IND-CPA vérifiant en plus l'anonymat de la clé de re-chiffrement. La base du chiffrement de ces deux schémas est une fois de plus le chiffrement ElGamal. Quant au re-chiffrement, il fonctionne toujours sur un principe similaire à ce que l'on a déjà vu. En détail, à partir d'un chiffré ElGamal de la forme $(g^r, e(g, g^{r \cdot sk_{\mathtt{A},1}}) \cdot m)$, on est ainsi capable d'obtenir $h^{r' \cdot sk_{\mathtt{B},2}} = e(g^r, g^{sk_{\mathtt{B},2} \cdot sk_{\mathtt{A},1}})$ où $r' = r \cdot sk_{\mathtt{A},2}$ et donc un chiffré ElGamal d'une autre forme, à savoir $(h^{r' \cdot sk_{\mathtt{B},2}}, h^{r'} \cdot m)$.

Dans [SC09], Shao et Cao ont essayé une autre approche en considérant un h qui convienne sans utiliser de couplages bilinéaires. Ils se sont placés pour cela, dans l'ensemble $\mathbb{Z}^*_{N^2}$ pour trouver un tel élément h, où N est un module RSA décrit dans [RSA78]. Cependant, leur schéma, bien qu'annoncé sûr sous l'hypothèse DDH et sous le problème difficile de la factorisation d'un module RSA, a été successivement démontré comme n'étant pas IND-CCA par Chow *et al.* dans [CWYD10] puis comme n'étant même pas IND-CPA par Zhang, Cheng et Li dans [CZL11] et aucune piste ne laisse présager qu'il soit réparable.

Libert et Vergnaud ont également proposé un schéma de ce type dans [LV08a] sûr sous l'hypothèse Aug-DBDH, permettant en plus de détecter si un serveur de re-chiffrement dévoile sa clé de re-chiffrement ou non en cas d'utilisation de nombreux serveurs. Leur idée consiste en quelque sorte à considérer une clé de re-chiffrement de A vers B comme une clé maître d'un schéma de chiffrement basé sur l'identité avec joker dont nous reparlerons au Chapitre 8 et qui consiste à chiffrer un message pour une large sélection d'identités. Chaque serveur à qui A souhaite donner cette clé recevra une clé dérivé de cette dernière pour une identité ω quelconque choisie par le délégant. C'est un peu comme si chaque clé de re-chiffrement étant en fait une clé de re-chiffrement entre un dérivé de la clé du délégant pour l'identité ω et la clé du délégué. Afin que chaque serveur, possédant une telle clé de re-chiffrement, puisse modifier un chiffré correctement, ce dernier est chiffré à destination de toutes les clés possibles dérivées à partir de la clé du délégant, c'est-à-dire pour toutes les identités possibles. Dans le cas où une clé de re-chiffrement serait dévoilée par un serveur, le délégant peut alors détecter quel serveur l'a dévoilée. Il lui suffit alors de l'utiliser afin de re-chiffrer des messages que l'on chiffre non plus à destination de toutes les clés possibles pour toutes les identités mais pour un ensemble plus restreint. Si cette clé permet le re-chiffrement, c'est que l'identité qui lui est associée est incluse dans cet ensemble restreint. En effectuant plusieurs de ces tests, le délégant peut ainsi retrouver l'identité associée à la clé. Ce schéma est également prouvé IND-CPA sûr sous l'hypothèse 2-3-CDH.

Troisième catégorie. L'idée de ce troisième groupe consiste à utiliser un chiffrement à seuil, de l'anglais *threshold encryption* introduit par Desmedt dans [Des87]. On appelle schéma de chiffrement à seuil (n, t), un schéma qui consiste à séparer une clé privée en n morceaux, chacun de ces morceaux permettant de déchiffrer partiellement le chiffré, de sorte qu'au moins t déchiffrements partiels soient nécessaires et suffisants pour retrouver le message en clair. L'utilisation d'un chiffrement à seuil pour réaliser un schéma de serveur de re-chiffrement fut exposée par Ivan et Dodis dans [ID03]. Le principe repose sur la considération d'un chiffrement à seuil $(2, 2)$, chaque clé de déchiffrement étant alors séparée en deux morceaux. Le premier morceau est donné en clair au serveur via la clé de re-chiffrement, ce dernier pouvant alors déchiffrer partiellement le chiffré. Le deuxième morceau est également joint à la clé de re-chiffrement, mais chiffré pour la clé publique du délégué. Le re-chiffrement consiste alors à fournir au délégué, le chiffré original, son déchiffrement partiel et le deuxième morceau de la clé de déchiffrement, ce qui permet au délégué de récupérer le message en clair. Cette idée fut étudiée par Hanaoka *et al.* dans [HKK+12] qui présentèrent une construction générique permettant d'obtenir un schéma IND-CCA à partir d'un tel chiffrement à seuil.

Association naturelle. Le re-chiffrement unidirectionnel se comporte un peu comme l'opposé du re-chiffrement bidirectionnel. On voit, dans tous les cas, qu'une clé de re-chiffrement n'est pas simplement, comme dans le cas bidirectionnel, une opération d'une loi de composition interne d'un groupe. Ceci ne va donc pas permettre, lors du re-chiffrement, de remplacer tout simplement les termes correspondant au délégant par ceux du délégué sans en changer la forme. Pour cette raison, la forme du chiffré obtenu après re-chiffrement sera a priori différente de celle du chiffré originel. On obtient ainsi un re-chiffrement entre deux types différents. De plus, on voit dans tous les cas, que l'on ne peut pas re-chiffrer n'importe comment et obtenir comme re-chiffré, le chiffré de son choix. Ainsi, il semble difficile de pouvoir obtenir sur ce chiffré, à la fois une vérifiabilité publique et une structure qui permette de le re-chiffrer à nouveau. Ceci justifie le fait qu'un re-chiffrement unidirectionnel soit bien adapté pour n'offrir qu'un re-chiffrement unique.

4.1.4 Re-chiffrement multiple de cardinal n supérieur à 1 et unidirectionnel

Pour les raisons que l'on vient d'expliquer, ce type de schéma n'est pas très répandu dans la littérature. Certains schémas ont cependant vu le jour. Les techniques utilisées pour le re-chiffrement rejoignent les techniques déjà présentées dans la section précédente.

Wang et Cao exposèrent un schéma de cette catégorie type dans [WC09] sans pour autant se soucier de sa sécurité. Ce dernier possède les ensembles : $\mathsf{Type} = \{1, 2, \cdots, n+1\}$, $\mathsf{Liaison} = \{(1,2),(2,3),\cdots,(n,n+1)\}$ et, pour tout $i \in [\![1;n]\!]$, $\mathsf{D}_{(i,i+1),de} = \{\texttt{clé privée}\}$ et $\mathsf{D}_{(i,i+1),vers} = \{\texttt{clé publique}\}$. Par la suite, Zhang et Wang [ZW12a] reprirent leurs travaux et démontrèrent qu'il n'était en fait pas sécurisé.

Shao *et al.* proposèrent également un schéma avec re-chiffrement multiple > 1 dans [SLCW11]. Il est cependant possible d'attaquer leur schéma. On remarque tout d'abord que la sécurité d'un re-chiffré doit prendre en compte certaines restrictions non naturelles étant donné qu'un re-chiffré est composé notamment du chiffré originel. Il est également possible de concevoir une attaque sur le chiffré originel prouvant que le schéma n'est pas sécurisé. En effet et avec leur notations, si l'on souhaite attaquer un chiffré non re-chiffré $C_{pk^*} = (A_1, B_1, C_1, D_1, E_1, S_1)$ pour pk^*, il suffit de demander un re-chiffrement de celui-ci pour une entité pk honnête non corrompue et on obtient ainsi un re-chiffré $C_{pk} = (R_1, C_{pk^*}, S_1, Q_1, A'_1, B'_1, C_t, C'_t)$ où C_t et C'_t sont deux chiffrés destinés à pk masquant respectivement deux valeurs $\bar{k}_{1,2}$ et $\widehat{k}_{1,2}$. Le re-chiffrement de ce chiffré C_{pk} consiste à re-chiffrer de manière indépendante C_t et C'_t. Les auteurs ont pris cependant une précaution, puisque si l'on modifie $(R_1, C_{pk^*}, S_1, Q_1, A'_1, B'_1, C_t)$, alors la signature $\tilde{S_2}$ contenue dans C'_t ne sera plus valide. Cependant, aucun mécanisme n'est inclus dans C_t pour empêcher de modifier C'_t et il n'est pas possible d'en mettre car C_t est constant et lié à la clé de re-chiffrement alors que C'_t varie et est créé au moment du re-chiffrement. En considérant donc n'importe quel chiffré C''_t à la place de C'_t, on pourra obtenir le re-chiffrement de C_t pour une entité corrompue, ce qui est permis puisque le chiffré $(R_1, C_{pk^*}, S_1, Q_1, A'_1, B'_1, C_t, C''_t)$ n'est pas le chiffré challenge ou un de ses re-chiffrés. Ce re-chiffrement de C_t nous permettra alors d'obtenir l'élément $\bar{k}_{1,2}$ qui combiné à la clé de re-chiffrement de pk^* vers pk permet d'obtenir h_1 et r_1, éléments utilisés dans toutes les clés de re-chiffrement calculées par pk^*. En regardant l'algorithme $\mathsf{Déchiffrer}$ sur le chiffré originel, on voit alors qu'en prenant $r' = r_1$ et $h' = h_1$, on dispose alors d'un moyen de déchiffrer le chiffré C_{pk^*} et ainsi obtenir le message challenge.

Nous pouvons ainsi voir qu'il n'existe actuellement aucun schéma de ce type atteignant une sécurité IND-CCA, voire même IND-RCCA.

Nouvelle direction de recherche pour atteindre une sécurité de type IND-CCA. Il semble pourtant possible de réaliser de tels schémas atteignant une sécurité de type IND-CCA en utilisant des outils mathématiques plus avancés, tels que les applications multi-linéaires, dont des propositions de réalisations ont été exposées dans [GGH13] dans lequel le problème Diffie-

Hellman multi-linéaire décisionnel [2] est difficile. Supposons que nous disposions d'une application $(n+1)$-linéaire symétrique d'un groupe $\mathbb{G}$ d'ordre p premier dans un groupe $\mathbb{G}_T$, pour $n \geq 2$, définie de la manière suivante :

$$e : \underbrace{\mathbb{G} \times \mathbb{G} \times \cdots \times \mathbb{G}}_{n+1 \text{ fois}} \to \mathbb{G}_T.$$

La propriété de $(n+1)$-linéarité signifie que, pour tout $(g_1, g_2, \cdots, g_{n+1}) \in \mathbb{G}^{n,+1}$, pour tout $(a_1, a_2, \cdots, a_{n+1}) \in \mathbb{Z}_p^{n+1}$, l'égalité suivante doit être vérifiée :

$$e(g_1^{a_1}, g_2^{a_2}, \cdots, g_{n+1}^{a_{n+1}}) = e(g, \cdots, g)^{a_1 \cdot a_2 \cdots a_{n+1}}.$$

En utilisant une telle fonction, nous pouvons envisager la construction de schémas offrant un re-chiffrement multiple de cardinal n et unidirectionnel. En effet, nous pouvons, par exemple, modifier le schéma [LV08b] afin qu'il puisse offrir un tel re-chiffrement. Pour simplifier la présentation du schéma, nous notons LV ce schéma, composé des différents algorithmes $\{\mathsf{Clé}, \mathsf{Délégation}, \mathsf{Chiffrer}, \mathsf{Re\text{-}chiffrer}, \mathsf{Déchiffrer}\}$, où nous ignorons, une fois de plus, les paramètres publics. Notre but est de concevoir un re-chiffrement undirectionnel et multiple de cardinal n, ce qui nous donne les deux ensembles $\mathsf{Type} = \{1, 2, \cdots, n+1\}$ et $\mathsf{Liaison} = \{(1,2), (2,3), \cdots, (n, n+1)\}$. Nous choisissons 1 et 2 comme les deux types de chiffrés offerts par le schéma LV et tel que $(1,2)$ soit la liaison de celui-ci. Ainsi, en conservant le même algorithme $\mathsf{Clé}$, nous pouvons reprendre comme base du chiffrement et du re-chiffrement ceux du schémas LV. C'est-à-dire que $\mathsf{Chiffrer}_1$, $\mathsf{Chiffrer}_2$, $\mathsf{Délégation}_{(1,2)}$, $\mathsf{Re\text{-}chiffrer}_{(1,2)}$, $\mathsf{Déchiffrer}_1$ et $\mathsf{Déchiffrer}_2$ sont définis comme dans le schéma LV en remplaçant le couplage bilinéaire par cette application $(n+1)$-linéaire. Afin de rendre notre idée plus compréhensible, nous modifions les notations utilisées dans la Figure 4.2 et remplaçons les éléments générés aléatoirement r et t par respectivement r_0 et r_1. Plus précisément, un chiffré de type 2, re-chiffré de $\mathtt{A}$ vers $\mathtt{B}$, est alors de la forme $(C_1, C_{2,\mathtt{B}}, D_\mathtt{B}, E_\mathtt{B}, C_3, C_4, \sigma)$ où $C_1 = svk$, $C_{2,\mathtt{B}} = C_{2,\mathtt{A}}^{r_1}$, $D_\mathtt{B} = pk_\mathtt{A}^{r_1}$, $E_\mathtt{B} = R_{\mathtt{A},1\to B,2}^{1/r_1}$, $C_3 = e(g, \cdots, g)^{r_0} \cdot m$, $C_4 = (u^{svk} \cdot v)^{r_0}$ et $\sigma = \mathsf{OTS.Signer}(ssk, C_3 \| C_4)$ s'il est re-chiffré de $\mathtt{A}$ vers $\mathtt{B}$, avec $C_{2,\mathtt{A}} = pk_\mathtt{A}^{r_0}$. Nous allons maintenant ajouter les re-chiffrements successifs à ce schéma.

Intéressons-nous en premier lieu à la liaison $(2,3)$. Nous définissons également l'élément $R_{\mathtt{B},1\to\mathtt{C},2} = pk_\mathtt{C}^{1/sk_\mathtt{B}}$, la clé de re-chiffrement de $\mathtt{B}$ pour le type 1 vers $\mathtt{C}$ pour le type 2, comme clé de re-chiffrement de $\mathtt{B}$ pour le type 2 vers $\mathtt{C}$ pour le type 3, notée $R_{\mathtt{B},2\to\mathtt{C},3}$. A partir de cette clé de re-chiffrement et du chiffré précédent, nous sommes capables, grâce à l'utilisation de l'application multi-linéaire, d'obtenir $e(g, \cdots, g)^{r_0 \cdot sk_\mathtt{C}} = e(C_{2,\mathtt{B}}, E_\mathtt{B}, R_{\mathtt{B},2\to\mathtt{C},3}, g, \cdots, g)$. Ainsi, avec ces différents éléments, l'utilisateur $\mathtt{C}$ est en mesure de récupérer, avec sa clé privée $sk_\mathtt{C}$, l'élément $e(g, \cdots, g)$ et donc de déchiffrer. Nous pouvons définir le re-chiffrement sur la liaison $(2,3)$ comme l'algorithme consistant à ajouter, au chiffré de type 2 précédemment exposé, l'élément $R_{\mathtt{B},2\to\mathtt{C},3}$.

Pour autant, cette idée n'est pas suffisante si l'on souhaite obtenir un schéma sûr. En effet, nous voyons bien dans ce chiffré, que la clé de re-chiffrement de $\mathtt{B}$ vers $\mathtt{C}$ n'est liée à aucune autre composante du chiffré. Cette particularité a déjà été prise en compte dans le re-chiffrement du schéma LV, qui pour éviter cela, transforme $C_{2,\mathtt{A}}$, $pk_\mathtt{A}$ et $R_{\mathtt{A},1\to\mathtt{B},2}$ en $C_{2,\mathtt{B}} = C_{2,\mathtt{A}}^{r_1}$, $D_\mathtt{B} = pk_\mathtt{A}^{r_1}$ et $E_\mathtt{B} = R_{\mathtt{A},1\to\mathtt{B},2}^{1/r_1}$ avec $r_1 \xleftarrow{\$} \mathbb{Z}_p^*$. Dans le cas du schéma LV avec un couplage bilinéaire, ces transformations permettent de lier la clé de re-chiffrement $R_{\mathtt{A},1\to\mathtt{B},2}$ au reste du chiffré en testant les deux équations suivantes :

$$e(D_\mathtt{B}, E_\mathtt{B}) = e(pk_\mathtt{B}, g),$$
$$e(C_{2,\mathtt{B}}, u^{C_1} \cdot v) = e(D_\mathtt{B}, C_4).$$

2. Variante du problème DBDH où l'on demande de distinguer un aléa d'une exponentiation de non plus un produit de 3 éléments mais de $n+1$ éléménts si la fonction est n-linéaire.

En réutilisant ce principe sur les éléments $C_{2,\mathtt{B}}$, $D_{\mathtt{B}}$ et $R_{\mathtt{B},2\to\mathtt{C},3}$ pour re-chiffrer de $\mathtt{B}$ vers $\mathtt{C}$, nous obtenons $C_{2,\mathtt{C}} = C_{2,\mathtt{B}}^{r_2}$, $D_{\mathtt{C}} = D_{\mathtt{B}}^{r_2}$ et $E_{\mathtt{C}} = R_{\mathtt{B},2\to\mathtt{C},3}^{1/r_2}$, avec $r_2 \xleftarrow{\$} \mathbb{Z}_p^*$. L'application $(n+1)$-linéaire et l'élément $E_{\mathtt{B}}$ permettent alors de vérifier si les deux équations suivantes sont vérifiées :

$$e(D_{\mathtt{C}}, E_{\mathtt{B}}, E_{\mathtt{C}}, g, \cdots, g) = e(pk_{\mathtt{C}}, g, \cdots, g),$$
$$e(C_{2,\mathtt{C}}, u^{C_1} \cdot v, g, \cdots, g) = e(D_{\mathtt{C}}, C_4, g, \cdots, g).$$

De plus, $\mathtt{C}$ est toujours capable de récupérer le message en clair, puisque $e(g, \cdots, g)^{r_0 \cdot sk_{\mathtt{C}}}$ s'obtient toujours en calculant $e(C_{2,\mathtt{C}}, E_{\mathtt{B}}, E_{\mathtt{C}}, g, \cdots, g)$. Nous définissons finalement le chiffré de type 3 destiné à $\mathtt{C}$ comme étant $(C_1, C_{2,\mathtt{C}}, D_{\mathtt{C}}, E_{\mathtt{B}}, E_{\mathtt{C}}, C_4, \sigma)$.

En utilisant ce principe sur les liaisons successives $(3,4), \cdots, (n, n+1)$, nous parvenons à un re-chiffrement multiple de cardinal n et unidirectionnel. Il est aussi possible de définir l'algorithme Chiffrer sur les types $3, 4, \cdots, n+1$ en utilisant une entité temporaire décrite dans le Remarque 5 sur les différents types successifs. Par exemple, un chiffré de type 3 pour $\mathtt{C}$ peut être défini comme un re-chiffré d'une entité virtuelle $\mathtt{V}_2$ vers $\mathtt{C}$, dont le chiffré modifié peut lui-même être défini comme un re-chiffré d'une entité virtuelle $\mathtt{V1}$ vers $\mathtt{V}_2$ et ainsi de suite jusqu'à tomber sur un chiffré de type 1. Nous remarquons qu'avec ce principe de re-chiffrement, il est impossible de re-chiffrer $n+1$ avec ce schéma puisque $e(g, \cdots, g)^{r_0 \cdot sk_{\mathtt{A}_{n+2}}}$ s'obtiendrait en utilisant le $(n+2)$-upplet $(C_{2,\mathtt{A}_{n+2}}, E_{\mathtt{A}_2}, \cdots, E_{\mathtt{A}_{n+2}})$ comme entrée de l'application $(n+1)$-linéaire qui n'accepte que les $(n+1)$-upplets. Ce schéma en version allégée est résumé dans la Figure 4.3 où nous omettons la signature à usage unique, ainsi que l'algorithme Chiffrer sur les types $2, 3, \cdots, n$.

Pour prouver la sécurité de ce schéma, nous devons en plus supposer que la généralisation de l'hypothèse $3-$QDBDH dans des groupes $\mathbb{G}$ et $\mathbb{G}_T$ munis de l'application $(n+1)$-linéaire soit difficile. Sous cette hypothèse, il est alors possible de démontrer que le schéma est sûr dans un scénario IND-RCCA avec cependant quelques restrictions non naturelles supplémentaires. En effet, nous avons considéré, que la clé de re-chiffrement d'une entité $\mathtt{A}$ vers une entité $\mathtt{B}$ était toujours la même, indépendamment de la liaison considérée. Cette propriété implique qu'en considérant, par exemple, un chiffré challenge de type n destiné à une entité cible $\mathtt{A}^*$, alors on ne peut donner à l'adversaire la clé de re-chiffrement de $\mathtt{A}^*$ pour le type 1 vers une entité corrompue $\mathtt{A}_c$ pour le type 2, puisque cette clé de re-chiffrement est exactement celle de $\mathtt{A}^*$ pour le type n vers $\mathtt{A}_c$ pour le type $n+1$. Si l'on souhaite obtenir un schéma sans ces restrictions non naturelles et obtenir une sécurité RCCA, il faut alors considérer, pour chaque entité, une paire de clés (privée/publique) par type de chiffré et donc des clés de re-chiffrements différentes selon la liaison considérée.

4.1.5 Re-chiffrement infini et unidirectionnel

Comme nous l'avons vu dans le chapitre précédent à la section 3.2.1, un re-chiffrement infini est caractérisé par l'existence d'une chaîne infinie de liaisons. Cela sous-entend l'existence d'une chaîne finie de liaisons de cardinal n pour tout n entier naturel. Nous avons vu dans le type précédent, que nous ne disposions pas encore de schémas sûrs avec re-chiffrement unidirectionnel ayant une telle chaîne dès que $n > 1$. Il semble ainsi difficile de pouvoir concevoir un schéma avec re-chiffrement unidirectionnel et infini possédant une chaîne infinie de Liaison.

Dans son mémoire de thèse [Gen09], Gentry a cependant eu l'idée d'utiliser un chiffrement totalement homomorphe, de l'anglais *fully homomorphic encryption* (FHE). En effet, un tel schéma permet d'effectuer n'importe quelle opération sur les chiffrés. Selon ce principe, le serveur pourrait effectuer les opérations de déchiffrement puis de chiffrement sur les données chiffrées [3], et ainsi

3. Ces opérations sont similaires à celles effectuées dans la notion de *bootstrapping* dont parle Gentry dans sa construction.

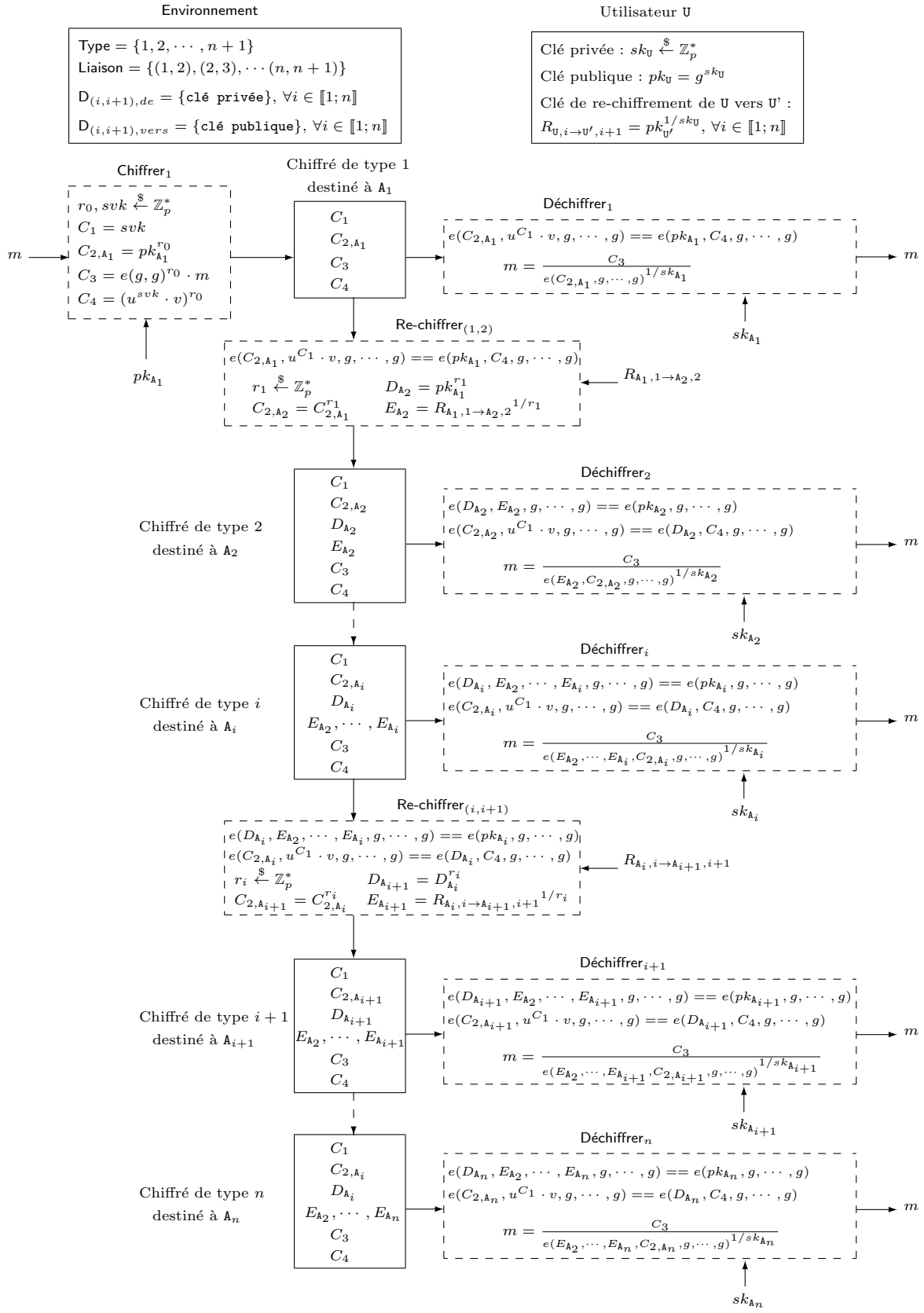

Figure 4.3 – Le schéma [LV08b] multiple de cardinal n allégé

transformer un chiffré pour A en un chiffré pour B, qui aurait la même forme que le chiffré originel et ce bien sûr sans avoir d'information sur le message en clair. Son idée va donc à l'encontre de tous les schémas unidirectionnels existants puisqu'elle permettrait de re-chiffrer sans modifier le type du chiffré, et ainsi d'obtenir un re-chiffrement infini. Il considère donc les ensembles $\mathsf{Type} = \{1\}$ et $\mathsf{Liaison} = \{(1,1)\}$.

En détail, son idée consiste à considérer un schéma de chiffrement totalement homomorphe FHE disposant des algorithmes (Initialiser, Chiffrer, Déchiffrer). Chaque utilisateur U génère sa clé privée $sk_{\mathtt{U}}$ et publique $pk_{\mathtt{U}}$ en utilisant l'algorithme FHE.Initialiser. La clé de re-chiffrement de A vers B est quant à elle, la réunion de la clé publique de B et du chiffré $\mathsf{FHE.Chiffrer}(pk_{\mathtt{B}}, sk_{\mathtt{A}})$ de la clé secrète de A pour B, notée $\overline{sk_{\mathtt{A}}}$. Ainsi $\mathsf{D}_{(1,1),de} = \{\texttt{clé privée}\}$ et $\mathsf{D}_{(1,1),vers} = \{\texttt{clé publique}\}$. Un chiffré pour A est calculé à partir de l'algorithme $C_{\mathtt{A}} = \mathsf{FHE.Chiffrer}(pk_{\mathtt{A}}, m)$. En utilisant ce chiffré et cette clé de re-chiffrement, le serveur est capable de produire le chiffré $\overline{C_{\mathtt{A}}} = \mathsf{FHE.Chiffrer}(pk_{\mathtt{B}}, C_{\mathtt{A}})$. Ensuite, en utilisant la possibilité de calculer sur des chiffrés, offerte par le schéma FHE, en particulier en faisant les calculs correspondants aux déchiffrement de $C_{\mathtt{A}}$ avec la clé privée $sk_{\mathtt{A}}$ mais sur les données chiffrées $\overline{C_{\mathtt{A}}}$ et $\overline{sk_{\mathtt{A}}}$, il peut obtenir $\overline{m} = \mathsf{FHE.Chiffrer}(pk_{\mathtt{B}}, m)$ qui correspond à un chiffré de la même forme que $C_{\mathtt{A}}$ mais à destination de B.

De cette manière, on obtient bien un re-chiffrement unidirectionnel qui ne modifie pas le type du chiffré, ce qui permet ainsi d'obtenir un re-chiffrement infini. Cependant, un schéma de chiffrement totalement homomorphe ne peut, de par ses fonctionnalités, atteindre une sécurité IND-CCA mais seulement IND-CPA. Pour cette raison, il sera difficile d'avoir de cette manière un serveur de re-chiffrement qui puisse atteindre la sécurité IND-CCA sans restrictions non naturelles. Il sera seulement possible d'obtenir une sécurité IND-CPA avec d'éventuelles restrictions non naturelles. L'efficacité d'un tel schéma dépend de celle du schéma de chiffrement totalement homomorphe qui, pour le moment, reste inexploitable dans des applications concrètes.

4.2 Relations entre les différents types

Nous avons examiné les différentes techniques permettant d'effectuer un re-chiffrement. Nous avons également vu que, dans la plupart des cas, certaines idées semblent récurrentes. Il est alors intéressant de se demander s'il est possible de concevoir un schéma d'un certain type à partir d'un autre. Pour cela, nous étudions maintenant les implications possibles entre ces différents types de façon non formelle que nous regroupons au sein de la Figure 4.4.

4.2.1 Le re-chiffrement unidirectionnel implique le re-chiffrement bidirectionnel

Cette implication est sûrement la plus simple à obtenir. Il suffit tout simplement de réunir, au sein d'une même clé de re-chiffrement, la clé de re-chiffrement unidirectionnelle de A vers B et celle de B vers A, ce qui donne bien une clé bidirectionnelle. Cette idée fut énoncée par Canetti et Hohenberger dans [CH07]. Le schéma obtenu possède les mêmes caractéristiques sur le nombre de re-chiffrements successifs possibles que le schéma avec re-chiffrement unidirectionnel. Il en est de même pour le type de sécurité et les restrictions non naturelles.

Comme nous l'avons évoqué à la Section 3.3.3, cette idée permet de concevoir des schémas avec re-chiffrement bidirectionnel sans aucune restriction non naturelle, notamment celles habituellement prises en compte par les re-chiffrements bidirectionnels.

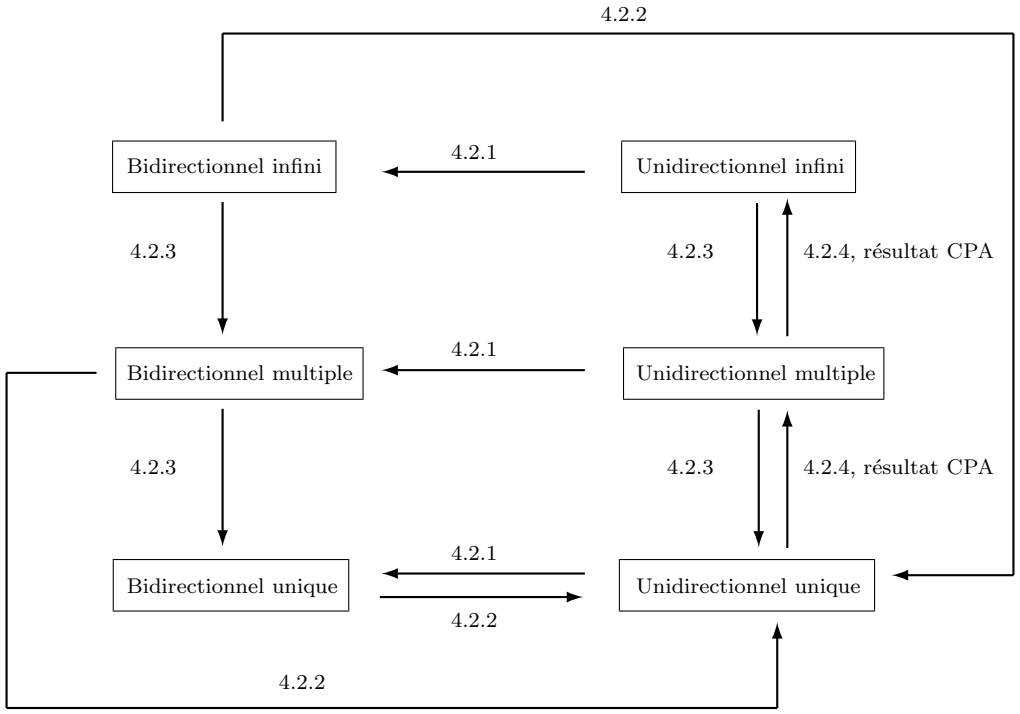

Figure 4.4 – Relations entre les différents types récurrents

4.2.2 Le re-chiffrement bidirectionnel implique le re-chiffrement unidirectionnel

Cette implication est plus compliquée que la précédente puisqu'il faut modifier le fonctionnement de la clé de re-chiffrement pour pouvoir supprimer le pouvoir de délégation du délégué vers le délégant inclus dans la clé de re-chiffrement bidirectionnelle. En particulier, il faut transformer la clé de re-chiffrement pour que son calcul ne nécessite plus d'élément privé du délégué.

Chow *et al.* se sont intéressés, dans l'article [CWYD10], à transformer le schéma [DWLC08], qui, comme nous l'avons vu à la Section 4.1.2, offre un re-chiffrement bidirectionnel et unique, afin d'en faire un schéma avec re-chiffrement unidirectionnel et unique. Ainsi, les environnements Type et Liaison de leur schéma sont inchangés par rapport au premier et sont toujours respectivement égaux à $\{1, 2\}$ et $\{(1, 2)\}$. Leur schéma est détaillé dans la Figure 4.5. Leur idée fut de transformer la clé de re-chiffrement de A vers B de la forme $sk_{\mathtt{B}}/sk_{\mathtt{A}}$, en une clé de re-chiffrement $sk_{\mathtt{V}}/sk_{\mathtt{A}}$ où $sk_{\mathtt{V}}$ est la clé privée d'une entité virtuelle choisie par A, et de transmettre cette clé privée $sk_{\mathtt{V}}$ à B en la chiffrant pour celui-ci et en l'incluant dans la clé de re-chiffrement. La clé de re-chiffrement de A vers B devient alors le couple $(sk_{\mathtt{V}}/sk_{\mathtt{A}}, \mathsf{Chiffrer}(pk_{\mathtt{B}}, sk_{\mathtt{V}}))$ où, en suivant les notations du schéma, Chiffrer est l'algorithme de chiffrement Hash-ElGamal, $sk_{\mathtt{A}} = sk_{\mathtt{A},1} \cdot \mathcal{H}_4(pk_{\mathtt{A},2}) + sk_{\mathtt{A},2}$ et $pk_{\mathtt{B}} = pk_{\mathtt{B},2}$. Il est évident que cette clé est bien unidirectionnelle puisqu'elle ne nécessite pas la clé privée $sk_{\mathtt{B}}$. En l'utilisant, le serveur peut transformer un chiffré pour A en un chiffré pour V et y ajouter le chiffré de $sk_{\mathtt{V}}$ pour B. Ainsi, B pourra obtenir $sk_{\mathtt{V}}$ en utilisant sa clé privée $sk_{\mathtt{B}}$, puis l'utiliser pour obtenir le message en clair. On voit bien sur la Figure 4.5 qu'un chiffré pour B est composé d'un chiffré Hash-ElGamal (E', F) de m pour V et d'un chiffré Hash-ElGamal (V, W) de $sk_{\mathtt{V}}$ pour B.

Pour autant, cette seule modification sur le re-chiffrement ne suffit pas à transformer le schéma [DWLC08] en un schéma unidirectionnel sûr contre une attaque CCA ou $\mathrm{CCA}_{\mathrm{B}}$ bien que ce dernier ait été annoncé par ses auteurs comme étant sûr contre une attaque $\mathrm{CCA}_{\mathrm{B}}$. Nous

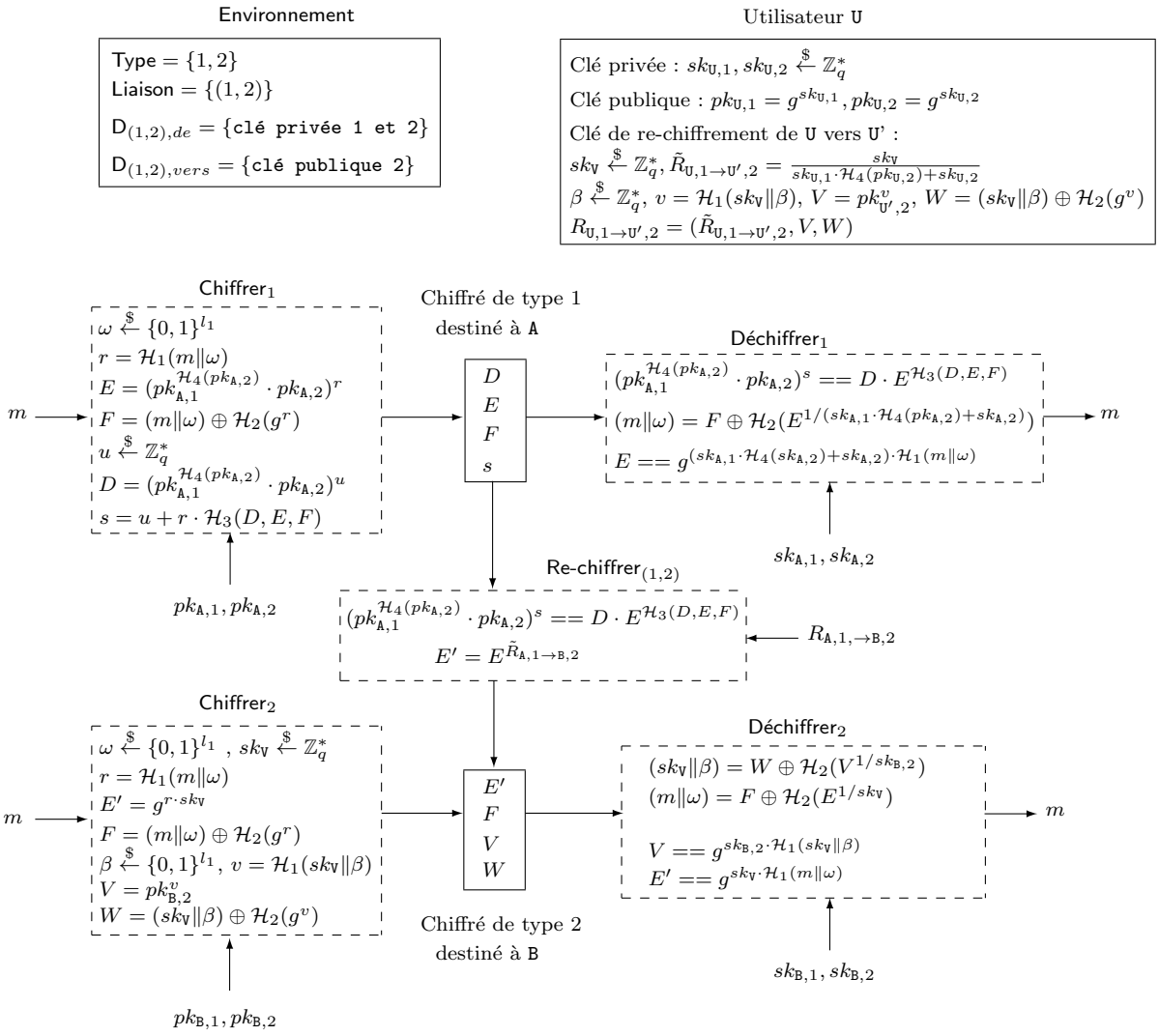

Figure 4.5 – Le schéma [CWYD10]

verrons en effet, au Chapitre 6, que le schéma obtenu impose des restrictions non naturelles, indépendantes de celles concernant le schéma [DWLC08]. Nous y présenterons une réparation adaptée à ce schéma pour le rendre sécurisé sans aucune restriction non naturelle. Cette réparation a également été publiée dans [CDL11].

Nous voyons bien que cette idée d'utiliser une entité virtuelle, telle que nous l'avons décrite, est indépendante du schéma [DWLC08] et n'utilise que le fait que la clé de re-chiffrement soit de la forme $sk_{\mathtt{B}}/sk_{\mathtt{A}}$. Ce raisonnement est donc valable pour d'autres schémas. Nous présenterons, dans le Chapitre 7, une construction générique qui généralisera le choix des différents éléments $(sk_{\mathtt{V}}/sk_{\mathtt{A}}, \mathsf{Chiffrer}(pk_{\mathtt{B}}, sk_{\mathtt{V}}))$ composant la clé de re-chiffrement unidirectionnelle évoquée ci-dessus et fonctionnera pour toute clé de re-chiffrement bidirectionnelle et pas seulement celles de la forme $sk_{\mathtt{B}}/sk_{\mathtt{A}}$. Nous y exposerons également des modifications supplémentaires à apporter à cette construction afin d'obtenir un schéma avec re-chiffrement unidirectionnel ne nécessitant en aucun cas de tenir compte de restrictions non naturelles. Il y a cependant une contrepartie à cette construction. Nous le voyons avec l'état de l'art, les schémas avec re-chiffrement bidirectionnel offrent régulièrement un re-chiffrement infini. En utilisant cette construction, nous perdons la possibilité de pouvoir re-chiffrer plusieurs fois de suite. En effet, nous modifions la structure du chiffré produit par le re-chiffrement bidirectionnel, puisque lors du re-chiffrement unidirectionnel

de A vers B, nous obtenons un chiffré pour V. Celui-ci ne peut donc pas être re-chiffré de B vers une autre entité. Ce re-chiffrement est donc unidirectionnel et unique.

4.2.3 Le re-chiffrement infini implique le re-chiffrement multiple de cardinal m, qui implique lui-même le re-chiffrement multiple de cardinal n avec m supérieur à n

Limiter le nombre de re-chiffrements possibles n'est pas si compliqué en soi. En effet, pour qu'un chiffré re-chiffrable ne le soit plus, il n'y a pas besoin de modifier le fonctionnement du re-chiffrement, il suffit juste d'empêcher qu'il soit possible. Ceci peut être réalisé en ajoutant, une fois le re-chiffrement effectué, une "couche" de chiffrement n'offrant pas de re-chiffrement à destination du délégué. Par exemple, supposons que l'on dispose d'un schéma de serveur de re-chiffrement sur des ensembles $\mathsf{Type} = \{1,2,3\}$ et $\mathsf{Liaison} = \{(1,2),(2,3)\}$ et que l'on souhaite le transformer en un schéma avec re-chiffrement unique sur des ensembles $\mathsf{Type} = \{1,2\}$ et $\mathsf{Liaison} = \{(1,2)\}$. Considérons deplus un schéma de chiffrement à clé publique PUB composé des algorithmes $\{\mathsf{Initialiser}, \mathsf{Clé}, \mathsf{Chiffrer}, \mathsf{Déchiffrer}\}$. Un utilisateur va alors posséder la clé privée $sk = (sk_{\mathsf{PRE}}, sk_{\mathsf{PUB}})$ et la clé publique $pk = (pk_{\mathsf{PRE}}, pk_{\mathsf{PUB}})$ où $(sk_{\mathsf{PRE}}, pk_{\mathsf{PRE}})$ est générée avec PRE.Initialiser et $(sk_{\mathsf{PRE}}, pk_{\mathsf{PUB}})$ est générée avec PUB.Chiffrer. Le chiffrement sur le type 1 correspond au chiffrement originel du PRE sur le type 1 ce qui produit un chiffré destiné à un utilisateur A égal à $C_{pk_{\mathtt{A}},1} = \mathsf{PRE.Chiffrer}_1(pk_{\mathsf{PRE},\mathtt{A}}, m)$. Quant au re-chiffrement du type 1 au type 2 de cet utilisateur vers un utilisateur B, il suffit alors de considérer l'algorithme effectuant l'opération :

$$C_{pk_{\mathtt{B}},2} = \mathsf{PUB.Chiffrer}(pk_{\mathsf{PUB},\mathtt{B}}, \mathsf{PRE.Re\text{-}chiffrer}_{(1,2)}(R_{\mathtt{A},1\to\mathtt{B},2}, C_{pk_{\mathtt{A}}})).$$

Le chiffré $C_{pk_{\mathtt{B}},2}$ étant un chiffré de l'algorithme PUB, il ne peut être re-chiffré à nouveau. On a donc bien limité le nombre de re-chiffrement possible. Il faut cependant ajouter à la clé de re-chiffrement $R_{\mathtt{A},1\to\mathtt{B},2}$ la clé publique $pk_{\mathsf{PUB},\mathtt{B}}$ pour que le serveur puisse ajouter cette couche de chiffrement. Dans le cas d'un re-chiffrement bidirectionnel, il faut y aussi ajouter la clé publique du délégant, à savoir $pk_{\mathsf{PUB},\mathtt{A}}$ pour des raisons de symétrie.

Il est facile de constater que la sécurité de cette construction repose sur la sécurité du schéma de serveur de re-chiffrement pour les chiffrés re-chiffrables et sur celle de ce chiffrement supplémentaire pour les chiffrés non re-chiffrables. Ainsi, si l'on souhaite un schéma atteignant la sécurité IND-CPA ou IND-CCA, il faut que le serveur de re-chiffrement et que ce chiffrement supplémentaire l'atteignent. Les restrictions non naturelles à considérer sont les mêmes que pour le serveur de re-chiffrement. Il peut cependant y en avoir des autres, puisque, si le chiffrement supplémentaire prend par exemple en compte la restriction RCCA, alors la sécurité du schéma de serveur de re-chiffrement la prendra également en compte.

4.2.4 Le re-chiffrement multiple de cardinal n implique le re-chiffrement multiple de cardinal m avec m supérieur à n, qui implique lui même le re-chiffrement infini

Cette implication ne semble pas être réalisable, puisque si un chiffré n'est pas re-chiffrable, comment serait-il possible, sans connaître le fonctionnement du re-chiffrement, de le transformer en un re-chiffrable de manière générique ? En effet, elle ne consiste plus, comme nous l'avons fait dans les implications précédentes, à réunir ou retirer des fonctionnalités offertes par le schéma considéré, mais à en ajouter de nouvelles.

Il y a cependant un moyen de concevoir une solution peu efficace. Plus précisément, il est possible de regrouper plusieurs schémas de re-chiffrement pour en obtenir un seul qui offre plus de re-chiffrements successifs. Cette idée a été exposée pour la première fois par Green et Ateniese

dans [GA07] qui ont construit un schéma de serveur de re-chiffrement basé sur l'identité [4] avec re-chiffrement infini et unidirectionnel à partir d'un schéma avec re-chiffrement unidirectionnel et unique. Nous la généralisons et l'appliquons à n'importe quel schéma de serveur de re-chiffrement.

Considérons pour commencer, le cas de schémas avec re-chiffrement unidirectionnel - le cas bidirectionnel se déduira à partir de celui-ci. L'idée est, si ce n'est pas déjà le cas, de transformer la clé de re-chiffrement en deux parties en reprenant l'idée décrite à la Section 4.2.2 pour passer du cas bidirectionnel au cas unidirectionnel. Le délégant `A` choisit une entité virtuelle `V` et crée une première partie composée d'une clé de re-chiffrement unidirectionnelle pour ce schéma de `A` vers `V`. Il joint à cette clé, un chiffré de la clé privée de `V` pour le délégué `B`. Nous savons que de cette manière, nous obtenons un re-chiffrement unique. Afin d'ajouter une étape de re-chiffrement, il suffit de considérer que ce chiffré de la clé privée de `V` pour `B` est cette fois-ci un chiffré re-chiffrable d'un schéma de serveur de re-chiffrement unidirectionnel. De cette manière, `B` pourra déléguer à une entité `C`, le fait de pouvoir obtenir la clé privée de `V`, qui lui permettra de récupérer le message. En appliquant cette idée de découpage de la clé de re-chiffrement en deux parties, il est possible de réunir les re-chiffrements de différents schémas. Si l'on considère n schémas de serveurs de re-chiffrement avec $n > 1$, il est ainsi possible d'obtenir un schéma avec un re-chiffrement multiple de cardinal n.

Cette description ne fournit que des schémas unidirectionnels même si à la base les n schémas considérés étaient bidirectionnels. Pour obtenir un re-chiffrement bidirectionnel, il faut alors appliquer la transformation de la Section 4.2.1.

On remarque que le schéma - bidirectionnel ou unidirectionnel - obtenu devient vite inefficace, puisqu'à chaque re-chiffrement, on ajoute au chiffré au moins une composante. De plus, il est facile de voir que chacune de ses nouvelles composantes est indépendante des autres. En effet, cette construction ne fournit pas de moyen pour vérifier publiquement qu'un re-chiffré est bien formé. On ne pourra donc pas, d'après la Section 3.3.4, obtenir simplement un schéma sûr contre des attaques, comme IND-CCA et IND-RCCA, où l'adversaire a accès à un oracle de re-chiffrement et de déchiffrement. La seule sécurité qu'un tel schéma pourra atteindre sera donc IND-CPA avec éventuellement quelques restrictions non naturelles sur les différentes clés données à l'adversaire.

4.2.5 Conclusion sur les différentes relations

En examinant ces différentes constructions, on voit qu'il est possible, dans la plupart des cas, de concevoir un schéma correspondant à un type précis à partir d'un quelconque autre, puisqu'au final, seul le fait d'augmenter le nombre de re-chiffrements semble être problématique. L'intérêt de ces constructions est avant tout théorique, puisque les transformations ne sont pas adaptées à un schéma spécifique et ne comporte donc pas d'optimisations. Pour autant, il est possible qu'elles donnent des résultats intéressants, comme le schéma [CWYD10] adapté de [DWLC08], que nous réparons au Chapitre 6. Cette réparation est notamment la seule réalisation d'un schéma atteignant la sécurité IND-CCA sans avoir recours au couplage bilinéaire.

Récapitulation des différents schémas

Nous regroupons au sein du Tableau 4.1, par ordre chronologique, les différents schémas existant dans la littérature, ainsi que les contributions de ce mémoire. Pour chacun, nous présentons les types de chiffrés et les liaisons de leur environnement. Nous précisons également les caractéristiques liées au nombre de re-chiffrements et au type de confiance qu'ils offrent, ainsi que leur sécurité. Nous ne présentons que les schémas principaux et les constructions générales. Il est donc possible, à partir des implications de la section précédente, d'en obtenir d'autres. Nous y mentionnons également nos propres contributions. En particulier, nous y rapportons des schémas

4. Nous nous intéresserons à cette primitive en fin de mémoire (page 129).

Schéma	Nbre de re-chiffrements	Confiance	Type	Liaison	Sécurité
[BBS98]	∞	Bi	$\{1\}$	$\{(1,1)\}$	$\mathrm{CPA_B}$
[ID03]	∞	Bi	$\{1\}$	$\{(1,1)\}$	$\mathrm{CPA_B}$
[AFGH06]	1	Uni	$\{1,2\}$	$\{(1,2)\}$	CPA
[CH07]	∞	Bi	$\{1\}$	$\{(1,1)\}$	$\mathrm{CCA_B}$
[LV08b]	1	Uni	$\{1,2\}$	$\{(1,2)\}$	RCCA
[DWLC08]	∞	Bi	$\{1,2\}$	$\{(1,2)\}$	$\mathrm{CCA_{B,REST}}$
[ABH09]	1	Uni	$\{1,2\}$	$\{(1,2)\}$	CPA
[SC09]	1	Uni	$\{1,2\}$	$\{(1,2)\}$	Cassé
[WC09]	n	Uni	$\{1,2,\cdots, n+1\}$	$\{(1,2),(2,3), \cdots,(n,n+1)\}$	Cassé
[Gen09]	∞	Uni	$\{1\}$	$\{(1,1)\}$	$\mathrm{CPA_{REST}}$
[CWYD10]	1	Uni	$\{1,2\}$	$\{(1,2)\}$	$\mathrm{CCA_{REST}}$
[WCY+10]	1	Uni	$\{1,2\}$	$\{(1,2)\}$	$\mathrm{CCA_{REST}}$
[SCL10]	1	Uni	$\{1,2\}$	$\{(1,2)\}$	CCA
[WWZ10]	1	Uni	$\{1,2\}$	$\{(1,2)\}$	CCA
[MNT10]	∞	Bi	$\{1\}$	$\{(1,1)\}$	Cassé
[SLCW11]	n	Uni	$\{1,2,\}$ $n+1\}$	$\{(1,2),(2,3),\}$ $\cdots,(n,n+1)\}$	Cassé
[HKK+12]	1	Uni	$\{1,2\}$	$\{(1,2)\}$	CCA
Nos propres contributions					
[CDL11] - [DWLC08] + Chap 6	∞	Bi	$\{1\}$	$\{(1,1)\}$	$\mathrm{CCA_B}$
[CDL11] - [CWYD10] + Chap 6	1	Uni	$\{1,2\}$	$\{(1,2)\}$	CCA
[LV08b] + Chap 5	1	Uni	$\{1,2\}$	$\{(1,2)\}$	CCA
BI/INF avec sécurité $\mathrm{CCA_B}$ + Chap 7	1	Uni	$\{1,2\}$	$\{(1,2)\}$	CCA
BI/INF avec sécurité $\mathrm{CCA_B}$ + Chap 7	∞	Bi et Uni	$\{1,2\}$	$\{(1,1),(1,2),(2,2)\}$	$\mathrm{RCCA_B}$
[LV08b] + Chap 7	∞	Bi et Uni	$\{1,2\}$	$\{(1,1),(1,2),(2,2)\}$	$\mathrm{RCCA_B}$
Section 4.1.4 - [LV08b] + App $(n+1)$-linéaire	n	Uni	$\{1,2,\cdots, \cdots,n+1\}$	$\{(1,2),(2,3), ,(n,n+1)\}$	RCCA
Section 4.1.4 - [LV08b] + App $(n+1)$-linéaire + Chap 7	∞	Bi et Uni	$\{1,2,\cdots, \cdots,n+1\}$	$\{(1,2),(2,3), ,(n,n+1)\}$ $\cup\{(i,i)\}_{i\in[\![1;n+1]\!]}$	$\mathrm{RCCA_B}$

Table 4.1 – Les différents schémas PRE possibles

d'un nouveau modèle que nous nommerons *combiné*. Ils offrent deux types de re-chiffrements, à savoir un unidirectionnel et un unique, avec la particularité de pouvoir être utilisés successivement dans une certaine mesure. Nous verrons au Chapitre 7 une définition précise de ces schémas et l'utilisation que l'on peut en faire.

Nous utilisons dans ce tableau les notations suivantes :
- $\mathrm{CPA_{REST}}$, la sécurité CPA avec des restrictions non naturelles autres que celles prises en compte par la sécurité $\mathrm{CPA_B}$;
- $\mathrm{CCA_{REST}}$, la sécurité CCA avec des restrictions non naturelles autres que celles prises en compte par les sécurités $\mathrm{CCA_B}$ et RCCA ;
- $\mathrm{CCA_{B,REST}}$, la sécurité $\mathrm{CCA_B}$ avec des restrictions non naturelles supplémentaires autres que celles prises en compte par la sécurité RCCA ;
- BI/INF, un schéma avec re-chiffrement bidirectionnel infini ;
- A + B, la réunion des idées/techniques de A et de B.

Troisième partie

Gestion de la sécurité

Chapitre 5

Amélioration de la sécurité RCCA

Sommaire

5.1 Préliminaires 69
5.1.1 La sécurité RCCA 69
5.1.2 Rafraîchissement public d'un chiffré 70
5.2 Amélioration de la sécurité RCCA pour le chiffrement à clé publique 71
5.2.1 Utilisation d'un chiffrement hybride 71
5.2.2 La construction Canetti-Krawczyk-Nielsen 71
5.3 Amélioration de la sécurité RCCA pour les schémas offrant un re-chiffrement unique et unidirectionnel 72
5.3.1 Une construction générique 72
5.3.2 Sécurité du schéma 75
5.3.3 Un exemple de construction plus pratique sur le schéma Libert-Vergnaud 77
5.4 À propos des autres types 79
5.4.1 Le type bidirectionnel et unique 79
5.4.2 Les schémas avec re-chiffrement multiple strictement supérieur à 1 ou infini 79

Nous nous intéressons dans ce chapitre à la restriction RCCA, évoquée brièvement au Chapitre 3, et qui consiste à restreindre certains des moyens mis à disposition de l'adversaire dans un scénario IND-CCA, ce qui donne la sécurité IND-RCCA. Nous y étudions plus particulièrement la relation entre RCCA et CCA, dans le cas de schéma de serveur de re-chiffrement, afin de proposer une construction générique pour transformer certains types de schéma RCCA en des schémas CCA. Nous montrons également qu'il est possible d'améliorer cette construction pour certains schémas afin que le résultat soit plus efficace. Ces résultats sont en cours de soumission.

5.1 Préliminaires

5.1.1 La sécurité RCCA

De manière générale et en ne se limitant pas seulement au contexte de ce mémoire et du serveur de re-chiffrement, il existe des cas d'usage où la sécurité IND-CCA peut apparaître comme difficile à atteindre. Selon le service visé, il est également possible que cette sécurité soit trop forte. Canetti *et al.* exposent notamment dans [CKN03] un ensemble de services où c'est le cas, comme par exemple l'authentification par mot de passe chiffré.

Ils y ont défini dans [CKN03] une attaque plus restrictive pour l'adversaire qui découlera sur une sécurité plus faible mais qui sera suffisante pour l'ensemble de ces services. Ils ont appelé cette attaque : attaque à messages chiffrés choisis rejouables - de l'anglais *replayable chosen ciphertext attacks* (RCCA). Le scénario correspondant est donc noté IND-RCCA. Dans celui-ci,

l'adversaire n'a plus le droit d'effectuer, par rapport au scénario d'IND-CCA, des requêtes de déchiffrement portant sur l'un des deux messages m_0 et m_1 qu'il a choisi lors de la phase Challenge (voir Section 2.4.2). Les auteurs de ce papier ont montré que cette sécurité était suffisante pour différents cas d'usage, comme par exemple l'authentification par mot de passe chiffré. Cette sécurité se situe en quelque sorte entre la sécurité CCA qui est la plus forte et la sécurité CPA qui est le minimum requis. Il paraît évident qu'atteindre une sécurité plus forte implique des calculs plus importants. Dans le cas où la sécurité CCA impliquerait trop de coûteux et ne serait pas nécessaire, cette sécurité peut représenter un bon compromis puisqu'elle résiste tout de même à certaines attaques actives.

La sécurité RCCA a été définie pour les schémas de chiffrement à clé publique. Elle a également un sens pour les schémas de serveur de re-chiffrement puisque, dans le cas d'une attaque CCA, l'adversaire a également accès à un oracle de déchiffrement. Elle intervient par exemple pour le schéma [LV08b]. Elle a d'abord été utilisée pour désigner la sécurité du schéma [CH07], bien qu'elle servait à expliquer les restrictions naturelles propres au serveur de re-chiffrement. Nous considérons que ce schéma n'est pas concerné par cette restriction et qu'il atteint donc la sécurité IND-CCA$_\mathrm{B}$. Par conséquent, le premier schéma de serveur de re-chiffrement évoquant de telles restrictions est le schéma [LV08b] cité précédemment. Elle est aussi prise en compte pour d'autres modèles de serveur de re-chiffrement dont nous parlerons dans la note en fin de mémoire (page 129), comme le schéma [CWC+09] [1]. Nous présenterons également, au cours de ce mémoire, d'autres constructions atteignant cette sécurité.

Bien que nous parlions de sécurité RCCA, il en existe en fait plusieurs qui dépendent des autres restrictions non naturelles prises en compte dans la sécurité du schéma. Le terme RCCA fait seulement référence à la restriction non naturelle évoquée ci-dessus. Ainsi nous pouvons avoir une sécurité où aucune autre restriction non naturelle n'est prise en compte et que nous appellerons sécurité RCCA. Il est possible d'en obtenir d'autres, comme par exemple la sécurité RCCA$_\mathrm{B}$ qui consiste à réunir, dans une même sécurité, les restrictions non naturelles des sécurités RCCA et CCA$_\mathrm{B}$. Nous préciserons alors, si besoin est, comme nous l'avons fait à la Section 4.2.5, quelle sécurité sera prise en compte.

5.1.2 Rafraîchissement public d'un chiffré

Plusieurs raisons connues, qui sont détaillées dans l'article [CKN03], peuvent expliquer la restriction RCCA. Nous détaillons ici une de ces causes récurrentes, à savoir le rafraîchissement d'un chiffré, sur laquelle nous serons vigilants dans nos constructions.

Rafraîchir un chiffré signifie le transformer en un nouveau chiffré qui correspond toujours au même message pour la même clé publique. Cette modification peut être faite de manière publique ou seulement par certaines entités. Par exemple, pour un serveur de re-chiffrement muni d'une liaison $(1, 2)$, le serveur est capable, s'il dispose d'une clé de re-chiffrement d'un utilisateur $\mathtt{A}$ pour le type 1 vers ce même utilisateur pour le type 2, de transformer un chiffré pour $\mathtt{A}$ en un chiffré toujours pour $\mathtt{A}$ mais différent. Ce rafraîchissement peut donc juste être effectué par un tiers identifié, le serveur - et éventuellement $\mathtt{A}$ s'il a gardé la clé de re-chiffrement. Comme exemple de rafraîchissement public, nous pouvons citer le chiffrement Elgamal qui permet un tel rafraîchissement à condition de disposer de la clé publique du destinataire. Si nous ne disposons que du chiffré, comme c'est le cas du schéma [LV08b], nous pouvons imaginer un chiffré, dans un environnement bilinéaire symétrique, de la forme $C = (C_1, C_2, C_3) = (m.e(g, g)^{rt}, pk_\mathtt{A}^r, pk_\mathtt{A}^t)$ où $pk_\mathtt{A} = g^{sk_\mathtt{A}}$ et qui se déchiffre en calculant $m = C_1/e(C_2, C_3)^{1/sk_\mathtt{A}^2}$. Il peut alors être modifié publiquement en tirant aléatoirement $z \in \mathbb{Z}_p^*$ et en calculant $C' = (C_1, C_2^z, C_3^{1/z})$ qui se déchiffrera de la même manière que C. Nous nous focalisons maintenant sur le rafraîchissement public d'un chiffré.

1. Ce schéma est un schéma de serveur de re-chiffrement dont le chiffrement des données n'est plus basé sur un chiffrement à clé publique mais sur un chiffrement de groupe.

Dans le cas où l'on conçoit un schéma qui devrait atteindre une sécurité CCA, l'existence de cette propriété de rafraîchissement public l'en empêchera systématiquement. En effet, si le challenge est rafraîchissable publiquement, il suffit à l'adversaire de demander le déchiffrement du challenge rafraîchi. La restriction RCCA permet par définition d'empêcher une telle attaque.

5.2 Amélioration de la sécurité RCCA pour le chiffrement à clé publique

Il est évident que, par définition, la sécurité RCCA est impliquée par la sécurité CCA. La réciproque est évidemment fausse. Dans cette section, nous nous intéressons, comme dans [CKN03], à la transformation de schémas IND-RCCA sûrs afin qu'ils ne prennent plus en considération cette restriction RCCA.

5.2.1 Utilisation d'un chiffrement hybride

Canetti *et al.* ont étudié cette possibilité dans leur article [CKN03], en proposant une construction générique pour les schémas de chiffrement à clé publique "standard". Leur solution se base essentiellement sur l'utilisation d'un chiffrement hybride. L'idée de ce chiffrement, comme nous l'avons déjà vu, consiste à tirer aléatoirement une clé de session lorsque l'on souhaite chiffrer un message. Seule cette clé de session est chiffrée avec le chiffrement à clé publique, alors que le message est chiffré à l'aide de cette clé de session. Ainsi deux chiffrés portant sur le même message auront, a priori, une clé de session différente. Ce principe permet d'outrepasser la restriction RCCA en demandant le déchiffrement d'un chiffré portant sur l'un des deux messages m_0 et m_1, à condition bien sûr que ce chiffré ne soit pas le challenge [2].

Cette idée est la base de la construction proposée par ces auteurs. En observant l'éventuelle propriété de rafraîchissement, dont nous avons parlé dans la section précédente, nous voyons que cette seule utilisation d'un chiffrement hybride n'empêche pas le rafraîchissement d'un chiffré, ainsi le schéma obtenu reste uniquement IND-RCCA sûr. Ils ont alors proposé une construction plus élaborée permettant d'y arriver, que nous détaillons maintenant.

5.2.2 La construction Canetti-Krawczyk-Nielsen

Soient un schéma de chiffrement à clé publique PUB sûr contre une attaque RCCA et un schéma de chiffrement à clé secrète SEC sûr contre une attaque CCA tels que définis dans le Chapitre 2 et pour lesquels l'ensemble des clés secrètes du schéma SEC soit inclus dans l'espace des messages du schéma PUB. La construction est décrite ci-dessous.

$\mathsf{Initialiser}(\lambda) \to \mathcal{P}$: éxecute $\mathsf{PUB.Initialiser}(\lambda)$ et $\mathsf{SEC.Initialiser}(\lambda)$ pour obtenir respectivement $\mathcal{P}_{\mathsf{PUB}}$ et $\mathcal{P}_{\mathsf{SEC}}$. Les paramètres publics sont $\mathcal{P} = (\mathcal{P}_{\mathsf{PUB}}, \mathcal{P}_{\mathsf{SEC}})$.

$\mathsf{Clé}(\mathcal{P})$: cet algorithme est le même que l'algorithme $\mathsf{PUB.Clé}$. Il génère donc, à partir de $\mathcal{P}_{\mathsf{PUB}}$, une paire de clés (privée/publique) (sk, pk).

$\mathsf{Chiffrer}(\mathcal{P}, pk, m)$: prend en entrée les paramètres publics $\mathcal{P} = (\mathcal{P}_{\mathsf{PUB}}, \mathcal{P}_{\mathsf{SEC}})$, une clé publique pk et un message m. Cet algorithme fonctionne ainsi :

1. Génère aléatoirement une clé de session K en utilisant l'algorithme $\mathsf{SEC.Clé}(\mathcal{P}_{\mathsf{SEC}})$.
2. Calcule $L_{pk} = \mathsf{PUB.Chiffrer}(\mathcal{P}_{\mathsf{PUB}}, pk, K)$ et $C_2 = \mathsf{SEC.Chiffrer}(\mathcal{P}_{\mathsf{SEC}}, K, L_{pk} \| m)$.
3. Définit $C_{pk} = (L_{pk}, C_2)$ comme chiffré de m pour pk.

2. Ce principe n'est pas compatible avec un chiffrement hybride où la clé de session dépend du message comme les travaux [DAB+02, BKR13] présenter au Chapitre 1 pour éviter la duplication des données chiffrées.

$\mathsf{Déchiffrer}(\mathcal{P}, sk, C_{pk})$: prend en entrée les paramètres publics $\mathcal{P} = (\mathcal{P}_{\mathsf{PUB}}, \mathcal{P}_{\mathsf{SEC}})$, une clé privée sk et un chiffré $C_{pk} = (L_{pk}, C_2)$ pour pk. Cet algorithme déchiffre comme suit :

1. Calcule $K = \mathsf{PUB.Déchiffrer}(\mathcal{P}_{\mathsf{PUB}}, sk, L_{pk})$ et $(L' \| m) = \mathsf{SEC.Déchiffrer}(\mathcal{P}_{\mathsf{SEC}}, K, C_2)$.
2. Vérifie si $L_{pk} = L'$, retourne $\perp$ et s'arrête là si ce n'est pas le cas.
3. Retourne m.

Les auteurs de cette construction ont prouvé que le schéma, ainsi obtenu, est sûr contre une attaque CCA sans aucune restriction non naturelle. Nous voyons que le fait de chiffrer L_{pk} dans C_2 permet de lier ces deux composantes, et notamment de vérifier, au moment du déchiffrement, la validité d'un chiffré. Celle-ci passe notamment par le test d'égalité $L_{pk} = L'$, qui empêche l'éventuel rafraîchissement d'un chiffré.

5.3 Amélioration de la sécurité RCCA pour les schémas offrant un re-chiffrement unique et unidirectionnel

Nous venons de voir comment transformer un chiffrement à clé publique atteignant la sécurité restrictive en un schéma ne l'atteignant pas. Le serveur de re-chiffrement étant composé d'un chiffrement à clé publique, il est possible de réaliser le même genre de transformation sur le serveur de re-chiffrement. Mais nous allons voir que ce n'est pas si évident et qu'il y a mêmes des cas où il ne sera pas possible d'atteindre la sécurité IND-CCA. Nous commençons par étudier dans cette section le cas particulier d'un schéma offrant un re-chiffrement unique et unidirectionnel. Nous exposons une méthode générique permettant d'y arriver. Celle-ci n'est pas efficace et est avant tout d'intérêt théorique. Elle fournit cependant un ensemble d'outils pouvant être utilisés indépendamment dans différents schémas et aboutissant éventuellement à des schémas efficaces. Nous verrons que l'usage de certains de ces outils permet de transformer le schéma [LV08b] en un schéma IND-CCA sûr qui reste efficace.

5.3.1 Une construction générique

L'idée d'utiliser le chiffrement hybride reste un bon point de départ pour cette transformation, puisqu'elle va toujours permettre d'outrepasser a priori les restrictions liées à la sécurité RCCA. Pour des raisons similaires à celles évoquées précédemment, la construction de Canetti *et al.*, qui consiste à encapsuler la composante L_{pk} dans le chiffré C_2, permettrait d'assurer la validité d'un chiffré. Cependant, cela nécessite de le déchiffrer et de tester si la composante L' que l'on récupère lors du déchiffrement est égale à L_{pk}. Sa validité n'est donc pas publiquement vérifiable, ce qui est problématique pour les chiffrés re-chiffrables. Seul l'algorithme de déchiffrement est capable de la vérifier alors que le serveur de re-chiffrement ne l'est pas. Le schéma peut donc éventuellement présenter une faille sur ce point. Il faut alors trouver un moyen d'utiliser cette construction, pour permettre de vérifier publiquement les chiffrés re-chiffrables produits. En particulier, il faut arriver à lier les deux composantes produites par le chiffrement hybride et montrer quelles sont calculées à partir de la même clé de session K.

D'un point de vue générique, nous ne disposons pas d'information sur les calculs effectués pour produire un chiffré. Il semble difficile de pouvoir arriver à prouver efficacement sa validité. Notre idée, pour le prouver, est alors d'utiliser une preuve de connaissance non interactive à divulgation nulle et avec validité de la simulation. Ce type de preuve est appelé en anglais *simulation-sound non-interactive zero-knowledge proof systems* que nous décrivons ci-dessous.

Définition 27 (Système de preuves de connaissance non interactive à divulgation nulle et avec validité de la simulation). *Un* système de preuves de connaissance non interactives à divulgation nulle*, notée* (NIZK)*, pour un language* $\mathcal{L}$ NP*-complet muni d'une relation polynomiale*

$\mathcal{R}$ est la donnée de deux algorithmes probabilistes $\mathtt{Pr}$ et $\mathtt{V}$ tels que les propriétés suivantes soient vérifiées.

***Complétude** : pour tout $y \in \mathcal{L}$ tel qu'il existe ω tel que $(y,\omega) \in \mathcal{R}$, $\mathtt{V}$ retourne 1 sur l'entrée (y,π), où π est la sortie de l'algorithme $\mathtt{Pr}$ sur (y,ω).*

***Validité** : pour tout $y \in \mathcal{L}$ et π, sortie de $\mathtt{Pr}$, si $\mathtt{V}$ retourne 1 sur l'entrée (y,π), alors il existe ω tel que $(y,\omega) \in \mathcal{R}$.*

***Divulgation nulle** : pour tout algorithme $\mathtt{V}^*$ s'exécutant en temps polynomial, il existe un algorithme $\mathtt{Sim}$ fonctionnant en temps polynomial tel que, pour toute suite $\{(y_n,\omega_n)\} \in \mathcal{R}^{\mathbb{N}}$, les ensembles $\{\mathtt{V}^*(y_n, \mathtt{Pr}(y_n,\omega_n))\}$ et $\{\mathtt{V}^*(y_n, S(y_n))\}$ sont indistinguables.*

***Validité de la simulation** : pour tout (y,π) accepté par $\mathtt{V}$, où π n'est pas forcément la sortie de $\mathtt{Pr}$ sur une entrée $(y,\omega) \in \mathcal{R}$, il n'existe pas d'algorithme s'exécutant en temps polynomial utilisant (y,π) pour produire (y',π') avec $y' \notin \mathcal{L}$ et $(y',\pi') \neq (y,\pi)$.*

Sahai a montré dans [Sah99], qu'il était possible de construire de tels systèmes pour n'importe quel language $\mathcal{L}$, NP-complet, et n'importe quelle relation polynomiale $\mathcal{R}$. Avec un tel système, nous allons être capable de prouver que les deux composantes évoquées précédemment sont liées, et plus précisémment qu'elles utilisent la même clé de session K. On retrouve ce genre d'idée dans les articles [NY90] et [FP01] où un message est chiffré deux fois et où l'on prouve que les deux chiffrés correspondent au même message.

Cas d'un re-chiffrement unique et unidirectionnel. Plaçons-nous maintenant dans le cas particulier où les ensembles Type et Liaison du schéma considéré sont respectivement égaux à $\{1,2\}$ et $\{(1,2)\}$. Ainsi, seuls les chiffrés de type 1 doivent disposer d'une vérifiabilité publique. Ces chiffrés ne peuvent pas être des re-chiffrés, ils sont créés à partir de l'algorithme $\mathsf{Chiffrer}_1$ et du message m. Il est donc possible, pour ces chiffrés, de réaliser une preuve NIZK prouvant que la clé de session qu'ils vont masquer est également utilisée pour chiffrer le message m, puisque l'on dispose du message et de la clé de session lors de leur création. Nous obtenons ainsi la vérifiabilité publique d'un chiffré de type 1. En plus de lier ces deux composantes, cette preuve empêche un éventuel rafraîchissement du chiffré, puisque si le chiffré est modifié, la preuve NIZK ne sera plus valable.

Le cas des chiffrés de type 2 est différent. Ils peuvent être des re-chiffrés, et dans ce cas, la preuve NIZK prouvant la validité du chiffré original n'est plus valable. Devenue inutile, cette preuve est donc supprimée lors du re-chiffrement. Cependant, comme nous l'avons expliqué précédemment, ces chiffrés n'ont pas besoin d'être publiquement vérifiables. La technique utilisée dans [CKN03] pour vérifier si les deux composantes L_{pk} et L' sont égales peut donc être suffisante. En considérant que ce chiffré ait été re-chiffré de $\mathtt{A}$ vers $\mathtt{B}$, la composante L' que l'on récupère est égale à $L_{pk_\mathtt{A}}$ qui n'est donc pas égale à $L_{pk_\mathtt{B}}$. Il faut donc, pour tester l'égalité de [CKN03], fournir à $\mathtt{B}$ la composante $L_{pk_\mathtt{A}}$. Nous décidons alors de lui fournir de manière confidentielle, afin d'éviter la prise en compte de considérations non naturelles, comme par exemple le fait que $\mathtt{A}$ soit capable de déchiffrer ce chiffré alors qu'il est destiné à $\mathtt{B}$. Cette confidentialité est atteinte en utilisant un algorithme de chiffrement à clé publique, noté PUB, atteignant la sécurité IND-CCA.

Cette modification permet de s'assurer de la vérifiabilité d'un chiffré de type 2, mais contrairement aux chiffrés de type 1, elle n'empêche pas un éventuel rafraîchissement du chiffré de type 2, puisque $L_{pk_\mathtt{B}}$ n'est lié à aucune autre composante du chiffré. Nous décidons donc de l'encapsuler dans le chiffrement de $L_{pk_\mathtt{A}}$.

La construction détaillée. Soit un paramètre de sécurité λ et un schéma de serveur de re-chiffrement PRE, avec re-chiffrement unique et unidirectionnel, sûr contre une attaque RCCA, dont les ensembles Type et Liaison sont respectivement notés $\{1,2\}$ et $\{(1,2)\}$. Soit également un schéma de chiffrement à clé publique PUB sûr contre une attaque CCA, tel que la réunion d'un chiffré de type 1 et d'un chiffré de type 2 soit incluse dans l'espace des messages de ce schéma. Soit de plus un schéma de chiffrement à clé secrète SEC sûr contre une attaque CCA, tel que

l'ensemble des clés secrètes du schéma soit inclus dans l'espace des messages du schéma PRE. Notre construction est détaillée ci-dessous.

Environnement : il est composé des mêmes ensembles Type et Liaison que le schéma PRE. Quant aux données requises pour calculer une clé de re-chiffrement sur la seule liaison possible, à savoir $(1,2)$, nous avons $\mathsf{D}_{(1,2),de} = \mathsf{PRE.D}_{(1,2),de}$ et $\mathsf{D}_{(1,2),vers} = \mathsf{PRE.D}_{(1,2),vers}$.

$\mathsf{Initialiser}(\lambda) \rightarrow \mathcal{P}$: éxecute $\mathsf{PRE.Initialiser}(\lambda)$, $\mathsf{PUB.Initialiser}(\lambda)$ et $\mathsf{SEC.Initialiser}(\lambda)$ pour obtenir respectivement $\mathcal{P}_{\mathsf{PRE}}$, $\mathcal{P}_{\mathsf{PUB}}$ et $\mathcal{P}_{\mathsf{SEC}}$. Les paramètres publics du schéma sont $\mathcal{P} = (\mathcal{P}_{\mathsf{PRE}}, \mathcal{P}_{\mathsf{PUB}}, \mathcal{P}_{\mathsf{SEC}})$.

$\mathsf{Clé}(\mathcal{P})$: cet algorithme prend en entrée les paramètres publics $\mathcal{P} = (\mathcal{P}_{\mathsf{PRE}}, \mathcal{P}_{\mathsf{PUB}}, \mathcal{P}_{\mathsf{SEC}})$ et exécute $\mathsf{PRE.Clé}(\mathcal{P}_{\mathsf{PRE}})$ et $\mathsf{PUB.Clé}(\mathcal{P}_{\mathsf{PUB}})$ afin d'obtenir la clé privée $sk = (sk_{\mathsf{PRE}}, sk_{\mathsf{PUB}})$ et la clé publique $pk = (pk_{\mathsf{PRE}}, pk_{\mathsf{PUB}})$.

$\mathsf{Chiffrer}_1(\mathcal{P}, pk, m)$: prend en entrée les paramètres publics $\mathcal{P} = (\mathcal{P}_{\mathsf{PRE}}, \mathcal{P}_{\mathsf{PUB}}, \mathcal{P}_{\mathsf{SEC}})$, une clé publique $pk = (pk_{\mathsf{PRE}}, pk_{\mathsf{PUB}})$ et un message m. Cet algorithme fonctionne ainsi.

1. Génère une clé de session K en utilisant l'algorithme $\mathsf{SEC.Clé}(\mathcal{P}_{\mathsf{SEC}})$.
2. Calcule le chiffré hybride (L_{pk}, C_2) où $L_{pk} = \mathsf{PRE.Chiffrer}_1(\mathcal{P}_{\mathsf{PRE}}, pk_{\mathsf{PRE}}, K)$ et $C_2 = \mathsf{SEC.Chiffrer}(\mathcal{P}_{\mathsf{SEC}}, K, L_{pk} \| m)$.
3. Calcule une NIZK, $\pi = \mathtt{Pr}((L_{pk}, C2), (K, m))$, pour la relation composée des deux égalités, $L_{pk} = \mathsf{PRE.Chiffrer}_1(\mathcal{P}_{\mathsf{PRE}}, pk_{\mathsf{PRE}}, K)$ et $C_2 = \mathsf{SEC.Chiffrer}(\mathcal{P}_{\mathsf{SEC}}, K, L_{pk} \| m)$.
4. Définit $C_{pk,1} = (L_{pk}, C_2, \pi)$ comme chiffré de type 1 de m pour pk.

$\mathsf{Déchiffrer}_1(\mathcal{P}, sk, C_{pk,1})$: prend en entrée les paramètres publics $\mathcal{P} = (\mathcal{P}_{\mathsf{PRE}}, \mathcal{P}_{\mathsf{PUB}}, \mathcal{P}_{\mathsf{SEC}})$, une clé privée $sk = (sk_{\mathsf{PRE}}, sk_{\mathsf{PUB}})$ et un chiffré $C_{pk,1} = (L_{pk}, C_2, \pi)$ pour pk, la clé publique associée à sk. Cet algorithme déchiffre ainsi.

1. Vérifie la NIZK, π, en testant si $\mathtt{V}((L_{pk}, C_2), \pi) = 1$ et retourne $\perp$ si ce n'est pas le cas [3].
2. Calcule $K = \mathsf{PRE.Déchiffrer}_1(\mathcal{P}_{\mathsf{PRE}}, sk, L_{pk})$. Il utilise ensuite cette clé de session K pour obtenir $(L' \| m) = \mathsf{SEC.Déchiffrer}(\mathcal{P}_{\mathsf{SEC}}, K, C_2)$.
3. Retourne le message m.

$\mathsf{Délégation}_{(1,2)}(\mathcal{P}, \mathsf{D}^{\mathtt{A}}_{(1,2),de}, \mathsf{D}^{\mathtt{B}}_{(1,2),vers})$: prend en entrée les paramètres publics $\mathcal{P} = (\mathcal{P}_{\mathsf{PRE}}, \mathcal{P}_{\mathsf{PUB}}, \mathcal{P}_{\mathsf{SEC}})$, les ensembles formels de données $\mathsf{D}^{\mathtt{A}}_{(1,2),de} = \mathsf{PRE.D}^{\mathtt{A}}_{(1,2),de}$ et $\mathsf{D}^{\mathtt{B}}_{(1,2),vers} = \mathsf{PRE.D}^{\mathtt{B}}_{(1,2),vers} \cup \{pk_{\mathsf{PUB},\mathtt{B}}\}$. Cet algorithme calcule la clé de re-chiffrement pour le schéma PRE : $\tilde{R}_{\mathtt{A},1\rightarrow\mathtt{B},2} = \mathsf{PRE.Délégation}(\mathcal{P}_{\mathsf{PRE}}, \mathsf{PRE.D}^{\mathtt{A}}_{(1,2),de}, \mathsf{PRE.D}^{\mathtt{B}}_{(1,2),vers})$. Il retourne la clé de re-chiffrement $R_{\mathtt{A},1\rightarrow\mathtt{B},2} = (\tilde{R}_{\mathtt{A},1\rightarrow\mathtt{B},2}, pk_{\mathsf{PUB},\mathtt{B}})$.

$\mathsf{Re\text{-}chiffrer}_{(1,2)}(\mathcal{P}, R_{\mathtt{A},1\rightarrow\mathtt{B},2}, C_{pk_{\mathtt{A}},1})$: prend en entrée les paramètres publics $\mathcal{P} = (\mathcal{P}_{\mathsf{PRE}}, \mathcal{P}_{\mathsf{PUB}}, \mathcal{P}_{\mathsf{SEC}})$, une clé de re-chiffrement $R_{\mathtt{A},1\rightarrow\mathtt{B},2} = (\tilde{R}_{\mathtt{A},1\rightarrow\mathtt{B},2}, pk_{\mathsf{PUB},\mathtt{B}})$ et un chiffré $C_{pk_{\mathtt{A}},1} = (L_{pk_{\mathtt{A}}}, C_2, \pi)$. Cet algorithme fonctionne ainsi.

1. Vérifie la NIZK, π, en testant si $\mathtt{V}((L_{pk_{\mathtt{A}}}, C_2), \pi) = 1$ et retourne $\perp$ si ce n'est pas le cas.
2. Calcule $L_{pk_{\mathtt{B}}} = \mathsf{PRE.Re\text{-}chiffrer}_{(1,2)}(\mathcal{P}_{\mathsf{PRE}}, \tilde{R}_{\mathtt{A},1\rightarrow\mathtt{B},2}, L_{pk_{\mathtt{A}}})$.
3. Chiffre $L_{pk_{\mathtt{A}}}$ et $L_{pk_{\mathtt{B}}}$ pour $\mathtt{B}$: $D_1 = \mathsf{PUB.Chiffrer}(\mathcal{P}_{\mathsf{PUB}}, pk_{\mathsf{PUB},\mathtt{B}}, L_{pk_{\mathtt{A}}} \| L_{pk_{\mathtt{B}}})$.
4. Retourne le chiffré $C_{pk_{\mathtt{B}},2} = (D_1, C_2)$ de type 2 pour $\mathtt{B}$.

$\mathsf{Chiffrer}_2(\mathcal{P}, pk, m)$: prend en entrée les paramètres publics $\mathcal{P} = (\mathcal{P}_{\mathsf{PRE}}, \mathcal{P}_{\mathsf{PUB}}, \mathcal{P}_{\mathsf{SEC}})$, une clé publique $pk = (pk_{\mathsf{PRE}}, pk_{\mathsf{PUB}})$ et un message m. Nous ne pouvons pas le définir comme la succession des algorithmes $\mathsf{Chiffrer}_1$ et $\mathsf{Re\text{-}chiffrer}_{(1,2)}$ en utilisant une entité temporaire $\mathtt{T}$ comme expliqué dans la Remarque 5, car rien ne nous dit que $\mathsf{D}_{(1,2),vers}$ n'est pas composé

3. La validité d'un chiffré pourrait être vérifiée en testant si $L_{pk} = L'$ comme pour [CKN03], sauf qu'il faut vérifier que π est également lié à ces deux composantes, d'où cette vérification.

d'au moins une partie de la clé privée du délégué. Il nous faut donc utiliser l'algorithme $\mathsf{PRE.Chiffrer}_2$.

1. Génère une clé de session K en utilisant l'algorithme $\mathsf{SEC.Clé}(\mathcal{P}_{\mathsf{SEC}})$.
2. Génère une paire de clés (privée/publique) $(sk_{\mathsf{PRE},\mathtt{E}}, pk_{\mathsf{PRE},\mathtt{E}})$ pour $\mathtt{E}$, une entité quelconque, en utilisant l'algorithme $\mathsf{PRE.Clé}(\mathcal{P})$.
3. Calcule $C_{1,pk_{\mathtt{E}}} = \mathsf{PRE.Chiffrer}_1(\mathcal{P}_{\mathsf{PRE}}, pk_{\mathsf{PRE},\mathtt{E}}, K)$. Cette composante correspond à la composante $L_{pk_{\mathtt{A}}}$ d'un re-chiffré de $\mathtt{A}$ vers $\mathtt{B}$. Il calcule également un chiffré du message et de $C_{1,pk_{\mathtt{E}}}$ pour cette entité $\mathtt{E}$, ce qui donne $C_2 = \mathsf{SEC.Chiffrer}(\mathcal{P}_{\mathsf{SEC}}, K, C_{1,pk_{\mathtt{E}}} \| m)$.
4. Calcule $L_{pk} = \mathsf{PRE.Chiffrer}_2(\mathcal{P}_{\mathsf{PRE}}, pk_{\mathsf{PRE}}, K)$.
5. Chiffre $C_{1,pk_{\mathtt{E}}}$ et $\mathcal{L}_{pk}$ pour pk : $D_1 = \mathsf{PUB.Chiffrer}(\mathcal{P}_{\mathsf{PUB}}, pk_{\mathsf{PUB}}, C_{1,pk_{\mathsf{PRE},\mathtt{E}}} \| L_{pk})$.
6. Retourne le chiffré $C_{pk,2} = (D_1, C_2)$ de type 2 pour pk.

$\mathsf{Déchiffrer}_2(\mathcal{P}, sk, C_{pk,2})$: prend en entrée les paramètres publics $\mathcal{P} = (\mathcal{P}_{\mathsf{PRE}}, \mathcal{P}_{\mathsf{PUB}}, \mathcal{P}_{\mathsf{SEC}})$, une clé privée $sk = (sk_{\mathsf{PRE}}, sk_{\mathsf{PUB}})$ et un chiffré $C_{pk,2} = (D_1, C_2)$ pour pk, la clé publique associée à sk. Cet algorithme déchiffre ainsi.

1. Calcule $(L \| L_{pk}) = \mathsf{PUB.Déchiffrer}(\mathcal{P}_{\mathsf{PUB}}, sk_{\mathsf{PUB}}, D_1)$. Selon les cas, L est égal à $L_{pk_{\mathtt{A}}}$ ou $L_{pk_{\mathtt{B}}}$.
2. Calcule $K = \mathsf{PRE.Déchiffrer}_2(\mathcal{P}_{\mathsf{PRE}}, sk, L_{pk})$. Il utilise ensuite cette clé de session K pour obtenir $(L' \| m) = \mathsf{SEC.Déchiffrer}(\mathcal{P}_{\mathsf{SEC}}, K, C_2)$.
3. Vérifie si $L = L'$, retourne $\perp$ et s'arrête si ce n'est pas le cas.
4. Retourne le message m.

Ce schéma est consistant, à condition que les différents schémas PRE, PUB et SEC le soient.

5.3.2 Sécurité du schéma

Théorème 28. *Le schéma atteint la sécurité* IND-CCA *pour un chiffré challenge re-chiffrable dans le modèle de l'oracle aléatoire, à condition que le schéma* PRE *atteigne la sécurité* IND-RCCA *pour un chiffré challenge de type 1, et que les schémas* PUB *et* SEC *atteignent chacun la sécurité* IND-CCA*.*

Preuve rapide. Nous ne donnons qu'une idée de la preuve. Plus précisément, nous ne décrivons que la simulation d'un algorithme $\mathcal{B}$, utilisant un adversaire $\mathcal{A}$ contre la sécurité de ce schéma afin de casser la sécurité du schéma SEC. Cet algorithme se décrit ainsi.

Phase d'initialisation. $\mathcal{B}$ étant un adversaire contre la sécurité du schéma SEC, il reçoit les paramètres publics $\mathcal{P}_{\mathsf{SEC}}$ de ce schéma. La sécurité qu'il attaque étant l'IND-CCA, il a également accès à un oracle de déchiffrement pour ce schéma noté SEC.ODéchiffrer. Il génère les paramètres publics $\mathcal{P}_{\mathsf{PRE}}$ et $\mathcal{P}_{\mathsf{PUB}}$ comme dans le schéma et retourne les paramètres publics $\mathcal{P} = (\mathcal{P}_{\mathsf{PRE}}, \mathcal{P}_{\mathsf{PUB}}, \mathcal{P}_{\mathsf{SEC}})$ à $\mathcal{A}$.

Il génère également les clés des différentes entités comme dans le schéma en utilisant l'algorithme Clé et retourne à $\mathcal{A}$ la paire de clé (privée/publique) (sk, pk) si l'entité est corrompue et seulement la clé publique pk si celle-ci est honnête. Il stocke (sk, pk) dans une liste $\mathcal{K}^{\mathsf{list}}$. Il génère, toujours en utilisant cet algorithme, la paire de clés (privée/publique) (sk^*, pk^*) de l'entité cible, fournit pk^* à $\mathcal{A}$ et stocke (sk^*, pk^*) dans $\mathcal{K}^{\mathsf{list}}$.

Phase 1. $\mathcal{B}$ répond aux différentes requêtes de $\mathcal{A}$ de la manière suivante.

Délégation$(pk_{\mathtt{A}}, pk_{\mathtt{B}})$: $\mathcal{B}$ génère la clé de re-chiffrement de $\mathtt{A}$ vers $\mathtt{B}$ comme dans le schéma et la fournit à l'adversaire.

ORe-chiffrer$(pk_{\mathtt{A}}, pk_{\mathtt{B}}, C_{pk_{\mathtt{A}},1})$: $\mathcal{B}$ re-chiffre en utilisant la clé de re-chiffrement de $\mathtt{A}$ vers $\mathtt{B}$ comme le fait l'algorithme $\mathsf{Re\text{-}chiffrer}_{(1,2)}$ et fournit le résultat à l'adversaire.

ODéchiffrer$_2(pk, C_{pk,2})$: $\mathcal{B}$ déchiffre en utilisant la clé privée sk associée à la clé publique pk comme le fait l'algorithme $\mathsf{Déchiffrer}_2$ et fournit le résultat à l'adversaire.

Challenge. $\mathcal{A}$ choisit deux messages m_0 et m_1 appartenant tous les deux à l'ensemble $\mathcal{M}$. $\mathcal{B}$ génère alors une clé de session K' en utilisant l'algorithme $\mathsf{SEC.Clé}$ et calcule $L^* = \mathsf{PRE.Chiffrer}_1(\mathcal{P}_{\mathsf{PRE}}, pk^*_{\mathsf{PRE}}, K')$. $\mathcal{B}$ fournit alors $L^*\|m_0$ et $L^*\|m_1$ comme messages challenges et reçoit un chiffré $C_2^* = \mathsf{SEC.Chiffrer}(\mathcal{P}_{\mathsf{SEC}}, K^*, L^*\|m_b)$ où $b \in \{0,1\}$ et K^* est une clé privée générée par l'algorithme $\mathsf{Clé}$; ces deux éléments étant inconnus de $\mathcal{B}$. $\mathcal{B}$ simule ensuite une preuve NIZK π^* sur les deux composantes L^* et C_2^*, qui sera valide pour $\mathcal{A}$ bien que ces deux chiffrés ne portent pas, a priori, sur la même clé de session [4]. Il retourne ensuite le challenge $C_{pk^*,1} = (L^*, C_2^*, \pi^*)$ à $\mathcal{A}$.

Phase 2. $\mathcal{B}$ répond aux différentes requêtes de $\mathcal{A}$ comme dans la Phase 1., avec une différence sur l'oracle de déchiffrement.

ODéchiffrer$_2(pk, C_{pk,2})$: $\mathcal{B}$ récupère la clé privée $sk = (sk_{\mathsf{PRE}}, sk_{\mathsf{PUB}})$ de $\mathcal{K}^{\mathsf{list}}$ et décompose $C_{pk,2}$ en (D_1, C_2). Il calcule $(L, L_{pk}) = \mathsf{PUB.Déchiffrer}(\mathcal{P}_{\mathsf{PUB}}, sk_{\mathsf{PUB}}, D_1)$, puis la clé de session $K = \mathsf{PRE.Déchiffrer}_2(\mathcal{P}_{\mathsf{PRE}}, sk, L_{pk})$. Il est alors confronté à deux possibilités.

- si $K \neq K'$, il calcule $(L'\|m) = \mathsf{SEC.Déchiffrer}(\mathcal{P}_{\mathsf{SEC}}, K, C_2)$. Si $L = L'$, alors il retourne m ou $\perp$ si ce n'est pas le cas.
- si $K = K'$, il requiert l'oracle $\mathsf{SEC.ODéchiffrer}$ sur l'entrée C_2 et obtient $(L'\|m) = \mathsf{SEC.Déchiffrer}(\mathcal{P}_{\mathsf{SEC}}, K^*, C_2)$. Si $L = L'$, alors il retourne m ou $\perp$ si ce n'est pas le cas.

Réponse. $\mathcal{A}$ retourne b', qui est retourné ensuite par $\mathcal{B}$. Si $\mathcal{B}$ gagne son challenge, alors $\mathcal{A}$ aussi.

Cet algorithme permet de voir que si l'adversaire a un avantage non négligeable pour casser la sécurité du schéma que nous avons présenté, alors $\mathcal{B}$ a le même avantage pour casser la sécurité du schéma SEC. Il est possible de se rendre compte que la simulation est parfaite, en se rappelant que les schémas PRE et PUB sont respectivement IND-RCCA et IND-CCA sûr. Cela permet par exemple de justifier le fait que l'adversaire ne pourra s'apercevoir que la clé K' chiffrée dans L^* ne correspond pas à la clé K^* qui est utilisée pour chiffrer le message m_b. □

Théorème 29. *Le schéma atteint la sécurité* IND-CCA *pour un chiffré challenge de type 2 re-chiffré ou pas dans le modèle de l'oracle aléatoire, à condition que le schéma* PRE *atteigne la sécurité* IND-RCCA *pour un chiffré challenge re-chiffré ou pas, et que les schémas* PUB *et* SEC *atteignent chacun la sécurité* IND-CCA*.*

Preuve informelle. La preuve suit les mêmes lignes que la précédente. On notera que l'algorithme à décrire sera globalement le même pour ces deux sécurités, chiffré challenge de type 2 re-chiffré et chiffré challenge de type 2 non re-chiffré. Il faudra seulement générer ce challenge différement selon la sécurité considérée afin qu'il soit re-chiffré pour la première et non re-chiffré pour la seconde. □

Corollaire 30. *Le schéma atteint la sécurité* IND-CCA *dans le modèle de l'oracle aléatoire, à condition que le schéma* PRE *atteigne la sécurité* IND-RCCA*, et que les schémas*PUB *et* SEC *atteignent chacun la sécurité* IND-CCA*.*

Amélioration éventuelle. Dans le cas éventuel où le schéma de serveur de re-chiffrement RCCA sûr ne permette pas le rafraîchissement d'un chiffré de type 2, il est possible de rendre la construction plus efficace. En effet, nous avons encapsulé la composante $L_{pk_{\mathtt{B}}}$ lors du re-chiffrement de $\mathtt{A}$ vers $\mathtt{B}$ dans le chiffré D_1 afin d'empêcher qu'elle ne puisse être rafraîchie. Il est alors inutile d'effectuer cette encapsulation - il est possible de s'en convaincre en observant la preuve informelle du

4. Comme expliqué dans [FP01], une telle simulation, qui soit valide, est notamment possible dans le modèle de l'oracle aléatoire.

Théorème 28 -, ce qui permet d'améliorer l'efficacité de la construction, même si le calcul le plus coûteux reste toujours le calcul de la NIZK. Nous ne pouvons pas, en revanche, améliorer cette construction dans le cas où les chiffrés de type 1 ne sont pas rafraîchissables, puisque, contrairement au chiffrement de type 2, nous n'avons pas ajouté de calculs additionnels au chiffrement de type 1 pour contrer cette éventuelle propriété.

5.3.3 Un exemple de construction plus pratique sur le schéma Libert-Vergnaud

Cette construction générique fournit les différents ingrédients qui peuvent être utilisés pour transformer un schéma RCCA sûr spécifique en un schéma CCA. Nous pouvons nous en servir pour rendre, par exemple, le schéma [LV08b] sûr contre une attaque CCA, tout en restant efficace.

En observant ce dernier (voir Figure 4.2), nous nous apercevons que la restriction RCCA sur ce schéma est due seulement au fait qu'un chiffré de type 2 est publiquement rafraîchissable. Nous l'avons déjà constaté dans la Figure 4.2 puisque nous avons considéré un chiffré de type 2 de la forme $C_{pk_{\mathtt{B}},2} = (C_1, C_{2,\mathtt{B}}, D_{\mathtt{B}}, E_{\mathtt{B}}, C_3, C_4, \sigma)$ avec, en particulier, $C_{2,\mathtt{B}} = g^{r\tilde{t}}$, $D_{\mathtt{B}} = g^{\tilde{t}}$ et $E_{\mathtt{B}} = pk_{\mathtt{B}}^{1/\tilde{t}}$ alors que le schéma le décrit initialement avec $C_{2,\mathtt{B}} = pk_{\mathtt{B}}^{r\tilde{t}}$, $D_{\mathtt{B}} = pk_{\mathtt{B}}^{\tilde{t}}$ et $E_{\mathtt{B}} = g^{1/\tilde{t}}$. De manière générale, un tel chiffré peut être rafraîchi en tirant aléatoirement un élément $z \xleftarrow{\$} \mathbb{Z}_q^*$ et en remplaçant ces trois éléments par $C_{2,\mathtt{B}}^z$, $D_{\mathtt{B}}^z$ et $E_{\mathtt{B}}^{1/z}$.

Pour empêcher une telle modification, nous avons vu, dans la construction générique, que nous pouvions utiliser le chiffrement à clé publique que nous avons appelé PUB. Chiffrer $C_{pk_{\mathtt{B}},2}$ est donc une solution possible. Cependant, nous ne sommes pas obligés de chiffrer toutes ses composantes. Nous voyons en effet que C_1, C_3, C_4 et σ n'interviennent pas lors d'un tel rafraîchissement et il n'y a alors aucune raison de les chiffrer. De plus, les trois composantes concernées par cette modification sont liées entre elles par le couplage bilinéaire et une modification sur une des composantes implique donc nécessairement une modification des autres. Il suffit donc qu'une seule de ces composantes ne puisse pas être modifiée pour obtenir un schéma CCA sûr. Ces trois composantes appartiennent au groupe $\mathbb{G}$, il peut donc être judicieux de choisir un chiffrement PUB qui soit CCA sûr et qui chiffre efficacement un élément de ce groupe. Nous restons dans un cas général et utilisons un schéma générique atteignant cette sécurité, bien qu'on pourrait par exemple utiliser le chiffrement Hash ElGamal (voir Section 6.1.1) qui permet de chiffrer n'importe quelle élément de $\{0,1\}^*$.

Nous détaillons maintenant le schéma [LV08b] modifié afin qu'il ne considère plus aucune restriction non naturelle. Dans notre description, nous gardons toujours l'idée du chiffrement hybride afin qu'il permette de chiffrer n'importe quel message indépendamment de sa longueur. Dans la construction générique de la section précédente, cette considération de chiffrement hybride impliquait l'utilisation d'une preuve NIZK. Concernant ce schéma, nous pouvons faire mieux et éviter l'utilisation d'une telle preuve qui en alourdit l'efficacité. En effet, on s'aperçoit que ce schéma lie les composantes C_3 et C_4 au reste du chiffré en utilisant une signature à usage unique. Notre idée est donc de joindre à ces deux éléments le chiffré produit par le chiffrement à clé secrète utilisé pour le chiffrement hybride, ce qui liera cette composante aux autres. Pour les mêmes raisons, nous n'avons pas non plus à encapsuler le chiffré de K au message m dans le chiffrement hybride.

Le schéma ainsi modifié est détaillé dans la Figure 5.1 où :
- OTS est une signature à usage unique composée des algorithmes Clé, Signer et Vérifier.
- u et v sont des paramètres publics.
- SEC est un chiffrement à clé secrète CCA sûr.
- PUB est un chiffrement à clé publique CCA sûr tel que l'espace des messages contienne l'espace des clés du schéma SEC.

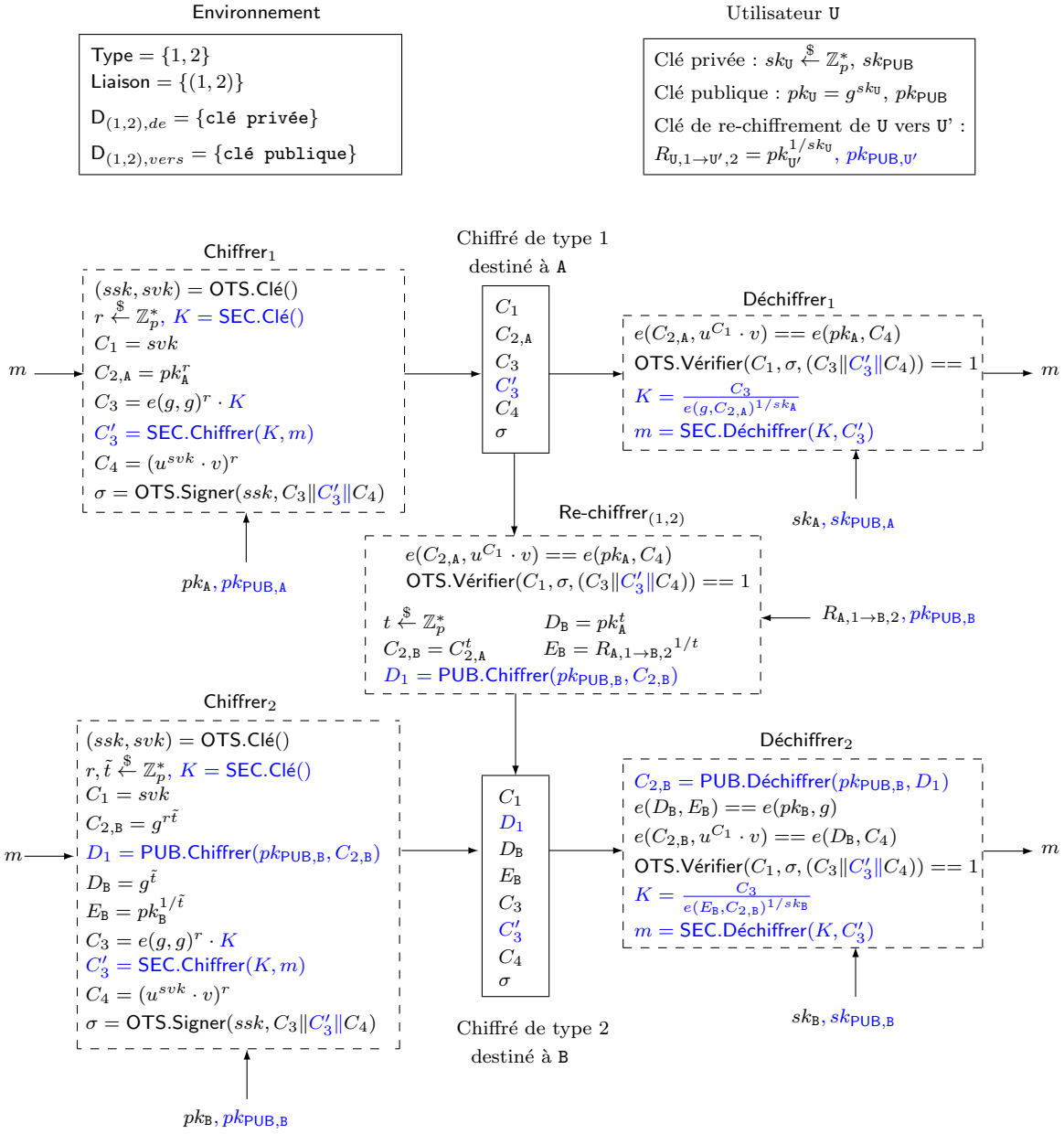

Figure 5.1 – Le schéma [LV08b] modifié CCA sûr

Théorème 31. *Le schéma atteint la sécurité* IND-CCA *dans le modèle standard, à condition que les schémas* PUB *et* SEC *atteignent chacun la sécurité* IND-CCA*.*

Preuve informelle. Il faut, pour prouver ce théorème, que le schéma soit sûr pour chaque manière de produire un chiffré. La preuve dans chacun des cas ressemble globalement à celle fournie dans l'article [LV08b] en y ajoutant les spécificités liées aux outils que nous avons utilisés et dont les utilisations sont décrites dans les Théorèmes 28 et 29. □

Efficacité. L'efficacité dépend bien évidemment du schéma PUB choisi. En supposant que son efficacité soit de l'ordre de celle du chiffrement bilinéaire utilisé comme base de ce schéma, nous obtenons au final un schéma dont l'efficacité est multiplié par 2, ce qui reste acceptable. Concernant le stockage des données, le serveur ne stocke, pour rappel, que des chiffrés non rechiffrés. L'espace de stockage reste donc inchangé par rapport au schéma [LV08b] originel s'il est utilisé avec un chiffrement hybride.

5.4 À propos des autres types

5.4.1 Le type bidirectionnel et unique

Si l'on regarde bien les opérations mentionnées dans notre construction, à aucun moment nous n'utilisons le fait que le re-chiffrement soit unidirectionnel. Notre construction reste donc valable pour un schéma unidirectionnel et unique. À un point près, puisqu'il faut que la clé de re-chiffrement obtenue soit également bidirectionnelle, ce qui n'est pas le cas puisqu'elle est de la forme $R_{\mathtt{A},1\to\mathtt{B},2} = (\tilde{R}_{\mathtt{A},1\to\mathtt{B},2}, pk_{\mathsf{PUB},\mathtt{B}})$. Pour la rendre bidirectionnelle, il faut donc lui ajouter l'élément $pk_{\mathsf{PUB},\mathtt{A}}$.

Pour rappel, beaucoup de schémas bidirectionnels considèrent des restrictions non naturelles que nous avons notées $\mathrm{ATT}_{\mathrm{B}}$ pour $\mathrm{ATT} \in \{\mathrm{CPA}, \mathrm{CCA}, \mathrm{RCCA}\}$. Dans ce cas, cette construction générique permet de modifier un schéma avec sécurité IND-$\mathrm{RCCA}_{\mathrm{B}}$ en un schéma avec sécurité IND-$\mathrm{CCA}_{\mathrm{B}}$.

5.4.2 Les schémas avec re-chiffrement multiple supérieur supérieur à 1 ou infini

Concevoir une construction générique dans un tel cas est plus difficile. L'utilisation d'un chiffrement hybride et même de la construction générique que nous avons présentée serait un bon départ. Elle permettrait en quelque sorte de "transformer le premier re-chiffrement RCCA en un re-chiffrement CCA". On s'aperçoit également que ce re-chiffrement modifie le type du chiffré. Ainsi, même si l'on considérait un re-chiffrement de base sur une liaison $(1,1)$, cela nous donnera après utilisation de la construction un re-chiffrement sur une liaison $(1,2)$, laissant sous-entendre l'impossibilité de re-chiffrer une seconde fois pour les schémas offrant une liaison $(1,1)$.

Plaçons nous dans le cas le plus simple avec un re-chiffrement de cardinal 2 sur un ensemble $\mathsf{Type} = \{1,2,3\}$ et $\mathsf{Liaison} = \{(1,2),(2,3)\}$. Lors d'un re-chiffrement de `A` vers `B` pour la liaison $(1,2)$, le chiffré $C_{pk_{\mathtt{A}},1} = (L_{pk_{\mathtt{A}}}, C_2, \pi)$ est transformé en $C_{pk_{\mathtt{B}},2} = (D_1, C_2)$ où $D_1 = \mathsf{PUB.Chiffrer}(\mathcal{P}_{\mathsf{PUB}}, pk_{\mathsf{PUB},\mathtt{B}}, L_{pk_{\mathtt{A}}} \| L_{pk_{\mathtt{B}}})$, $L_{pk_{\mathtt{B}}} = \mathsf{PRE.Re\text{-}chiffrer}_{(1,2)}(\mathcal{P}_{\mathsf{PRE}}, \tilde{R}_{\mathtt{A},1\to\mathtt{B},2}, L_{pk_{\mathtt{A}}})$. Le problème avec ce chiffré est que la composante qui est re-chiffrable, à savoir $L_{pk_{\mathtt{B}}}$, n'est pas disponible du serveur qui effectuerait le re-chiffrement. Il n'est donc pas possible de réaliser un second re-chiffrement et donc d'obtenir un re-chiffrement multiple ou infini de cette manière.

Dans l'éventualité où la composante $L_{pk_{\mathtt{A}}}$ (correspondant à un chiffré de type 2 du schéma originel) ne pourrait être re-chiffrée, il n'est pas nécessaire de l'encapsuler dans le chiffré D_1. Cette composante est donc à disposition du serveur qui peut calculer $L_{pk_{\mathtt{C}}} = \mathsf{PRE.Re\text{-}chiffrer}_{(2,3)}(\mathcal{P}_{\mathsf{PRE}}, \tilde{\ })$. On est ramené au même problème de rafraîchissement puisque si $L_{pk_{\mathtt{C}}}$ est rafraîchissable, il faut le chiffrer avec un chiffrement à clé publlique atteignant la sécurité IND-CCA. S'il ne l'est pas, ce chiffrement n'est pas nécessaire. Si tous se passe bien et que le schéma n'est en aucun cas concerné par le rafraîchissement d'un chiffré, il est alors possible d'obtenir un re-chiffrement successif, à condition bien sûr de disposer d'une vérifiabilité publique de chaque chiffré dont nous ne disposons pas encore. Il faudrait que le serveur produise lors de chaque re-chiffrement une preuve NIZK supplémentaire sur chaque liaison successive afin de prouver la validité du re-chiffré. En particulier, cette preuve devrait par exemple prouver sur la liaison $(1,2)$ que le chiffré $L_{pk_{\mathtt{B}}}$ est bien un re-chiffré d'un certain chiffré $L_{pk_{\mathtt{A}}}$, que le chiffré D_1 est bien un chiffré de $L_{pk_{\mathtt{A}}} \| L_{pk_{\mathtt{B}}}$ et qu'il existe une certaine preuve NIZK prouvant que $L_{pk_{\mathtt{A}}}$ et C_2 sont formés comme dans l'algorithme $\mathsf{Chiffrer}_1$. Il faudrait ensuite itérer cette idée aux preuves des re-chiffrés successifs.

Cette construction permet au final d'obtenir un schéma avec re-chiffrement multiple ou infini et atteignant la sécurité IND-CCA que dans des cas très précis où aucun des différents chiffrés produit ne doit être rafraîchissable. Le rafraîchissement d'un chiffré étant pourtant une des causes principales de la restriction RCCA (comme en témoigne par exemple le schéma [LV08b]), l'intérêt de cette construction en est donc très limité. Elle devient de plus très vite laborieuse par l'uti-

lisation de preuves NIZK successives. La problématique de trouver une construction générique fonctionnant pour tous les schémas de serveur de re-chiffrement atteignant la sécurité IND-RCCA avec re-chiffrement multiple de cardinal n ou infini reste donc ouverte.

Chapitre 6

Conception de schémas sûrs et fondés sur le chiffrement Hash ElGamal

Sommaire

6.1 Revue des schémas existants 81
6.1.1 Boîte à outils 81
6.1.2 Le schéma de Deng *et al.* 82
6.1.3 Le schéma de Chow *et al.* 83
6.1.4 Analyse de la sécurité 84
6.2 Un nouveau schéma 86
6.2.1 Preuve de connaissance avec extracteur 86
6.2.2 Description du nouveau schéma 87
6.2.3 Sécurité du schéma 88

Nous présentons dans ce chapitre nos résultats autour du schéma de Chow *et al.* exposé dans [CWYD10], basé sur le chiffrement Hash ElGamal. En particulier, nous nous apercevons que la sécurité du schéma nécessite de prendre en considération des restrictions non naturelles en plus de celles annoncées par ses auteurs. Ce schéma étant basé sur le schéma [DWLC08] de Deng *et al.*, il est normal de se demander si ce dernier ne présente pas les mêmes problèmes. Nous proposons, pour chacun, une réparation afin qu'il n'y ait aucune restriction non naturelle à prendre en compte [1]. Ces travaux ont été publiés dans [CDL11].

6.1 Revue des schémas existants

Nous avons présenté dans le Chapitre 4, la base du chiffrement des deux schémas ainsi que le fonctionnement de leur re-chiffrement. Afin d'étudier leur sécurité, nous devons maintenant les décrire en détail pour mettre en évidence les opérations qui posent problème.

6.1.1 Boîte à outils

Avant de présenter ces schémas, nous détaillons les outils sur lesquels ceux-ci sont fondés. En particulier, nous présentons le chiffrement Hash ElGamal, ainsi que la preuve de connaissance non interactive qui permet une vérifiabilité publique de ces chiffrés. Les ensembles, dans lesquels nous allons considérer les différents éléments de ces outils, seront précisés lors de la description du schéma de Deng *et al.* dans la section suivante.

1. Concernant le schéma [DWLC08], il faudra tout de même considérer les restrictions non naturelles récurrentes des schémas avec re-chiffrement bidirectionnel que nous avons déjà évoquées à la Section 3.3.3.

Le chiffrement Hash-Elgamal. Le chiffrement Hash ElGamal consiste à utiliser la technique de Fujisako-Okamoto [FO99] sur le chiffrement ElGamal [Gam84] afin de le rendre IND-CCA (dans le modèle de l'oracle aléatoire) sous l'hypothèse CDH. Le chiffrement d'un message m pour la clé publique $pk = g^{sk}$, associée à la clé privée sk, consiste à tirer un aléa ω et à calculer $r = \mathcal{H}_1(m\|\omega)$, $E = pk^r$ et $F = (m\|\omega) \oplus \mathcal{H}_2(g^r)$ où $\mathcal{H}_1$ et $\mathcal{H}_2$ sont deux fonctions de hachage modélisées comme des oracles aléatoires. La clé privée sk permet de retrouver g^r via E et donc $(m\|\omega)$ via F. Avant de retourner m, il faut cependant vérifier que le chiffré est valide en s'assurant que $E = g^{sk.\mathcal{H}_1(m\|\omega)}$.

La preuve de connaissance non interactive à la Schnorr. Pour obtenir une vérifiabilité publique de leurs chiffrés, Deng *et al.* ont utilisé une variante d'une preuve de connaissance basée sur le protocole d'identification de Schnorr [Sch89], rendue non interactive par l'heuristique de Fiat-Shamir [FS86], ce qui fournit ainsi un protocole de signature. Cela permet de prouver la connaissance d'un logarithme discret, en effectuant une signature valide d'un message, où la clé de signature utilisée est ce logarithme discret.

Cette variante se décrit de la manière suivante : pour prouver la connaissance d'un logarithme discret r vérifiant la relation $E = pk^r$, nous calculons la signature (D, s) où $D = pk^u$ et $s = u + r.\mathcal{H}_3(D\|E\|m)$ avec u un élément tiré aléatoirement et m le message à signer.

6.1.2 Le schéma de Deng *et al.*

Pour mémoire, le schéma [DWLC08] offre un re-chiffrement bidirectionnel et unique. Ce schéma est le seul de la littérature à proposer un tel re-chiffrement sans passer par les techniques présentées à la Section 4.2. Ce re-chiffrement unique est rendu possible via l'utilisation d'une preuve de connaissance à la Schnorr, qui dépend du destinataire du chiffré et qui n'est pas transformable. La bidirectionnalité du re-chiffrement est assurée par un quotient de la clé privée du délégant et de celle du délégué. Nous détaillons maintenant ce schéma.

Environnement : il est composé d'un ensemble $\mathsf{Type} = \{1, 2\}$ et d'un ensemble $\mathsf{Liaison} = \{(1, 2)\}$. Les structures de données, indiquant les données requises pour calculer la clé de re-chiffrement, sont toutes composées de la clé privée et de la clé publique de l'entité concernée.

Initialiser$(\lambda) \to \mathcal{P}$: soient p et q deux nombres premiers tels que λ soit la taille de q et que $q|p-1$. Soit $\mathbb{G} = \langle g \rangle$ un sous-groupe de $\mathbb{Z}_p^*$ d'ordre q. De plus, soient l_0 et l_1, deux paramètres de sécurité, polynomiaux en λ. Considérons alors $\mathcal{M} = \{0,1\}^{l_0}$ comme espace des messages, ainsi que trois fonctions de hachage résistantes aux collisions, $\mathcal{H}_1 : \{0,1\}^{l_0} \times \{0,1\}^{l_1} \to \mathbb{Z}_q^*$, $\mathcal{H}_2 : \mathbb{G} \to \{0,1\}^{l_0+l_1}$ et $\mathcal{H}_3 : \{0,1\}^* \to \mathbb{Z}_q^*$. Les paramètres publics sont $\mathcal{P} = (q, \mathbb{G}, l_0, l_1, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3)$.

Clé$(\mathcal{P})$: prend en entrée les paramètres publics $\mathcal{P}$, tire aléatoirement $sk \xleftarrow{\$} \mathbb{Z}_q^*$ et définit $pk = g^{sk}$.

Délégation$_{1,2}(\mathcal{P}, (sk_{\mathtt{A}}, pk_{\mathtt{A}}), (sk_{\mathtt{B}}, pk_{\mathtt{B}}))$: prend en entrée les paramètres publics $\mathcal{P}$, la paire de clés $(sk_{\mathtt{A}}, pk_{\mathtt{A}})$ du délégant et la paire de clés $(sk_{\mathtt{B}}, pk_{\mathtt{B}})$ du délégué. Cet algorithme calcule $\tilde{R}_{\mathtt{A},1\to\mathtt{B},2} = sk_{\mathtt{B}}/sk_{\mathtt{A}}$ et retourne la clé de re-chiffrement bidirectionnelle $R_{\mathtt{A},1\to\mathtt{B},2} = (\tilde{R}_{\mathtt{A},1\to\mathtt{B},2}, pk_{\mathtt{A}}, pk_{\mathtt{B}})$.

Chiffrer$_1(\mathcal{P}, pk, m)$: prend en entrée les paramètres publics $\mathcal{P}$, une clé publique pk et un message $m \in \mathcal{M}$. Cet algorithme fonctionne comme suit.

1. Tire aléatoirement $\omega \xleftarrow{\$} \{0,1\}^{l_1}$ et calcule $r = \mathcal{H}_1(m\|\omega)$.
2. Calcule $E = pk^r$ et $F = (m\|\omega) \oplus \mathcal{H}_2(g^r)$.
3. Tire aléatoirement $u \xleftarrow{\$} \mathbb{Z}_q^*$ et calcule $D = pk^u$ et $s = u + r.\mathcal{H}_3(D\|E\|F)$.

4. Définit $C_{pk,1} = (D, E, F, s)$ comme chiffré de type 1.

$\mathsf{Déchiffrer}_1(\mathcal{P}, sk, C_{pk,1})$: prend en entrée les paramètres publics $\mathcal{P}$, une clé privée sk et un chiffré $C_{pk,1} = (D, E, F, s)$ de type 1 pour pk. Cet algorithme déchiffre comme suit.

1. Vérifie si $pk^s = D.E^{\mathcal{H}_3(D\|E\|F)}$, retourne $\perp$ si ce n'est pas le cas.
2. Calcule $(m\|\omega) = F \oplus \mathcal{H}_2(E^{1/sk})$.
3. Vérifie si $E = g^{sk.\mathcal{H}_1(m\|\omega)}$ et retourne m, ou $\perp$ si ce n'est pas le cas.

$\mathsf{Re\text{-}chiffrer}_{1,2}(\mathcal{P}, R_{\mathtt{A},1\to\mathtt{B},2}, C_{\mathtt{A},1})$: prend en entrée les paramètres publics $\mathcal{P}$, une clé de re-chiffrement $R_{\mathtt{A},1\to\mathtt{B},2} = (\tilde{R}_{\mathtt{A},1\to B,2}, pk_{\mathtt{A}}, pk_{\mathtt{B}})$ de $\mathtt{A}$ vers $\mathtt{B}$, un chiffré $C_{\mathtt{A},1} = (D, E, F, s)$ de type 1 pour $\mathtt{A}$. Cet algorithme re-chiffre comme suit.

1. Vérifie si $pk_{\mathtt{A}}^s = D.E^{\mathcal{H}_3(D\|E\|F)}$, retourne $\perp$ si ce n'est pas le cas.
2. Calcule $E' = E^{\tilde{R}_{\mathtt{A},1\to\mathtt{B},2}} = (g^{r.sk_{\mathtt{A}}})^{(sk_{\mathtt{B}}/sk_{\mathtt{A}})} = g^{r.sk_{\mathtt{B}}}$.
3. Définit $C_{\mathtt{B},2} = (E', F)$ comme chiffré de type 2.

$\mathsf{Chiffrer}_2(\mathcal{P}, pk, m)$: prend en entrée les paramètres publics $\mathcal{P}$, une clé publique pk et un message $m \in \mathcal{M}$. Cet algorithme comme suit.

1. Tire aléatoirement $\omega \xleftarrow{\$} \{0,1\}^{l_1}$ et calcule $r = \mathcal{H}_1(m\|\omega)$.
2. Calcule $E' = pk^r$ et $F = (m\|\omega) \oplus \mathcal{H}_2(g^r)$.
3. Définit $C_{pk,2} = (E', F)$ comme chiffré de type 2.

$\mathsf{Déchiffrer}_2(\mathcal{P}, sk, C_{pk,2})$: prend en entrée les paramètres publics $\mathcal{P}$, une clé privée sk et un chiffré $C_{pk,2} = (E', F)$ de type 2 pour pk. Cet algorithme déchiffre comme suit.

1. Calcule $(m\|\omega) = F \oplus \mathcal{H}_2(E'^{1/sk})$.
2. Vérifie si $E' = g^{sk.\mathcal{H}_1(m\|\omega)}$ et retourne m, ou $\perp$ si ce n'est pas le cas.

Nous obtenons ainsi un schéma avec re-chiffrement bidirectionnel et unique. Sa consistance est évidente. Avant d'étudier sa sécurité, nous décrivons aussi le schéma de Chow *et al.* [CWYD10] qui est construit sur la base de ce schéma.

6.1.3 Le schéma de Chow *et al.*

Nous avons vu à la Section 4.2.2 le fonctionnement du schéma [CWYD10]. En observant les calculs des différents algorithmes qui le composent, nous nous apercevons qu'il est en fait très proche du schéma de Deng *et al.* que nous venons de présenter et qui offre un re-chiffrement bidirectionnel et unique. En effet, l'idée qu'ont eu Chow *et al.* a été de transformer ce dernier en un schéma avec un re-chiffrement unidirectionnel et unique. Ces deux schémas offrant un re-chiffrement unique, ils auront ainsi les mêmes ensembles Type et Liaison. Nous détaillons maintenant les différences par rapport au schéma initial de Deng *et al.*

L'idée de Chow *et al.*, pour transformer un re-chiffrement bidirectionnel en un re-chiffrement unidirectionnel, fut d'introduire une technique consistant à utiliser une entité virtuelle, que nous avons généralisée dans la Section 4.2. Le seul élément permettant réellement de re-chiffrer dans le schéma [DWLC08] est de la forme $sk_{\mathtt{B}}/sk_{\mathtt{A}}$. En effet, les autres éléments de la clé de re-chiffrement, à savoir $pk_{\mathtt{A}}$ et $pk_{\mathtt{B}}$, ne servent qu'à assurer la validité publique d'un chiffré re-chiffrable. Les auteurs ont donc eu l'idée de transformer cette clé de re-chiffrement bidirectionnelle en une clé de re-chiffrement unidirectionnelle de la forme $sk_{\mathtt{V}}/sk_{\mathtt{A}}$, à laquelle on joint un chiffré de $sk_{\mathtt{V}}$ à destination de $\mathtt{B}$. Comme nous l'avons vu, le chiffrement du schéma [DWLC08] est basé sur un Hash Elgamal et donc les clés privées sont conçues pour ce chiffrement. Ainsi, pour ne pas modifier la clé privée $sk_{\mathtt{B}}$, les auteurs ont également chiffré cette clé privée virtuelle $sk_{\mathtt{V}}$ avec un Hash ElGamal. Cependant, par cette transformation, on voit bien qu'une coalition entre un serveur et un délégué permettrait de retrouver la clé privée $sk_{\mathtt{A}}$ et donc de déchiffrer tous les chiffrés à destination de $\mathtt{A}$. Pour éviter cela, les auteurs ont considéré deux paires de clés par utilisateur, permettant de déchiffrer chacune un type de chiffré. De cette manière, même si une

telle coalition arrivait à retrouver la clé privée utilisée pour les chiffrés re-chiffrables - puisque c'est elle qui est liée à $sk_{\mathtt{V}}$ dans la clé de re-chiffrement -, celle-ci ne permettrait pas de déchiffrer les chiffrés non re-chiffrables. Et bien que cette coalition soit capable d'obtenir cette clé privée, cela ne lui apporterait rien puisque le délégué peut déjà obtenir l'information à destination de cette clé via, un re-chiffrement.

Pour générer les deux paires de clés par utilisateur, les auteurs ont considéré une fonction de hachage supplémentaire $\mathcal{H}_4 : \mathbb{G} \to \mathbb{Z}_q^*$ qui est ajoutée aux paramètres publics. Un utilisateur $\mathtt{U}$ génère deux paires de clés (privée/publique) $(sk_{\mathtt{U},1}, pk_{\mathtt{U},1})$ et $(sk_{\mathtt{U},2}, pk_{\mathtt{U},2})$ en utilisant l'algorithme Initialiser. La clé privée globale de l'utilisateur U est en fait la concaténation de ces deux clés privées. Il en est de même pour la clé publique globale.

Le chiffrement de type 1 pour $\mathtt{A}$ se fait toujours de la même manière, sauf que l'élément à considérer comme clé publique de $\mathtt{A}$ pour ce chiffrement est $(pk_{\mathtt{A},1})^{\mathcal{H}_4(pk_{\mathtt{A},2})}.pk_{\mathtt{A},2}$. Le déchiffrement d'un tel chiffré se fait par la clé privée correspondante $sk_{\mathtt{A},1}.\mathcal{H}_4(pk_{\mathtt{A},2}) + sk_{\mathtt{A},2}$. La clé de re-chiffrement de $\mathtt{A}$ vers $\mathtt{B}$ est donc de la forme $sk_{\mathtt{V}}/(sk_{\mathtt{A},1}.\mathcal{H}_4(pk_{\mathtt{A},2}) + sk_{\mathtt{A},2})$ à laquelle on joint $V = pk_{\mathtt{B},2}^v$ et $W = (sk_{\mathtt{V}} \| \beta) \oplus \mathcal{H}_2(g^v)$ où :

- $sk_{\mathtt{V}}$ est un élément aléatoire de $\mathbb{Z}_q^*$, de taille l_0. Cela permet de le voir à la fois comme une clé privée et comme un élément appartenant à l'espace des messages $\mathcal{M}$. Cet élément peut donc être utilisé comme clé privée pour déchiffrer mais peut aussi être chiffré en V, W avec un chiffrement Hash ElGamal pour la clé publique $pk_{\mathtt{B},2}$.
- β est tiré aléatoirement parmi $\{0,1\}^{l_1}$ et v est défini comme étant $\mathcal{H}_1(sk_{\mathtt{V}} \| \beta)$.

Le re-chiffrement se passe de la même manière que pour le schéma [DWLC08] mais en utilisant cette nouvelle clé. Ce qui nous donne un re-chiffré de la forme (E', F), qui est par conséquent un chiffré pour la clé publique associée à $sk_{\mathtt{V}}$. On concatène à cette clé le chiffré Hash ElGamal (V, W) de $sk_{\mathtt{V}}$ pour la clé publique $pk_{\mathtt{B},2}$ afin que $\mathtt{B}$ puisse récupérer $sk_{\mathtt{V}}$. Lors du déchiffrement d'un tel chiffré, il faut évidemment s'assurer que celui-ci est bien formé. Cette vérification se fait en deux étapes. Une première partie, où on s'assure de la vérifiabilité de (E', F) en utilisant $sk_{\mathtt{V}}$, se déroule comme pour le schéma initial. La deuxième partie, quant à elle, s'assure de la vérifiabilité de (V, W) en utilisant $sk_{\mathtt{B},2}$, de la même façon que pour la première partie en remplaçant (E', F) par (V, W) et $sk_{\mathtt{V}}$ par $sk_{\mathtt{B},2}$.

Pour finir, l'algorithme permettant de produire directement un chiffré de type 2 - et donc non re-chiffrable - peut être décrit en utilisant la Remarque 5. Il procède alors à un chiffrement de type 1 du message pour une entité temporaire $\mathtt{T}$, puis le re-chiffre de $\mathtt{T}$ vers l'entité souhaitée.

6.1.4 Analyse de la sécurité

Les deux schémas sont annoncés comme atteignant une sécurité de type IND-CCA dans le modèle de l'oracle aléatoire sous l'hypothèse CDH. Les raisons évoquées sont que les différents chiffrés considérés sont des chiffrés Hash ElGamal, schéma de chiffrement prouvé IND-CCA. En ce qui concerne la vérifiabilité publique d'un chiffré de type 2, elle est assurée par une preuve de connaissance non interactive.

Restrictions annoncées. Le premier schéma est annoncé comme étant IND-CCA$_{\text{REST}}$ sûr. Il est en effet concerné par une restriction non naturelle et qui concerne la sécurité d'un chiffré re-chiffré de $\mathtt{A}$ vers $\mathtt{B}$, où $\mathtt{B}$ joue le rôle de l'entité cible du challenge. En effet, l'adversaire a le droit d'obtenir la clé de re-chiffrement de $\mathtt{A}$ vers $\mathtt{B}$, si et seulement si, $\mathtt{A}$ n'est pas corrompu. Ceci est dû au fait qu'à partir de la clé privée de $\mathtt{A}$ et de la clé de re-chiffrement de $\mathtt{A}$ vers $\mathtt{B}$, il est facile d'obtenir la clé privée $sk_{\mathtt{V}}$ qui permet de déchiffrer.

Commentaires sur ces schémas.

(1) Nous venons de voir que le schéma [CWYD10] ne produisait pas, lors du re-chiffrement,

un chiffré à destination exclusive du délégué, puisque le délégant est capable, à partir de la clé de re-chiffrement, d'obtenir la clé $sk_{\mathtt{V}}$ qui permet de déchiffrer. Nous nous proposons de modifier ce schéma afin que le re-chiffrement produise un chiffré destiné uniquement au délégué. En particulier, cela permet de ne plus considérer la restriction non naturelle évoquée dans la sécurité d'un chiffré re-chiffré. Pour cela, nous rendons l'algorithme de re-chiffrement probabiliste, en chiffrant en Hash ElGamal, pour la clé publique $pk_{\mathtt{B},2}$ du délégué, une partie du re-chiffré.

(2) Examinons maintenant les sécurités de ce schéma concernant les chiffrés de type 2 - cela inclut celle concernée par le point précédent. En analysant les preuves de sécurité correspondantes, exposées dans l'article, nous nous apercevons qu'elle comporte d'autres restrictions non naturelles, contrairement à ce qui est annoncé par les auteurs. En effet, certaines clés de re-chiffrement, notamment celles de l'entité cible du challenge vers une entité corrompue, ne peuvent être données à l'adversaire. Pourtant dans une telle sécurité, il n'y a aucune raison de ne pas le faire puisque le challenge n'est pas re-chiffrable. Nous nous proposons de fournir une preuve de sécurité différente qui va permettre de corriger les restrictions observées dans celle-ci.

(3) Un troisième point, concernant toujours le schéma de Chow *et al.*, porte sur la manière de générer les clés des différents utilisateurs. En effet, les auteurs utilisent comme clé privée, pour un chiffré de type 2, une clé de la forme $sk_{\mathtt{U},1}.\mathcal{H}_4(pk_{\mathtt{U},2}) + sk_{\mathtt{U},2}$. Cette combinaison linéaire empêche un individu de retrouver, à partir de celle-ci, les clés privées $sk_{\mathtt{U},1}$ et $sk_{\mathtt{U},2}$. Ceci est dû au fait qu'il y a autant de solutions possibles à l'équation $c = x.\mathcal{H}_4(pk_{\mathtt{U},2})+y$, où c est n'importe quelle constante, que le cardinal de G, groupe auquel appartiennent les clés privées. Nous remarquons que la clé privée $sk_{\mathtt{U},1}$ n'est utilisée nulle part ailleurs. Pour cette raison, nous considérons tout simplement $sk_{\mathtt{U},1}$ comme clé privée pour un chiffré de type 1 à la place de $sk_{\mathtt{U},1}.\mathcal{H}_4(pk_{\mathtt{U},2}) + sk_{\mathtt{U},2}$. Cette modification permet de simplifier la génération des différentes clés privées dans les preuves de sécurité. Ainsi, les calculs à exécuter sont réduits et il est inutile de considérer la fonction de hachage $\mathcal{H}_4$.

(4) Pour terminer, nous nous intéressons à la vérifiabilité publique d'un chiffré et, plus particulièrement, à la preuve de connaissance utilisée. Cette remarque ne concerne, cette fois-ci, pas que le schéma [CWYD10] mais également le schéma [DWLC08] [2]. En observant les algorithmes de re-chiffrement, nous nous apercevons que la vérifiabilité publique des chiffrés repose entièrement sur cette preuve de connaissance, preuve qui est également prise en compte lors du déchiffrement d'un chiffré de type 1. À la différence près, que lors de ce déchiffrement, il y a un test supplémentaire à effectuer une fois le déchiffrement effectué, pour s'assurer que le chiffré est valide : est-ce que $E = g^{sk.\mathcal{H}_1(m\|\omega)}$? Ce test s'assure que l'aléa utilisé pour masquer le message lors du chiffrement est bien de la forme $\mathcal{H}_1(m\|\omega)$. La preuve de connaissance, quant à elle, ne concerne pas ce test. Ceci signifie qu'elle ne permet pas de conclure si un chiffré est bien formé ou non, en tout cas pas entièrement.
Pour autant, il ne semble pas possible d'exploiter cette faille pour casser directement l'indistinguabilité de messages chiffrés pour ces deux schémas. Il est en revanche très facile de l'utiliser pour démontrer que les preuves de sécurité, données pour ces schémas par leurs auteurs, ne conviennent pas. En particulier, nous sommes capables de concevoir des requêtes de re-chiffrement pour lesquelles l'oracle de re-chiffrement n'est pas en mesure de répondre correctement. Il suffit pour cela de concevoir un chiffré, destiné à l'entité cible de la sécurité considérée et re-chifffrable, de telle manière que l'aléa utilisé r ne soit pas calculé comme un haché $\mathcal{H}_1(m\|\omega)$ mais comme un véritable aléa. Un tel chiffré ne peut pas être re-chiffré par l'oracle de re-chiffrement, tel que décrit dans les preuves de sécurité. Ceci implique que tout

2. Cette remarque n'est pas propre à ces schémas. Cette preuve étant bien adaptée au chiffrement Hash ElGamal, il est possible que tout schéma de serveur de re-chiffrement, basé sur ce chiffrement, l'utilise.

adversaire effectuant cette requête ne sera pas pris en compte par la preuve de sécurité. Pour rectifier ce problème, il faudrait être capable de disposer d'une vérifiabilité publique complète d'un chiffré. Il est certainement possible de concevoir théoriquement une telle preuve, au détriment de l'efficacité du schéma, comme nous l'avons fait au Chap 5 en utilisant une preuve NIZK. Pour autant, on peut se limiter à cette vérifiabilité publique incomplète d'un chiffré. En effet, la faille ne semble pas mettre en danger les clés privées des utilisateurs ou les messages chiffrés. Il suffit, dans les preuves de sécurité, de pouvoir récupérer, pour ces requêtes qui posent problème, les aléas r utilisés afin de re-chiffrer correctement. Il faut, pour cela, être capable d'extraire l'aléa utilisé. Cette extraction est possible en utilisant des techniques de rembobinage telles que décrites dans [PS00], avec l'inconvénient que la réduction offerte par la preuve de sécurité ne soit alors plus polynomiale. Lors de la création de la signature, une demande à l'oracle aléatoire est nécessaire afin d'obtenir $h_1 = \mathcal{H}_3(D\|E\|m)$ et produire une signature s_1. En considérant le même adversaire sur le même scénario, l'oracle aléatoire répond par $h_2 = \mathcal{H}_3(D\|E\|m)$ (en le choisissant différent de h_1) pour produire une signature s_2. Il est alors possible d'utiliser ces deux scénarios pour retrouver le secret r utilisé : $r = \frac{s_1 - s_2}{h_1 - h_2}$. Le problème est que, pour k signature à produire, il faut considérer 2^k scénarios ce qui implique le temps du scénario (puisque l'on est obligé d'attendre que les autres scénarios se déroulent aussi pour pouvoir extraire ces données). La réduciton que l'on obtient fait alors apparaître un facteur exponentiel. Nous proposons d'utiliser une preuve de connaissance légèrement différente. Celle-ci permet toujours de démontrer la connaissance d'un logarithme discret, mais en plus d'extraire en direct le secret quand on en a besoin, sans faire appel aux techniques de rembobinage. Nous détaillons ce type de preuve dans la section suivante.

6.2 Un nouveau schéma

6.2.1 Preuve de connaissance avec extracteur

Comme nous venons de le voir, nous avons besoin d'une preuve de connaissance permettant d'extraire le secret lors de certaines requêtes effectuées à l'oracle de re-chiffrement. Pour cela, nous utilisons la construction présentée par Fischlin dans [Fis05]. Celle-ci permet de transformer une preuve de connaissance non interactive, prouvant la connaissance de logarithmes discrets en une même preuve offrant en plus la possibilité d'extraire le secret en direct via l'utilisation de l'oracle aléatoire.

En prenant comme base la preuve de connaissance non-interactive à la Schnorr, ceci nous donne la preuve de connaissance énoncée ci-après [3]. Pour rappel, le but est de prouver la connaissance d'un logarithme discret r dans une relation du type $E = T^r$ en signant un message m, où E et T sont publics. Pour cela, nous employons certains paramètres utilisés dans les schémas que l'on vient de présenter, comme λ, r, q et $\mathcal{H}_3$, à la différence près que la fonction de hachage $\mathcal{H}_3$ sera désormais définie de $\{0,1\}^*$ dans $\{0,1\}^u$ où $u = O(\log(\lambda))$. Soit $t = O(\log(\lambda))$ tel que $t \geq u$ et $2^t \leq p$. Pour effectuer une preuve, nous choisissons l engagements $\mathtt{eng}_1, \ldots, \mathtt{eng}_l \in \mathbb{Z}_p^*$, où $l = O(\log(\lambda))$ et nous calculons $T_k = T^{\mathtt{eng}_k}$, $\forall k \in [\![1; l]\!]$. Ensuite, pour chaque $k \in [\![1; l]\!]$, nous calculons $\mathtt{rep}_k = \mathtt{eng}_k + r.\mathtt{ch}_k$, où $\mathtt{ch}_k$ est un challenge parcourant $[\![1; 2^t]\!]$ et $\mathtt{rep}_k$ est la réponse à ce challenge, jusqu'à trouver un couple (challenge/réponse) $(\mathtt{ch}_k, \mathtt{rep}_k)$ tel que $\mathcal{H}_3(E\|T_1\| \ldots \|T_l\|k\|\mathtt{ch}_k\|\mathtt{rep}_k\|m)$ soit égal au u-uplet nul de $\{0,1\}^u$. Si aucun couple n'est trouvé, alors nous choisissons le couple $(\mathtt{ch}_k, \mathtt{rep}_k)$ pour lequel la valeur de la fonction de hachage $\mathcal{H}_3$ est la plus petite possible [4]. La preuve de connaissance consiste alors en $(T_k, \mathtt{ch}_k, \mathtt{rep}_k)_{k=1,\ldots,l}$. Afin de simplifier la description de notre schéma, nous notons cette

3. Le lecteur souhaitant obtenir plus de détails sur le choix de certains paramètres est invité à se référer à [Fis05].
4. On peut toujours voir l'ensemble $\{0,1\}^u$ comme la représentation en bits de $[\![0, 2^u - 1]\!]$.

exécution : $(T_k, \mathtt{ch}_k, \mathtt{rep}_k)_{k=1,\dots,l} = \text{NIZKOE}(r : E = T^r)(m)$.

La procédure de vérification d'une telle preuve consiste à s'assurer que, pour chaque $k \in [\![1; l]\!]$, $T^{\mathtt{rep}_k} = T_k.E^{\mathtt{ch}_k}$ et que $\sum_{k=1}^{l} \mathcal{H}_3(E\|T_1\| \dots \|T_l\|k\|\mathtt{ch}_k\|\mathtt{rep}_k\|m) \leq S$ où $S = O(l)$. Nous dirons alors que la preuve est valide pour m sous E et T.

L'extraction en direct est rendue possible du fait que l'on force $\mathtt{ch}_k$ à parcourir $[\![1; 2^t]\!]$ jusqu'à trouver une valeur de hachage de $\mathcal{H}_3$ qui convienne. De cette manière, si l'on dispose d'une preuve $(T_k, \mathtt{ch}_k, \mathtt{rep}_k)_{k=1,\dots,l}$ valide, alors il existe, avec une probabilité non négligeable, $k \in [\![1; l]\!]$ pour lequel une requête $(E\|T_1\| \dots \|T_l\|k\|\mathtt{ch}'_k\|\mathtt{rep}'_k\|m)$ a été faite à l'oracle aléatoire $\mathcal{H}_3$ avec $\mathtt{ch}_k \neq \mathtt{ch}'_k$ et tel que $T^{\mathtt{rep}'_k} = T_k.E^{\mathtt{ch}'_k}$. On se retrouve ainsi avec un système à deux équations à deux inconnues, à savoir $\mathtt{eng}_k$ et r, de déterminant $\mathtt{ch}_k - \mathtt{ch}'_k \neq 0$, ce qui permet de retrouver $r = \frac{\mathtt{rep}_k - \mathtt{rep}'_k}{\mathtt{ch}_k - \mathtt{ch}'_k}$.

6.2.2 Description du nouveau schéma

La seule modification à apporter sur le schéma [DWLC08] est de remplacer la preuve de connaissance initialement utilisée par celle de Fischlin qui offre une extraction en direct du secret. Pour cette raison, nous ne la présentons pas et nous contentons de présenter le schéma [CWYD10] modifié. Nous apportons sur ce dernier toutes les modifications énoncées ci-dessus. Les différents paramètres publics utilisés ainsi que l'environnement restent inchangés et ne sont pas reprécisés.

Clé($\mathcal{P}$) : prend en entrée les paramètres publics, tire aléatoirement $sk = (sk_1, sk_2) \xleftarrow{\$} (\mathbb{Z}_q^*)^2$ et définit $pk = (g^{sk_1}, g^{sk_2})$.

Délégation$_{1,2}(\mathcal{P}, (sk_{\mathtt{A},1}, pk_{\mathtt{A},1}), pk_{\mathtt{B},2})$: prend en entrée les paramètres publics, la clé privée $sk_{\mathtt{A},1}$ de $\mathtt{A}$ ainsi que la clé publique associée $pk_{\mathtt{A},1}$ et la clé publique $pk_{\mathtt{B},2}$ de $\mathtt{B}$. La clé de re-chiffrement est calculée de la manière suivante :

1. Tire aléatoirement $sk_{\mathtt{V}} \in \mathbb{Z}_q^*$ tel que sa représentation en bits soit un élément de $\{0,1\}^{l_0}$ et calcule $\tilde{R}_{\mathtt{A},1\to B,2} = sk_{\mathtt{V}}/sk_{\mathtt{A},1}$.
2. Chiffre $sk_{\mathtt{V}}$ grâce au chiffrement Hash ElGamal pour $pk_{\mathtt{B},2}$: tire aléatoirement $\beta \in \{0,1\}^{l_1}$ et calcule $v = \mathcal{H}_1(sk_{\mathtt{V}}\|\beta)$, $V = pk_{\mathtt{B},2}^v$ et $W = (sk_{\mathtt{V}}\|\beta) \oplus \mathcal{H}_2(g^v)$.
3. Retourne la clé de re-chiffrement : $R_{\mathtt{A},1\to\mathtt{B},2} = (\tilde{R}_{\mathtt{A},1\to\mathtt{B},2}, V, W, pk_{\mathtt{A},1}, pk_{\mathtt{B},2})$.

Chiffrer$_1(\mathcal{P}, pk, m)$: prend en entrée les paramètres publics, une clé publique $pk = (pk_1, pk_2)$ et un message $m \in \mathcal{M}$. Cet algorithme fonctionne ainsi :

1. Chiffre m en Hash ElGamal : tire aléatoirement $\omega \xleftarrow{\$} \{0,1\}^{l_1}$ et calcule $r = \mathcal{H}_1(m\|\omega)$, $E = pk_1^r$ et $F = (m\|\omega) \oplus \mathcal{H}_2(g^r)$.
3. Définit $(T_k, \mathtt{ch}_k, \mathtt{rep}_k)_{k=1,\dots,l} = \text{NIZKOE}(r : E = pk_1^r)(F)$.
4. Retourne le chiffré de type 1 : $C_{pk,1} = (E, F, (T_k, \mathtt{ch}_k, \mathtt{rep}_k)_{k=1,\dots,l})$.

Déchiffrer$_1(\mathcal{P}, sk, C_{pk,1})$: prend en entrée les paramètres publics, une clé privée $sk = (sk_1, sk_2)$ et un chiffré $C_{pk,1} = (E, F, (T_k, \mathtt{ch}_k, \mathtt{rep}_k)_{k=1,\dots,l})$ de type 1 pour pk. Cet algorithme déchiffre comme suit :

1. Vérifie si la preuve $(T_k, \mathtt{ch}_k, \mathtt{rep}_k)_{k=1,\dots,l}$ est valide pour F sous E et pk_1 ; retourne $\perp$ et s'arrête là, si ce n'est pas le cas.
2. Déchiffre (E, F) en utilisant sk_1 : calcule $(m\|\omega) = F \oplus \mathcal{H}_2(E^{1/sk_1})$, vérifie si $E = g^{sk.\mathcal{H}_1(m\|\omega)}$ et retourne m, ou $\perp$ si ce n'est pas le cas.

Re-chiffrer$_{1,2}(\mathcal{P}, R_{\mathtt{A},1\to\mathtt{B},2}, C_{\mathtt{A},1})$: prend en entrée les paramètres publics $\mathcal{P}$, une clé de re-chiffrement $R_{\mathtt{A},1\to\mathtt{B},2} = (\tilde{R}_{\mathtt{A},1\to\mathtt{B},2}, V, W, pk_{\mathtt{A},1}, pk_{\mathtt{B},2})$ de $\mathtt{A}$ vers $\mathtt{B}$, un chiffré de type 1 pour $\mathtt{A}$: $C_{\mathtt{A},1} = (E, F, (T_k, \mathtt{ch}_k, \mathtt{rep}_k)_{k=1,\dots,l})$. Cet algorithme re-chiffre de la manière suivante :

1. Vérifie si $(T_k, \mathtt{ch}_k, \mathtt{rep}_k)_{k=1,\dots,l}$ est valide pour F sous E et $pk_{\mathtt{A},1}$; retourne $\perp$ et s'arrête là, si ce n'est pas le cas.
2. Calcule $E' = E^{\tilde{R}_{\mathtt{A},1\to\mathtt{B},2}} = (g^{r.sk_\mathtt{A}})^{(sk_\mathtt{V}/sk_\mathtt{A})} = g^{r.sk_\mathtt{V}}$.
3. Tire aléatoirement $z \in \mathbb{Z}_q^*$ tel que sa représentation en bits soit un élément de $\{0,1\}^{l_0}$ et calcule $F' = \mathcal{H}_2(g^z) \oplus F$.
4. Chiffre z en Hash ElGamal pour $pk_{\mathtt{B},2}$: tire aléatoirement $\alpha \in \{0,1\}^{l_1}$ et calcule $x = \mathcal{H}_1(z\|\alpha)$, $X = pk_{\mathtt{B},2}^x$ et $Y = (z\|\alpha) \oplus \mathcal{H}_2(g^x)$.
5. Retourne le chiffré de type 2 : $C_{\mathtt{B},2} = (E', F', V, W, X, Y)$.

$\mathsf{Chiffrer}_2(\mathcal{P}, pk, m)$: prend en entrée les paramètres publics $\mathcal{P}$, une clé publique $pk = (pk_1, pk_2)$ et un message $m \in \mathcal{M}$. Cet algorithme peut être décrit via l'utilisation d'une entité temporaire telle que décrite dans la Remarque 5. Il est pour autant possible de le définir sans utiliser une succession de l'algorithme Chiffrer pour le type 1 et de l'algorithme Re-chiffrer tout en obtenant le même chiffré. Ceci permet notamment d'accélérer les calculs à effectuer, puisque, par exemple, il ne nécessite plus le calcul d'une preuve de connaissance. Il peut être décrit ainsi :

1. Tire aléatoirement $sk_\mathtt{V} \in \mathbb{Z}_q^*$ tel que sa représentation en bits soit un élément de $\{0,1\}^{l_0}$.
2. Chiffre m en Hash ElGamal pour $g^{sk_\mathtt{V}}$: tire aléatoirement $\omega \xleftarrow{\$} \{0,1\}^{l_1}$ et calcule $r = \mathcal{H}_1(m\|\omega)$, $E' = g^{r.sk_\mathtt{V}}$ et $F = \mathcal{H}_2(g^r) \oplus (m\|\omega)$.
3. Chiffre $sk_\mathtt{V}$ en Hash ElGamal pour pk_2 : tire aléatoirement $\beta \in \{0,1\}^{l_1}$ et calcule $v = \mathcal{H}_1(sk_\mathtt{V}\|\beta)$, $V = pk_2^v$ et $W = (sk_\mathtt{V}\|\beta) \oplus \mathcal{H}_2(g^v)$.
4. Tire aléatoirement $z \in \mathbb{Z}_q^*$ tel que sa représentation en bits soit un élément de $\{0,1\}^{l_0}$ et calcule $F' = \mathcal{H}_2(g^z) \oplus F$.
5. Chiffre z en Hash ElGamal pour pk_2 : tire aléatoirement $\alpha \in \{0,1\}^{l_1}$ et calcule $x = \mathcal{H}_1(z\|\alpha)$, $X = pk_2^x$ et $Y = (z\|\alpha) \oplus \mathcal{H}_2(g^x)$.
6. Retourne le chiffré de type 2 : $C_{\mathtt{B},2} = (E', F', V, W, X, Y)$.

$\mathsf{Déchiffrer}_2(\mathcal{P}, sk, C_{pk,2})$: prend en entrée les paramètres publics, une clé privée $sk = (sk_1, sk_2)$ et un chiffré $C_{pk,2} = (E', F', V, W, X, Y)$ de type 2. Cet algorithme déchiffre comme suit :

1. Déchiffre (X, Y) pour obtenir z en utilisant sk_2 : calcule $(z\|\alpha) = Y \oplus \mathcal{H}_2(X^{1/sk_2})$. Si $X \neq g^{sk_2.\mathcal{H}_1(z\|\alpha)}$, alors l'algorithme retourne $\perp$ et s'arrête.
2. Calcule $F = F' \oplus \mathcal{H}_2(g^z)$.
3. Déchiffre (V, W) pour obtenir $sk_\mathtt{V}$ en utilisant sk_2 : calcule $(sk_\mathtt{V}\|\beta) = W \oplus \mathcal{H}_2(V^{1/sk_2})$. Si $V \neq g^{sk_2.\mathcal{H}_1(sk_\mathtt{V}\|\beta)}$, alors l'algorithme retourne $\perp$ et s'arrête.
4. Déchiffre (E', F) pour obtenir m en utilisant sk_V : calcule $(m\|\omega) = F \oplus \mathcal{H}_2(E'^{1/sk_\mathtt{V}})$. Si $E \neq g^{sk_2.\mathcal{H}_1(m\|\omega)}$, alors l'algorithme retourne $\perp$ et s'arrête.
5. Retourne finalement m.

Consistance. La consistance est facile à vérifier.

6.2.3 Sécurité du schéma

Nous nous intéressons maintenant à la sécurité de ce schéma. D'après l'environnement de ce schéma, nous voyons qu'il y a trois manières de produire un chiffré, à savoir : un chiffré de type 1, un chiffré de type 2 rechiffré et un chiffré de type 2 non rechiffré. Il y a donc trois sécurités à regarder : une pour chaque manière de produire un chiffré.

Les preuves permettant de prouver ces trois sécurités sont intéressantes car elles permettent de prendre conscience des problèmes que l'on peut rencontrer pour étudier la sécurité d'un schéma de serveur de re-chiffrement. Il est naturel, lors de la conception d'un schéma, de considérer que l'utilisation de différentes techniques cryptographiques, connues et étudiées [5], vont permettre

5. Comme par exemple le chiffrement Hash ElGamal.

d'obtenir un schéma sécurisé, sans avoir besoin d'examiner en détail la sécurité sous-jacente. C'est le cas pour certaines sécurités dont on peut se convaincre facilement qu'elles seront vérifiées. Les sécurités étudiées dans les Théorèmes 32 et 33 en sont un bon exemple. En revanche, il y en a certaines pour lesquelles ce n'est pas aussi simple. Il faut alors effectuer une preuve de sécurité minutieuse afin d'être certain de prendre en considération toutes les restrictions naturelles et non naturelles susceptibles d'intervenir. C'est le cas de la sécurité étudiée dans le Théorème 34. Nous présentons désormais ces différents théorèmes.

Théorème 32. *Le schéma atteint la sécurité* IND-CCA *pour un chiffré challenge re-chiffré, sous l'hypothèse* CDH *dans le modèle de l'oracle aléatoire.*

Preuve informelle. La manière la plus simple de prouver ce théorème est d'effectuer une réduction permettant de ramener cette sécurité à celle du chiffrement Hash ElGamal, qui est également prouvé IND-CCA sous l'hypothèse CDH. En effet, quand on observe l'algorithme de re-chiffrement, on s'aperçoit que l'on cumule, dans un même chiffré, plusieurs chiffrés Hash ElGamal. En particulier, dans un re-chiffré $C_{pk,2} = (E', F', V, W, X, Y)$, (X, Y) est un chiffré Hash ElGamal de l'aléa z pour la clé publique pk_2. C'est cet aléa qui fait que le déchiffrement n'est possible que pour le délégué, puisque $F' = (m\|\omega) \oplus \mathcal{H}_2(g^r) \oplus \mathcal{H}_2(g^z)$. Ainsi pour pouvoir distinguer à partir du chiffré, si F' masque un message m_0 ou un message m_1, il faut obligatoirement pouvoir récupérer cet aléa z afin d'obtenir $\mathcal{H}_2(g^z)$ qui masque parfaitement le message chiffré. L'aléa n'intervenant que dans F' et (X, Y), il faut, pour le récupérer, casser le chiffré Hash ElGamal (X, Y). Il suffit alors d'introduire la sécurité du chiffrement Hash ElGamal sur ce chiffré (X, Y). Cette réduction est facile à concevoir et possible car la clé sk_2 qui permet de le déchiffrer, n'intervient pas dans les clés de re-chiffrement. Ce n'est, en revanche, pas le cas pour la clé sk_1, qui elle intervient dans les clés de re-chiffrement via les quotients de la forme $sk_{\mathtt{V}}/sk_1$. Pour autant, nous avons modifié le schéma pour que ces deux clés soient indépendantes. Ainsi, sk_1 ne dévoile aucune information sur sk_2. Il n'y a, de ce fait, aucune restriction non naturelle à considérer sur la partie re-chiffrement et, de par la sécurité du chiffrement Hash ElGamal, aucune restriction non naturelle à considérer sur l'oracle de déchiffrement. □

Théorème 33. *Le schéma atteint la sécurité* IND-CCA *pour un chiffré challenge de type 2 non rechiffré sous l'hypothèse* CDH *dans le modèle de l'oracle aléatoire.*

Preuve informelle. Nous énonçons, une fois de plus, uniquement l'idée permettant de démontrer le théorème. La description de l'algorithme $\mathsf{Chiffrer}_1$ est très importante ici. Comme nous l'avons évoqué, un chiffré non rechiffré de type 1 pour `B` correspond exactement à un chiffré issu d'un re-chiffrement d'une entité temporaire `T` vers `B`. Un chiffré challenge pour cette sécurité peut alors être vu comme un re-chiffré challenge pour la sécurité concernée par le théorème précédent. Il est donc possible de prouver ce théorème en effectuant une réduction au théorème précédent, qui ne considère aucune restriction sur le déléguant, rôle assumé ici par l'entité temporaire `T`. D'où l'énoncé de ce théorème. □

Théorème 34. *Le schéma atteint la sécurité* IND-CCA *pour un chiffré challenge de type 1 sous l'hypothèse* CDH *dans le modèle de l'oracle aléatoire.*

Preuve. Nous ne pouvons pas simplement nous rapporter à la sécurité du chiffrement Hash ElGamal comme nous l'avons fait dans les théorèmes précédents, puisqu'ici la clé sk_1, qui permet de déchiffrer un chiffré de type 1, intervient dans les clés de re-chiffrement via $sk_{\mathtt{V}}/sk_1$. Pour prouver ce théorème, nous détaillons un algorithme $\mathcal{B}$ qui va utiliser un adversaire $\mathcal{A}$, attaquant la sécurité du schéma, pour casser l'hypothèse DCDH qui, pour rappel, est équivalente à l'hypothèse CDH.

Soit $(A = g^a, B = g^b)$ une instance DCDH, le but de $\mathcal{B}$ est de calculer $g^{b/a}$.

Phase d'initialisation. $\mathcal{B}$ commence par choisir les éléments $(l_0, l_1, l, t, u, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3)$ comme paramètres publics. Les entiers l, t, u et la fonction de hachage $\mathcal{H}_3$ sont définis comme dans la preuve de connaissance avec extracteur. Les autres éléments sont, quant à eux, définis comme dans le schéma originel.

Cet algorithme maintient trois listes $\mathcal{K}^{\mathsf{list}}$, $\mathcal{R}^{\mathsf{list}}$ et $\mathcal{C}^{\mathsf{list}}$ initialement vides et qui vont stocker respectivement : la liste des paires de clés (privée/publique), la liste des clés de re-chiffrement et une liste de chiffrés particuliers et de leurs re-chiffrés.

Les clés des différentes entités sont générées de la manière suivante.

Clé : $\mathcal{B}$ génère les clés publiques en tirant aléatoirement $(sk_1, sk_2) \in \mathbb{Z}_q^* \times \mathbb{Z}_q^*$.

- si l'entité concernée est l'entité cible, il calcule $pk^\star = (pk_1^\star, pk_2^\star) = (A^{sk_1}, g^{sk_2})$, ce qui définit implicitement $sk^\star = (sk_1^\star, sk_2^\star) = (a.sk_1, sk_2)$, et ajoute $((\perp, sk_2^\star), pk^*)$ à $\mathcal{K}^{\mathsf{list}}$. $\mathcal{B}$ stocke en mémoire sk_1 qui sera utilisé par la suite et que l'on notera $\tilde{sk}_1^\star$.
- sinon, il définit $sk = (sk_1, sk_2)$, calcule $pk = (g^{sk_1}, g^{sk_2})$ et ajoute (sk, pk) à $\mathcal{K}^{\mathsf{list}}$.

Pour la suite, nous avons besoin de décrire la modélisation par $\mathcal{B}$ des oracles aléatoires $\mathcal{H}_1$, $\mathcal{H}_2$ et $\mathcal{H}_3$. Pour cela, il maintient trois listes $\mathcal{H}_1^{\mathsf{list}}$, $\mathcal{H}_2^{\mathsf{list}}$ et $\mathcal{H}_3^{\mathsf{list}}$ qui sont initialement vides et répond aux requêtes de $\mathcal{A}$ à l'oracle aléatoire comme suit.

- $(m\|\omega)$ est demandé à $\mathcal{H}_1$: s'il existe une entrée $(m\|\omega, r)$ dans $\mathcal{H}_1^{\mathsf{list}}$, alors retourne r, sinon tire aléatoirement $r \xleftarrow{\$} \mathbb{Z}_q^*$, ajoute le couple $(m\|\omega, r)$ à $\mathcal{H}_1^{\mathsf{list}}$ et retourne r.
- R est demandé à $\mathcal{H}_2$: s'il existe une entrée (R, S) dans $\mathcal{H}_2^{\mathsf{list}}$, alors retourne S, sinon tire aléatoirement $S \xleftarrow{\$} \{0,1\}^{l_0+l_1}$, ajoute l'upplet (R, S) à $\mathcal{H}_2^{\mathsf{list}}$ et retourne S.
- $(E\|T_1\| \ldots \|T_l\|k\|\mathtt{ch}_k\|\mathtt{rep}_k\|F)$ est demandé à $\mathcal{H}_3$: s'il existe l'upplet $(E\|T_1\| \ldots \|T_l\|k\|\mathtt{ch}_k\|\mathtt{rep}_k\|F, c)$, comme entrée dans $\mathcal{H}_3^{\mathsf{list}}$, alors retourne c. Sinon, $\mathcal{B}$ ne peut pas simplement tirer un élément aléatoire et le renvoyer comme dans les modélisations précédentes. Comme expliqué dans [Fis05], il faut que $\mathcal{B}$ soit capable d'utiliser cet oracle aléatoire pour simuler correctement le challenge qui va être donner à $\mathcal{A}$. Pour cela, $\mathcal{B}$ doit commencer à préparer le chiffré challenge qu'il donnera à $\mathcal{A}$ avant de définir la modélisation de $\mathcal{H}_3$ car cette dernière dépendra de ce chiffré.

  Préparation du challenge : $\mathcal{B}$ calcule $E^\star = B^{\tilde{sk}_1^\star}$ et choisit $F^\star$, aléatoirement dans $\{0,1\}^{l_0+l_1}$. Le couple $(E^\star, F^\star)$ correspondra, dans la phase de Challenge, à un chiffré Hash Elgamal de m_δ. Ensuite, il prépare la preuve de connaissance, qu'il utilisera pour ce chiffré, en choisissant pour chaque $k \in [\![1; l]\!]$, une application $\tau_k : [\![1; 2^t]\!] \to \{0,1\}^u$. Il considère $\mathtt{ch}_k^\star$, l'élément pour lequel τ_k atteint son minimum. Puis, il tire aléatoirement $\mathtt{rep}_k^\star \in \mathbb{Z}_q^*$ et calcule $T_k^\star = A^{\mathtt{rep}_k^\star.\tilde{sk}_1^\star}.B^{-\mathtt{ch}_k^\star.\tilde{sk}_1^\star}$. Ceci définit implicitement $\mathtt{eng}_k^\star = \mathtt{rep}_k^\star - \frac{b}{a}.\mathtt{ch}_k^\star$. Il définit $C^\star = (E^\star, F^\star, (T_k^\star, \mathtt{ch}_k^\star, \mathtt{rep}_k^\star)_{k=1,\ldots,l})$, qui sera donné plus tard à $\mathcal{A}$ comme chiffré challenge.

  $\mathcal{B}$ modélise alors l'oracle aléatoire $\mathcal{H}_3$ en fonction de ce chiffré challenge et des applications $\tau_1, \ldots, \tau_l$. D'après l'exécution de cette preuve NIZKOE dans l'algorithme $\mathsf{Chiffrer}_1$, il ne faut pas que $\mathcal{H}_3$ réponde à une requête $(E^\star, T_1^\star, \ldots, T_l^\star, k, \mathtt{ch}_k, \mathtt{rep}_k, F^\star)$, telle que $(\mathtt{ch}_k, \mathtt{rep}_k) \neq (\mathtt{ch}_k^\star, \mathtt{rep}_k^\star)$ et $(E^\star)^{\mathtt{rep}_k} = T_k^\star.(pk_1^\star)^{\mathtt{ch}_k}$, par une valeur plus petite que celle retournée à la requête $(E^\star, T_1^\star, \ldots, T_l^\star, k, \mathtt{ch}_k^\star, \mathtt{rep}_k^\star, F^\star)$. Il répond à ces requêtes en utilisant l'application τ_k et en définissant $(E^\star, T_1^\star, \ldots, T_l^\star, k, \mathtt{ch}_k, \mathtt{rep}_k, F^\star) = \tau_k(\mathtt{ch}_k)$, ce qui assure la minimalité pour le couple $(\mathtt{ch}_k^\star, \mathtt{rep}_k^\star)$. $(E^\star, T_1^\star, \ldots, T_l^\star, k, \mathtt{ch}_k, \mathtt{rep}_k, F^\star, \tau_k(\mathtt{ch}_k))$ est ajouté à $\mathcal{H}_3^{\mathsf{list}}$. Pour toute autre requête $(E\|T_1\| \ldots \|T_l\|k\|\mathtt{ch}_k\|\mathtt{rep}_k\|F)$, il tire aléatoirement $c \xleftarrow{\$} \{0,1\}^u$, ajoute l'upplet $(E\|T_1\| \ldots \|T_l\|k\|\mathtt{ch}_k\|\mathtt{rep}_k\|F, c)$ à $\mathcal{H}_3^{\mathsf{list}}$ et retourne c.

Phase 1. $\mathcal{B}$ répond aux différentes requêtes de $\mathcal{A}$ de la manière suivante.

Délégation$(pk_{\mathtt{A}}, pk_{\mathtt{B}})$: Il commence par récupérer les entrées correspondantes à $pk_{\mathtt{A}}$ et $pk_{\mathtt{B}}$ dans

la liste $\mathcal{K}^{\mathsf{list}}$ afin d'obtenir $(sk_{\mathtt{A}}, pk_{\mathtt{A}})$ et $(sk_{\mathtt{B}}, pk_{\mathtt{B}})$. Il génère la clé de re-chiffrement de $\mathtt{A}$ vers $\mathtt{B}$ de la manière suivante :

- si $sk_{\mathtt{A}} = (\perp, sk_{\mathtt{A},2})$, alors $\mathcal{B}$ génère une fausse clé de re-chiffrement puisqu'il ne connaît pas $sk_{A,1}$. Pour cela, il tire aléatoirement $\tilde{R}_{\mathtt{A},1\to\mathtt{B},2} \xleftarrow{\$} \mathbb{Z}_q^*$ ainsi que $sk_{\mathtt{V}} \xleftarrow{\$} \mathbb{Z}_q^*$, tel que sa représentation en bits soit un élément de $\{0,1\}^{l_0}$. Ensuite, il calcule un chiffré Hash ElGamal de $sk_{\mathtt{V}}$ pour $pk_{B,2}$ et obtient (V, W). Pour finir, il définit la clé de re-chiffrement comme $R_{\mathtt{A},1\to\mathtt{B},2} = (\tilde{R}_{\mathtt{A},1\to\mathtt{B},2}, V, W)$ et ajoute $(pk_A, pk_B, \tilde{R}_{\mathtt{A},1\to\mathtt{B},2}, sk_{\mathtt{V}}, V, W)$ à $\mathcal{R}^{\mathsf{list}}$. Dans le cas où $\mathtt{B}$ est corrompu, $\mathcal{B}$ retourne $\perp$ à $\mathcal{A}$ puisque ce dernier n'a pas le droit d'obtenir cette clé.
- sinon, $\mathcal{B}$ effectue les mêmes calculs que dans l'algorithme Délégation et considère le résultat comme la clé de re-chiffrement. Il ajoute $(pk_A, pk_B, \tilde{R}_{\mathtt{A},1\to\mathtt{B},2}, sk_{\mathtt{V}}, V, W)$ à $\mathcal{R}^{\mathsf{list}}$ où $sk_{\mathtt{V}}$ est un élément tiré aléatoirement lors de la génération de cette clé.

ORe-chiffrer$(pk_{\mathtt{A}}, pk_{\mathtt{B}}, C_{\mathtt{A},1})$: Il commence à répartir le chiffré $C_{\mathtt{A},1}$ en $(E, F, (T_k, \mathtt{ch}_k, \mathtt{rep}_k)_{k=1..l})$ et $pk_{\mathtt{A}} = (pk_{\mathtt{A},1}, pk_{\mathtt{A},2})$. Il vérifie ensuite la preuve de connaissance $(T_k, \mathtt{ch}_k, \mathtt{rep}_k)_{k=1..l}$ pour E et $pk_{\mathtt{A},1}$ et retourne $\perp$ si elle n'est pas valide. Sinon, il récupère les entrées (sk_A, pk_A) et (sk_B, pk_B) dans la liste $\mathcal{K}^{\mathsf{list}}$ ainsi que l'entrée $(pk_A, pk_B, \tilde{R}_{\mathtt{A},1\to\mathtt{B},2}, sk_{\mathtt{V}}, V, W)$ dans la liste $\mathcal{R}^{\mathsf{list}}$. Il exécute Délégation$(pk_{\mathtt{A}}, pk_{\mathtt{B}})$ décrit précédemment si une telle entrée n'existe pas. Il procède ensuite selon différents cas.

- si $sk_{\mathtt{A}} = (\perp, sk_{\mathtt{A},2})$ et si $\mathtt{B}$ est honnête, il exécute l'algorithme Re-chiffrer$_{1,2}$ en utilisant la clé de re-chiffrement $(\tilde{R}_{\mathtt{A},1\to\mathtt{B},2}, sk_{\mathtt{V}}, V, W)$. Il obtient ainsi un chiffré $C_{\mathtt{B},2}$ qui est donné à $\mathtt{A}$. Dans ce cas, le chiffré donné à l'adversaire n'est pas un chiffré valide, puisque la clé de re-chiffrement utilisée est incorrecte. Cependant, nous verrons que ce n'est pas un problème, car $\mathcal{A}$ ne peut pas s'apercevoir de la supercherie. Il faut, pour autant, faire attention avec ce re-chiffré. En effet, si un adversaire demande son déchiffrement, il faut lui fournir le message correspondant au chiffré originel et non à ce re-chiffré, afin que la simulation soit correcte. Pour cela, $\mathcal{B}$ ajoute $(pk_{\mathtt{A}}, pk_{\mathtt{B}}, C_{\mathtt{A},1}, C_{\mathtt{B},2})$ à $\mathcal{C}^{\mathsf{list}}$ afin de se souvenir du chiffré originel.
- si $sk_{\mathtt{A}} = (\perp, sk_{\mathtt{A},2})$ et si $\mathtt{B}$ est corrompu, alors dans ce cas là, $\mathcal{B}$ ne peut utiliser la fausse clé de re-chiffrement comme dans le précédent cas. En effet, $\mathcal{A}$ peut déchiffrer l'information destinée à $\mathtt{B}$, il s'apercevrait alors de la supercherie. Il faut donc lui fournir un re-chiffré correct. En particulier, il faut être capable de transformer $E = pk_{A,1}^r$ en $E' = g^{sk_{\mathtt{V}} \cdot r}$. En parcourant la liste $\mathcal{H}_1^{\mathsf{list}}$, $\mathcal{B}$ peut retrouver r tel que $E = pk_{\mathtt{A}}^r$ et calculer E', à condition que r ait bien été calculé comme $\mathcal{H}_1(m\|\omega)$. La preuve de connaissance NIZKOE permet, quant à elle, de pouvoir extraire cet élément, qu'il ait été bien formé ou non. De cette manière, $\mathcal{B}$ est capable de produire un E' correct. Il peut compléter le re-chiffrement en tirant tout d'abord aléatoirement $z \in \mathbb{Z}_q^*$ tel que sa représentation en bits soit un élément de $\{0,1\}^{l_0}$ et en calculant $F' = \mathcal{H}_2(g^z) \oplus F$. Ensuite, il chiffre z en Hash ElGamal pour $pk_{\mathtt{B},2}$ et obtient (X, Y). Au final, ces manipulations donnent un chiffré $C_{\mathtt{B},2} = (E', F', V, W, X, Y)$ correctement re-chiffré.
- sinon, il suffit de procéder comme dans l'algorithme Re-chiffrer en utilisant la clé de re-chiffrement, qui, dans ce cas, est valide.

ODéchiffrer$_2(pk, C_{pk,2})$: Il commence à répartir le chiffré $C_{pk,2}$ en (E', F', V, W, X, Y). Il récupère ensuite l'entrée (sk, pk) dans la liste $\mathcal{K}^{\mathsf{list}}$ et procède selon différents cas.

- S'il existe une entrée $(pk', pk, \tilde{R}_{pk',1\to pk,2}, sk_{\mathtt{V}}, V, W)$ dans $\mathcal{R}^{\mathsf{list}}$, plusieurs cas sont alors possibles.
  - Si pk' est l'entité cible du challenge :
    A) s'il existe une entrée $(pk', pk, C_{pk',1}, C_{pk,2})$ dans $\mathcal{C}^{\mathsf{list}}$, cela signifie que $C_{pk,2}$ n'est pas valide, qu'il a été produit par une requête de re-chiffrement et correspond à un re-chiffrement de $C_{pk',1}$. Il suffit alors de déchiffrer $C_{pk',1}$ avec la clé privée correspon-

dante sk'. Cependant, sk' est de la forme $(\perp, sk'_2)$. $\mathcal{B}$ ne connaît donc pas la clé de déchiffrement utilisée. Il répartit alors le chiffré $C_{pk',1}$ en $(E, F, (T_k, \mathtt{ch}_k, \mathtt{rep}_k)_{k=1..l})$. La preuve de connaissance NIZKOE de ce chiffré a déjà été vérifiée lors de la requête de re-chiffrement, il n'est donc pas nécessaire de la vérifier à nouveau. Comme nous l'avons déjà expliqué dans les requêtes de re-chiffrement, l'oracle aléatoire peut être utile pour retrouver r tel que $E = pk'^r_1$, où pk'_1 est la première composante de pk', en parcourant $\mathcal{H}_1^{\mathsf{list}}$ et à condition que r soit de la forme $\mathcal{H}_1(m\|\omega)$. Sinon, $\mathcal{B}$ peut utiliser l'extracteur de la preuve de connaissance pour le retrouver. Il calcule ensuite $(m\|\omega) = F \oplus \mathcal{H}_2(g^r)$. Il vérifie finalement que $r = \mathcal{H}_1(m\|\omega)$ et retourne m ou $\perp$ si ce n'est pas le cas.

B) sinon, il utilise l'oracle aléatoire $\mathcal{H}_1$ pour retrouver r et x tel que $E = (pk'_1)^r$ et $X = pk_2^x$, où $E = (E')^{1/\tilde{R}_{pk',1\to pk,2}}$, pk'_1 et pk_2 sont respectivement la première composante de pk' et la deuxième de pk. Cela lui permet d'obtenir successivement $(z\|\alpha) = Y \oplus \mathcal{H}_2(g^x)$, et $(m\|\omega) = F' \oplus \mathcal{H}_2(g^r) \oplus \mathcal{H}_2(g^z)$. $\mathcal{B}$ n'a pas à vérifier que (V, W) est un chiffré valide, puisqu'il sait déjà qu'il chiffre $sk_{\mathtt{V}}$. Pour cette raison, il vérifie seulement que $r = \mathcal{H}_1(m\|\omega)$ et $x = \mathcal{H}_1(z\|\alpha)$ et retourne m ou $\perp$ si ce n'est pas le cas.

- La clé de re-chiffrement est correcte et n'a donc pas pu introduire de faux calculs comme dans le 1er cas. Pour cette raison, il suffit de déchiffrer comme dans l'algorithme $\mathsf{Déchiffrer}_2$ en utilisant la clé privée sk_2, qui est toujours connue de $\mathcal{B}$.

– S'il n'y a aucune entrée $(pk', pk, \tilde{R}_{pk',1\to pk,2}, sk_{\mathtt{V}}, V, W)$ dans $\mathcal{R}^{\mathsf{list}}$, alors $\mathcal{B}$ déchiffre comme dans l'algorithme $\mathsf{Déchiffrer}_2$ en utilisant la clé privée sk_2, qui est toujours connue de $\mathcal{B}$.

Challenge. $\mathcal{A}$ choisit deux messages m_0 et m_1 appartenant tous les deux à l'ensemble $\mathcal{M}$. $\mathcal{B}$ lui fournit en retour le chiffré challenge $C^\star = (E^\star, F^\star, (T_k^\star, \mathtt{ch}_k^\star, \mathtt{rep}_k^\star)_{k=1,\ldots,l})$. Cependant, ce chiffré doit correspondre à un chiffré de m_δ, où δ est tiré aléatoirement parmi $\{0, 1\}$, pour la clé publique pk_1^*. En particulier, $E^\star = B^{\tilde{sk}_1^\star}$ et $F^\star$ qui est tiré aléatoirement, doivent être respectivement de la forme $(pk_1^\star)^{\mathcal{H}_1(m_\delta\|\omega^\star)}$ et $(m_\delta\|\omega^\star) \oplus \mathcal{H}_2(g^{b/a})$. Ceci est rendu possible via l'utilisation de l'oracle aléatoire. Ainsi :

– il tire aléatoirement $\omega^\star \xleftarrow{\$} \{0,1\}^{l_1}$, ce qui définit implicitement $\mathcal{H}_1(m_\delta\|\omega^\star) = b/a$ et $\mathcal{H}_2(g^{b/a}) = F^\star \oplus (m_\delta\|\omega^\star)$. S'il existe déjà une entrée $((m_\delta, \|\omega^\star), r)$ dans $\mathcal{H}_1^{\mathsf{list}}$ ou une entrée $((R, F^\star \oplus (m_\delta\|\omega^\star))$ dans $\mathcal{H}_2^{\mathsf{list}}$, alors il recommence cette étape.

Phase 2. $\mathcal{B}$ répond toujours aux requêtes de $\mathcal{A}$ de la même manière que dans la Phase 1, en tenant compte évidemment des restrictions naturelles de cette sécurité. Cela donne lieu à quelques différences de fonctionnement, en particulier sur les oracles $\mathsf{ORe\text{-}chiffrer}$ et $\mathsf{ODéchiffrer}$.

– $\mathbf{ORe\text{-}chiffrer}(pk_{\mathtt{A}}, pk_{\mathtt{B}}, C_{\mathtt{A},1})$: si $pk_{\mathtt{A}} = pk^\star$, $C_{\mathtt{A},1} = C^\star$ et $pk_{\mathtt{B}}$ corrompu, alors $\mathcal{B}$ retourne $\perp$ puisque $\mathcal{A}$ ne doit pas obtenir un re-chiffré du challenge pour une entité corrompue. Autrement, il procède comme dans la Phase 1.

– $\mathbf{ODéchiffrer}(pk, C_{pk,2})$: s'il existe une entrée $(pk^\star, pk, C^\star, C_{pk,2})$ dans $\mathcal{C}^{\mathsf{list}}$, alors $\mathcal{B}$ retourne $\perp$ puisque $\mathcal{A}$ ne doit pas obtenir un déchiffrement d'un re-chiffré du challenge. Autrement, il procède comme dans la Phase 1.

Réponse. $\mathcal{A}$ retourne δ' et gagne son challenge si $\delta = \delta'$. $\mathcal{B}$, quant à lui, sélectionne un upplet $(R, S) \in \mathcal{H}_2^{\mathsf{list}}$ et retourne R comme solution à l'instance DCDH.

Analyse de la simulation. Nous commençons tout d'abord par analyser la simulation que l'on vient de présenter. En effet, pour que $\mathcal{B}$ puisse utiliser $\mathcal{A}$ pour casser l'hypothèse DCDH, il faut que celui-ci ne détecte pas la simulation.

1) La simulation de l'oracle aléatoire $\mathcal{H}_1$ est parfaite à moins que $(m_\delta\|\omega^\star)$ ne soit demandé à cet oracle, puisque la valeur correspondante est b/a qui est inconnue. Il en est de même

avec $\mathcal{H}_2$ à moins que $g^{b/a}$ ne soit demandé à $\mathcal{H}_2$, puisque cette valeur est inconnue, donc impossible à détecter, et que la valeur correspondante est fixée. La simulation de $\mathcal{H}_3$ est, quant à elle, parfaite.

2) La simulation des clés de chiffrement est parfaite.

3) Les clés de re-chiffrement sont bien formées, exceptées celles qui permettent de re-chiffrer de l'entité cible vers une entité honnête, puisque dans ce cas, $\mathcal{B}$ produit une fausse clé de re-chiffrement. En effet, l'élément $\tilde{R}_{A,1\to B,2}$ devrait être égal à $sk_{\mathsf{V}}/sk_1^\star$ alors qu'il est aléatoire. Cependant, $\mathcal{A}$ n'est pas capable de faire cette distinction. En effet, il n'a aucune information sur $sk_1^\star$. De plus, sk_{V} n'apparaît que chiffrée pour la clé privée sk_2^*, qui lui est inconnue, avec le schéma Hash ElGamal prouvé IND-CCA sous l'hypothèse CDH. Le quotient $sk_{\mathsf{V}}/sk_1^\star$ ressemble donc à un élément aléatoire pour $\mathcal{A}$. Pour cette raison, la simulation de ces clés est également parfaite.

4) Concernant l'oracle de re-chiffrement, il est facile de voir que, si la clé de re-chiffrement concernée par la requête est correcte, alors la simulation est parfaite. Quand la clé de re-chiffrement est incorrecte, nous sommes amenés à distinguer deux situations :
   - le délégué est honnête, alors la clé de re-chiffrement est une clé valide du point de vue de $\mathcal{A}$ et l'oracle rechiffre en l'utilisant comme dans l'algorithme Re-chiffrer. La simulation est donc parfaite.
   - le délégué est malhonnête, alors l'oracle de re-chiffrement produit un rechiffré valide en utilisant l'oracle aléatoire $\mathcal{H}_1$ et l'extracteur de la preuve NIZKOE pour récupérer l'élément r. La simulation est donc parfaite à condition que l'extracteur parvienne à récuperer le secret (événement dont la probabilité est à distance négligeable de 1).

5) Il nous reste à voir la simulation de l'oracle de déchiffrement. Nous ne détaillons pas toute l'analyse de cet oracle. Il est facile dans la plupart des cas de voir qu'il répond aux requêtes de $\mathcal{A}$, en procédant comme dans l'algorithme $\mathsf{Déchiffrer}_2$. En revanche, il y a deux points qu'il faut préciser. Ils concernent la simulation de l'oracle de déchiffrement quand il existe une entrée $(pk', pk, \tilde{R}_{pk',1\to pk,2}, sk_{\mathsf{V}}, V, W)$ dans $\mathcal{R}^{\mathsf{list}}$ avec pk' l'entité cible du challenge mais pas d'entrée $(pk', pk, C_{pk',1}, C_{pk,2})$ dans $\mathsf{C}^{\mathsf{list}}$.
   - Le premier point, qui est celui considéré par l'oracle aléatoire, correspond à la situation où la fausse clé de re-chiffrement considérée a été utilisée pour calculer le chiffré $C_{pk,2}$ mais sans passer par l'oracle de re-chiffrement - on ne dispose alors pas de preuve de connaissance NIZKOE pour aider l'oracle aléatoire $\mathcal{H}_1$ à retrouver r. Il est facile de voir que si r a bien été choisi comme $\mathcal{H}_1(m\|\omega)$, alors $\mathsf{ODéchiffrer}_2$ et $\mathsf{Déchiffrer}_2$ répondent de la même manière via l'utilisation de $\mathcal{H}_1^{\mathsf{list}}$. En revanche, lorsque r n'a pas été choisi de cette manière, $\mathsf{ODéchiffrer}_2$ retourne $\perp$ puisqu'il ne peut pas le retrouver. Ce n'est, pour autant, pas toujours le cas pour $\mathsf{Déchiffrer}_2$, qui lui retrouve $(m\|\omega)$, le demande en requête à $\mathcal{H}_1$ afin d'obtenir r et vérifie ensuite que $E' = g^{r.sk_{\mathsf{V}}}$. Cette dernière égalité ne sera a priori pas vérifiée, à moins que, par un heureux hasard, $\mathcal{H}_1$ retourne la valeur permettant de vérifier cette égalité. Cet heureux hasard arrive avec une probabilité de $1/q$, qui est négligeable. L'oracle de re-chiffrement répond donc, sauf dans certains cas négligeables, comme dans le schéma. Il est à noter que ceci n'arrive pas, a priori, dans les autres cas, puisque la preuve NIZKOE permet de récupérer ce r (quand $\mathcal{H}_1$ ne le peut pas) et donc de retrouver $(m\|\omega)$.
   - Le deuxième point, qui n'est pas considéré par l'oracle aléatoire, correspond à la situation où la fausse clé de re-chiffrement considérée n'a pas été utilisée pour créer le chiffré $C_{pk,2}$. Dans ce cas, l'oracle de re-chiffrement considère que le chiffré a été formé en utilisant cette fausse clé et qu'il n'est donc pas valide. Il récupère alors $E = (E')^{\tilde{R}_{pk',1\to pk,2}}$ afin de pouvoir déchiffrer correctement. Cette fausse clé n'ayant pas été utilisée, l'opération va produire un déchiffrement incorrect et donc renvoyer

$\perp$. Pour autant, il est possible que $\mathsf{Déchiffrer}_2$ renvoie m et non $\perp$, à condition que ce chiffré ait été produit en connaissant sk_{V} qui n'apparaît que chiffré en (V, W) pour l'entité pk honnête. $\mathcal{A}$ doit donc trouver, pour être dans une telle situation, la clé sk_{V}. Cet événement a une probabilité de $1/2^{l_0}$, donc négligeable, ce qui justifie le fait que l'oracle de déchiffrement n'en tienne pas compte.

6) Concernant la génération du challenge, il est évident qu'elle est parfaite.

Résolution de l'instance DCDH. Nous voyons ainsi que la simulation offerte par $\mathcal{B}$ à $\mathcal{A}$ est parfaite, sauf dans certains cas apparaissant avec une probabilité négligeable. Il nous faut maintenant voir comment $\mathcal{B}$ peut utiliser $\mathcal{A}$ pour casser l'hypothèse DCDH. Pour simplifier les explications, nous omettons le paramètre de sécurité λ dans les différentes probabilités et avantages que nous considérons. Désignons alors $\mathsf{Err} = \mathcal{H}_1^* \vee \mathcal{H}_2^* \vee \mathsf{Err.O_2}$, l'événement que l'un de ces cas particulier se produise, où :

- $\mathcal{H}_1^*$ est l'événement que $(m_\delta \| \omega^\star)$ soit demandé à $\mathcal{H}_1$.
- $\mathcal{H}_2^*$ est l'événement que $g^{b/a}$ soit demandé à $\mathcal{H}_2$ - pour rappel, c'est la valeur que $\mathcal{B}$ recherche.
- $\mathsf{Err.O_2}$ est l'événement que l'un des cas particuliers dans l'analyse de $\mathsf{ODéchiffrer}_2$ ait lieu.

Nous voyons bien que c'est le cas où $\mathcal{H}_2^*$ se produit qui va intéresser $\mathcal{B}$ et aucun autre, pour attaquer l'hypothèse DCDH. Dans ce cas, $\mathcal{B}$ choisit un couple (R, S) dans $\mathcal{H}_2^{\mathsf{list}}$ et renvoie R. En notant $q_{\mathcal{H}_2}$ le nombre de requêtes à l'oracle aléatoire $\mathcal{H}_2$, il a une chance sur $q_{\mathcal{H}_2}$ de choisir le bon et ainsi de trouver la solution à l'instance DCDH. Il faut donc arriver à relier cet événement au fait que l'adversaire a un avantage non négligeable de gagner son challenge.

Si l'événement Err n'intervient pas - et donc $\mathcal{H}_2^*$ non plus -, alors $\mathcal{A}$ a exactement une chance sur deux de gagner son challenge. Nous avons donc $\mathtt{Pr}(\delta = \delta' | \neg \mathsf{Err}) = \frac{1}{2}$. Notons $\mathtt{Pr}_{(\mathcal{B},\mathrm{DCDH})}$, la probabilité que $\mathcal{B}$ ait de trouver la solution à l'instance DCDH. Comme expliqué dans [CWYD10], en utilisant le fait que $\mathsf{Adv}_{\mathsf{PRE},\mathcal{A},(1)} = |2.\mathtt{Pr}(\delta = \delta') - 1|$ et que $\mathtt{Pr}(\delta = \delta') = \mathtt{Pr}(\delta = \delta' | \mathsf{Err}) + \mathtt{Pr}(\delta = \delta' | \neg \mathsf{Err})$, nous obtenons $\mathtt{Pr}_{(\mathcal{B},\mathrm{DCDH})} \geq \frac{\mathtt{Pr}(\mathcal{H}_2^\star)}{q_{\mathcal{H}_2}} \geq \frac{\mathtt{Pr}(\mathsf{Err})}{q_{\mathcal{H}_2}} - \frac{\mathtt{Pr}(\mathcal{H}_1^\star)}{q_{\mathcal{H}_2}} - \frac{\mathtt{Pr}(\mathsf{Err.O_2})}{q_{\mathcal{H}_2}} \geq \frac{\mathsf{Adv}_{\mathsf{PRE},\mathcal{A},(1)}}{q_{\mathcal{H}_2}} - \frac{\mathtt{Pr}(\mathcal{H}_1^\star)}{q_{\mathcal{H}_2}} - \frac{\mathtt{Pr}(\mathsf{Err.O_2})}{q_{\mathcal{H}_2}}$.

Il nous reste à calculer les probabilités restantes. Concernant $\mathcal{H}_1^*$, il faut que $(m_\delta \| \omega^\star)$ soit demandé à $\mathcal{H}_1$ avec $\omega^\star$ qui n'est utilisé que dans le chiffré challenge et inconnu de $\mathcal{A}$. Par un calcul de loi binomiale, nous obtenons $\mathtt{Pr}(\mathcal{H}_1^*) \leq \frac{q_{\mathcal{H}_1}}{2^{l_1}}$ qui est négligeable, où $q_{\mathcal{H}_1}$ est le nombre de requêtes effectuées à $\mathcal{H}_1$. Nous avons également vu, dans l'analyse de l'oracle de déchiffrement, que la probabilité d'une requête provoquant l'événement $\mathsf{Err.O_2}$ était inférieure ou égale à $\max(\frac{1}{q}, \frac{1}{2^{l_0}})$. En notant $q_{\mathsf{ODéchiffrer}_2}$ le nombre de requêtes à cet oracle, nous obtenons, par le même calcul, que $\mathtt{Pr}(\mathsf{Err.O_2}) \leq q_{\mathsf{ODéchiffrer}_2}.\max(\frac{1}{q}, \frac{1}{2^{l_0}})$ qui est aussi négligeable. En définitive, d'après l'hypothèse DCDH, il n'existe donc pas de tel adversaire avec un avantage non négligeable, ce qui prouve le théorème.

En réalité, et pour ne pas compliquer davantage la preuve et sa compréhension, nous avons omis les cas où la preuve NIZKOE ne réussissait pas à extraire r. En effet, dans une telle situation, qui arrive avec une probabilité négligeable notée e, $\mathcal{B}$ doit retourner $\perp$ et arrêter la simulation puisqu'il n'est pas capable de répondre correctement. En notant Abort l'événement que la preuve NIZKOE ne réussisse pas à effectuer une extraction du secret et $q_{\mathsf{ORe\text{-}chiffrer}}$ le nombre de requêtes à l'oracle de re-chiffrement, nous obtenons, toujours par le même calcul, que $\mathtt{Pr}(\mathsf{Abort}) \leq e.(q_{\mathsf{ORe\text{-}chiffrer}} + q_{\mathsf{ODéchiffrer}_2})$. Cette probabilité est négligeable et l'événement contraire $\neg\mathsf{Abort}$ a donc une probabilité à distance négligeable de 1. $\mathcal{B}$ ne peut alors utiliser l'adversaire que si Abort n'intervient pas. Ainsi en considérant $\mathsf{Err} = \mathcal{H}_1^* \vee \mathcal{H}_2^* \vee \mathcal{H}_3^* \vee \mathsf{Err.O_2} | \neg\mathsf{Abort}$ et en suivant le même raisonnement que précédemment,

nous obtenons $\mathtt{Pr}_{(\mathcal{B},\mathrm{DCDH})} \geq \frac{\mathtt{Pr}(\neg\mathsf{Abort}).\mathsf{Adv}_{\mathsf{PRE},\mathcal{A},(1)}}{q_{\mathcal{H}_2}} - \frac{\mathtt{Pr}(\mathcal{H}_1^\star)}{q_{\mathcal{H}_2}} - \frac{\mathtt{Pr}(\mathcal{H}_3^\star)}{q_{\mathcal{H}_2}} - \frac{\mathtt{Pr}(\mathsf{Err}.\mathsf{O}_2)}{q_{\mathcal{H}_2}}$. $\mathtt{Pr}(\neg\mathsf{Abort})$ étant négligeablement proche de 1, le théorème reste valide.

□

Le corollaire suivant se déduit directement de ces trois théorèmes.

Corollaire 35. *Le schéma atteint la sécurité* IND-CCA *sous l'hypothèse* CDH *et sans aucune restriction non naturelle.*

Conclusion. D'un point de vue anecdotique, il est intéressant de remarquer que les deux schémas que nous avons modifiés dans ce chapitre possèdent une vérifiabilité publique qui ne s'assure pas complètement de la validité d'un chiffré. Ce sont même les seuls schémas sûrs contre des attaques à chiffrés choisis qui offrent cette fonctionnalité. Cette particularité est rendue possible grâce à l'utilisation massive de l'oracle aléatoire. Pour cette raison, elle ne semble pas réalisable dans le modèle standard. En particulier, dans un tel modèle, il faudrait concevoir et considérer un moyen de vérifier publiquement et complètement un chiffré, ce qui peut affecter fortement l'efficacité du schéma. Nous remarquons également que ces schémas sont les seuls à ne pas utiliser de couplage bilinéaire tout en étant IND-CCA sûr (ou IND-CCA_B sûr pour les schémas avec re-chiffrement bidirectionnel), ce qui peut justifier leur utilisation si l'on dispose par exemple de dispositifs n'implémentant pas ces couplages bilinéaires.

Quatrième partie

Gestion des droits d'accès

Chapitre 7

Gestion dynamique des droits d'accès

Sommaire

7.1 Un cas d'usage particulier 99
7.1.1 Problématique 99
7.1.2 Un nouveau type de schéma - Le serveur de re-chiffrement combiné 101
7.1.3 Fonctionnalités 102
7.2 Résultats génériques 103
7.2.1 Le re-chiffrement combiné implique le re-chiffrement bidirectionnel infini et le re-chiffrement unidirectionnel unique 103
7.2.2 La réciproque est-elle vraie ? 103
7.2.3 Trois re-chiffrements bidirectionnels et infinis impliquent le re-chiffrement combiné 104
7.3 Une construction pratique 107
7.3.1 Une version combinée du schéma Libert-Vergnaud 107
7.3.2 Propriétés additionnelles 109
***Note sur un serveur de re-chiffrement combiné généralisé* 111**

Nous présentons dans ce chapitre un nouveau modèle de serveur de re-chiffrement. Afin de bien le comprendre, nous reprenons la motivation proposée au Chapitre 1 proposant un stockage de données basé sur le serveur de re-chiffrement. Nous nous intéressons ici à la gestion dynamique des droits d'accès. Après avoir exposé les fonctionnalités souhaitées sur ce service, nous étudions les solutions existantes dans la littérature. Nous présentons ensuite ce nouveau modèle où l'on réunit au sein d'un même schéma deux modèles de re-chiffrement. Nous fournissons alors une construction générique de ces schémas ainsi qu'une solution pratique basée sur le schéma [LV08b]. Ces travaux ont été publiés dans [CD13].

7.1 Un cas d'usage particulier

7.1.1 Problématique

Nous avons vu au Chapitre 1 que le système de stockage de données à base de serveur de re-chiffrement offrait une certaine dynamique sur les droits d'accès. Nous nous attachons maintenant à étudier s'il est possible d'obtenir une dynamique complète qui soit indépendante des données stockées, i.e que les utilisateurs n'aient pas à récupérer les données stockées pour pouvoir en ajouter ou en supprimer d'autres. Nous passons en revue les différentes opérations sur le dynamisme des droits d'accès auquel le système peut être confronté. Pour cela, nous reprenons les notations utilisées dans notre exemple de stockage considéré dans le Chapitre 1. Nous avons donc une donnée chiffrée et stockée pour un dispositif $\mathtt{A}_1$ appartenant à Alice, donnée chiffrée en

$C_{\mathtt{A}_1}$. Nous ne précisons pas le type de schéma considéré ni le type de ce chiffré, afin de rester le plus général possible. Les différentes opérations possibles sont :

- **Ajout d'un utilisateur.** Cette opération traduit la situation où `A` a confiance en un nouvel utilisateur. Comme nous l'avons déjà vu, les clés de re-chiffrement sont indépendantes du chiffré $C_{\mathtt{A}_1}$. Il suffit alors, pour ajouter un utilisateur, `U`, que $\mathtt{A}_1$ calcule une clé de re-chiffrement de $\mathtt{A}_1$ vers `U` et la donne au serveur. Ainsi l'ajout d'un utilisateur se fait très facilement, l'utilisateur `U` devenant alors un délégué de $\mathtt{A}_1$.
- **Suppression d'un délégué de $\mathtt{A}_1$.** Cette opération traduit la situation où $\mathtt{A}_1$ n'a plus confiance en un de ses délégués, ce qui pourrait arriver par exemple si le délégué en question se faisait voler son smartphone. Evidemment, il est possible que ce délégué ait déjà eu accès à la donnée stockée, mais si ce n'est pas le cas, il faut qu'il ne puisse pas le faire. Cette suppression permet également de l'empêcher d'accéder aux données que $\mathtt{A}_1$ décidera de stocker par la suite. Soit `U` ce délégué, il existe donc une clé de re-chiffrement de $\mathtt{A}_1$ vers `U`. Il suffit de supprimer cette clé (pour rappel, on fait confiance au serveur pour qu'il fasse ce qu'on lui demande) pour que `U` n'ait plus accès aux données de $\mathtt{A}_1$.
- **Suppression de l'accès du dispositif maître.** Cette opération traduit la situation où Alice ne posséderait plus son dispositif $\mathtt{A}_1$, comme par exemple si elle avait donné ou vendu son ordinateur à un autre utilisateur. Pour un smartphone, cela pourrait signifier également qu'Alice se l'est fait voler ou l'a perdu. Dans ce cas, on ne peut supprimer, comme précédemment, une clé de re-chiffrement. La donnée est stockée et chiffrée pour ce dispositif, il faut donc la supprimer. Cependant cela n'est pas suffisant, puisqu'il faut que les délégués puissent toujours accéder à la donnée. Le serveur doit donc avant d'effectuer cette suppression, re-chiffrer $C_{\mathtt{A}_1}$ pour chaque délégué et conserver chacun de ces re-chiffrés. Les clés de re-chiffrement entre $\mathtt{A}_1$ et ses délégués ne sont plus d'aucune utilité, le serveur peut donc les supprimer. Cette opération est illustrée dans la Figure 7.1.

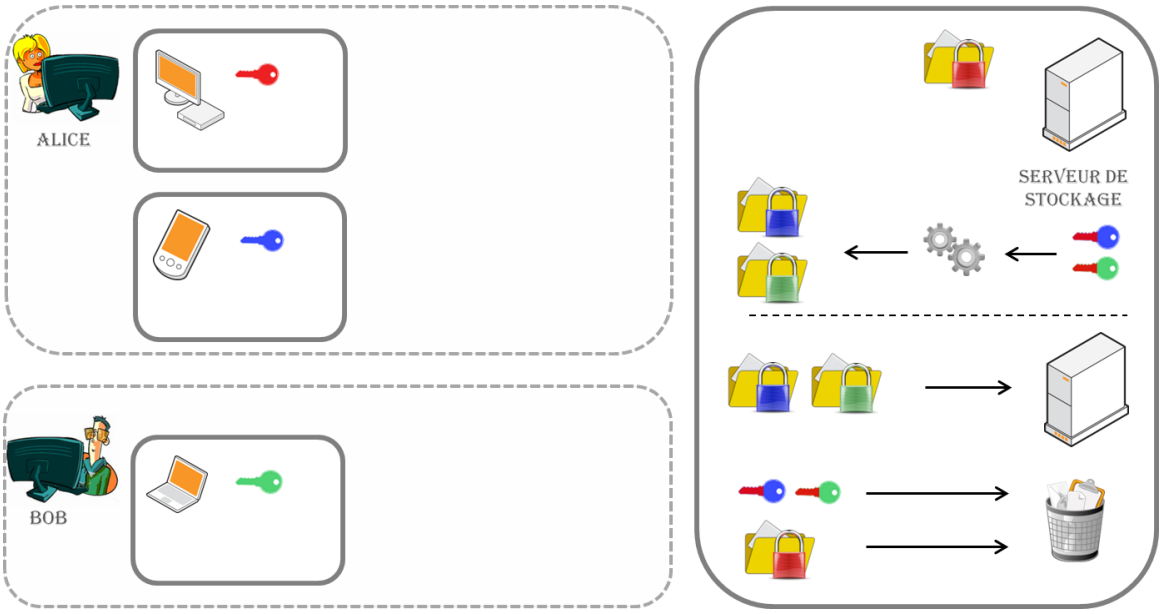

Figure 7.1 – Suppression de l'accès du dispositif $\mathtt{A}_1$

Ainsi, il est possible d'ajouter ou de supprimer n'importe quel utilisateur. De plus, nous n'avons pas précisé le type de serveur de re-chiffrement pris en compte, ce qui permet de réaliser toutes ces opérations, quelque soit le schéma utilisé. En revanche, il reste un point essentiel : le système doit rester fonctionnel une fois ces opérations effectuées, notamment si l'on souhaite une gestion dynamique. Il est facile de voir que les deux premières opérations ne changeront pas cette

gestion, puisqu'elles ne consistent qu'à ajouter ou supprimer des clés de re-chiffrement et n'utilise pas la fonctionnalité du re-chiffrement contrairement à la dernière. Les influences qu'aura celle-ci dépendront des fonctionnalités de re-chiffrement offertes par le schéma utilisé. Nous examinons maintenant les différentes caractéristiques que nous avons énoncées au Chapitre 3 sur lesquelles sont basés les types récurrents.

- **Type de confiance.** De par sa définition, le type de confiance n'a pas d'impact sur le re-chiffrement effectué. Il faut pour autant être attentif sur l'éventuelle bidirectionnalité des clés de re-chiffrement. En effet, le serveur doit impérativement supprimer celles impliquant l'utilisateur $\mathtt{A}_1$, afin qu'un individu disposant du dispositif $\mathtt{A}_1$ ne puisse obtenir les données accessibles par les délégués de ce dernier. Nous avons également vu qu'il était fréquent qu'à partir d'une clé de re-chiffrement bidirectionnelle et d'une clé privée impliquée dans cette dernière, il était possible de retrouver la deuxième clé privée la composant. Ainsi, il faut être certain que cette clé de re-chiffrement ne puisse être obtenue par un individu malveillant puisque s'il dispose également du dispositif $\mathtt{A}_1$, il pourrait alors obtenir les clés privées des autres dispositifs d'Alice. De plus, lors de la conception du système, il ne faut pas oublier que les clés bidirectionnelles impliquent une confiance mutuelle entre le délégant et le délégué. Elles ne peuvent être utilisées pour n'importe quel utilisateur.
- **Nombre de re-chiffrements.** Contrairement au point précédent, la fonctionnalité offerte par cette caractéristique va impacter différemment le système. En effet, on voit que la suppression de l'accès de $\mathtt{A}_1$ entraîne la production de re-chiffrés. Nous voyons tout de suite que si ce re-chiffré ne peut être re-chiffré à nouveau, alors il sera impossible d'ajouter par la suite de nouveaux délégués. La gestion ne sera donc plus dynamique. Le re-chiffrement infini permet, quant à lui, de toujours garder une gestion dynamique, quelque soit le nombre de suppressions de droits d'accès effectuées. Le re-chiffrement multiple se situe entre les deux et autorise, s'il est de cardinal n, n opérations de suppression du dispositf maître avant de ne plus disposer d'une gestion dynamique.

Modèle de base idéal. Nous voyons ainsi que la solution idéale parmi les modèles de bases serait de disposer d'un schéma offrant un re-chiffrement infini et unidirectionnel. En effet, la caractéristique de re-chiffrement infini offre une gestion dynamique, peu importe les opérations effectuées. Le re-chiffrement unidirectionnel permet aux utilisateurs de ne pas se faire, de manière systématique, mutuellement confiance entre eux. D'après le Chapitre 4, le seul schéma correspondant à ce type est celui de Gentry, qui utilise un chiffrement totalement homomorphe. Même si son efficacité était amélioré, ce schéma resterait donc trop coûteux par rapport aux autres schémas de serveur de re-chiffrement. Ainsi, nous ne disposons pas de schéma efficace qui soit idéal pour la problématique exposée.

7.1.2 Un nouveau type de schéma - Le serveur de re-chiffrement combiné

Nous souhaitons trouver une solution qui permette la gestion dynamique exposée ci-dessus. Il nous faut ainsi concevoir et réaliser un nouveau type de schéma de serveur de re-chiffrement. D'après la littérature, nous disposons de deux types courants de schéma : celui avec re-chiffrement bidirectionnel et infini avec un ensemble de liaisons $\{(1,1)\}$ (ou $(2,2)$) et celui avec re-chiffrement unidirectionnel et unique sur un ensemble de liaisons $\{(1,2)\}$. Notre idée est alors de combiner ces deux types de schémas pour n'en faire qu'un seul, ce qui nous donne les ensembles $\mathsf{Type} = \{1,2\}$ et $\mathsf{Liaison} = \{(1,1),(1,2),(2,2)\}$. Chaque re-chiffrement sera utilisé dans un cas précis dépendant du couple (délégant/délégué) décrit ainsi :

- le re-chiffrement bidirectionnel et infini sur une liaison (i,i) pour $i \in \{1,2\}$ sera utilisé dans le cas où le délégant et le délégué correspondent à deux dispositifs d'un même utilisateur. Ces dispositifs peuvent être, par exemple, le smartphone et l'ordinateur d'un même utilisateur. Ce type de re-chiffrement n'est pas problématique dans ce cas là, puisqu'il est évident que

l'utilisateur souhaite accéder à ses données depuis son ordinateur mais également depuis son smartphone ;
– le re-chiffrement unidirectionnel et unique sur la liaison $(1, 2)$ sera utilisé dans le cas où le délégant et le délégué correspondent à deux dispositifs appartenant à des utilisateurs différents.

7.1.3 Fonctionnalités

Avant de présenter des exemples de réalisation de cette brique, nous détaillons ses particularités.

Type de délégation. Dans un tel schéma et par la description succinte que nous venons d'en faire, il ne sera pas possible de déléguer à son tour les droits d'accès entre utilisateurs, puisque le re-chiffrement unidirectionnel utilisé est unique. Ainsi, les données stockées ne pourront être transférées de Alice vers Charlie en passant par Bob. Pour que Charlie puisse les obtenir, il faudra que Bob intervienne en récupérant les données, puis les envoie à Charlie, ou que Bob arrive à convaincre Alice d'accepter Charlie comme délégué. L'utilisation de ce type de schéma est détaillée dans la Figure 7.2.

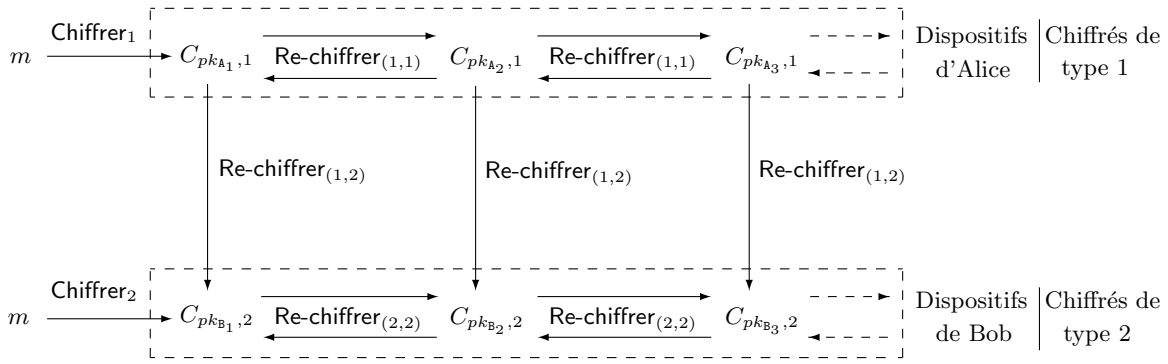

Figure 7.2 – Utilisation du serveur de re-chiffrement combiné

Remarque 7. Les deux suites de dispositifs $(\mathtt{A}_1, \mathtt{A}_2, \mathtt{A}_3, \cdots,)$ et $(\mathtt{B}_1, \mathtt{B}_2, \mathtt{B}_3, \cdots)$ représentées sont indépendantes. Pour passer de Alice à Bob, nous n'avons mentionné sur cette figure que les re-chiffrements entre le dispositif $\mathtt{A}_i$ et le dispositif $\mathtt{B}_i$ pour $i \in [\![1; 3]\!]$ mais il est également possible de passer directement de $\mathtt{A}_i$ à $\mathtt{B}_j$ pour $i \neq j$ à condition qu'il existe une clé de re-chiffrement entre ces deux dispositifs.

Gestion dynamique des droits d'accès. Nous avons vu que seule la suppression de l'accès du dispositif maître était problématique pour obtenir une gestion dynamique des droits d'accès. La nature d'un schéma de serveur de re-chiffrement combiné la rend possible en raison de sa structure particulière. En effet, lors de la suppression des droits d'accès du dispositif $\mathtt{A}_1$, le serveur modifie toutes les données stockées pour $\mathtt{A}_1$ et les remplace par leurs équivalents pour $\mathtt{A}_2$ (ou tout autre dispositif d'Alice) et les re-chiffrements autorisés impliquant $\mathtt{A}_1$ seront remplacés par des re-chiffrements équivalents impliquant $\mathtt{A}_2$, comme par exemple la clé de $\mathtt{A}_1$ vers $\mathtt{B}_1$ sera remplacée par la clé de $\mathtt{A}_2$ vers $\mathtt{B}_1$. Cette opération ne modifie donc pas la nature du stockage et de la délégation, si ce n'est de supprimer les droits d'accès de $\mathtt{A}_1$. Par exemple, le dispositif B_1 peut toujours accéder aux données qui étaient au départ destinées à $\mathtt{A}_1$ avant d'être destinées à $\mathtt{A}_2$. De plus, rien n'empêche de supprimer par la suite le dispositif $\mathtt{A}_2$, de le remplacer par $\mathtt{A}_3$, puis de supprimer à nouveau celui-ci et ainsi de suite jusqu'à ce qu'Alice ne possède plus de dispositif.

Ce schéma étant un modèle particulier de serveur de re-chiffrement, il est naturel que son modèle de sécurité corresponde à celui du modèle unifié exposé dans le Chapitre 3.

7.2 Résultats génériques

En observant un serveur de re-chiffrement combiné, nous nous apercevons qu'il est composé de deux types de schéma : le re-chiffrement bidirectionnel inifini et le re-chiffrement unidirectionnel unique. Il est alors naturel de s'intéresser aux relations entre ces deux types de re-chiffrement et celui combiné. Nous présentons également une construction générique permettant d'obtenir un schéma avec re-chiffrement combiné à partir de schémas avec re-chiffrements bidirectionnels infinis.

7.2.1 Le re-chiffrement combiné implique le re-chiffrement bidirectionnel infini et le re-chiffrement unidirectionnel unique

Cette implication signifie qu'à partir d'un schéma de serveur de re-chiffrement combiné, il est possible d'en obtenir deux avec re-chiffrement bidirectionnel et infini et un avec re-chiffrement unidirectionnel et unique. Soit un schéma de serveur de re-chiffrement combiné défini sur un ensemble $\mathsf{Type} = \{1, 2\}$ et $\mathsf{Liaison} = \{(1, 1), (1, 2), (2, 2)\}$. Il est alors possible de définir, par restriction sur celui-ci, trois schémas ainsi décrits :

- un schéma avec re-chiffrement bidirectionnel et infini sur un ensemble $\mathsf{Type} = \{1\}$ et $\mathsf{Liaison} = \{(1, 1)\}$;
- un schéma avec re-chiffrement unidirectionnel et unique sur un ensemble $\mathsf{Type} = \{1, 2\}$ et $\mathsf{Liaison} = \{(1, 2)\}$;
- un schéma avec re-chiffrement unidirectionnel et infini sur un ensemble $\mathsf{Type} = \{2\}$ et $\mathsf{Liaison} = \{(2, 2)\}$.

Propriété 36. *Chaque schéma défini ci-dessus atteint la même sécurité que le schéma de serveur de re-chiffrement.*

Preuve informelle. La preuve est évidente puisqu'un adversaire, ayant un avantage non négligeable pour attaquer l'un de ces trois schémas, aura le même avantage pour attaquer le schéma de serveur de re-chiffrement combiné. □

7.2.2 La réciproque est-elle vraie ?

Cette non-implication signifie que la réciproque à l'implication précédente est fausse et pour cela il suffit de trouver un contre-exemple. Pour autant, un tel contre-exemple est ici difficile à obtenir, puisqu'il faut réussir à trouver trois schémas sécurisés qui soient compatibles entre eux pour qu'ils puissent re-chiffrer successivement un même message. Nous nous apercevons qu'une telle compatibilité n'est, à notre connaissance, jamais atteinte par les schémas de la littérature. Pour autant, nous pouvons expliquer les raisons qui nous permettent d'imaginer pourquoi cette implication serait fausse.

En effet, un schéma dit sûr l'est dans un scénario précis avec un ensemble de moyens mis à disposition de l'adversaire. Ici, rien ne dit que si l'on donne des moyens supplémentaires à l'adversaire, alors le schéma restera sûr. Par exemple, un schéma avec re-chiffrement bidirectionnel n'offre généralement pas à l'adversaire de clés de re-chiffrement qui lui permettent d'obtenir un lien entre une entité honnête et une entité corrompue, contrairement à un re-chiffrement unidirectionnel. Nous ne pouvons donc pas affirmer que la considération des clés unidirectionnelles entre de telles entités puisse être acceptées par le modèle de sécurité du schéma avec re-chiffrement bidirectionnel.

7.2.3 Trois re-chiffrements bidirectionnels et infinis impliquent le re-chiffrement combiné

Nous présentons maintenant une construction générique permettant de transformer un schéma avec re-chiffrement bidirectionnel et infini - en réalité trois schémas - en un schéma de re-chiffrement combiné.

Idée de la construction. La première étape pour obtenir une construction générique de schéma de re-chiffrement combiné est de trouver trois schémas - deux avec re-chiffrement bidirectionnel infini et un avec re-chiffrement unidirectionnel unique - qui soient compatibles [1]. Pour cela, notre idée est de prendre comme point de départ un schéma bidirectionnel infini sur des ensembles $\mathsf{Type} = \{1\}$ et $\mathsf{Liaison} = \{(1,1)\}$ et de construire à partir de celui-ci un schéma avec re-chiffrement unidirectionnel unique [2] sur des ensembles $\mathsf{Type} = \{(1,2)\}$ et $\mathsf{Liaison} = \{(1,2)\}$ qui sera compatible avec le re-chiffrement bidirectionnel infini considéré. Nous reprenons donc l'idée d'utiliser une entité virtuelle `V` et obtenons qu'un re-chiffré de `A` vers `B` de manière unidirectionnelle soit de la forme $(\mathsf{Chiffrer}(pk_{\mathtt{V}}, m), \mathsf{Chiffrer}(pk_{\mathtt{B}}, sk_{\mathtt{V}}))$ où `V` est l'entité virtuelle utilisée dans la clé de re-chiffrement de `A` vers `B`. Nous avons mentionné au Chapitre 4 qu'un schéma produisant un tel chiffré ne pouvait atteindre une sécurité IND-CCA sans restriction non naturelle et l'avons constaté sur le schéma [CWYD10] au Chapitre 6. Pour rectifier ce problème, nous avions alors chiffré dans ce dernier schéma la composante F en F' en utilisant un chiffrement Hash-ElGamal pour `B`. Nous généralisons cette méthode en chiffrant la partie correspondant au chiffré de m destiné à `V` avec un schéma de chiffrement à clé publique IND-CCA sûr. Il n'y a aucune raison de chiffrer la deuxième composante, à savoir le chiffré de $sk_{\mathtt{V}}$ pour `B`, puisque cette dernière est déjà chiffrée pour B. Dans la construction que nous détaillerons ensuite, nous généraliserons certains éléments :

- les chiffrés Hash-ElGamal utilisés dans la construction du Chapitre 6.
- les clés de re-chiffrement bidirectionnelles qui ne sont pas forcément de la forme $sk_{\mathtt{B}}/sk_{\mathtt{A}}$.

Nous obtenons de cette manière deux des trois re-chiffrements nécessaires pour notre construction. Il ne manque plus qu'à concevoir un re-chiffrement sur la liaison $\{(2,2)\}$. Pour l'obtenir, nous choisissons d'utiliser, à la place des chiffrements à clé publique que l'on a considéré, des chiffrements de serveur de re-chiffrement bidirectionnels qui peuvent alors être re-chiffrés. Ces chiffrement sont ceux utiliser pour protéger la clé $sk_{\mathtt{V}}$ pour `B` et le chiffré de m pour `V`. Nous avons ainsi trois re-chiffrements compatibles qui vont nous permettre de construire un re-chiffrement combiné sécurisé. Nous détaillons maintenant cette construction.

La construction générique.

Soit PRE_α, PRE_β et PRE_γ, trois schémas de serveur de re-chiffrement bidirectionnel infini dont les ensembles $(\mathsf{Type}, \mathsf{Liaison})$ sont respectivement $\big(\{1_\alpha\}, \{(1_\alpha, 1_\alpha)\}\big)$, $\big(\{1_\beta\}, \{(1_\beta, 1_\beta)\}\big)$ et $\big(\{1_\gamma\}, \{(1_\gamma, 1_\gamma)\}\big)$ tels que :

- l'espace des chiffrés de type 1_α soit inclus dans l'espace des messages du schéma PRE_β ;
- l'espace des clés privées du schéma PRE_α soit inclus dans l'espace des messages du schéma PRE_γ.

Nous omettons une fois de plus les paramètres publics afin de faciliter la compréhension de notre construction, qui se compose comme suit.

$\mathsf{Environnement}$: il est composé des ensembles $\mathsf{Type} = \{1,2\}$ et $\mathsf{Liaison} = \{(1,2)\}$ - en particulier le type 1 correspond au type 1_α. Quant aux données requises pour calculer une clé de re-chiffrement, nous avons trois liaisons donc six ensembles formels de données décrits

1. Ceci ne contredit en aucun cas le point précédent, puisqu'ici en plus d'être compatible avec les re-chiffrements bidirectionnels infinis, le re-chiffrement unidirectionnel unique aura une forme particulière.
2. Cette étape nous donnera la construction générique pour passer d'un re-chiffrement bidirectionnel à un re-chiffrement unidirectionnel dont nous avons déjà mentionné l'existence au Chapitre 4.

ainsi :
- pour la liaison $(1,1)$, $\mathsf{D}_{(1,1),de} = \mathsf{PRE}_\alpha.\mathsf{D}_{(1_\alpha,1_\alpha),de}$ et $\mathsf{D}_{(1,1),vers} = \mathsf{PRE}_\alpha.\mathsf{D}_{(1_\alpha,1_\alpha),de}$;
- pour la liaison $(1,2)$, $\mathsf{D}_{(1,2),de} = \mathsf{PRE}_\alpha.\mathsf{D}_{(1_\alpha,1_\alpha),de}$ et $\mathsf{D}_{(1,2),vers} = \{\texttt{clé publique}\}$;
- pour la liaison $(2,2)$, $\mathsf{D}_{(2,2),de} = \mathsf{PRE}_\beta.\mathsf{D}_{(1_\beta,1_\beta),de} \cup \mathsf{PRE}_\gamma.\mathsf{D}_{(1_\gamma,1_\gamma),de}$ et $\mathsf{D}_{(2,2),vers} = \mathsf{PRE}_\beta.\mathsf{D}_{(1_\beta,1_\beta),vers} \cup \mathsf{PRE}_\gamma.\mathsf{D}_{(1_\gamma,1_\gamma),vers}$.

$\mathsf{Clé}()$: cet algorithme exécute $\mathsf{PRE}_\alpha.\mathsf{Clé}$, $\mathsf{PRE}_\beta.\mathsf{Clé}$ et $\mathsf{PRE}_\gamma.\mathsf{Clé}$ pour obtenir respectivement une paire de clés (privée/publique) $(sk_{\mathsf{PRE}_\alpha}, pk_{\mathsf{PRE}_\alpha})$, $(sk_{\mathsf{PRE}_\beta}, pk_{\mathsf{PRE}_\beta})$ et $(sk_{\mathsf{PRE}_\gamma}, pk_{\mathsf{PRE}_\gamma})$. Il retourne la clé privée sk et la clé publique pk où $sk = (sk_{\mathsf{PRE}_\alpha}, sk_{\mathsf{PRE}_\beta}, sk_{\mathsf{PRE}_\gamma})$ et $pk = (pk_{\mathsf{PRE}_\alpha}, pk_{\mathsf{PRE}_\beta}, pk_{\mathsf{PRE}_\gamma})$.

Les quatres algorithmes suivants correspondent aux algorithmes du schéma PRE_α.

$\mathsf{Chiffrer}_1(pk, m)$: prend en entrée une clé publique $pk = (pk_{\mathsf{PRE}_\alpha}, pk_{\mathsf{PRE}_\beta}, pk_{\mathsf{PRE}_\gamma})$ et un message m. Cet algorithme retourne un chiffré $C_{pk,1} = \mathsf{PRE}_\alpha.\mathsf{Chiffrer}_{1_\alpha}(pk_{\mathsf{PRE}_\alpha}, m)$ de type 1 pour pk.

$\mathsf{Délégation}_{(1,1)}(\mathsf{D}^{\mathtt{A}}_{(1,1),de}, \mathsf{D}^{\mathtt{B}}_{(1,1),vers})$: prend en entrée les ensembles $\mathsf{D}^{\mathtt{A}}_{(1,1),de} = \mathsf{PRE}_\alpha.\mathsf{D}^{\mathtt{A}}_{(1_\alpha,1_\alpha),de}$ et $\mathsf{D}^{\mathtt{B}}_{(1,1),vers} = \mathsf{PRE}_\alpha.\mathsf{D}^{\mathtt{B}}_{(1_\alpha,1_\alpha),vers}$. Cet algorithme calcule la clé de re-chiffrement :

$$R_{\mathtt{A},1\to\mathtt{B},2} = \mathsf{PRE}_\alpha.\mathsf{Délégation}_{(1_\alpha,1_\alpha)}(\mathsf{PRE}_\alpha.\mathsf{D}^{\mathtt{A}}_{(1_\alpha,1_\alpha),de}, \mathsf{PRE}_\alpha.\mathsf{D}^{\mathtt{B}}_{(1_\alpha,1_\alpha),vers}).$$

$\mathsf{Re\text{-}chiffrer}_{(1,1)}(R_{\mathtt{A},1\to\mathtt{B},1}, C_{pk_\mathtt{A},1})$: prend en entrée une clé de re-chiffrement $R_{\mathtt{A},1\to\mathtt{B},2}$ et un chiffré $C_{pk_\mathtt{A},1}$. Cet algorithme retourne le chiffré :

$$C_{pk_\mathtt{B},1} = \mathsf{PRE}_\alpha.\mathsf{Re\text{-}chiffrer}_{(1_\alpha,1_\alpha)}(R_{\mathtt{A},1\to\mathtt{B},2}, C_{pk_\mathtt{A},1}).$$

$\mathsf{Déchiffrer}_1(sk, C_{pk,1})$: prend en entrée une clé privée $sk = (sk_{\mathsf{PRE}_\alpha}, sk_{\mathsf{PRE}_\beta}, sk_{\mathsf{PRE}_\gamma})$ et un chiffré $C_{pk,1}$ pour pk, la clé publique associée à sk. Cet algorithme retourne le message $m = \mathsf{PRE}_\alpha.\mathsf{Déchiffrer}_{1_\alpha}(sk_{\mathsf{PRE}_\alpha}, C_{pk,1})$.

$\mathsf{Délégation}_{(1,2)}(\mathsf{D}^{\mathtt{A}}_{(1,2),de}, \mathsf{D}^{\mathtt{B}}_{(1,2),vers})$: prend en entrée les ensembles $\mathsf{D}^{\mathtt{A}}_{(1,2),de} = \mathsf{PRE}_\alpha.\mathsf{D}^{\mathtt{A}}_{(1,2),de}$ et $\mathsf{D}^{\mathtt{B}}_{(1,2),vers} = \{pk_\mathtt{B}\}$. Cet algorithme calcule la clé de re-chiffrement ainsi :
1. Génère une paire de clés $(sk_\mathtt{V}, pk_\mathtt{V})$ pour une entité virtuelle $\mathtt{V}$ en utilisant l'algorithme $\mathsf{PRE}_\alpha.\mathsf{Clé}$;
2. Calcule la clé de re-chiffrement de $\mathtt{A}$ vers $\mathtt{V}$: $\tilde{R}_{\mathtt{A},1\to\mathtt{B},2} = \mathsf{PRE}_\alpha.\mathsf{Délégation}_{(1_\alpha,1_\alpha)}(\mathsf{D}^{\mathtt{A}}_{(1_\alpha,1_\alpha),de}, \mathsf{D}^{\mathtt{V}}_{(1_\alpha,1_\alpha),vers})$;
3. Chiffre $sk_\mathtt{V}$ pour $\mathtt{B}$ en $R_{\mathtt{A},\mathtt{B}} = \mathsf{PRE}_\gamma.\mathsf{Chiffrer}_{1_\gamma}(pk_{\mathsf{PRE}_\gamma,\mathtt{B}}, sk_\mathtt{V})$;
4. Retourne la clé de re-chiffrement $\mathcal{R}_{\mathtt{A},1\to\mathtt{B},2} = (\tilde{R}_{\mathtt{A},1\to\mathtt{B},2}, R_{\mathtt{A},\mathtt{B}}, pk_{\mathsf{PRE}_\beta,\mathtt{B}})$.

$\mathsf{Re\text{-}chiffrer}_{(1,2)}(R_{\mathtt{A},1\to\mathtt{B},2}, C_{pk_\mathtt{A},1})$: prend en entrée une clé de re-chiffrement $R_{\mathtt{A},1\to\mathtt{B},2} = (\tilde{R}_{\mathtt{A},1\to\mathtt{B},2}, R_{\mathtt{A},\mathtt{B}}, pk_{\mathsf{PRE}_\beta,\mathtt{B}})$ et un chiffré $C_{pk_\mathtt{A},1}$. Cet algorithme fonctionne ainsi :
1. Re-chiffre $C_{pk_\mathtt{A},1}$ de $\mathtt{A}$ vers $\mathtt{V}$ en $C_{pk_\mathtt{V},1} = \mathsf{PRE}_\alpha.\mathsf{Re\text{-}chiffrer}_{(1_\alpha,1_\alpha)}(\tilde{R}_{\mathtt{A},1\to\mathtt{B},2}, C_{pk_\mathtt{A},1})$;
2. Chiffre ce dernier pour $\mathtt{B}$ en $D_{pk_\mathtt{B}} = \mathsf{PRE}_\beta.\mathsf{Chiffrer}(pk_{\mathsf{PRE}_\beta,\mathtt{B}}, C_{pk_\mathtt{V},1})$;
3. Retourne le chiffré $C_{pk_\mathtt{B},2} = (D_{pk_\mathtt{B}}, R)$ de type 2 pour $\mathtt{B}$ où $R = R_{\mathtt{A},\mathtt{B}}$.

$\mathsf{Chiffrer}_2(pk, m)$: cet algorithme peut être défini en utilisant une entité temporaire $\mathtt{T}$. Le message est ainsi chiffré pour $\mathtt{T}$ et re-chiffré ensuite de $\mathtt{T}$ vers $\mathtt{B}$ comme exposé dans la Remarque 5 du Chapitre 3 puisque $\mathsf{D}_{(1,2),vers}$ n'est composé que d'éléments publics.

$\mathsf{Délégation}_{(2,2)}(\mathsf{D}^{\mathtt{A}}_{(2,2),de}, \mathsf{D}^{\mathtt{B}}_{(2,2),vers})$: prend en entrée les ensembles $\mathsf{D}^{\mathtt{A}}_{(2,2),de} = \mathsf{PRE}_\beta.\mathsf{D}^{\mathtt{A}}_{(1_\beta,1_\beta),de} \cup \mathsf{PRE}_\gamma.\mathsf{D}^{\mathtt{A}}_{(1_\gamma,1_\gamma),de}$ et $\mathsf{D}^{\mathtt{B}}_{(2,2),vers} = \mathsf{PRE}_\beta.\mathsf{D}^{\mathtt{B}}_{(1_\beta,1_\beta),vers} \cup \mathsf{PRE}_\gamma.\mathsf{D}^{\mathtt{B}}_{(1_\gamma,1_\gamma),vers}$. Cet algorithme fonctionne ainsi :
1. Calcule $R_{\mathtt{A},1_\beta\to\mathtt{B},1_\beta} = \mathsf{PRE}_\beta.\mathsf{Délégation}(\mathsf{PRE}_\beta.\mathsf{D}^{\mathtt{A}}_{(1_\beta,1_\beta),de}, \mathsf{PRE}_\beta.\mathsf{D}^{\mathtt{B}}_{(1_\beta,1_\beta),vers})$;
2. Calcule $R_{\mathtt{A},1_\gamma\to\mathtt{B},1_\gamma} = \mathsf{PRE}_\gamma.\mathsf{Délégation}(\mathsf{PRE}_\gamma.\mathsf{D}^{\mathtt{A}}_{(1_\gamma,1_\gamma),de}, \mathsf{PRE}_\gamma.\mathsf{D}^{\mathtt{B}}_{(1_\gamma,1_\gamma),vers})$;
3. Retourne la clé de re-chiffrement $R_{\mathtt{A},2\to\mathtt{B},2} = (R_{\mathtt{A},1_\beta\to\mathtt{B},1_\beta}, R_{\mathtt{A},1_\gamma\to\mathtt{B},1_\gamma})$.

$\mathsf{Re\text{-}chiffrer}_{(2,2)}(R_{\mathtt{A},2\to\mathtt{B},2}, C_{pk_\mathtt{A},2})$: prend en entrée $R_{\mathtt{A},2\to\mathtt{B},2} = (R_{\mathtt{A},1_\beta\to\mathtt{B},1_\beta}, R_{\mathtt{A},1_\gamma\to\mathtt{B},1_\gamma})$, une clé de re-chiffrement et un chiffré $C_{pk_\mathtt{A},2} = (D_{pk_\mathtt{A}}, R)$. Cet algorithme fonctionne ainsi :

1. $D_{pk_\mathtt{B}} = \mathsf{PRE}_\beta.\mathsf{Re\text{-}chiffrer}(1_\beta, 1_\beta)(R_{\mathtt{A},1_\beta\to\mathtt{B},1_\beta}, D_{pk_\mathtt{A}})$;
2. $R' = \mathsf{PRE}_\gamma.\mathsf{Re\text{-}chiffrer}_{(1_\gamma,1_\gamma)}(R_{\mathtt{A},1_\gamma\to\mathtt{B},1_\gamma}, R)$;
3. Retourne le chiffré $C_{pk_\mathtt{B},2} = (D_{pk_\mathtt{B}}, R')$.

$\mathsf{Déchiffrer}_2(\mathcal{P}, sk, C_{pk,2})$: prend en entrée une clé privée $sk = (sk_{\mathsf{PRE}_\alpha}, sk_{\mathsf{PRE}_\beta}, sk_{\mathsf{PRE}_\gamma})$ et un chiffré $C_{pk,2} = (D_{pk}, R)$ de type 2 pour pk, la clé publique associée à sk. Cet algorithme déchiffre comme suit :

1. Déchiffre R en utilisant sk_{PRE_γ} : $sk_\mathtt{V} = \mathsf{PRE}_\gamma.\mathsf{Déchiffrer}_{1_\gamma}(sk_{\mathsf{PRE}_\gamma}, R)$;
2. Déchiffre D_{pk} en utilisant sk_{PRE_β} : $C_{pk_\mathtt{V},1} = \mathsf{PRE}_\beta.\mathsf{Déchiffrer}_{1_\beta}(sk_{\mathsf{PRE}_\beta}, D_{pk})$;
3. Déchiffre $C_{pk_\mathtt{V},1}$ en utilisant $sk_\mathtt{V}$: $m = \mathsf{PRE}_\alpha.\mathsf{Déchiffrer}_{1_\alpha}(sk_\mathtt{V}, C_{pk_\mathtt{V},1})$;
4. Retourne le message m.

Ce schéma est consistant si les différents schémas PRE_α, PRE_β et PRE_γ le sont.

Théorème 37. *Le schéma atteint respectivement la sécurité* IND-RCCA$_\mathrm{B}$*, à condition que les schémas* PRE_α*,* PRE_β *et* PRE_γ *atteignent chacun respectivement la sécurité* IND-RCCA$_\mathrm{B}$.

Preuve informelle. Il est facile de voir que le schéma est sûr contre une attaque CCA (avec d'éventuelles restrictions non naturelles) puisque chaque chiffré produit est composé respectivement d'au moins un chiffrement CCA$_\mathrm{B}$. Les preuves à effectuer - une pour chaque chaîne de liaisons possible - sont classiques et il ne nous paraît pas utile de donner plus de détails.

La restriction RCCA vient du fait que les deux composantes d'un chiffré de type 2 ne sont pas reliées par des moyens qui peuvent être publiquement vérifiés. Cette caractéristique implique que l'adversaire peut utiliser l'oracle de re-chiffrement sur la liaison $(2, 2)$ pour attaquer le schéma. En effet, un chiffré challenge de type 2 pour pk^* est de la forme $C^* = (D_{pk^*}, R^*)$. L'algorithme de chiffrement ne nécessitant que des éléments publics, l'adversaire peut alors utiliser chacune des composantes pour créer deux chiffrés non valides de type 2 pour l'entité $\mathtt{A}$:

$$C_1 = (D_{pk^*}, \mathsf{PRE}_\gamma.\mathsf{Chiffrer}_{1_\gamma}(pk^*_{\mathsf{PRE}_\gamma}, sk)) \text{ et}$$
$$C_2 = (\mathsf{PRE}_\beta.\mathsf{Chiffrer}_{1_\beta}(pk^*_{\mathsf{PRE}_\beta}, \mathsf{PRE}_\alpha.\mathsf{Chiffrer}_{1_\alpha}(pk, m)), R^*)$$

où (sk, pk) est généré avec $\mathsf{PRE}_\alpha.\mathsf{Clé}$ et m est un message quelconque. Ces deux chiffrés C_1 et C_2 ne sont pas valides pour l'algorithme de déchiffrement $\mathsf{Déchiffrer}_2$. En revanche ils le sont pour l'algorithme $\mathsf{Re\text{-}chiffrer}_{(2,2)}$. D'après le scénario IND-CCA$_\mathrm{B}$, le seul type de requête que l'adversaire peut effectuer sur la liaison $(2, 2)$ et qui implique l'entité pk^*, est de demander des re-chiffrements de pk^* pour pk honnête. Il peut ainsi obtenir les re-chiffrements de C_1 et C_2 pour pk, ce qui lui donne deux chiffrés de la forme $C'_1 = (\mathsf{PRE}_\beta.\mathsf{Re\text{-}chiffrer}_{(1_\beta,1_\beta)}(R_{pk^*_{\mathsf{PRE}_\beta},1_\beta\to pk_{\mathsf{PRE}_\beta},1_\beta}, D_{pk^*}), *)$ et $C'_2 = (*, \mathsf{PRE}_\gamma.\mathsf{Re\text{-}chiffrer}_{(1_\gamma,1_\gamma)}(R_{pk^*_{\mathsf{PRE}_\gamma},1_\gamma\to pk_{\mathsf{PRE}_\gamma},1_\gamma}, R^*)$. Ces deux composantes lui permettent d'obtenir un re-chiffré du challenge :

$$C'^* = \big(\mathsf{PRE}_\beta.\mathsf{Re\text{-}chiffrer}_{(1_\beta,1_\beta)}(R_{pk^*_{\mathsf{PRE}_\beta},1_\beta\to pk_{\mathsf{PRE}_\beta},1_\beta}, D_{pk^*}),$$
$$\mathsf{PRE}_\gamma.\mathsf{Re\text{-}chiffrer}_{(1_\gamma,1_\gamma)}(R_{pk^*_{\mathsf{PRE}_\gamma},1_\gamma\to pk_{\mathsf{PRE}_\gamma},1_\gamma}, R^*)\big),$$

bien qu'il n'ait pas été produit par l'algorithme $\mathsf{Re\text{-}chiffrer}_{(2,2)}$. Ainsi, l'adversaire a théoriquement le droit d'obtenir son déchiffrement, ce qui lui permet de retrouver le message challenge.

Afin d'empêcher cette éventualité, nous pouvons nous placer dans le scénario qui prend en compte la restriction RCCA, puisque l'adversaire n'aura pas le droit de demander le déchiffrement de C'^*. Une autre solution serait de considérer un scénario CCA$_\mathrm{B}$ avec en plus la restriction non naturelle que l'adversaire n'ait pas le droit de demander un re-chiffrement pour une requête portant sur un chiffré de type 2 qui soit différent du challenge et dont l'une des deux composantes corresponde à D_{pk^*} ou à R^*. □

Remarque 8. Il est bien sûr possible d'obtenir un schéma avec re-chiffrement combiné IND-CCA$_{\mathrm{B}}$ sûr sans la restriction RCCA. Pour cela, il faudrait modifier la construction et trouver un moyen qui permette de vérifier publiquement que les deux composantes d'un chiffré de type 2 sont liées entre elles. Nous pourrions nous servir du même genre de preuve que celle utilisée dans le Chapitre 5 au détriment de l'efficacité du schéma.

Construction générique de schémas avec re-chiffrement unidirectionnel et unique. En restreignant le schéma obtenu à son re-chiffrement sur la liaison $(1, 2)$, nous obtenons ainsi un schéma avec re-chiffrement unidirectionnel et unique. On s'aperçoit que les re-chiffrements des schémas PRE_β et PRE_γ n'interviennent que dans les algorithmes Délégation et Re-chiffrer de la liaison $(2, 2)$ et sont donc inutiles si l'on ne s'intéresse qu'au re-chiffrement unidirectionnel et unique. C'est également le cas du re-chiffrement infini du schéma PRE_α. En ne considérant que l'algorithme Chiffrer de ces deux schémas, nous obtenons une construction générique qui transforme un schéma avec re-chiffrement bidirectionnel - ici PRE_α - en un schéma avec re-chiffrement unidirectionnel.

Théorème 38. *Le schéma unidirectionnel ainsi obtenu atteint la sécurité* IND-CCA*, à condition que le schéma* PRE_α *atteigne la sécurité* IND-CCA$_{\mathrm{B}}$ *et que les restrictions respectives des schémas* PRE_β *et* PRE_γ *en des schémas de chiffrement à clé publique sans re-chiffrement atteignent la sécurité* IND-CCA*.*

Preuve informelle. La preuve de ce théorème s'obtient de la même manière que dans le théorème précédent. □

Remarque 9. Nous remarquons qu'en effectuant cette transformation, nous obtenons un schéma avec re-chiffrement unidirectionnel et unique **sans restriction non naturelle** bien que le schéma PRE_α prenne en compte les restrictions récurrentes liées au re-chiffrement bidirectionnel. Ce théorème prouve l'implication de la Section 4.2.2.

7.3 Une construction pratique

7.3.1 Une version combinée du schéma Libert-Vergnaud

Bien qu'efficace, la construction précédente ne propose pas de re-chiffrements qui s'intègrent les uns aux autres, puisqu'elle utilise trois schémas de re-chiffrement indépendants (si ce n'est que certaines clés privées ou certains chiffrés d'un schéma doivent être inclus dans l'espace des messages d'un autre). Par exemple, le re-chiffrement sur la liaison $(1, 2)$ ne se contente, en quelque sorte, que de superposer les chiffrements du schéma PRE_α et PRE_β. Nous proposons maintenant une construction plus élégante basée sur le schéma [LV08b].

Cette construction consiste à ajouter à ce schéma les deux re-chiffrements bidirectionnels infinis qui lui font défaut pour en faire un schéma avec re-chiffrement combiné. Comme nous l'avons vu, le schéma [LV08b] est notamment composé d'un chiffrement ElGamal en environnement bilinéaire. Notre idée consiste à reprendre l'idée du re-chiffrement du schéma [BBS98] et de l'adapter aux chiffrés dont nous disposons. Plus précisémment, l'opération de re-chiffrement bidirectionnel remplace un élément de la forme $h^{r.sk_{\mathtt{A}}}$ en un élément $h^{r.sk_{\mathtt{B}}}$ en utilisant la clé de re-chiffrement $sk_{\mathtt{B}}/sk_{\mathtt{A}}$, où $h \in \{g, e(g, g)\}$ selon le type considéré. Afin que ce re-chiffrement soit correct et puisse atteindre une sécurité RCCA, sécurité du schéma [LV08b], nous ajoutons les mécanismes de vérification publique d'un chiffré exposé par Libert et Vergnaud. Le schéma alors obtenu est détaillé dans la Figure 7.3 où nous avons omis l'algorithme Déchiffrer puisqu'il est inchangé par rapport au schéma de base. Il est évident que, par la nature des clés $sk_{\mathtt{B}}/sk_{\mathtt{A}}$, il faut prendre en compte les restrictions non naturelles récurrentes pour les re-chiffrements bidirectionnels, d'où la sécurité RCCA$_{\mathrm{B}}$.

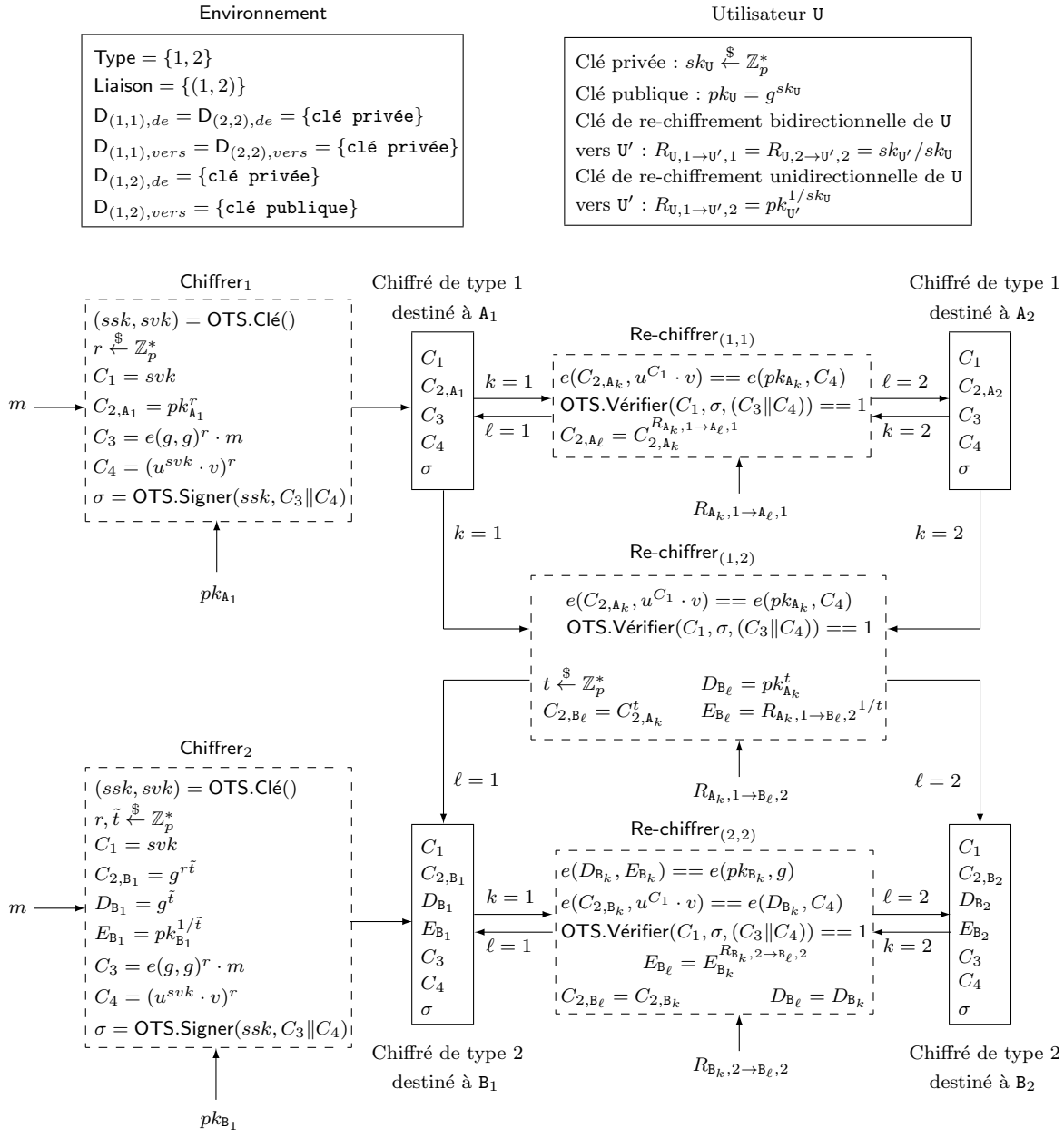

Figure 7.3 – Le schéma [LV08b] combiné sans l'algorithme Déchiffrer

Théorème 39. *Le schéma atteint la sécurité* IND-RCCA$_B$ *sous l'hypothèse* 3-QDBDH.

Preuve informelle. La preuve de ce théorème suit le même raisonnement que celle exposée dans [LV08b]. La grande différence est que, contrairement au schéma de base, nous n'avons pas à considérer trois manières de produire un chiffré mais une infinité. Par exemple, un chiffré peut être obtenu en suivant la chaîne de liaisons $(1, 2)$ ou $(1, 1, 2)$,... Il faudrait donc, a priori, effectuer une preuve pour chacune de ces chaînes. Nous disons bien a priori, car les re-chiffrements bidirectionnels infinis sont d'une certaine forme et ne changent pas celle du chiffré. Par exemple, nous voyons bien que les chiffrés de type 1 produits par l'algorithme Chiffrer$_1$ ou par l'algorithme Re-chiffrer$_{(1,1)}$ sont indiscernables et n'impliquent que la donnée du destinataire - il n'y pas dans un re-chiffré de type 1 d'éléments liés au délégant. Ceci permet de réduire le nombre de sécurités à prouver.

Observons par exemple la sécurité pour la chaîne de liaisons $(1, 1)$. Dans ce cas et en notant B

l'entité cible habituellement notée pk^*, le challenge est de la forme $C_{pk_{\mathtt{B}},2}$ et est produit comme un re-chiffré de $C_{pk_{\mathtt{A}},2}$. Ce dernier peut également être vu comme un re-chiffré de $C_{pk_{\mathtt{B}},2}$ en utilisant la clé $R_{\mathtt{B},2\to\mathtt{A},2}$, c'est pourquoi en considérant l'un des deux challenges, on obtient aussi l'autre. De plus, les clés de re-chiffrement étant bidirectionnelles, les restrictions non naturelles qui les concernent sont similaires - il faut juste remplacer `A` par `B` et vice-versa. Ainsi, il est possible de démontrer que le schéma est sûr pour la chaîne (1), si et seulement si il l'est pour la chaîne $(1,1)$, si et seulement si il l'est pour la chaîne $(1,1,1)$ et ainsi de suite. En appliquant ce principe à toutes les chaînes possibles, nous ne devons effectuer une preuve de sécurité que pour trois chaînes de liaison, à savoir (1), $(1,2)$ et (2). □

Remarque 10. Le schéma obtenu prend donc en considération la restriction RCCA. Nous avons présenté au Chapitre 5 une amélioration du schéma [LV08b] afin de le rendre CCA tout en restant efficace. Cette amélioration n'est pas compatible avec cette version combinée, puisqu'un chiffré de niveau 2 de la version CCA n'est pas re-chiffrable, même avec une clé bidirectionnelle. En détail et avec les notations que nous avons utilisées, la composante $C_{2,\mathtt{B}}$ est chiffrée dans D_1 et le serveur ne peut donc la re-chiffrer. En revanche, on s'aperçoit qu'un chiffré de type 1 de la version CCA est re-chiffrable bidirectionnellement. Ainsi si nous souhaitons avoir un schéma combiné "restreint" (sans le re-chiffrement bidirectionnel sur le type 2), nous pouvons alors mélanger la version CCA du schéma [LV08b] avec sa version combinée.

7.3.2 Propriétés additionnelles

Minimalisation des clés de re-chiffrement nécessaires. De par la description d'un schéma de serveur de re-chiffrement combiné, il y a plusieurs chemins possibles pour passer d'un dispositif à un autre. Par exemple pour passer de $\mathtt{A}_1$ à $\mathtt{B}_2$, le serveur peut choisir de transiter par $\mathtt{A}_2$, par $\mathtt{B}_1$, mais également par $\mathtt{A}_2$, puis $\mathtt{B}_1$ pour enfin arriver à $\mathtt{B}_2$. Il y a donc tout un ensemble de clés de re-chiffrement possibles si l'on souhaite obtenir une gestion dynamique des droits d'accès. Cela peut être problématique, puisque si Alice souhaite que Charlie devienne son délégué, il faut qu'elle ajoute une clé de re-chiffrement de chacun de ses dispositifs vers chaque dispositif de Charlie, ce qui peut s'avérer fastidieux. Le schéma que nous venons de présenter permet de limiter le nombre de clés de re-chiffrement à prendre en compte. En effet, ces clés ont des formes particulières, ce qui nous permet d'en déduire facilement, pour tout dispositif `A`, `B`, `C` et `D`, les formules suivantes :

$$\begin{aligned} R_{\mathtt{A},1\to\mathtt{C},1} &= R_{\mathtt{A},1\to\mathtt{B},1}\cdot R_{\mathtt{B},1\to\mathtt{C},1}\\ R_{\mathtt{A},2\to\mathtt{C},2} &= R_{\mathtt{A},2\to\mathtt{B},2}\cdot R_{\mathtt{B},1\to\mathtt{C},2}\\ R_{\mathtt{A},1\to\mathtt{D},2} &= R_{\mathtt{B},1\to\mathtt{C},2}^{R_{\mathtt{C},2\to\mathtt{D},2}/R_{\mathtt{A},1\to\mathtt{B},1}} \end{aligned}$$

Les deux premières formules sont très fréquentes pour les re-chiffrements bidirectionnels. La troisième signifie que la clé de re-chiffrement unidirectionnelle sur la liaison $(2,2)$ de `A` vers `D` peut être calculée à partir de la clé bidirectionnelle sur la liaison $(1,1)$ de `A` vers `B`, de la clé unidirectionnelle sur la liaison $(1,2)$ de `B` vers `C` et de la clé bidirectionnelle sur la liaison $(2,2)$ de `C` vers `D`.

Ces formules permettent de réduire le nombre de clés de re-chiffrement à calculer et à fournir au serveur. Par exemple, si Alice dispose de n dispositifs $\mathtt{A}_1,\cdots,\mathtt{A}_n$, il suffit alors que le serveur dispose de $n-1$ clés de re-chiffrement pour relier tous ces dispositifs entre eux de n'importe quelle manière. Ces $n-1$ clés ne doivent par contre pas être quelconques et il faut que l'ensemble soit bien représentatif. En effet, chaque dispositif doit intervenir au moins une fois dans une de ces clés. Les ensembles de clés de $\mathtt{A}_i$ vers $\mathtt{A}_{i+1}$ pour $i\in[\![1;n-1]\!]$ ou de $\mathtt{A}_1$ vers $\mathtt{A}_i$ pour $i\in[\![2;n]\!]$ sont des exemples de tels ensembles représentatifs. La mise en œuvre d'un tel choix est très facile, puisqu'il suffit, à chaque fois qu'Alice décide d'enregistrer un nouveau dispositif dans le système, qu'elle ajoute une clé de ce nouveau dispositif vers un dispositif déjà enregistré. Ce raisonnement est également valable pour chacun de ses délégués - ici Bob. En utilisant la troisième formule, on

voit bien qu'Alice n'aura également qu'à calculer une seule et unique clé de re-chiffrement à partir de n'importe lequel de ses dispositifs vers n'importe lequel des dispositifs de Bob pour que tous les chemins possibles de re-chiffrement puissent être réalisés. Il faut cependant être prudent lors de la suppression des droits d'accès. Considérons, par exemple, que le serveur dispose de deux clés : celle de $\mathtt{A}_1$ vers $\mathtt{A}_2$ et celle de $\mathtt{A}_2$ vers $\mathtt{B}_1$. Dans ce cas, si Alice veut supprimer les droits d'accès de $\mathtt{A}_2$, nous avons vu que le serveur supprimait les clés impliquant $\mathtt{A}_2$, ce qui empêcherait aussi $\mathtt{B}_1$ d'accéder aux données d'Alice. Le serveur doit donc calculer et stocker, en utilisant les formules précédentes, la clé de $\mathtt{A}_1$ vers $\mathtt{B}_1$ avant de supprimer celles qui impliquent $\mathtt{A}_2$.

Extension : Un serveur par utilisateur. Les clés de re-chiffrement bidirectionnelles sont plus sensibles que les clés de re-chiffrement unidirectionnelles, puisque, comme nous l'avons vu, il est possible à partir de x et y/x de retrouver y. Il faut donc être sûr, lorsque l'on perd un dispositif - typiquement lors d'un vol -, que ces clés soient supprimées et qu'elles ne finissent pas dans les mains d'un individu malveillant. Une confiance de chaque utilisateur dans le serveur de stockage est donc requise, ce qui n'est pas nécessairement le cas. Il est en effet possible que Bob ne sache même pas quel serveur Alice utilise.

Pour répondre à cette problématique, nous pouvons adapter le système de stockage des clés en considérant un serveur par utilisateur (en lequel ce dernier a confiance). Chaque utilisateur peut alors donner les clés bidirectionnelles impliquant ses dispositifs à son serveur de confiance. Le fonctionnement d'un tel système est détaillé dans la Figure 7.4.

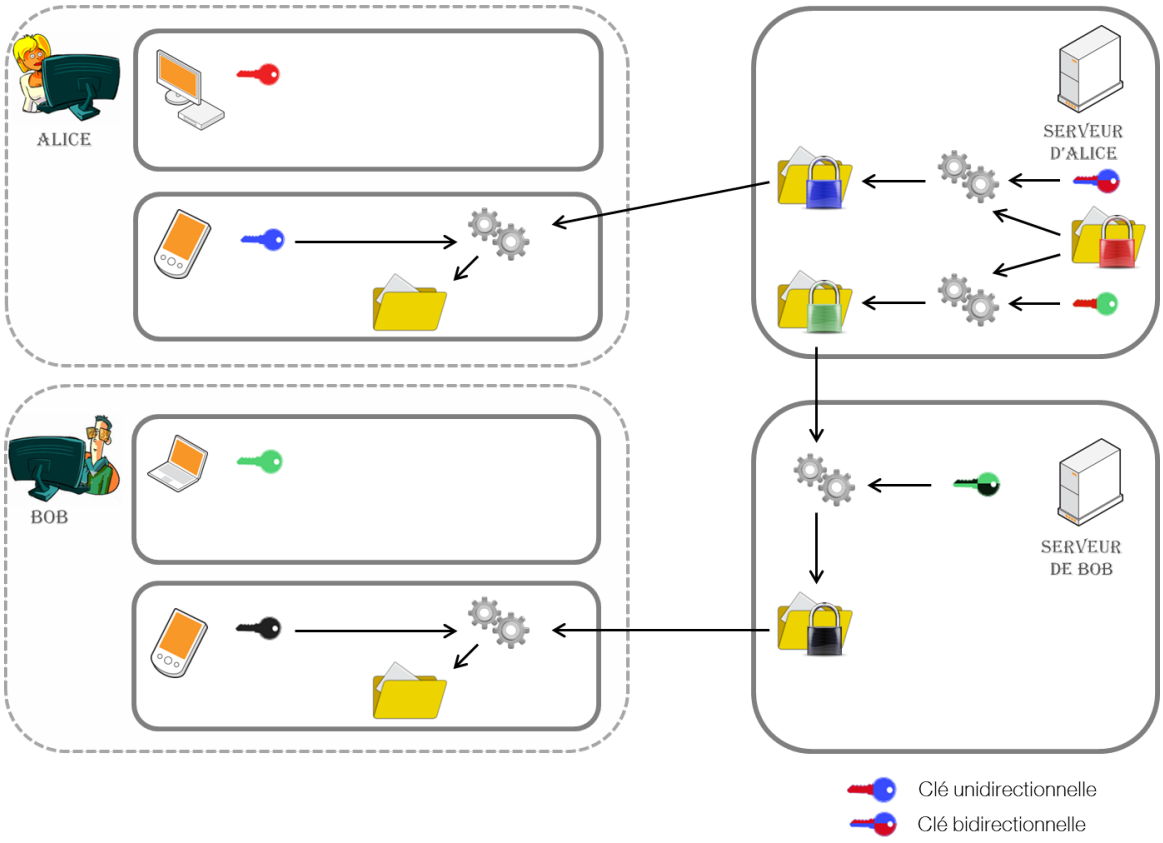

Figure 7.4 – Stockage de données avec un serveur par utilisateur

Cette fonctionnalité est possible avec tout schéma de serveur de re-chiffrement combiné. La particularité qu'offre le schéma [LV08b] combiné est que lors de la suppression du droit d'accès du dispositif $\mathtt{A}_1$ (clé rouge), Alice n'a pas besoin de fournir une clé de son deuxième dispositif vers un des dispositifs de Bob, comme ce serait le cas avec un schéma ne possédant pas une telle

minimisation du nombre de clés. En effet, en utilisant les formules sur les clés de re-chiffrement exposées ci-dessus, le serveur d'Alice est capable de calculer la clé de $\mathtt{A}_2$ (clé bleue) vers le dispositif $\mathtt{B}_1$ (clé verte) à partir des clés mises à disposition.

Note sur un serveur de re-chiffrement combiné généralisé

Nous venons de voir que le serveur de re-chiffrement combiné était intéressant et permettait d'envisager une gestion dynamique des droits d'accès sans accroîssement de la complexité. Il peut cependant y avoir des cas d'usage où le type de partage offert ne soit pas suffisant. Nous constatons que le partage est limité, car il n'est pas transitif. En effet, Bob ne peut transmettre ses droits d'accès à un délégué en qui il a confiance, disons Charlie. Si Bob veut que Charlie obtienne les données, il devra intervenir à chaque requête de Charlie pour récupérer les données en clair et les lui envoyer.

Il serait alors d'intéressant d'étendre le principe du serveur de re-chiffrement combiné à ce cas d'usage, que nous appellerons serveur de re-chiffrement combiné généralisé. Il serait constitué de deux types de re-chiffrement décrits ainsi :

- un unidirectionnel qui ne serait pas unique et qui serait utilisé pour le re-chiffrement entre deux dispositifs appartenant à des utilisateurs différents. Cela pourrait être un re-chiffrement multiple de cardinal n avec $n \geq 2$ ou un re-chiffrement infini.
- un bidirectionnel infini, qui serait utilisé pour le re-chiffrement entre deux dispositifs appartenant à un même utilisateur.

L'utilisation d'un tel schéma est décrite dans la Figure 7.5. Le nombre de re-chiffrements possibles entre des utilisateurs différents dépend du nombre de re-chiffrements offert par le re-chiffrement unidirectionnel considéré.

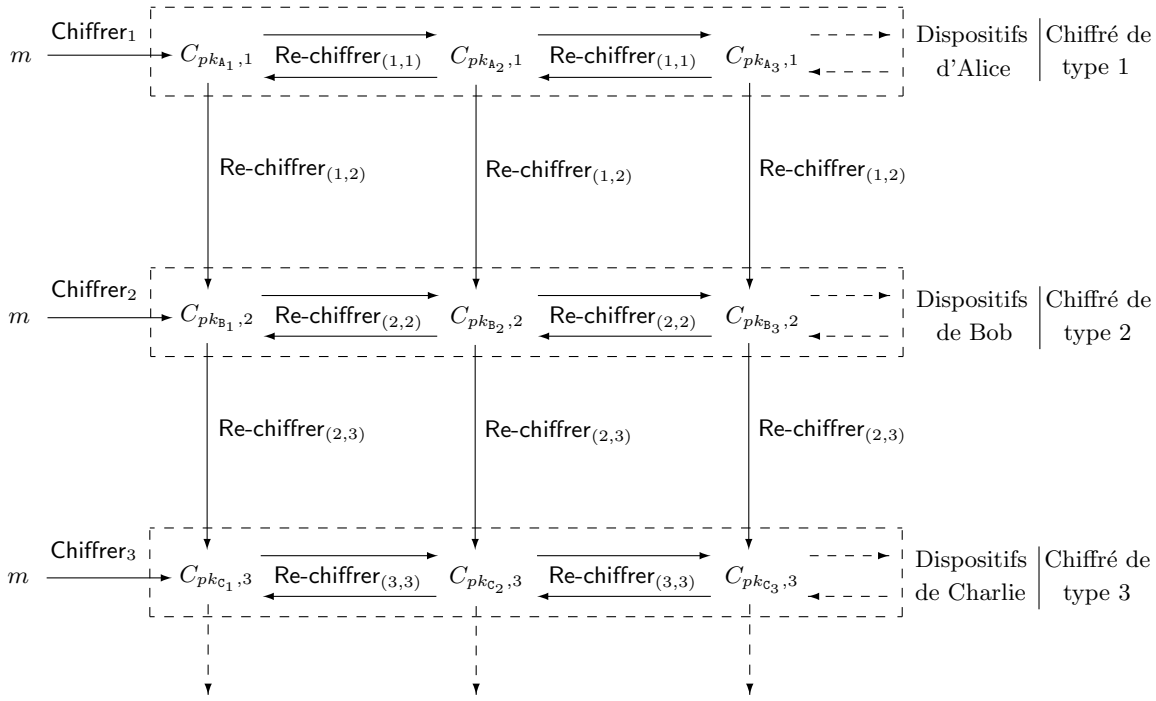

Figure 7.5 – Utilisation du serveur de re-chiffrement combiné généralisé

Remarquons que le serveur de re-chiffrement combiné généralisé - qui utilise un re-chiffrement unidirectionnel et non unique - présente un intérêt indéniable, par rapport à un serveur de re-

chiffrement unidirectionnel et infini qui permettrait de résoudre également cette problématique. En effet, un schéma de re-chiffrement bidirectionnel, non construit à partir de la réunion de deux clés unidirectionnelles, offre en général de meilleures performances qu'un schéma avec re-chiffrement unidirectionnel. Par exemple, un facteur d'efficacité 3 existe entre les schémas présentés dans ce chapitre. De plus, nous avons vu qu'il est possible, grâce à la forme des clés de re-chiffrement bidirectionnelles, de minimiser le nombres de clés de re-chiffrement à stocker. C'est pourquoi l'utilisation d'un re-chiffrement bidirectionnel est préférable entre deux dispositifs d'un même utilisateur au re-chiffrement unidirectionnel et donne donc tout son sens au serveur de re-chiffrement combiné généralisé.

Pour réaliser un tel système, il faut donc disposer de schémas avec re-chiffrement unidirectionnel multiple de cardinal strictement supérieur à 1 ou infini, schémas dont nous ne disposons pas encore de réalisation efficace et sécurisée. Nous pouvons cependant concevoir théoriquement un tel schéma en se basant sur la construction que nous avons évoquée à la Section 4.1.4 et qui utilise notamment des applications multi-linéaires. En effet, le re-chiffrement bidirectionnel que nous avons utilisé sur le schéma [LV08b] est également valable sur tout type de chiffré de ce schéma. Le schéma ainsi obtenu est détaillé dans la Figure 7.6 où nous ne détaillons que l'algorithme Re-chiffrer sans les opérations de vérification d'un chiffré. La sécurité du schéma serait la même que celle du schéma vu à la Section 4.1.4 en considérant en plus les restrictions non naturelles récurrentes liées au re-chiffrement bidirectionnel.

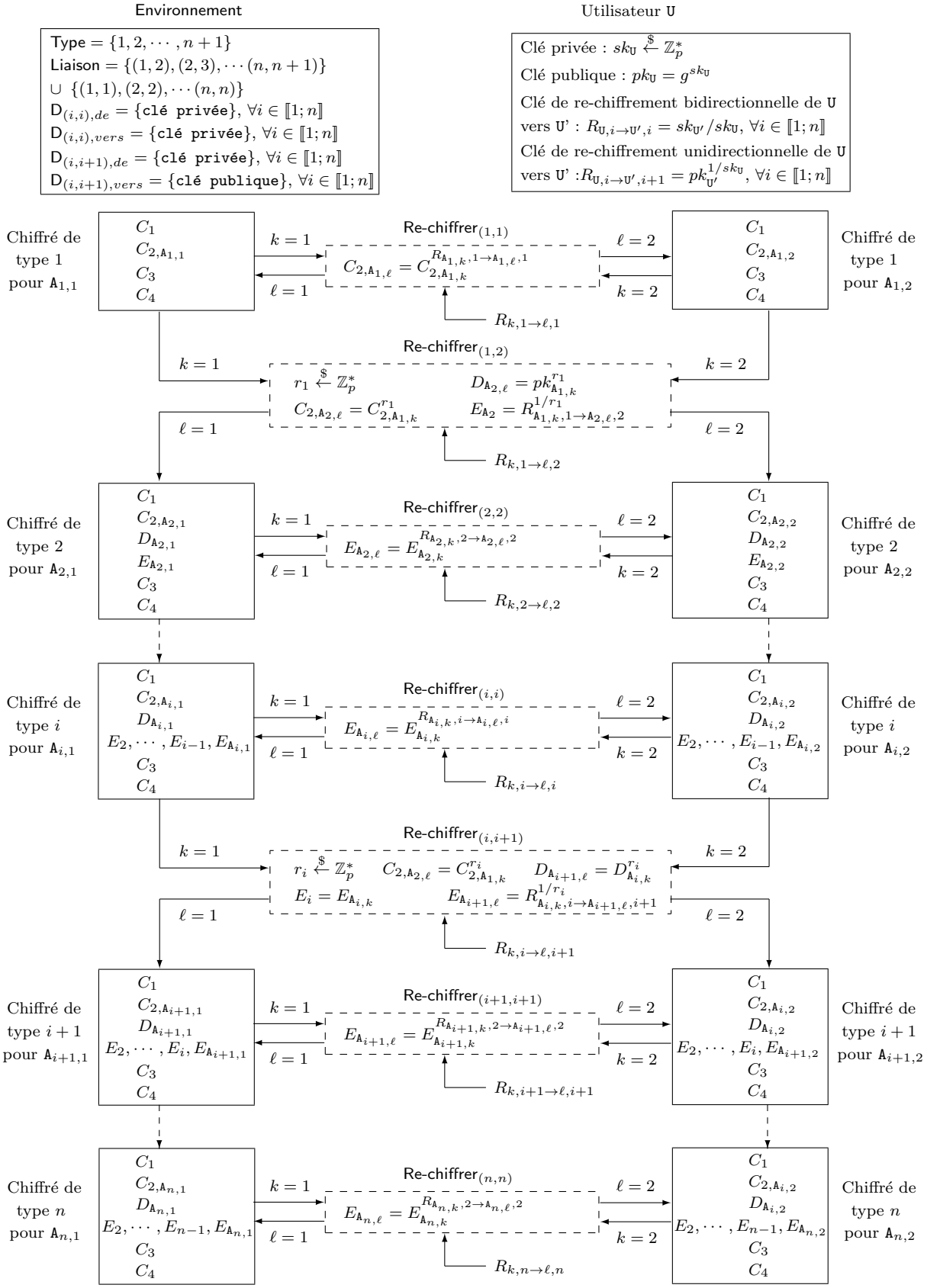

Figure 7.6 – Re-chiffrement combiné généralisé sur le schéma [LV08b] multiple de cardinal n

Chapitre 8

Gestion fine des droits d'accès

Sommaire

8.1 Le serveur de re-chiffrement conditionnel 115
8.1.1 Délégation de type "tout ou rien" 115
8.1.2 Définition 116
8.1.3 État de l'art 118
8.1.4 Compatibilité avec la gestion dynamique des droits d'accès 119
8.2 Une gestion des droits d'accès plus fine 119
8.2.1 Limite de la granularité des conditions 120
8.2.2 Le serveur de re-chiffrement sélectif 121
8.2.3 Notre schéma avec re-chiffrement sélectif et unidirectionnel 124

Nous nous intéressons dans ce chapitre à la gestion fine des droits d'accès dans un système de stockage de données en étudiant le modèle de serveur de re-chiffrement conditionnel. Nous exposons l'idée derrière ce modèle et expliquons ensuite comment ce modèle peut être défini à partir de notre modèle unifié présenté au Chapitre 3. Nous passons ensuite en revue son état de l'art. Nous expliquons ensuite que la granularité des droits d'accès que l'on peut obtenir pour ce modèle peut être améliorée et proposons un nouveau modèle pour lequel nous exhibons un schéma avec re-chiffrement unidirectionnel et unique. Ces résultats sont en cours de soumission.

8.1 Le serveur de re-chiffrement conditionnel

8.1.1 Délégation de type "tout ou rien"

Considérons un système de stockage de données, avec un ensemble d'utilisateurs et de fichiers à partager. Il est évident qu'un utilisateur ne souhaite pas forcément partager l'ensemble de ses données avec tous les utilisateurs en qui il a confiance, ni offrir le même partage à tous. Si nous examinons un tel système fondé sur un serveur de re-chiffrement, nous nous apercevons que la délégation offerte entre un délégant et un délégué est en quelque sorte du "tout ou rien". Examinons par exemple deux cas :

- le re-chiffrement infini sur des ensembles $\mathsf{Type} = \{1\}$ et $\mathsf{Liaison} = \{(1,1)\}$. Ainsi, s'il existe une clé de re-chiffrement $R_{\mathtt{A},1\to\mathtt{B},1}$, alors tous les chiffrés à destination de `A` pourront être accessibles de `B` via un re-chiffrement. Il n'est donc pas possible d'effectuer une sélection des données partagées avec `B`.
- le re-chiffrement unique sur des ensembles $\mathsf{Type} = \{1,2\}$ et $\mathsf{Liaison} = \{(1,2)\}$. Ici, s'il existe une clé de re-chiffrement $R_{\mathtt{A},1\to\mathtt{B},2}$, alors tous les chiffrés de type 1 à destination de `A` seront accessibles à `B` via un re-chiffrement. En revanche, les chiffrés de type 2 ne le seront pas puisqu'ils sont non re-chiffrables. Cela permet donc d'effectuer une certaine sélection

des données que A souhaite partager avec B. Cependant, les données non accessibles à B, puisque chiffrées en type 2, ne pourront pas non plus être partagées avec d'autres délégués et finalement les autres délégués auront accès aux mêmes données que B.

En appliquant le même raisonnement aux autres modèles de schémas étudiés jusqu'à présent, nous en déduisons qu'aucun n'est idéal pour un partage de données sélectif. Pour répondre à ce besoin, un modèle particulier de serveur de re-chiffrement a vu le jour, appelé serveur de re-chiffrement conditionnel - de l'anglais *conditional proxy re-encryption* (CPRE). L'idée est d'inclure une condition - qui peut être considérée comme une clé - à la fois dans le chiffré et dans la clé de re-chiffrement de telle manière que le re-chiffrement ne puisse se faire que si la condition utilisée dans ces deux éléments est la même. Ainsi, une clé de re-chiffrement de A vers B portant sur la condition "urgent" ne pourra traiter que les chiffrés comportant cette même condition "urgent". B n'aura donc accès qu'aux messages conditionnés comme "urgent". Cette condition doit être indépendante du couple (délégant/délégué) afin de pouvoir être utilisée pour tous les couples possibles.

8.1.2 Définition

Nous allons maintenant donner une définition de ces schémas. Ces schémas offrant naturellement les fonctionnalités d'un PRE, elle se rapproche de la Définition 24 à laquelle on ajoute des propriétés. Tout comme pour cette dernière, nous donnons une définition généralisée qui regroupe tous les modèles de PRE conditionnel rencontrés dans la littérature et qui laisse la possibilité d'en envisager d'autres.

Intuition. L'idée de base est d'inclure une condition lorsque l'on chiffre un message. Nous pouvons alors être confronté à deux types de re-chiffrement, qui dépendront du schéma considéré.

1. Le premier tient compte de la condition dans la modification du destinataire du chiffré. Une fois le re-chiffrement effectué, cette condition disparaît du chiffré. Selon les schémas, il sera possible de modifier à nouveau le chiffré ainsi obtenu de la manière habituelle, le re-chiffrement étant alors identique au cas suivant. Il est évident que la clé permettant ce re-chiffrement dépend de la condition incluse dans le chiffré.
2. Le deuxième ne prend pas en compte cette éventuelle condition et modifie seulement le destinataire du chiffré, ce qui correspond à un re-chiffrement tel que nous l'avons considéré jusqu'à présent. Ainsi, une clé de re-chiffrement portant sur un couple (délégant/délégué) (A/B) permettra de modifier un chiffré de A en un chiffré de B, qu'il comporte une condition ou non. Dans le cas où le chiffré de A en considère une, alors le chiffré de B la considérera également. Les clés de re-chiffrement correspondantes ne dépendent donc d'aucune condition puisqu'elles ne les modifient en aucun cas.

La définition que nous donnons ci-dessous prend en compte ces deux types de re-chiffrement.

Définition 40 (Serveur de re-chiffrement conditionnel)**.** *Un* schéma de serveur de re-chiffrement conditionnel CPRE *est un schéma correspondant à la Définition 24 et possédant de plus les propriétés décrites ci-dessous.*

- *L'ensemble* Type *est de la forme* $\mathsf{Type}_1 \times \mathsf{Cond} \cup \mathsf{Type}_2$ *où* $\mathsf{Type}_1 \times \mathsf{Cond}$ *décrit un ensemble de types de chiffrés portant une condition et* Type_2 *un ensemble de types de chiffrés n'en portant pas.* Type_1 *et* Cond *doivent être non vides si l'on veut disposer d'un re-chiffrement conditionnel. Sinon on se retrouve avec un schéma habituel.*
- *L'ensemble* Liaison *est la réunion d'ensembles de la forme suivante :*
  - $\{((\eta, c), (\eta', c))\}_{c \in \mathsf{Cond}}$ *avec* $\eta, \eta' \in \mathsf{Type}_1$*. La liaison* $((\eta, c), (\eta', c))$ *sera notée* $(\eta, \eta')_c$*.*
  - $\{((\eta, c), \eta')\}_{c \in \mathsf{Cond}}$ *avec* $(\eta, \eta') \in \mathsf{Type}_1 \times \mathsf{Type}_2$*.*
  - $\{(\eta, \eta')\}$ *avec* $\eta, \eta' \in \mathsf{Type}_2$*.*
- *Pour toute liaison* $(\eta, \eta')_c \in$ Liaison*, l'algorithme* Délégation *sur cette liaison entre un délégant* A *et un délégué* B *fournit une clé de re-chiffrement indépendante de* c *qui pourra*

être utilisée par l'algorithme de re-chiffrement Re-chiffrer *sur toute liaison* $(\eta, \eta')_{c'}$ *entre* $\mathtt{A}$ *et* $\mathtt{B}$ *pour toute condition* $c' \in \mathsf{Cond}$. *Cette clé de re-chiffrement sera notée* $R_{\mathtt{A},\eta\to\mathtt{B},\eta'}$ *et les algorithmes* Délégation *et* Re-chiffrer *sur cette liaison seront respectivement notés* $\mathsf{Délégation}_{(\eta,\eta')}$ *et* $\mathsf{Re\text{-}chiffrer}_{(\eta,\eta')}$. *De plus, les ensembles de données* $\mathsf{D}_{(\eta,\eta')_c,de}$ *et* $\mathsf{D}_{(\eta,\eta')_c,vers}$ *sont égaux pour tous* $c \in \mathsf{Cond}$ *et seront alors notés respectivement* $\mathsf{D}_{(\eta,\eta'),de}$ *et* $\mathsf{D}_{(\eta,\eta'),vers}$.

Pour résumé, l'algorithme Délégation *peut prendre trois types d'entrées différentes et est décrit comme suit.*

- $\mathsf{Délégation}_{(\eta,\eta')}(\mathcal{P}, \mathsf{D}^{\mathtt{A}}_{(\eta,\eta'),de}, \mathsf{D}^{\mathtt{B}}_{(\eta,\eta'),vers})$ *sur un couple* $(\eta, \eta') \in \mathsf{Type}_1^2$ *: prend en entrée les paramètres publics* $\mathcal{P}$ *et les deux ensembles* $\mathsf{D}^{\mathtt{A}}_{(\eta,\eta'),de}, \mathsf{D}^{\mathtt{B}}_{(\eta,\eta'),vers}$ *et retourne une clé de re-chiffrement* $R_{\mathtt{A},\eta\to\mathtt{B},\eta'}$ *qui permet de transformer , pour tout* $c \in \mathsf{Cond}$*, tout chiffré de type* (η, c) *pour* $\mathtt{A}$ *en un chiffré de* (η', c) *pour* $\mathtt{B}$ *;*
- $\mathsf{Délégation}_{((\eta,c),\eta')}(\mathcal{P}, \mathsf{D}^{\mathtt{A}}_{((\eta,c),\eta'),de}, \mathsf{D}^{\mathtt{B}}_{((\eta,c),\eta'),vers})$ *sur une liaison* $((\eta, c), \eta')$ *avec* $\eta \in \mathsf{Type}_1$, $\eta' \in \mathsf{Type}_2$ *et* $c \in \mathsf{Cond}$ *: prend en entrée les paramètres publics* $\mathcal{P}$ *et les deux ensembles* $\mathsf{D}^{\mathtt{A}}_{((\eta,c),\eta'),de}, \mathsf{D}^{\mathtt{B}}_{((\eta,c),\eta'),vers}$ *et retourne une clé de re-chiffrement* $R_{\mathtt{A},(\eta,c)\to\mathtt{B},\eta'}$ *qui permet de transformer tout chiffré de type* (η, c) *pour* $\mathtt{A}$ *en un chiffré de* η' *pour* $\mathtt{B}$ *;*
- $\mathsf{Délégation}_{(\eta,\eta')}(\mathcal{P}, \mathsf{D}^{\mathtt{A}}_{(\eta,\eta'),de}, \mathsf{D}^{\mathtt{B}}_{(\eta,\eta'),vers})$ *sur une liaison* $(\eta, \eta') \in \mathsf{Type}_2^2$ *: prend en entrée les paramètres publics* $\mathcal{P}$ *et les deux ensembles* $\mathsf{D}^{\mathtt{A}}_{(\eta,\eta'),de}, \mathsf{D}^{\mathtt{B}}_{(\eta,\eta'),vers}$ *et retourne une clé de re-chiffrement* $R_{\mathtt{A},\eta\to\mathtt{B},\eta'}$ *qui permet de transformer tout chiffré de type* η *pour* $\mathtt{A}$ *en un chiffré de* η' *pour* $\mathtt{B}$*.*

En conséquence, l'algorithme Re-chiffrer *peut prendre également trois types d'entrées différentes et est décrit comme suit.*

- $\mathsf{Re\text{-}chiffrer}_{(\eta,\eta')}(\mathcal{P}, R_{\mathtt{A},\eta\to\mathtt{B},\eta'}, C_{pk_{\mathtt{A}},(\eta,c)})$ *sur un couple* $(\eta, \eta') \in \mathsf{Type}_1^2$ *: prend en entrée les paramètres publics* $\mathcal{P}$*, une clé de re-chiffrement* $R_{\mathtt{A},\eta\to\mathtt{B},\eta'}$ *de* $\mathtt{A}$ *vers* $\mathtt{B}$ *pour toutes les liaisons* $(\eta, \eta')_{c'}$ *avec* $c' \in \mathsf{Cond}$ *et un chiffré* $C_{pk_A,(\eta,c)}$ *de type* (η, c) *pour* $\mathtt{A}$ *et retourne un chiffré* $C_{pk_B,(\eta',c)}$ *du même message en clair de type* (η', c) *pour* $\mathtt{B}$ *ou un symbole* $\perp$ *signifiant que le chiffré n'est pas valide ;*
- $\mathsf{Re\text{-}chiffrer}_{((\eta,c),\eta')}(\mathcal{P}, R_{\mathtt{A},(\eta,c)\to\mathtt{B},\eta'}, C_{pk_{\mathtt{A}},(\eta,c)})$ *sur une liaison* $((\eta, c), \eta')$ *avec* $\eta \in \mathsf{Type}_1$, $\eta' \in \mathsf{Type}_2$ *et* $c \in \mathsf{Cond}$ *: prend en entrée les paramètres publics* $\mathcal{P}$*, une clé de re-chiffrement* $R_{\mathtt{A},(\eta,c)\to\mathtt{B},\eta'}$ *de* (η, c) *pour* $\mathtt{A}$ *vers* η' *pour* $\mathtt{B}$ *et un chiffré* $C_{pk_A,(\eta,c)}$ *de type* (η, c) *pour* $\mathtt{A}$ *et retourne un chiffré* $C_{pk_B,\eta'}$ *du même message en clair de type* η' *pour* $\mathtt{B}$ *ou un symbole* $\perp$ *signifiant que le chiffré n'est pas valide ;*
- $\mathsf{Re\text{-}chiffrer}_{(\eta,\eta')}(\mathcal{P}, R_{\mathtt{A},\eta\to\mathtt{B},\eta'}, C_{pk_{\mathtt{A}},\eta})$ *sur une liaison* $(\eta, \eta') \in \mathsf{Type}_2^2$ *: prend en entrée les paramètres publics* $\mathcal{P}$*, une clé de re-chiffrement* $R_{\mathtt{A},\eta\to\mathtt{B},\eta'}$ *de* η *pour* $\mathtt{A}$ *vers* η' *pour* $\mathtt{B}$ *et un chiffré* $C_{pk_A,\eta}$ *de type* η *pour* $\mathtt{A}$ *et retourne un chiffré* $C_{pk_B,\eta'}$ *du même message en clair de type* η' *pour* $\mathtt{B}$ *ou un symbole* $\perp$ *signifiant que le chiffré n'est pas valide.*

Cette définition permet d'envisager différentes situations, comme les deux données par exemple dans la Figure 8.1.

Utilisation. Les conditions de l'ensemble Cond sont indépendantes des utilisateurs du système. En plus d'être utilisables pour chaque couple (délégant/délégué), elles pourront avoir des significations différentes selon les délégants considérés. Par exemple, considérons $\mathsf{Type}_1 = \{1\}$, $\mathsf{Type}_2 = \{2\}$, $\mathsf{Cond} = \{c_1, c_2\}$ et $\mathsf{Liaison} = \{((1, c_1), 2), ((1, c_2), 2)\}$. Ces conditions étant indépendantes des utilisateurs, il est alors possible, pour un délégant A, qu'elles correspondent respectivement à "amis" et "famille" alors que pour un utilisateur B, elles correspondent respectivement à "collègues" et "amis". Ainsi, si A possède un ami C à qui il souhaite offrir ses droits de délégation, il calculera une clé de re-chiffrement de A vers C sur la liaison $((1, c_1), 2)$, pour B, une telle clé vers un délégué D signifiera que D est un de ses collègues. Pour des raisons d'indépendance, un utilisateur E pourrait être un membre de la famille de A tout en étant un collègue de B.

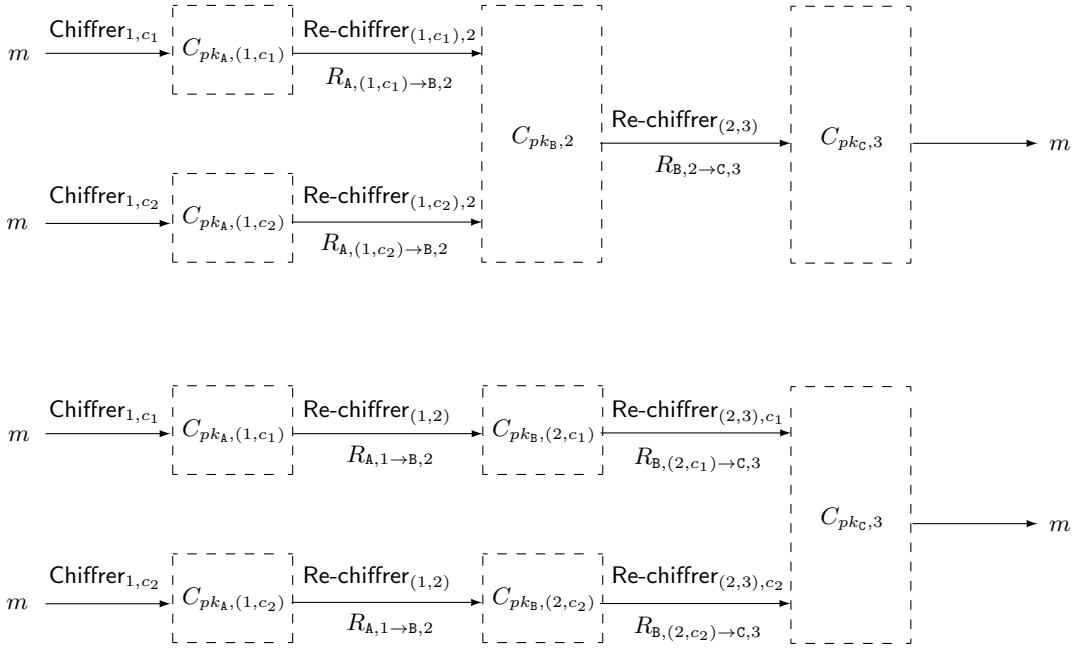

Figure 8.1 – Exemples de re-chiffrement conditionnel

Sécurité. Par définition, un schéma CPRE est un PRE. La sécurité que nous avons donc vu au Chapitre 3 reste valable. Cela donne cependant lieu à des spécificités intéressantes dont voici quelques exemples.

- Étant donné qu'un type (η, c) est différent d'un type (η, c'), l'adversaire, recevant un challenge de type (η^*, c^*) pour pk^*, peut alors obtenir une clé de re-chiffrement de pk^* vers une entité corrompue sur une liaison $((\eta^*, c), \eta')$ à condition que c soit différent de c^*. Pour les mêmes raisons, il peut également obtenir pour le même challenge, une clé de re-chiffrement de pk^* vers une entité corrompue sur une liaison $((\eta, c^*), (\eta'))$ tant que η est différent de η^*.
- À partir d'une clé de re-chiffrement portant sur une liaison $((\eta, c), \eta')$ pour un couple (délégant/délégué) (A/B), l'adversaire ne peut calculer une clé de re-chiffrement portant sur une liaison $((\eta, c'), \eta')$ pour le même couple (A/B) avec $c' \neq c$.

8.1.3 État de l'art

Cette notion de re-chiffrement conditionnel a été introduite par Tang dans l'article [Tan08] qui a proposé un schéma unidirectionnel atteignant la sécurité CCA sous l'hypothèse BDH. Libert et Vergnaud ont proposé une version de leur schéma [LV08b] dans le même article avec re-chiffrement conditionnel possédant les mêmes propriétés et la même sécurité. Weng *et al.* dans [WYT+09] et Jia *et al.* dans [JSJL10] ont également conçu un schéma avec les mêmes caractéristiques et atteignant la sécurité IND-CCA sous l'hypothèse DBDH. Il faudra cependant être prudent dans l'utilisation du schéma [JSJL10], car il présente le même genre de failles potentielles que celles que nous avons étudiées au Chapitre 6.

Fan *et al.* ont élaboré dans [FSW09] un schéma avec re-chiffrement unique, unidirectionnel et conditionnel utilisant notamment la transformation [CHK04] de Canetti *et al.* pour obtenir une sécurité CCA sous l'hypothèse 3-QDBDH. Weng *et al.* ont également construit dans [WDD+09] un schéma du même type en utilisant la construction de Fujisaki-Okamoto basée sur le chiffrement ElGamal. Par la suite, Vivek *et al.* ont amélioré l'efficacité de ce dernier schéma dans l'article [VSRR11] en utilisant des techniques présentées dans [CWYD10]. Ces trois schémas sont annoncés comme atteignant la sécurité CCA. Pour autant, ils ne sont prouvés en réalité sûrs que dans des modèles très restrictifs. En effet, ils ne considèrent en aucun cas les fonc-

tionnalités de partage offertes par les conditions du CPRE. Dans ces schémas, la clé de re-chiffrement d'un délégant vers un délégué est de la forme : $R_{\mathtt{A},(1,c)\to\mathtt{B},(2,c)} = (\tilde{R}_{\mathtt{A},1\to\mathtt{B},2}, R_c)$ où $\tilde{R}_{\mathtt{A},1\to\mathtt{B},2}$ correspond à une clé de re-chiffrement unidirectionnelle de A vers B sans condition et R_C une composante liée à la condition c indépendante de A et de B. Ainsi, à partir d'une clé de re-chiffrement $R_{\mathtt{A},(1,c')\to\mathtt{C},(2,c')} = (\tilde{R}_{\mathtt{A},1\to\mathtt{C},2}, R_{c'})$, on peut produire la clé de re-chiffrement $R_{\mathtt{A},(1,c')\to\mathtt{B},(2,c')} = (\tilde{R}_{\mathtt{A},1\to\mathtt{B},2}, R_{c'})$. A pense n'offrir à B que les droits d'accès pour les chiffrés portant sur la condition c, mais le serveur dispose des informations permettant à B d'obtenir ceux portant sur la condition c'. Ces trois schémas n'offrent donc pas un re-chiffrement conditionnel fonctionnel et suffisamment sécurisé.

8.1.4 Compatibilité avec la gestion dynamique des droits d'accès

Afin d'obtenir le système le plus fonctionnel possible, il serait intéressant de pouvoir cumuler la gestion fine des droits d'accès à leur gestion dynamique que nous avons abordée au chapitre précédent. D'un point de vue théorique, rien n'interdit de mélanger le re-chiffrement combiné avec le re-chiffrement conditionnel. Ces deux types de schémas s'intègrent même plutôt bien entre eux. En effet, le re-chiffrement conditionnel prenant en compte la condition lors du re-chiffrement pourrait s'effectuer sur le re-chiffrement unidirectionnel du combiné sans en modifier le re-chiffrement bidirectionnel. Ceci permettrait de mettre un droit d'accès soumis à condition entre les appareils d'utilisateurs différents, alors qu'entre les appareils d'un même utilisateur, aucune condition ne serait requise. L'utilisation d'un tel type de schéma est décrite dans la Figure 8.2.

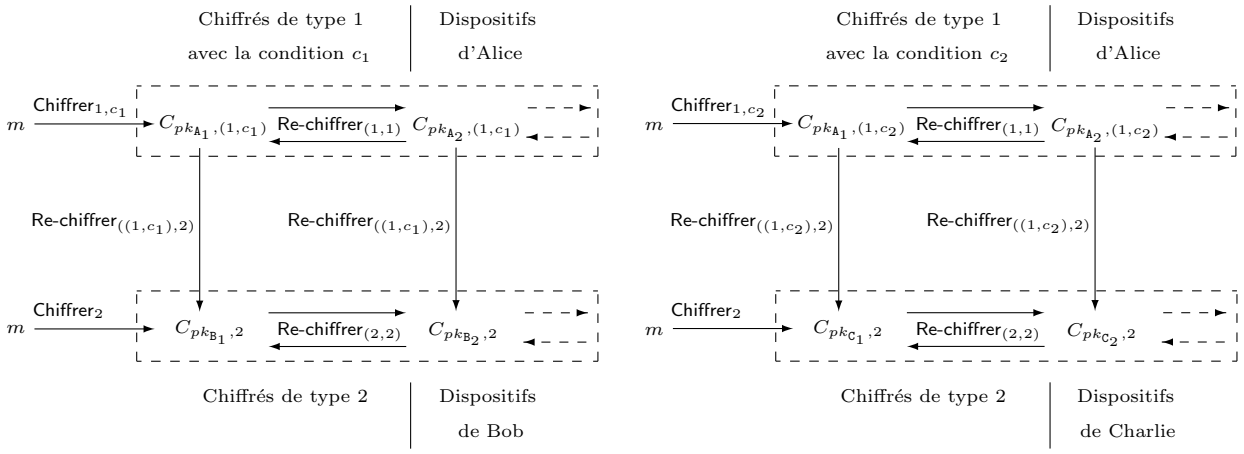

Figure 8.2 – Le re-chiffrement combiné et conditionnel

Il est possible de concevoir un schéma de ce type en utilisant :
- le schéma [LV08b] combiné présenté au chapitre précédent.
- le schéma [LV08b] conditionnel proposé par Libert-Vergnaud.

Il suffit de remplacer le re-chiffrement unidirectionnel du premier par le conditionnel du second pour obtenir un schéma avec re-chiffrement combiné et conditionnel. Etant donné que ces deux re-chiffrement sont très proches, cette modification ne remet pas en cause la sécurité du schéma et on obtient un schéma IND-RCCA sûr.

8.2 Une gestion des droits d'accès plus fine

Comme nous venons de le voir, les schémas avec re-chiffrement conditionnel permettent d'obtenir une gestion fine des droits d'accès afin d'offrir un stockage de données le plus fonctionnel possible. En effet, les droits d'accès, et donc les délégués, différent selon la donnée considérée. Étudions en détail ces différents droits d'accès.

8.2.1 Limite de la granularité des conditions

Nous avons déjà mentionné que chaque délégant fera une interprétation personnelle des conditions qu'il utilisera. Par exemple, pour un délégant `A`, la condition c_1 correspondra à ses amis tandis que la condition c_2 correspondra à sa famille. De cette manière, chaque condition représente, en quelque sorte, un cercle de délégués choisi par `A`. Si l'on souhaite utiliser une gestion très fine des droits d'accès, il est alors très probable que ces différents cercles s'intersectent. Ainsi, si on ajoute aux cercles précédents des cercles attrayant à une notion géographique comme "habitant la ville de Caen" en condition c_3, un délégué peut appartenir à ce dernier cercle tout en appartenant à celui des "amis". `A` aura alors pris soin de créer deux clés de re-chiffrement pour ce délégué en question : une sur c_1 et une sur c_3. Cela revient un peu à donner au délégué deux attributs vis-à-vis de `A`, à savoir "amis" et "habitant la ville de Caen".

Chaque délégué se retrouve donc doté d'un ensemble d'attributs choisis par le délégant, qui lui donneront différents accès aux données de ce dernier. En ne considérant par exemple que deux des attributs précédents, à savoir "amis" et "habitant la ville de Caen", il peut y avoir alors plusieurs cercles possibles que souhaite utiliser le délégant pour gérer ses droits d'accès et découlant de celles-ci. Ces différents cercles sont :

- "amis".
- "habitant la ville de Caen".
- "amis" et "habitants la ville de Caen" qui serait l'intersection des deux précédents
- le cercle ne portant sur aucun attribut et auquel tout le monde aurait accès.

Afin de disposer de la granularité la plus fine possible, il serait alors intéressant de pouvoir disposer au sein d'une clé de re-chiffrement d'une telle gestion des droits d'accès. Une clé de re-chiffrement, incluant les deux attributs "amis" et "habitant à Caen", permettrait au délégué concerné de bénéficier des droits d'accès des 4 cercles mentionnés ci-dessus. Quant au chiffrement d'un message, il consisterait alors à choisir un ensemble d'attributs et à produire un chiffré dont la condition serait cet ensemble d'attributs. Tous les délégués possédant au moins les attributs de cet ensemble pourraient alors accéder à la donnée chiffrée. Cela rejoint l'idée du chiffrement basé sur l'identité avec joker, de l'anglais *wildcard identity-based encryption* noté (WIBE), présentée indépendamment par Abdalla *et al.* dans [ACD+06] et Birkett *et al.* dans [BDNS07] et dont les travaux ont été fusionnés dans [ABC+11].

Ce chiffrement est une généralisation du chiffrement hiérarchique basé sur l'identité, de l'anglais *hierarchical identity-based encryption* (HIBE), où contrairement à ce dernier, un chiffré n'est pas destiné à un ensemble d'utilisateurs ayant tous le même ensemble d'attributs mais à un ensemble d'utilisateurs ayant un certain nombre d'attributs en commun. Considérons par exemple que chaque utilisateur possède deux types d'attributs : un attribut portant sur la localisation et un attribut portant sur le type de personne. Dans le cas d'un HIBE, un chiffré ne peut pas porter sur un seul attribut comme "habitants à Caen" signifiant que l'attribut sur le type de personne n'est pas pris en compte. Le WIBE offre cette possibilité en laissant en quelque sorte certains attributs ouverts à tout le monde. Dans ce cas, cette disponibilité universelle serait représentée par un joker, symbolisé communément par une chaîne de caractère du type "$\star\star\star\star\star$".

Objectif. Le but est d'offrir une sélection très fine des droits d'accès en utilisant un système d'attributs non pas sur les identités des utilisateurs comme dans le cas d'un HIBE et d'un WIBE mais sur les droits d'accès des délégués et donc sur l'ensemble Cond. En effet, il ne faut pas oublier que les conditions sont indépendantes des différents délégués et que chaque délégant peut

décider d'attribuer certains attributs à un délégué indépendamment des autres délégants. Nous présentons maintenant ce type de schéma que nous appelons, pour plus de simplicité, *serveur de re-chiffrement sélectif*, noté SPRE. De plus et afin d'offrir un large choix sur le nombre d'attributs sélectionnés lors du chiffrement, qui déterminera l'ensemble des délégués qui pourront accéder aux données chiffrées, nous y incluons également un système de joker "$\star\star\star\star\star\star$" comme dans un WIBE.

8.2.2 Le serveur de re-chiffrement sélectif

Un SPRE reprend les principes d'un CPRE, à la différence près qu'un chiffré incluera un ensemble de conditions et qu'une clé de re-chiffrement ne portera que sur une seule condition. Plus précisément un chiffré comportera une condition définie par un attribut possédant d'éventuels jokers, alors que la clé de re-chiffrement portera sur un attribut sans jokers. Ainsi, la définition d'un SPRE ne peut reprendre la définition d'un CPRE à laquelle on ajoute de nouvelles propriétés, étant donné que la condition incluse dans un chiffré ne sera plus la même que la condition incluse au sein d'une clé de re-chiffrement. Pour cela, nous commençons par décrire la façon dont nous représenterons les différents attributs d'un délégant.

Choix des attributs et interprétation selon le délégant. L'ensemble des attributs, que nous notons Att, est égal à $\{0,1\}^*$ et représente l'ensemble potentiel des attributs qu'un délégant peut donner à ses délégués. Chaque délégant, Alice dans l'exemple ci-dessous, peut représenter les attributs qu'il prend en compte dans un arbre labellisé dont un exemple est donné dans la Figure 8.3.

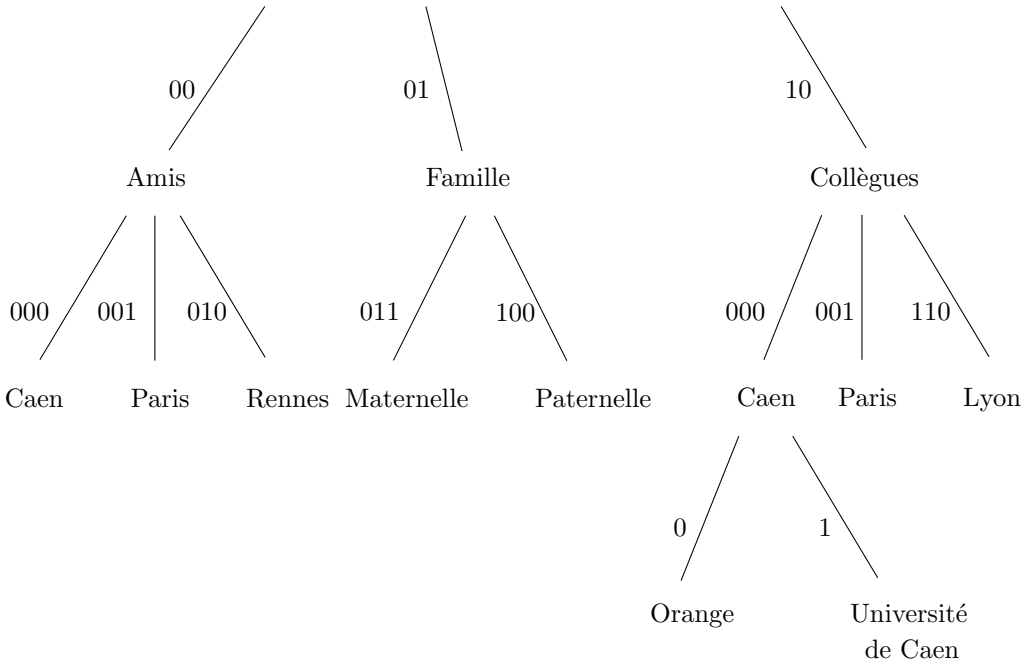

Figure 8.3 – Exemple d'arbre labellisé pour Alice

L'ensemble Cond est, quant à lui, composé des attributs de l'ensemble Att mais également des attributs avec joker, il est donc de la forme $\{0,1,\star\}^*$. Le symbôle $\star$ représente un bit joker correspondant à un bit d'attribut non précisé. Par exemple, la condition $0\star$ réunit les attributs 00 et 01. Nous utiliserons att pour représenter un élément de l'ensemble Att et **c** pour représenter un élément de l'ensemble Cond. Sur l'arbre d'Alice que nous venons de considérer, les conditions suivantes auront alors le sens suivant :

- 00000 pour "Amis,Caen".
- 00001 pour "Amis,Paris".
- $00\star\star\star$ pour "Amis,$\star$", qui regroupe "Amis,Caen", "Amis,Paris" et "Amis,Rennes".
- $10000\star$ pour "Collègues,Caen,$\star$", qui regroupe les collègues de Caen qu'ils soient d'Orange ou de l'Université de Caen.
- $\star\star001$ pour "$\star$,Paris", qui regroupe les amis et collègues de Paris.
- $\star\star000\star$ pour "$\star$,Caen,$\star$", qui regroupe les collègues de Caen d'Orange et de l'Université de Caen. Les amis de Caen ne rentrent pas dans cette catégorie étant donné qu'ils ne possèdent que deux attributs alors que $\star\star000\star$ porte sur trois attributs.

Dans notre exemple, "Caen", "Paris" et "Maternelle" sont représentés avec 3 bits, bien qu'ils sont attribués à des branches différentes. Si ce n'était pas le cas, il serait difficile de savoir exactement dans l'arbre quel ensemble de délégués serait concerné par une condition du type $\star\star1\star\star$. Pour cette raison, les attributs d'un niveau donné doivent avoir le même nombre de bits.

Nous noterons par $\mathcal{T}_{\mathtt{A}}$ l'arbre de $\mathtt{A}$. L'entier $l_{\mathtt{A}}$ représentera le nombre maximal de niveaux de l'arbre $\mathcal{T}_{\mathtt{A}}$ et $n_{\mathtt{A},i}$, pour $i \in [\![1; l_{\mathtt{A}}]\!]$, désignera le nombre de bits composant le niveau i. Dans cet exemple, $l_{\mathtt{A}}$ vaut 3 et $n_{\mathtt{A},1} = 2$. Les attributs considérés par $\mathtt{A}$, noté $\mathsf{Att}_{\mathtt{A}}$, seront donc ceux représentés dans l'arbre. L'ensemble de ces attributs auquel on ajoute ces mêmes attributs avec d'éventuels jokers sera noté $\mathsf{Cond}_{\mathtt{A}}$.

Nous précisons désormais l'environnement du modèle de serveur de re-chiffrement sélectif.

Environnement du modèle. L'environnement est le même que celui d'un CPRE, à la différence près que l'ensemble Cond est composé d'attributs avec éventuellement des jokers.

Nous donnons maintenant la définition d'un SPRE, qui reprend l'idée de la définition d'un CPRE en tenant compte des différences que nous avons évoquées.

Définition 41 (Serveur de re-chiffrement sélectif)**.** *Un* schéma de serveur de re-chiffrement sélectif SPRE *est la donnée d'un espace des messages $\mathcal{M}$, d'un environnement* Environnement *tel que décrit précédemment, ainsi que d'un ensemble de six algorithmes*

$$\{\mathsf{Initialiser}, \mathsf{Clé}, \mathsf{Délégation}, \mathsf{Chiffrer}, \mathsf{Re\text{-}chiffrer}, \mathsf{Déchiffrer}\}$$

décrits comme suit.

$\mathsf{Initialiser}(\lambda) \to \mathcal{P}$ *: prend en entrée un paramètre de sécurité $\lambda \in \mathbb{N}$ et retourne les paramètres publics $\mathcal{P}$ du schéma.*

$\mathsf{Clé}(\mathcal{P})$ *: prend en entrée les paramètres publics $\mathcal{P}$ et retourne une paire de clés (privée/publique) (sk, pk), telle que pk permette de chiffrer et sk de déchiffrer.*

L'algorithme Délégation *peut prendre trois types d'entrées différentes dépendant des liaisons considérées et est décrit ainsi :*

$\mathsf{Délégation}(\mathcal{P}, (\eta, \eta'), \mathsf{D}^{\mathtt{A}}_{(\eta,\eta'),de}, \mathsf{D}^{\mathtt{B}}_{(\eta,\eta'),vers})$ *: prend en entrée les paramètres publics $\mathcal{P}$, un couple (η, η') avec $\eta, \eta' \in \mathsf{Type}_1$ et les deux ensembles $\mathsf{D}^{\mathtt{A}}_{(\eta,\eta'),de}$ et $\mathsf{D}^{\mathtt{B}}_{(\eta,\eta'),vers}$ et retourne une clé de re-chiffrement $R_{\mathtt{A},\eta\to\mathtt{B},\eta'}$ de η pour $\mathtt{A}$ vers η' pour $\mathtt{B}$ qui permet de transformer, pour tout $\mathbf{c} \in \mathsf{Cond}$, tout chiffré de type $(\eta, \mathbf{c})$ pour $\mathtt{A}$ en un chiffré de type $(\eta', \mathbf{c})$ pour $\mathtt{B}$.*

$\mathsf{Délégation}(\mathcal{P}, ((\eta, \mathsf{att}), \eta'), \mathsf{D}^{\mathtt{A}}_{((\eta,\mathsf{att}),\eta'),de}, \mathsf{D}^{\mathtt{B}}_{((\eta,\mathsf{att}),\eta'),vers})$ *: prend en entrée les paramètres publics $\mathcal{P}$, un couple $((\eta, \mathsf{att}), \eta') \in (\mathsf{Type}_1 \times \mathsf{Att}) \times \mathsf{Type}_2$ et les deux ensembles $\mathsf{D}^{\mathtt{A}}_{((\eta,\mathsf{att}),\eta'),de}$ et $\mathsf{D}^{\mathtt{B}}_{((\eta,\mathsf{att}),\eta'),vers}$ et retourne une clé de re-chiffrement $R_{\mathtt{A},(\eta,\mathsf{att})\to\mathtt{B},\eta'}$ de η pour $\mathtt{A}$ avec l'attribut* att *vers η' pour $\mathtt{B}$ qui permet de transformer, pour tout $\mathbf{c} \in \mathsf{Cond}$ tel que $\mathbf{c}$ contienne l'attribut* att*, tout chiffré de type $(\eta, \mathbf{c})$ pour $\mathtt{A}$ en un chiffré de type η' pour $\mathtt{B}$.*

$\mathsf{Délégation}(\mathcal{P}, (\eta, \eta'), \mathsf{D}^{\mathtt{A}}_{(\eta,\eta'),de}, \mathsf{D}^{\mathtt{B}}_{(\eta,\eta'),vers})$ *: prend en entrée les paramètres publics* $\mathcal{P}$*, un couple* (η, η') *avec* $\eta, \eta' \in \mathsf{Type}_2$ *et les deux ensembles* $\mathsf{D}^{\mathtt{A}}_{(\eta,\eta'),de}$ *et* $\mathsf{D}^{\mathtt{B}}_{(\eta,\eta'),vers}$ *et retourne une clé de re-chiffrement* $R_{\mathtt{A},\eta\to\mathtt{B},\eta'}$ *de* η *pour* $\mathtt{A}$ *vers* η *pour* $\mathtt{B}$ *qui permet de transformer tout chiffré de type* η *pour* $\mathtt{A}$ *en un chiffré de type* η' *pour* $\mathtt{B}$*.*

L'algorithme Chiffrer *peut prendre deux types d'entrées différentes dépendant des liaisons considérées et est décrit ainsi :*

$\mathsf{Chiffrer}(\mathcal{P}, (\eta, \mathbf{c}), pk, m)$ *: prend en entrée les paramètres publics* $\mathcal{P}$*, un type* $(\eta, \mathbf{c})$ *appartenant à* $\mathsf{Type}_1 \times \mathsf{Cond}$*, une clé publique* pk *et un message* m *et retourne un chiffré* $C_{pk,(\eta,\mathbf{c})}$ *de type* $(\eta, \mathbf{c})$ *pour la clé publique* pk*.*

$\mathsf{Chiffrer}(\mathcal{P}, (\eta, \mathbf{c}), pk, m)$ *: prend en entrée les paramètres publics* $\mathcal{P}$*, un type* η *appartenant à* Type_2*, une clé publique* pk *et un message* m *et retourne un chiffré* $C_{pk,\eta}$ *de type* η *pour la clé publique* pk*.*

L'algorithme Re-chiffrer *peut prendre trois types d'entrées différentes dépendant des liaisons considérées et est décrit ainsi :*

$\mathsf{Re\text{-}chiffrer}(\mathcal{P}, (\eta, \eta'), R_{\mathtt{A},\eta\to\mathtt{B},\eta'}, C_{pk_{\mathtt{A}},(\eta,\mathbf{c})})$ *: prend en entrée les paramètres publics* $\mathcal{P}$*, un couple* (η, η') *avec* $\eta, \eta' \in \mathsf{Type}_1$*, une clé de re-chiffrement* $R_{\mathtt{A},\eta\to\mathtt{B},\eta'}$ *de* η *pour* $\mathtt{A}$ *vers* η' *pour* $\mathtt{B}$ *et un chiffré* $C_{pk_{\mathtt{A}},(\eta,\mathbf{c})}$ *de type* $(\eta, \mathbf{c})$ *pour* $\mathtt{A}$ *et retourne un chiffré* $C_{pk_{\mathtt{B}},(\eta',\mathbf{c})}$ *du même message en clair de type* $(\eta', \mathbf{c})$ *pour* $\mathtt{B}$ *ou un symbole* $\perp$ *signifiant que le chiffré n'est pas valide.*

$\mathsf{Re\text{-}chiffrer}(\mathcal{P}, ((\eta, \mathbf{c}), \eta'), R_{\mathtt{A},(\eta,\mathsf{att})\to\mathtt{B},\eta'}, C_{pk_{\mathtt{A}},(\eta,\mathbf{c})})$ *: prend en entrée les paramètres publics* $\mathcal{P}$*, une liaison* $((\eta, \mathbf{c}), \eta') \in (\mathsf{Type}_1 \times \mathsf{Cond}) \times \mathsf{Type}_2$*, une clé de re-chiffrement de* η *pour* $\mathtt{A}$ *avec l'attribut* att *vers* η' *pour* $\mathtt{B}$ *et un chiffré* $C_{pk_{\mathtt{A}},(\eta,\mathbf{c})}$ *de type* $(\eta, \mathbf{c})$ *pour* $\mathtt{A}$ *et retourne, à condition que* $\mathbf{c}$ *contienne l'attribut* att*, un chiffré* $C_{pk_{\mathtt{B}},\eta'}$ *du même message en clair de type* η' *pour l'entité* $\mathtt{B}$ *ou un symbole* $\perp$ *signifiant que le chiffré n'est pas valide ou que* $\mathbf{c}$ *ne contient pas* att*.*

$\mathsf{Re\text{-}chiffrer}(\mathcal{P}, (\eta, \eta'), R_{\mathtt{A},\eta\to\mathtt{B},\eta'}, C_{pk_{\mathtt{A}},\eta})$ *: prend en entrée les paramètres publics* $\mathcal{P}$*, un couple* (η, η') *avec* $\eta, \eta' \in \mathsf{Type}_2$*, une clé de re-chiffrement* $R_{\mathtt{A},\eta\to\mathtt{B},\eta'}$ *de* η *pour* $\mathtt{A}$ *vers* η' *pour* $\mathtt{B}$ *et un chiffré* $C_{pk_{\mathtt{A}},\eta}$ *de type* η *pour* $\mathtt{A}$ *et retourne un chiffré* $C_{pk_{\mathtt{B}},\eta'}$ *du même message en clair de type* η' *pour* $\mathtt{B}$ *ou un symbole* $\perp$ *signifiant que le chiffré n'est pas valide.*

L'algorithme Déchiffrer *peut prendre deux types d'entrées différentes dépendant des liaisons considérées et est décrit ainsi :*

$\mathsf{Déchiffrer}(\mathcal{P}, (\eta, \mathbf{c}), sk, C_{pk,(\eta,\mathbf{c})})$ *: prend en entrée les paramètres publics* $\mathcal{P}$*, un type* $(\eta, \mathbf{c})$ *appartenant à* $\mathsf{Type}_1 \times \mathsf{Cond}$*, une clé privée* sk *et un chiffré* $C_{pk,(\eta,\mathbf{c})}$ *de type* $(\eta, \mathbf{c})$ *pour la clé publique* pk *associée à la clé privée* sk *et retourne un message* m *ou un symbole* $\perp$ *signifiant que le chiffré n'est pas valide.*

$\mathsf{Déchiffrer}(\mathcal{P}, \eta, sk, C_{pk,\eta})$ *: prend en entrée les paramètres publics* $\mathcal{P}$*, un type* η *appartenant à* Type_2*, une clé privée* sk *et un chiffré* $C_{pk,\eta}$ *de type* η *pour la clé publique* pk *associée à la clé privée* sk *et retourne un message* m *ou un symbole* $\perp$ *signifiant que le chiffré n'est pas valide.*

Sécurité. La sécurité de ce type de schéma ne peut être exactement la même que celle d'un PRE ou d'un CPRE puisqu'une clé de re-chiffrement sur un attribut att permet de modifier un chiffré portant sur une condition $\mathbf{c}$ qui contient l'attribut att. Cependant, les idées restent exactement les mêmes, il faut juste les adapter en remplaçant le test d'égalité sur la condition d'un chiffré et d'une clé de re-chiffrement par l'appartenance d'un attribut d'une clé de re-chiffrement à une condition d'un chiffré. Ainsi, l'adversaire pourra obtenir la clé de re-chiffrement de l'entité cible vers une entité corrompue dès que l'attribut de la clé ne sera pas contenu dans la condition $\mathbf{c}^*$ portée par le challenge.

Utilisation. Afin d'utiliser correctement ce type de schéma, il est recommandé d'ajouter l'ensemble $\mathsf{Cond}_{\mathtt{A}}$ à la clé publique $pk_{\mathtt{A}}$ d'un utilisateur $\mathtt{A}$, ce qui revient à y décrire l'arbre $\mathcal{T}_{\mathtt{A}}$. Ceci permet de sélectionner, lors du chiffrement, l'ensemble des attributs qui seront nécessaires pour accéder aux données. Cette particularité n'est cependant pas nécessaire puisqu'il est possible de chiffrer un message pour n'importe quelle condition $\mathbf{c} \in \mathsf{Cond}$ même si celle-ci ne fait pas partie des conditions considérées par $\mathtt{A}$. Cela signifierait seulement qu'aucun délégué ne pourrait accéder à la donnée en l'état [1].

Etat de l'art. Fang *et al.* ont présenté dans [FSGW11] un schéma avec re-chiffrement sélectif, unidirectionnel et unique annoncé comme étant IND-CCA sûr sous l'hypothèse DBDH. Contrairement à ce qui est annoncé, ce schéma présente des restrictions non naturelles contraignantes qui impliquent que sa sécurité ne prend pas totalement en compte les fonctionnalités du re-chiffrement sélectif. En utilisant leurs notations, la paire de clé (privée/publique) d'un utilisateur est de la forme (sk, pk) avec $pk = g^{sk}$ et la clé de re-chiffrement de $\mathtt{A}$ avec attribut ω vers $\mathtt{B}$ est composée d'éléments $a_x = Z^{q_x(0)}\mathcal{H}_4(pk_i, \omega)^{r_x}$ et $b_x = g^{r_x}$, avec $q_x(0) = sk_{\mathtt{B}} - sk_{\mathtt{A}}$, Z un paramètre public et $r_x \xleftarrow{\$} \mathbb{Z}_p^*$. Le chiffré d'un message m pour $\mathtt{A}$ avec la condition ω est notamment composé des éléments $B = g^s$, $C_\omega = \mathcal{H}_4(pk_{\mathtt{A}}, \omega)^s$, $D = e(pk_{\mathtt{A}}, Z)^s \cdot R$ et $E = m \oplus \mathcal{H}_2(R)$ où $s \in\xleftarrow{\$} \mathbb{Z}_p^*$ et $\mathcal{R} \xleftarrow{\$} \mathbb{G}_2^*$. Il est facile de voir que pour déchiffrer, il faut récupérer R qui est masqué dans D par $e(pk_{\mathtt{A}}, Z)^s$. Le re-chiffrement permet de modifier cette composante D en $D' = e(pk_{\mathtt{B}}, Z)^s$ en utilisant $e(a_x, B)/e(b_x, C_\omega) = e(g, Z)^{s \cdot q_x(0)}$. En utilisant l'inverse de cette dernière valeur, il est alors possible de re-chiffrer un chiffré destiné à $\mathtt{B}$ pour la condition ω en un chiffré destiné à $\mathtt{A}$. Ceci nous donne donc un re-chiffrement bidirectionnel et non unidirectionnel qui prend en compte les restrictions non naturelles liées au re-chiffrement bidirectionnel tout en étant moins efficace que ce dernier puisqu'il n'utilise pas directement de clés de la forme $sk_{\mathtt{B}} - sk_{\mathtt{A}}$ mais $Z^{sk_{\mathtt{B}} - sk_{\mathtt{A}}}$. Plus problématique, on voit également que le délégant et le délégué partagent ainsi les mêmes attributs. Ainsi, si $\mathtt{A}$ fait confiance à $\mathtt{B}$ et considère que ce dernier habite à Caen, alors celui-ci considérera également que $\mathtt{A}$ habite à Caen... On voit de cette manière qu'un re-chiffrement bidirectionnel n'est pas idéal pour le re-chiffrement sélectif prenant en compte les attributs.

8.2.3 Notre schéma avec re-chiffrement sélectif et unidirectionnel

Intuition. Notre schéma est basé sur le schéma [LV08a] dont nous avons parlé au Chapitre 4 et qui permet de détecter des serveurs corrompus. Pour rappel, chaque clé de re-chiffrement se voit attribuer une identité différente pour chaque serveur à qui elle sera donnée. En cas de suspicion, cette identité limite la possibilité d'effectuer le re-chiffrement et permet de cette manière de retrouver le serveur à incriminer. Notre idée consiste à en reprendre les fonctionnalités et à considérer cette identité comme un attribut qui limitera le re-chiffrement. À la différence de ce schéma, qui chiffre un message pour toutes ces identités possibles, notre schéma ne le chiffrera que pour un ensemble d'attributs afin que toute clé de re-chiffrement ne puisse le transformer. De plus, nous autorisons les identités à prendre des longueurs différentes pour permettre la considération d'arbres dont les branches peuvent avoir un nombre de niveaux différents comme dans la Figure 8.3.

Notation. Le schéma sur lequel nous nous basons est lui-même construit à partir du schéma de chiffrement basé sur l'identité de Waters [Wat05]. Nous serons donc amenés à utiliser la fonction de Waters définie comme suit : pour tout $n \in \mathbb{N}^*$, pour tout $n+1$-upplet $\omega = (\omega_0, \cdots, \omega_n)$ et tout n-bits $b = (b_1, \cdots, b_n) \in \{0,1\}^n$, la fonction F de Waters sur b pour ω est définie par $F_\omega(b) = \omega_0 \cdot \prod_{i=1}^n \omega_i^{b_i}$.

Description du schéma.

1. Nous parlerons ensuite de la possibilité de modifier l'arbre $\mathcal{T}_{\mathtt{A}}$ et donc l'ensemble $\mathsf{Cond}_{\mathtt{A}}$.

$\mathsf{Environnement}$: il est composé d'un ensemble $\mathsf{Type} = \mathsf{Type}_1 \times \mathsf{Cond} \cup \mathsf{Type}_2$ où $\mathsf{Type}_1 = \{1\}$, $\mathsf{Type}_2 = \{2\}$. L'ensemble $\mathsf{Liaison}$ est égal à $\{((1, \mathsf{att}), 2)\}_{\mathsf{att} \in \mathsf{Att}}$. Les structures de données, indiquant les données requises pour calculer les différentes clés de re-chiffrement, sont toutes composées de la clé privée du délégant et de la clé publique du délégué.

$\mathsf{Initialiser}(\lambda) \to \mathcal{P}$: soit $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, e, \Psi)$ un couplage bilinéaire asymétrique de $\mathbb{G}_1 \times \mathbb{G}_2$ dans $\mathbb{G}_t$ où $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t$ sont trois groupes d'ordre q premier et Ψ un morphisme de groupe de $\mathbb{G}_2$ dans $\mathbb{G}_1$ qui n'est pas publiquement inversible. Soit $h \in \mathbb{G}_2$ et $g = \Psi(h)$. Considérons $\mathcal{M} = \mathbb{G}_t$ comme ensemble des messages. Les paramètres publics sont

$$\mathcal{P} = ((q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \Psi), g, h, \mathcal{M}).$$

$\mathsf{Clé}(\mathcal{P})$: prend en entrée les paramètres publics $\mathcal{P}$, tire aléatoirement $y, z \xleftarrow{\$} \mathbb{Z}_q^*$ et calcule $Y = h^y$ et $Z = e(g, h)^z$. L'utilisateur spécifie alors le nombre de niveaux l qu'il souhaite avoir dans son arbre de délégation ainsi que pour chaque niveau i, le nombre de bits n_i. Pour chaque $i \in [\![1; l]\!]$, pour chaque $j \in [\![0; n_i]\!]$, cet algorithme tire aléatoirement $v_{i,j} \in \mathbb{Z}_q^*$, puis calcule le vecteur V (en fait, un vecteur de vecteurs) tel que la (i, j)-ème composante $\mathtt{V}_{i,j}$ soit égale à $= h^{v_{i,j}}$. Il définit le vecteur U comme étant égal à $\Psi(\mathsf{V})$, signifiant que U est calculé comme l'image de $\mathtt{V}$ composante par composante. Finalement il définit $sk = (y, z, \mathsf{V})$ et $pk = (Y, Z, \mathsf{U})$.

$\mathsf{Délégation}_{(1,2)}(\mathcal{P}, \mathsf{att}, sk_{\mathtt{A}}, pk_{\mathtt{B}})$: prend en entrée les paramètres publics $\mathcal{P}$, un attribut $\mathsf{att} = (\mathsf{att}_{i,j})_{i \in [\![1;k]\!], j \in [\![1; n_{\mathtt{A},i}]\!]}$ où $k \leq l_{\mathtt{A}}$ et $\mathsf{att}_{i,j} \in \{0, 1\}$, la clé privée $sk_{\mathtt{A}} = (y_{\mathtt{A}}, z_{\mathtt{A}}, \mathsf{V}_{\mathtt{A}})$ du délégant et la clé publique $pk_{\mathtt{B}} = (Y_{\mathtt{B}}, Z_{\mathtt{B}}, \mathsf{U}_{\mathtt{B}})$ du délégué. Cet algorithme tire aléatoirement $r \in \mathbb{Z}_q^*$ et calcule $\tilde{R}_{\mathtt{A},(\eta,\mathsf{att}) \to \mathtt{B},\eta'} = Y_{\mathtt{B}}^{z_{\mathtt{A}}} \cdot \left(\prod_{i=1}^{k} F_{\mathsf{V}_{\mathtt{A},i}}(\mathsf{att}_i)\right)^r$ et $R = h^r$. Il retourne ensuite la clé de re-chiffrement $R_{\mathtt{A},(\eta,\mathsf{att}) \to \mathtt{B},\eta'} = (\mathsf{att}, \tilde{R}_{\mathtt{A},(\eta,\mathsf{att}) \to \mathtt{B},\eta'}, R)$ du type 1 pour $\mathtt{A}$ avec l'attribut att vers le type 2 pour $\mathtt{B}$.

$\mathsf{Chiffrer}_1(\mathcal{P}, (1, \mathbf{c}), pk, m)$: prend en entrée les paramètres publics $\mathcal{P}$, un type $(1, \mathbf{c})$ appartenant à $\mathsf{Type}_1 \times \mathsf{Cond}$ avec $\mathbf{c} = (\mathbf{c}_i)_{i \in [\![1;k]\!]}$ où $k \leq l$ et $\mathbf{c}_i \in \{0, 1\}^{n_i} \cup \{\star\}^{n_i}$, une clé publique $pk = (Y, Z, \mathsf{U})$ et un message $m \in \mathcal{M}$. Cet algorithme fonctionne ainsi :

1. Tire aléatoirement $s \xleftarrow{\$} \mathbb{Z}_q^*$ et définit $I^\star = \{i \in [\![1; k]\!] \ / \ \mathbf{c}_i = \{\star\}^{n_i}\}$.
2. Calcule $C_1 = m \cdot Z^s$, $C_2 = g^s$.
3. Calcule $C_3 = \left(\prod_{i \notin I^\star} F_{\mathtt{U}_i}(\mathbf{c}_i) \cdot \prod_{i \in I^\star} \mathtt{U}_{i,0}\right)^s$, ce qui correspond aux blocs d'attributs fixés par $\mathbf{c}$. Le produit de termes $\mathtt{U}_{i,0}$ pour $i \in I^\star$ permet de restreindre le déchiffrement aux délégués ayant exactement k attributs.
4. Calcule $\mathbf{C}^{(4)} = \left((\mathtt{U}_{i,j})^s\right)_{i \in I^\star, j \in [\![1; n_i]\!]}$, ce qui correspond aux attributs avec jokers.
4. Définit $C_{pk,(1,\mathbf{c})} = (C_1, C_2, C_3, \mathbf{C}^{(4)})$ comme chiffré de type $(1, \mathbf{c})$.

$\mathsf{Déchiffrer}_{(1,\mathbf{c})}(\mathcal{P}, sk, C_{pk,(1,\mathbf{c})})$: prend en entrée les paramètres publics $\mathcal{P}$, une clé privée $sk = (y, z, \mathsf{V})$ et un chiffré $C_{pk,(1,\mathbf{c})} = (C_1, C_2, C_3, \mathbf{C}^{(4)})$ de type $(1, \mathbf{c})$ pour la clé publique pk associée à sk. Cet algorithme retourne $m = \frac{C_1}{e(C_2, h)^z}$.

$\mathsf{Re\text{-}chiffrer}_{((1,\mathbf{c}),2)}(\mathcal{P}, R_{\mathtt{A},(\eta,\mathsf{att}) \to \mathtt{B},\eta'}, C_{pk_{\mathtt{A}},(1,\mathbf{c})})$: prend en entrée les paramètres publics $\mathcal{P}$, une clé de re-chiffrement $R_{\mathtt{A},(\eta,\mathsf{att}) \to \mathtt{B},\eta'} = (\mathsf{att}, \tilde{R}_{\mathtt{A},\eta \to \mathtt{B},\eta',\mathsf{att}}, R)$ de $\mathtt{A}$ avec l'attribut att vers $\mathtt{B}$ et un chiffré $C_{pk,(1,\mathbf{c})} = (C_1, C_2, C_3, \mathbf{C}^{(4)})$ de type $(1, \mathbf{c})$ pour $\mathtt{A}$. Cet algorithme re-chiffre de la manière suivante :

1. Vérifie si la condition $\mathbf{c}$ contient l'attribut att et renvoie $\perp$ si ce n'est pas le cas.
2. Définit $I^\star = \{i \in [\![1; k]\!] \ / \ \mathbf{c}_i = \{\star\}^{n_{\mathtt{A},i}}\}$
3. Calcule $C_3' = C_3 \cdot \prod_{i \in I^\star} F_{\mathbf{C}_i^{(4)}}(\mathsf{att}_i)$.
4. Calcule $C_2' = \frac{e(C_2, \tilde{R}_{\mathtt{A},\eta \to \mathtt{B},\eta',\mathsf{att}})}{e(C_3', R)}$.

5. Définit $C_{pk_{\mathtt{B}},2} = (C_1, C_2')$ comme chiffré de type 2.

$\mathsf{Chiffrer}_2(\mathcal{P}, pk, m)$: prend en entrée les paramètres publics $\mathcal{P}$, une clé publique $pk = (Y, Z, \mathsf{U})$ et un message $m \in \mathcal{M}$. Cet algorithme fonctionne ainsi :

1. Tire aléatoirement $s \in \mathbb{Z}_q^*$.
2. Calcule $C_1 = m \cdot e(g,h)^s$ et $C_2' = e(g,Y)^s$.
3. Définit $C_{pk,2} = (C_1, C_2')$ comme chiffré de type 2.

$\mathsf{Déchiffrer}_2(\mathcal{P}, sk, C_{pk,2})$: prend en entrée les paramètres publics $\mathcal{P}$, une clé privée $sk = (y, z, \mathsf{V})$ et un chiffré $C_{pk,2} = (C_1, C_2')$ de type 2 pour la clé publique pk associée à sk. Cet algorithme retourne $m = \frac{C_1}{C_2'^{1/y}}$.

Consistance. Seul le re-chiffrement n'est pas directement observable. Sa base s'appuie sur le terme $Y_{\mathtt{B}}^{z_{\mathtt{A}}}$ contenu dans la clé de re-chiffrement qui permet de passer, via l'utilisation d'un couplage bilinéaire, d'un chiffré pour $\mathtt{A}$ portant sur le terme $z_{\mathtt{A}}$ à un chiffré pour $\mathtt{B}$ portant sur le terme $y_{\mathtt{B}}$. Ce terme est cependant masqué par un produit de fonction de Waters portant sur l'attribut att. L'utilisation du couplage bilinéaire permet de retirer ce masque en utilisant les termes C_3', H, C_2.

Théorème 42. *Le schéma atteint la sécurité* IND-CPA*, sous l'hypothèse* DBDH *dans le modèle standard.*

Preuve informelle. Le principe de la preuve est le même que celui de la preuve donnée par Bellare et Ristenpart dans [BR09] sur le schéma de Waters. L'idée est de considérer, lors de la création d'une clé de re-chiffrement, que chaque élément $Y_{\mathtt{B}}^{z_{\mathtt{A}}}$ est la clé maître d'une autorité générant les identités d'un chiffrement basé sur l'identité. Ensuite, on obtient la clé de re-chiffrement de $\mathtt{A}$ vers $\mathtt{B}$ pour un attribut att comme la clé générée pour l'identité att par une autorité possédant la clé maître $Y_{\mathtt{B}}^{z_{\mathtt{A}}}$. □

Compatibilité avec la gestion dynamique des droits d'accès. Tout comme le CPRE, le SPRE s'adapte, d'un point de vue théorique, avec le re-chiffrement combiné en considérant le re-chiffrement sélectif tenant compte de l'attribut sur le re-chiffrement unidirectionnel du combiné. Pour notre schéma, il est facile de s'apercevoir que le fonctionnement du chiffrement de Waters repose sur le chiffrement ElGamal, sur lequel nous pouvons appliquer un re-chiffrement bidirectionnel comme exposé au Chapitre 3. Il n'est donc pas étonnant que l'on puisse également adapter ce schéma en un re-chiffrement combiné. Il y a cependant quelques précautions à prendre. Il est tout d'abord évident que le re-chiffrement bidirectionnel entre un dispositif $\mathtt{A}$ en un dispositif $\mathtt{B}$ ne peut se faire que si leurs clés respectives (publiques et privées) sont de même taille. Cela signifie que leur arbre de délégation respectif et que tous les niveaux de ces arbres possèdent le même nombre de bits. Dans le contraire, il y aurait forcément un couple (i,j) pour lequel $v_{\mathtt{B},i,j}$ ou $v_{\mathtt{A},i,j}$ n'existe pas et auquel cas on ne pourrait calculer de quotient entre ces deux éléments, quotient qui correspondrait à une clé de re-chiffrement bidirectionnelle habituelle. Cette restriction n'est en soit pas un problème, puisque dans le cas du re-chiffrement combiné, nous n'utilisons le re-chiffrement bidirectionnel qu'entre deux dispositifs d'un même utilisateur. Dans ce cas précis, il est donc naturel que l'utilisateur en question crée un arbre de délégation qui lui corresponde et qui soit valable pour tous ces dispositifs. Cette précaution n'est pas suffisante, puisque les clés de re-chiffrement bidirectionnelles correspondent alors aux quotients $r_{i,j} = v_{\mathtt{B},i,j}/v_{\mathtt{A},i,j}$, pour tout i et j. À la différence des schémas exposés jusqu'à présent, les exponentiations considérées ne sont plus isolées mais réunies en produit. On ne peut donc pas élever à une puissance une seule de ces exponentiations mais toutes à la fois. Cette puissance que l'on va appliquer lors du re-chiffrement bidirectionnel, qui correspond évidemment à la clé de re-chiffrement bidirectionnelle, doit donc être indépendante de i et de j tout en correspondant au quotient $r_{i,j} = v_{\mathtt{B},i,j}/v_{\mathtt{A},i,j}$, et ce pour tout i et j afin d'obtenir un re-chiffré valide. Cette particularité est possible en liant les clés privées de $\mathtt{A}$ et de $\mathtt{B}$. Supposons par exemple que la clé privée de $\mathtt{A}$ ait déjà été générée, $\mathtt{B}$

commence alors par générer de manière aléatoire un élément $v_{\mathtt{B},1,1}$ (ou n'importe quel élément $v_{\mathtt{B},i',j'}$) puis calcule les autres éléments via la formule $v_{\mathtt{B},i,j} = v_{\mathtt{A},i,j} \cdot v_{\mathtt{B},1,1}/v_{\mathtt{A},1,1}$ (formule qui devient $v_{\mathtt{B},i,j} = v_{\mathtt{A},i,j} \cdot v_{\mathtt{B},i',j'}/v_{\mathtt{A},i',j'}$ dans le cas de l'élément $v_{\mathtt{B},i',j'}$). Nous obtenons ainsi un re-chiffrement combiné sur ce schéma. Il peut donc bien bénéficier d'une gestion dynamique des droits d'accès.

Nous pouvons cependant aller plus loin dans cette réflexion. En effet, puisqu'un délégant peut ajouter des délégués à n'importe quel moment, il est naturel de lui offrir la possibilité de modifier son arbre de délégation et en particulier lui permettre de modifier les attributs de ses délégués. Il serait donc souhaitable que son arbre de délégation puisse être modifié à sa guise. Cela peut se faire de différentes manières :

- le délégant peut ajouter de nouveaux niveaux dans l'arbre ce qui permet d'augmenter le nombre d'attributs des délégués et donc modifie la valeur l ;
- le délégant peut ajouter des bits à un niveau de l'arbre, ce qui permet ainsi d'ajouter des attributs sur un niveau donné et donc modifier les entiers n_i.

Cette mise à jour de l'arbre est possible dans notre système. Il suffit pour cela, que le délégant, lorsqu'il le souhaite, mette à jour sa paire de clés (privée/publique) ainsi que les clés de re-chiffrement qu'il a calculées. Les modifications à apporter au schéma suivent la même idée que la dérivation de clés expliquée dans [ACD+06].

Note sur les modèles étendus de serveur de re-chiffrement

Tout au long de ce mémoire, nous nous sommes focalisés sur un modèle de serveur de re-chiffrement utilisant un schéma de chiffrement à clé publique tel que présenté dans la Définition 19 donnée au Chapitre 2 et sans aucune autre propriété. Il existe cependant d'autres modèles utilisant des chiffrements différents. Sur le principe, le re-chiffrement consiste à modifier le destinataire du chiffré et ne dépend pas des fonctionnalités offertes par le chiffrement considéré. D'un point de vue théorique, tous les chiffrements existants sont donc compatibles avec cette notion de re-chiffrement. Nous passons ici en revue certains de ces autres modèles comme celui basé sur le chiffrement basé sur l'identité. Nous ne nous attardons pas sur la sécurité de ces différents modèles, l'idée pour l'obtenir consistant à reprendre la sécurité correspondant au re-chiffrement présenté au Chapitre 3 et d'y inclure les éventuelles spécificités du chiffrement considéré.

Le chiffrement basé sur l'identité

Le concept du chiffrement basé sur l'identité ou IBE, de l'anglais *identity-based encryption*, a été proposé par Shamir dans [Sha84]. Cependant la réalisation de schémas sûrs resta un problème ouvert durant presque deux décennies jusqu'à ce qu'il soit résolu par Boneh et Franklin [BF01] ainsi que par Cocks [Coc01] - ce dernier ne permettant de chiffrer qu'un bit à la fois. Ce genre de chiffrement consiste à prendre comme clé publique un élément identifiant un utilisateur, comme par exemple son nom, prénom et/ou date de naissance. L'identifiant retenu doit bien sûr être différent pour chaque utilisateur du système, car si plusieurs d'entre eux possédaient le même, ils auraient une clé publique identique. Quant aux clés privées, puisqu'elles sont liées aux clés publiques dépendantes d'un élément identifiant, elles sont générées par une autorité de confiance qui utilise alors une clé maître qu'elle est la seule à connaître. La différence entre ce chiffrement et un chiffrement classique se fait au niveau de la génération des clés secrètes et publiques. La partie chiffrement/déchiffrement reste inchangée.

Etat de l'art des serveurs de re-chiffrement. Un schéma de serveur de re-chiffrement dont le chiffrement est un IBE s'appelle un serveur de re-chiffrement basé sur l'identité, de l'anglais *identity-based proxy re-encryption* (IBPRE). Les premiers schémas de ce type sont dûs à Green et Ateniese qui ont présenté dans [GA07] deux schémas avec re-chiffrement unidirectionnel et unique, l'un atteignant la sécurité IND-CPA et l'autre atteignant la sécurité IND-CCA. Ils y ont également introduit la technique que nous avons généralisé dans la Section 4.2.4 afin de transformer le premier en un schéma avec re-chiffrement unidirectionnel et infini atteignant également la sécurité IND-CPA. Ces trois schémas utilisent des techniques similaires et nous nous contentons d'expliquer le fonctionnement du re-chiffrement du schéma atteignant la sécurité IND-CCA. Ce schéma est basé sur une version modifiée de l'IBE de Boneh et Franklin [BF01]. Le mécanisme du re-chiffrement considéré consiste à calculer une clé d'un délégant avec identité $\mathsf{ID_A}$ pour un délégué avec identité $\mathtt{B}$ égale à $R_{\mathsf{ID_A}\to\mathsf{ID_B}} = sk_{\mathsf{ID_A}} \cdot \mathcal{H}_2(\mathsf{ID_B})$ où $sk_{\mathsf{ID_A}} = \mathcal{H}_1(\mathsf{ID_A})^{msk}$,

où $\mathcal{H}_1$ et $\mathcal{H}_2$ sont deux fonctions de hachage. Il n'est cependant pas suffisant pour garantir que seul B pourra accéder aux données, puisque le serveur peut par exemple transformer cette clé en $sk_{\mathsf{ID}} \cdot H_2(\mathsf{ID}'') = R_{\mathsf{ID_A} \to \mathsf{ID_B}} \cdot H_2(\mathsf{ID}'')/H_2(\mathsf{ID}')$ qui permet ainsi à C d'avoir accès aux données de A. Pour empêcher cette éventuelle attaque, l'idée consiste à ajouter à la fonction $\mathcal{H}_2$ une entrée additionnelle qui n'est connue que de A. Ainsi, ce schéma considère la clé de re-chiffrement $R_{\mathsf{ID_A} \to \mathsf{ID_B}} = sk_{\mathsf{ID_A}} \cdot \mathcal{H}_2(K \| \mathsf{ID_A} \| \mathsf{ID_B} \| N)$ où $K = e(sk_{\mathsf{ID_A}}, H_1(\mathsf{ID_B}))$ et $N \xleftarrow{\$} \{0,1\}^n$. Pour autant et bien qu'il soit annoncé comme atteignant la sécurité IND-CCA, ce schéma n'atteint en réalité que la sécurité IND-CPA. En effet, il est facile de voir que l'algorithme Re-chiffrer ne vérifie pas la validité d'un chiffré avant de le re-chiffrer. En détail et avec leurs notations, un adversaire peut modifier un chiffré $C_{\mathsf{ID_A}} = (C_1, C_2, C_3, S)$ en $C'_{\mathsf{ID_A}} = (C_1, C_2, C_3 \oplus R, S)$, où $R \xleftarrow{\$} \{0,1\}^n$, qui ne chiffre plus m mais $m \oplus R$. Celui-ci est valide pour l'algorithme Re-chiffrer qui accepte alors de le transformer pour pour une entité corrompue par l'adversaire, ce qui lui permet ainsi d'obtenir $m \oplus R$ et donc m.

D'autres schémas basés sur ce dernier ont également vu le jour comme celui de Wang *et al.* qui ont proposé, dans [WWMO10], un schéma empêchant une telle attaque. Cependant, il est possible qu'il présente de potentielles failles de sécurité. En effet, l'algorithme Re-chiffrer ne vérifie pas par exemple, selon leurs notations, qu'un chiffré est calculé avec un élément r égal à $\mathcal{H}(m \| \sigma)$ où $\mathcal{H}$ est une fonction de hachage et σ un aléa. Le lecteur attentif aura remarqué que nous avons étudié ce genre de situation dans le Chapitre 6. Hang *et al.* en ont proposé un dans [WCW10] avec re-chiffrement multiple annoncé comme atteignant la sécurité IND-CCA qui fut par la suite démontré comme ne l'étant pas par Zhang et Wang dans [ZW12b].

Chu et Tzeng ont conçu dans [CT07] un IBPRE en se basant sur l'IBE de Waters [Wat05] ainsi que sur les techniques présentées par Boyen *et al.* dans [BMW05] qui permettent de le transformer en un schéma IND-CCA sûr. Ce schéma utilise la technique que nous avons considérée à la Section 4.2.4 afin d'obtenir un schéma avec re-chiffrement multiple et unidirectionnel. Comme nous l'avons mentionné dans cette section, la sécurité que l'on pourra obtenir de cette manière ne sera que IND-CPA sûr, ce qui contredit la sécurité annoncée par les auteurs. Il est en effet facile de voir que leur re-chiffrés successifs ne disposent plus de vérifiabilité publique, ce qui va permettre à un adversaire de pouvoir aisément casser le schéma. Pour autant, si l'on se restreint à un re-chiffrement unique et unidirectionnel en utilisant le même genre de clés publiques comme nous l'avons fait au Chapitre 6, alors le schéma semble atteindre la sécurité IND-CCA sous l'hypothèse DBDH. Lai *et al.* ont également proposé un schéma avec re-chiffrement unidirectionnel et unique en se basant sur l'IBE de Waters dans [LZD+10]. On remarque que le mécanisme de leurs clés de re-chiffrement utilise notamment une entité virtuelle comme nous l'avons présenté au Chapitre 7 pour transformer un re-chiffrement bidirectionnel en un unidirectionnel. Ce schéma est IND-CCA sûr sous l'hypothèse DBDH.

Luo *et al* ont proposé dans [LSC11] un schéma avec re-chiffrement infini et unidirectionnel basé sur le chiffrement IBE de Lewko et Waters décrit dans [LW10], qui utilise notamment des groupes d'ordre composite $n = p_1 p_2 p_3$ où p_1, p_2, p_3 sont trois nombres premiers distincts. Il est également annoncé comme atteignant la sécurité IND-CPA. Il est cependant facile de concevoir une attaque sur ce schéma signifiant qu'il n'est même pas IND-CPA sûr. En détail et avec les notations des auteurs, il transforme un chiffré pour $C_{\mathsf{ID_A}} = (C, C_1, C_2, C_3)$ en $C_{\mathsf{ID_B}} = (C, C'_1, C_2, C_3)$ où $C'_1 = C_1^{rk_1} \cdot e(C_2, rk_2)^{-1}$ et (rk_1, rk_2) la clé de re-chiffrement unidirectionnelle de A vers B. Il présente en plus les particularités suivantes :

- C'_1 est exactement de la même forme que C_1 où seule l'identité destinatrice a été modifiée de $\mathsf{ID_A}$ en ID_B ;
- la clé de A vers B ne nécessite pas la clé privée de B pour sa génération.

La première permet ainsi au serveur disposant de la clé (rk_1, rk_2) de A vers B de re-chiffrer tout chiffré (C, C_1, C_2, C_3) destiné à $\mathsf{ID_B}$ en un chiffré (C, C'_1, C_2, C_3) destiné à A, en calculant

$C'_1 = (C_1 \cdot e(C_2, rk_2))^{1/rk_1}$ obtenu en inversant le calcul effectué par l'algorithme Re-chiffrer. Ceci signifie que le re-chiffrement est bidirectionnel et non unidirectionnel. Le point important est, d'après la seconde particularité, que B n'a jamais eu à intervenir dans le calcul de la clé de re-chiffrement considérée. Le schéma ne peut être sécurisé puisqu'il est possible pour A de se créer une clé de re-chiffrement de B vers A sans l'accord de B.

Shao *et al.* se sont intéressés à ajouter le re-chiffrement conditionnel à ce modèle d'IBPRE. Ils ont ainsi proposé dans [SWLX11] un schéma avec re-chiffrement unidirectionnel et unique, en utilisant comme base l'IBE de Boneh-Franklin. Ce schéma est annoncé comme atteignant la sécurité IND-CCA sous l'hypothèse DBDH mais présente les mêmes failles que celles que nous avons vues au Chapitre 6. Liang *et al.* ont, quant à eux, proposé dans [LLT+12], un schéma avec re-chiffrement conditionnel, unidirectionnel et unique avec comme base l'IBE de Waters. Il n'atteint également que la sécurité IND-CPA sous l'hypothèse DBDH, contrairement à ce qui est annoncé, puisque leur composante C_1 n'est pas incluse dans la vérifiabilité publique.

Dans [Mat07], Matsuo a exploré une autre voie en décrivant une primitive à mi-chemin entre la primitive de PRE étudiée tout au long de ce mémoire et la primitive d'IBPRE. Son idée consiste à transformer un chiffré destiné à une clé publique $pk_{\mathtt{A}}$, générée en utilisant un schéma de chiffrement à clé publique conforme à la Définition 19, en un chiffré destiné à une identité $\mathsf{ID}_{\mathtt{B}}$, correspondant à un chiffré d'un IBE. Sa construction transforme notamment un chiffré ElGamal en un chiffré IBE de Boneh-Boyen [BB04]. Wang et Yang ont cependant montré dans [WY10] que la sécurité de ce schéma présentait plusieurs failles. Tang *et al.* ont eu une idée similaire dans [THJ08] en proposant un re-chiffrement ne considérant que des chiffrés IBE mais dont les identités du délégant et du délégué sont enregistrées auprès d'autorités de génération de clés différentes. Leur construction est basée sur l'IBPRE de Green et Ateniese présenté ci-dessus et atteint la sécurité IND-CPA sous l'hypothèse DBDH. Ces deux directions de recherche permettent ainsi à un délégant de déléguer ses droits d'accès à un utilisateur non enregistré dans le même système que le sien.

Nous voyons par cette étude que la notion de vérifiabilité publique est très importante, puisque la plupart des schémas présentés ici ne la prenne pas en compte et n'atteignent que la sécurité IND-CPA. Il en résulte, pour le moment, que seuls les schémas d'IBPRE basés sur le chiffrement de Waters semblent atteindre la sécurité IND-CCA.

Le chiffrement de groupe

Nous avons déjà présenté au Chapitre 1 la notion de chiffrement de groupe que nous avons notée BE. Elle est, elle aussi, compatible avec le serveur de re-chiffrement.

État de l'art des serveurs de re-chiffrement. Un schéma de serveur de re-chiffrement dont le chiffrement est un BE s'appelle un serveur de re-chiffrement de groupe, de l'anglais *proxy broadcast re-encryption* (PBRE). Il y a en réalité plusieurs types de PBRE. Le re-chiffrement peut s'effectuer d'un utilisateur vers un groupe d'utilisateurs peu importe le groupe auquel appartient ce premier ou alors d'un groupe vers un autre groupe, la clé de re-chiffrement étant dans ce cas calculée par une autorité de groupe.

Sun et Hu ont proposé dans [SH12] un système offrant un re-chiffrement entre un schéma de chiffrement à clé publique classique et le chiffrement de groupe. Ainsi un chiffré destiné à un utilisateur peut être re-chiffré en un chiffré destiné à tout un groupe. Leur schéma est essentiellement basé sur le schéma [LV08b] et est annoncé comme atteignant la sécurité IND-CCA. Il présente cependant une attaque évidente, puisqu'un chiffré de la forme (t, C_1, C_2, C_3, C_4) est transformé en (t, C_1, C'_2, C_3, C_4). Le problème est que la vérification publique d'un chiffré re-chiffrable ne s'assure pas que la composante C_3 soit liée aux autres. Ainsi en remplaçant temporairement C_3

par n'importe quelle autre valeur, l'adversaire peut obtenir un re-chiffrement du chiffré challenge vers un utilisateur corrompu.

Mao et Ao ont eu une approche différente en considérant que le re-chiffrement se déroule d'un groupe vers un autre. Leur schéma étant bidirectionnel, chaque délégation d'un groupe vers un autre est alors décidée par les autorités respectives du groupe "délégant" et du groupe "délégué". Ils ont présenté un schéma CCA sûr sous l'hypothèse V-DDH dans [MA09a] qu'ils ont ensuite légèrement modifié dans [MA09b]. Cependant, on s'aperçoit vite que leur schéma est malléable et ne peut donc atteindre la sécurité IND-CCA. En effet, aucune vérification n'est faite sur la validité d'un chiffré, même lors du déchiffrement, ce qui permet de les modifier très facilement. Très récemment, Purushothama et Amberker ont exhibé une attaque utilisant cette non-vérifiabilité des chiffrés dans [PA13].

Chu *et al.* ont présenté dans [CWC+09] un exemple de schéma offrant un re-chiffrement d'un utilisateur délégant vers un groupe d'utilisateurs qui permet de modifier tout chiffré à destination de tous les ensembles d'utilisateurs auquels appartient ce délégant. Ils y ont également ajouté le re-chiffrement conditionnel. Leur schéma est basé sur le schéma d'Attrapadung *et al.* publié dans [AFI06] qui se base lui même sur le BE de Boneh *et al.* [BGW05] et l'IBE de Boneh et Boyen [BB04]. Ce schéma atteint la sécurité IND-CPA sous l'hypothèse n-DBDHE. Ils en ont également proposé une version IND-RCCA sûre en se servant en plus des techniques utilisées dans [BMW05]. Liang *et al.* ont aussi proposé un schéma de ce type dans [LHS+13] en se basant sur le schéma BE de Delerablée [Del07] qui est IND-CCA sûr sous l'hypothèse $(P, Q, f)-$GDDDHE.

Le chiffrement basé sur les attributs

Nous avons déjà parlé au Chapitre 3 de chiffrement basé sur les attributs noté ABE.

État de l'art des serveurs de re-chiffrement. Un schéma de serveur de re-chiffrement dont le chiffrement est un ABE s'appelle un serveur de re-chiffrement basé sur les attributs, de l'anglais *attribute-based proxy re-encryption* (ABPRE). Certains travaux se sont intéressés à proposer des ABPRE. Liang *et al.* ont été les premiers dans [LCLS09] à explorer cette voie en présentant un schéma avec re-chiffrement unidirectionnel et unique qui permet à un utilisateur de déléguer ses droits d'accès à un ensemble de délégués possédant un certain nombre d'attributs. Leur schéma est basé sur le schéma [CN07] et atteint la sécurité IND-CPA sous l'hypothèse ADBDH. Liang *et al.* ont proposé également un schéma de ce type en se basant sur le schéma [Wat11] sous l'hypothèse $q-$parallèle BDHE.

Le chiffrement totalement homomorphe

Le chiffrement totalement homomorphe, que nous avons noté FHE et que nous avons abordé au Chapitre 4, est différent des autres chiffrements que nous venons de présenter. Pour rappel, il permet d'effectuer n'importe quelle opération sur les chiffrés, ce qui peut être utile pour réaliser un service de calcul dématérialisé garantissant la confidentialité des données. Le FHE est plus que compatible avec la primitive de PRE puisque le re-chiffrement n'est, comme nous l'avons déjà vu, rien d'autre que l'exécution d'une opération sur les chiffrés pour en changer le destinataire. Utiliser un FHE plutôt qu'un PRE présente l'avantage de pouvoir disposer d'une informatique dématérialisée pas seulement de stockage mais aussi de calcul qui permet de partager des données avec des utilisateurs de confiance tout en préservant la vie privée. Néanmoins, les schémas FHE dont nous disposons sont encore loin d'être efficaces par rapport aux schémas PRE existants (exception faite pour celui de Gentry qui utilise un FHE).

Conclusion sur les modèles étendus

Comme nous venons de le voir sur les schémas [SWLX11, LLT+12, CWC+09], le re-chiffrement conditionnel semble bien s'adapter à ces modèles étendus de PRE. Pour des raisons similaires, il est sûrement possible de transposer les idées et résultats présentés au cours de ce mémoire à ces derniers, en effectuant quelques adaptations afin de tenir compte des particularités du chiffrement considéré.

Le re-chiffrement semble donc être compatible avec tout un ensemble de chiffrements possédant différentes particularités. Nous avons vu quelques uns des modèles étendus mais rien n'interdit, selon les besoins, d'en concevoir de nouveaux. Surtout si l'on considère des re-chiffrements multiples, le choix devient plus large. On peut par exemple imaginer un premier re-chiffrement qui transforme un chiffré basé sur l'identité en un chiffré pour un groupe et un deuxième re-chiffrement transformant, avec condition, ce dernier en un chiffré basé sur les attributs. La notion de serveur de re-chiffrement est donc très malléable, ce qui permet de l'envisager dans de nombreuses situations en fonction notamment des relations qu'ont les différents utilisateurs entre eux (s'ils font partie d'un groupe ou non par exemple). La tendance serait même d'essayer de lui ajouter toutes les propriétés possibles que l'on peut rencontrer autour de la primitive de chiffrement sans pour autant justifier l'apport du re-chiffrement. Il faut donc rester prudent sur l'intérêt des modèles exotiques que l'on peut rencontrer.

Conclusion

Dans cette thèse nous avons étudié la primitive de serveur de re-chiffrement ainsi que son utilisation pour la réalisation d'un stockage de données sécurisé sur serveur. Cette conclusion est l'occasion de rappeller les différentes contributions de ce mémoire et de donner quelques perspectives pour la recherche future.

Contributions

Dans une première partie, nous nous sommes focalisés sur la formalisation de la primitive de serveur de re-chiffrement afin de proposer un modèle unifié regroupant tous les modèles possibles, peu importe le re-chiffrement considéré. Comme nous l'avons vu, il serait très laborieux de décrire précisément le modèle de sécurité à prendre en considération dans le cas générique mais nous avons fourni toutes les idées qui permettront facilement d'obtenir le modèle d'un schéma en particulier en fonction de son "re-chiffrement". Nous avons également décrit les différentes techniques de conception existantes et étudié les relations entre les principaux modèles.

Dans une seconde partie, nous nous sommes intéressés à la sécurité de certains schémas. Une première étude a mis en évidence que la transformation générique [CKN03], appliquée au serveur de re-chiffrement, n'était pas suffisante pour passer d'une sécurité IND-RCCA à une sécurité IND-CCA. Nous avons ensuite présenté, pour les schémas avec re-chiffrement unique, les outils supplémentaires à y ajouter pour y parvenir. Nous en avons profité pour montrer comment les utiliser en pratique afin d'améliorer la sécurité d'un schéma existant. Par la suite, nous nous sommes concentrés sur la réalisation de schémas de serveur de re-chiffrement fondés sur le chiffrement Hash-ElGamal et de preuves de connaissance "à la Schnorr". Même si nous n'avons étudié précisément que deux schémas basés sur ces outils, les résultats obtenus peuvent être étendus à tous les schémas les utilisant. Nous avons alors montré que ces deux outils n'étaient pas suffisants pour obtenir des schémas qui atteignent la sécurité IND-CCA, à moins de dégrader fortement le modèle. Nous avons proposé des modifications en utilisant notamment des preuves "à la Schnorr" avec extracteur en direct du secret afin d'obtenir une telle sécurité sans restrictions non naturelles (excepté pour les schémas avec re-chiffrement bidirectionnel qui atteignent alors la sécurité IND-CCA$_{\mathrm{B}}$).

Dans la dernière partie, nous nous sommes interrogés sur la mise en application de ces schémas afin de proposer le stockage le plus fonctionnel possible. Nous nous sommes d'abord intéressés à la gestion dynamique des droits d'accès. Par nature, le serveur de re-chiffrement offre une dynamique partielle de ces droits. Afin de parvenir à une gestion dynamique complète, nous avons élaboré un nouveau modèle de serveur de re-chiffrement, le serveur de re-chiffrement combiné, réunissant les deux modèles les plus courants, à savoir le re-chiffrement unique et unidirectionnel et le re-chiffrement inifini et bidirectionnel. Après avoir analysé les relations existantes entre ces différents modèles, nous avons démontré qu'il était possible de réaliser des schémas avec re-chiffrement combiné efficaces et sécurisés. Enfin, nous nous sommes intéressés au serveur de re-chiffrement conditionnel. Afin dobtenir une gestion encore plus fine des droits d'accès, nous

avons proposé le modèle avec re-chiffrement sélectif. Nous en avons exposé un schéma avec re-chiffrement unidirectionnel unique basé sur le chiffrement de Waters. Comme mentionné à la fin du Chapitre 8, ce schéma est également compatible avec le re-chiffrement combiné.

Travaux futurs

D'un point de vue théorique, l'un des problèmes majeurs évoqué dans de nombreux articles est de trouver un schéma avec re-chiffrement unidirectionnel unique, qui n'utilise pas de couplages (ou fonctions multi-linéaires) et qui soit IND-CCA sûr dans le modèle standard, sans la considération d'autres hypothèses que des hypothèses algorithmiques (les schémas [SCL10, WWZ10] utilisent notamment un schéma à chiffrement à clé secrète atteignant la sécurité IND-CCA).

Un autre problème est de concevoir des schémas avec re-chiffrement multiple et unidirectionnel atteignant une sécurité IND-CCA (dans le modèle standard ou même de l'oracle aléatoire) tout en étant efficace (le schéma de Gentry offre une solution à base de chiffrement totalement homomorphe). Il serait alors intéressant d'étudier la cryptographie fondée sur les réseaux euclidiens, afin d'essayer de l'adapter au cas du serveur de re-chiffrement. L'opération de re-chiffrement consiste à effectuer justement une opération particulière sur le chiffré, à savoir changer son destinataire. La cryptographie basée sur ces réseaux permettant d'effectuer, via le schéma de Gentry, n'importe quelle opération sur les chiffrés, il est donc théoriquement envisageable de l'utiliser pour ne changer que le destinataire d'un chiffré. De cette manière, on pourrait espérer obtenir un schéma avec re-chiffrement multiple et unique qui soit plus efficace que celui de Gentry. Ce type de cryptographie permet également de construire des fonctions multi-linéaires dont nous avons montré l'intérêt pour le serveur de re-chiffrement au Chapitre 4. Il faudrait étudier ces fonctions pour voir si les hypothèses algorithmiques dont nous aurions besoin pour prouver la sécurité de nos schémas seraient difficiles dans les environnements multi-linéaires dont on dispose.

Concernant le re-chiffrement sélectif, seul le schéma que nous avons proposé est unidirectionnel. Il pourrait être intéressant d'en construire d'autres, atteignant si besoin une meilleure sécurité, puisque le schéma que nous avons proposé atteint la sécurité IND-CPA. En observant les techniques utilisées pour concevoir ce schéma, on pourrait se demander s'il n'est pas possible de concevoir une construction générique à partir d'un serveur de re-chiffrement sans condition et d'un chiffrement basé sur l'identité avec jokers, sous certaines conditions de compatibilité entre eux. En effet, on voit dans notre construction qu'une clé de re-chiffrement sélective entre un délégant `A` et un délégué `B` n'est rien d'autre qu'une clé de re-chiffrement entre `A` et `B`, que l'on a dérivée pour l'identité correspondant à la condition sélective.

Afin de pouvoir offrir d'autres fonctionnalités aux utilisateurs de ces systèmes, nous sommes en train de mener une étude pour réfléchir à la gestion fine des droits d'accès non plus sur les délégués mais sur les fichiers à partager. En effet, sur un ordinateur, les fichiers font généralement partie d'une arborescence avec des dossiers et sous-dossiers... Il est naturel que ces différents dossiers ne soient pas partagés avec les mêmes délégués : ils vont donc posséder des droits d'accès différents. En revanche, il paraît évident que, si un délégué obtient les droits d'accès à un dossier, il puisse également accéder à ses sous-dossiers et ainsi de suite. Pour répondre à ce besoin, une idée serait d'effectuer un re-conditionnement de la condition incluse dans un chiffré. Un délégué, qui aurait accès à la condition correspondant à un dossier D_1, pourrait alors obtenir un fichier se trouvant dans un sous-dossier D_2 du dossier D_1 bien que celui-ci porte sur une condition différente. Ce re-conditionnement permettrait également au délégant d'effectuer des opérations sur son arborescence, de la même manière qu'il le ferait sur son ordinateur, sans avoir à récupérer les données à chaque opération, comme par exemple le déplacement, la copie et la fusion de dossiers.

Autres contributions non présentées dans ce mémoire

D'autres contributions non présentées dans ce mémoire ont été réalisées au cours de ces trois années de doctorat. Elles s'articulent autour de la délégation sécurisée de calculs cryptographiques afin d'améliorer l'efficacité des schémas dont les calculs nécessitent encore trop de puissance. Le principe de la délégation repose sur l'exécution de calculs, initialement à la charge d'un dispositif peu puissant (comme le smartphone ou même une carte SIM), par un dispositif plus puissant (un serveur connecté au smartphone par la 3G ou même, dans le cas de la carte SIM, du smartphone la contenant).

Amélioration de l'efficacité de schémas cryptographiques utilisant les couplages. Dans [CDDT12], nous nous sommes focalisés sur les schémas cryptographiques utilisant les couplages. Bien que couramment utilisés pour la conception de nouveaux schémas, leurs calculs sont encore considérés comme coûteux. Après avoir présenté des mesures de performances de l'implémentation des couplages sur smartphone, nous avons étudié les moyens pour améliorer les schémas les utilisant. Nous avons ensuite montré leur efficacité au travers de la signature de groupe [BBS04].

Délégation de calculs cryptographiques. Dans [CCD+13], nous nous sommes intéressés à la manière d'utiliser génériquement les délégations existantes de calculs cryptographiques. Nous y avons fourni une méthode permettant d'automatiser certains choix de délégations à utiliser selon la sécurité voulue et les performances des dispositifs considérés. Cette méthode retourne également le meilleur moyen de répartir les différents calculs à exécuter entre le dispositif peu puissant et celui qui l'est plus afin d'avoir une exécution globale la plus efficace possible. Nous y avons aussi présenté un exemple d'utilisation de cette méthode sur la signature de groupe XSGS [DP06]. Nos résultats coïncident avec ceux obtenus par l'étude détaillée [CCdMP10] qui ne se focalisait que sur ce schéma, ce qui permet de confirmer le bon fonctionnement de notre méthode.

Délégation de calculs de couplages. Nous nous sommes également penchés sur la manière de déléguer un calcul de couplage. En particulier, nous avons recherché un moyen pour déléguer ce calcul qui soit vérifiable et au besoin confidentiel [2]. Les délégations les plus efficaces avec vérifiabilité ont un inconvénient : la puissance de calculs requise par le dispositif de faible puissance est de l'ordre de trois à quatre fois celle requise pour effectuer le calcul du couplage sans délégation. Leur utilisation n'est donc pas intéressante. Nous avons mis au point une méthode de délégation avec vérifiabilité ne requiérant qu'une seule exponentiation finale devant être effectuée par le dispositif de faible puissance, ce qui correspond environ à la moitié de la puissance requise pour l'exécution complète du calcul du couplage. Cette méthode fournit donc une délégation intéressante. Elle est en plus compatible avec la méthode de Girault-Lefranc [GL05] qui permet d'obtenir la confidentialité, ce qui fournit ainsi une méthode avec vérifiabilité et confidentialité avec deux exponentiations finales. Dans ce cas, elle a tout de même un intérêt puisqu'il n'est pas nécessaire d'implémenter l'algorithme de Miller dans le dispositif de faible puissance. Ces résultats sont en cours de soumission.

2. Si les données d'entrées du couplage sont secrètes, alors elles doivent le rester même après exécution de la délégation.

Liste des figures

1.1 Vue d'ensemble du stockage dématérialisé souhaité 6
1.2 Fonctionnement général d'un PRE 10
1.3 Stockage des données et des droits d'accès 11
1.4 Récupération des données 12

4.1 Protocole pour calculer une clé de re-chiffrement bidirectionnelle 52
4.2 Le schéma [LV08b] 54
4.3 Le schéma [LV08b] multiple de cardinal n allégé 59
4.4 Relations entre les différents types récurrents 61
4.5 Le schéma [CWYD10] 62

5.1 Le schéma [LV08b] modifié CCA sûr 78

7.1 Suppression de l'accès du dispositif $\mathtt{A}_1$ 100
7.2 Utilisation du serveur de re-chiffrement combiné 102
7.3 Le schéma [LV08b] combiné sans l'algorithme Déchiffrer 108
7.4 Stockage de données avec un serveur par utilisateur 110
7.5 Utilisation du serveur de re-chiffrement combiné généralisé 111
7.6 Re-chiffrement combiné généralisé sur le schéma [LV08b] multiple de cardinal n . . 113

8.1 Exemples de re-chiffrement conditionnel 118
8.2 Le re-chiffrement combiné et conditionnel 119
8.3 Exemple d'arbre labellisé pour Alice 121

Liste des tableaux

4.1 Les différents schémas PRE possibles 65

Publications personnelles

[DJ11] Julien Devigne and Marc Joye. Binary Huff curves. In *CT-RSA'11*, volume 6558 of *Lecture Notes in Computer Science*, pages pages 340-355. Springer, 2011.

[CDL11] Sébastien Canard, Julien Devigne and Fabien Laguillaumie. Improving the security of an efficient unidirectional proxy re-encryption scheme. *Journal of Internet Services and Information Security (JISIS)*, 1(2/3) :140-160, 8 2011.

[CDDT12] Sébastien Canard, Nicolas Desmoulins, Julien Devigne and Jacques Traoré. On the implementation of a pairing-based cryptographic protocol in a constrained device. In *Pairing'12*, volume 7708 of *Lecture Notes in Computer Science*, pages 210-217. Springer, 2012.

[CCD+13] Sébastien Canard, Iwen Coisel, Julien Devigne, Cécilia Gallais, Thomas Peters and Olivier Sanders. Toward generic method for server-aided cryptography. In *ICICS'13*, volume 8233 of *Lecture Notes in Computer Science*,pages 373-392. Springer, 2013.

[CD13] Sébastien Canard and Julien Devigne. Combined proxy re-encryption. In *ICISC'13*. Springer, 2013. *To appear*.

[CD13a] Sébastien Canard and Julien Devigne. Generic RCCA-to-CCA transformation for proxy re-encryption scheme. In submission, 16 pages.

[CDS13] Sébastien Canard, Julien Devigne and Olivier Sanders. Delegating an elliptic-curve pairing can be both secure and efficient. In submission, 14 pages.

[DGL13] Julien Devigne, Eleonora Guerrini and Fabien Laguillaumie. Proxy re-encryption scheme supporting a selection of delegatees. In submission, 18 pages.

Brevets déposés

Procédé et processeur pour réaliser des opérations arithmétiques sur des courbes Huff binaires, avec Marc Joye, 2010.

Procédé de délégation de calculs cryptographiques au cloud, avec Sébastien Canard, 2011.

Secure distributed file storage using proxy re-encryption with management of lost devices, avec Sébastien Canard, Boris Chazalet et Ahmed Tansir, 2011.

Fine grain management of access rights for a secured cloud storage system, avec Sébastien Canard, 2013.

Procédé de délégation du calcul d'un couplage, avec Sébastien Canard et Olivier Sanders, 2013.

Bibliographie

[ABC+11] Michel Abdalla, James Birkett, Dario Catalano, Alexander W. Dent, John Malone-Lee, Gregory Neven, Jacob C. N. Schuldt, and Nigel P. Smart. Wildcarded identity-based encryption. *J. Cryptology*, 24(1) :42–82, 2011.

[ABH09] Giuseppe Ateniese, Karyn Benson, and Susan Hohenberger. Key-private proxy re-encryption. In *CT-RSA'09*, volume 5473 of *Lecture Notes in Computer Science*, pages 279–294. Springer, 2009.

[ACD+06] Michel Abdalla, Dario Catalano, Alexander W. Dent, John Malone-Lee, Gregory Neven, and Nigel P. Smart. Identity-based encryption gone wild. In *ICALP (2)'06*, volume 4052 of *Lecture Notes in Computer Science*, pages 300–311. Springer, 2006.

[Adl79] Leonard Adleman. A subexponential algorithm for the discrete logarithm problem with applications to cryptography. In *Proceedings of the 20th Annual Symposium on Foundations of Computer Science*, SFCS '79, pages 55–60, Washington, DC, USA, 1979. IEEE Computer Society.

[AFGH06] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Trans. Inf. Syst. Secur.'06*, 9(1) :1–30, 2006.

[AFI06] Nuttapong Attrapadung, Jun Furukawa, and Hideki Imai. Forward-secure and searchable broadcast encryption with short ciphertexts and private keys. In *ASIACRYPT'06*, volume 1403 of *Lecture Notes in Computer Science*, pages 161–177. Springer, 2006.

[Ama] Amazon cloud drive. `http://www.amazon.fr/`.

[And] Andromède. `http://fr.wikipedia.org/wiki/Andromede_(cloud)`.

[ANS10] ANSSI. Mécanismes cryptographiques : Règles et recommandations concernant le choix et le dimensionnement des mécanismes cryptographiques, version 1.20. 2010.

[BB04] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *EUROCRYPT'04*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 2004.

[BBDP01] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In *ASIACRYPT'01*, volume 2248 of *Lecture Notes in Computer Science*, pages 566–582. Springer, 2001.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *EUROCRYPT'05*, volume 3494 of *Lecture Notes in Computer Science*, pages 440–456. Springer, 2005.

[BBP04] Mihir Bellare, Alexandra Boldyreva, and Adriana Palacio. An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In *EUROCRYPT'04*, volume 3027 of *Lecture Notes in Computer Science*, pages 171–188. Springer, 2004.

[BBS98] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In *EUROCRYPT'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 127–144. Springer, 1998.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *CRYPTO'04*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, 2004.

[BDNS07] James Birkett, Alexander W. Dent, Gregory Neven, and Jacob C. N. Schuldt. Efficient chosen-ciphertext secure identity-based encryption with wildcards. In *ACISP'07*, volume 4586 of *Lecture Notes in Computer Science*, pages 274–292. Springer, 2007.

[BDPA11] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. The Keccak SHA-3 submission. 2011. `http://keccak.noekeon.org/`.

[BDPR98] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In *CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45. Springer, 1998.

[BDZ03] Feng Bao, Robert H. Deng, and Huafei Zhu. Variations of Diffie-Hellman problem. In *ICICS'03*, volume 2836 of *Lecture Notes in Computer Science*, pages 301–312. Springer, 2003.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In *CRYPTO'01*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In *TCC'05*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341. Springer, 2005.

[BGW05] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *CRYPTO'05*, volume 3621 of *Lecture Notes in Computer Science*, pages 258–275. Springer, 2005.

[BKR13] Mihir Bellare, Sriram Keelveedhi, and Thomas Ristenpart. Message-locked encryption and secure deduplication. In *EUROCRYPT'13*, volume 7881 of *Lecture Notes in Computer Science*, pages 296–312. Springer, 2013.

[BL07] Philippe Barbé and Michel Ledoux. *Probabilité*. EDP SCIENCES, 2007.

[BMW05] Xavier Boyen, Qixiang Mei, and Brent Waters. Direct chosen ciphertext security from identity-based techniques. In *ACM Conference on Computer and Communications Security'05*, pages 320–329. ACM, 2005.

[BN08] Mihir Bellare and Chanathip Namprempre. Authenticated encryption : Relations among notions and analysis of the generic composition paradigm. *J. Cryptology*, 21(4) :469–491, 2008.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical : A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73. ACM, 1993.

[BR09] Mihir Bellare and Thomas Ristenpart. Simulation without the artificial abort : Simplified proof and improved concrete security for waters' ibe scheme. In *EUROCRYPT'09*, volume 5479 of *Lecture Notes in Computer Science*, pages 407–424. Springer, 2009.

[CCD+13] Sébastien Canard, Iwen Coisel, Julien Devigne, Cécilia Gallais, Thomas Peters, and Olivier Sanders. Toward generic method for server-aided cryptography. In *ICICS'13*, volume 8233 of *Lecture Notes in Computer Science*, pages 373–392. Springer, 2013.

[CCdMP10] Sébastien Canard, Iwen Coisel, Giacomo de Meulenaer, and Olivier Pereira. Group signatures are suitable for constrained devices. In *ICISC'10*, volume 6829 of *Lecture Notes in Computer Science*, pages 133–150. Springer, 2010.

[CD13] Sébastien Canard and Julien Devigne. Combined proxy re-encryption. In *ICISC'13*. Springer, 2013. *To appear*.

[CDDT12] Sébastien Canard, Nicolas Desmoulins, Julien Devigne, and Jacques Traoré. On the implementation of a pairing-based cryptographic protocol in a constrained device. In *Pairing'12*, volume 7708 of *Lecture Notes in Computer Science*, pages 210–217. Springer, 2012.

[CDL11] Sébastien Canard, Julien Devigne, and Fabien Laguillaumie. Improving the security of an efficient unidirectional proxy re-encryption scheme. *Journal of Internet Services and Information Security (JISIS)*, 1(2/3) :140–160, 8 2011.

[CFA+05] Henri Cohen, Gerhard Frey, Roberto Avanzi, Christophe Doche, Tanja Lange, Kim Nguyen, and Frederik Vercauteren. *Handbook of Elliptic and Hyperelliptic Curve Cryptography*. Chapman & Hall/CRC Press, 1st edition, 2005.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *STOC'98*, pages 209–218. ACM, 1998.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. On the random-oracle methodology as applied to length-restricted signature schemes. In *TCC'04*, volume 2951 of *Lecture Notes in Computer Science*, pages 40–57. Springer, 2004.

[CH07] Ran Canetti and Susan Hohenberger. Chosen-ciphertext secure proxy re-encryption. In *ACM Conference on Computer and Communications Security'07*, pages 185–194. ACM, 2007.

[CHK04] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In *EUROCRYPT'04*, volume 3027 of *Lecture Notes in Computer Science*, pages 207–222. Springer, 2004.

[CKN03] Ran Canetti, Hugo Krawczyk, and Jesper Buus Nielsen. Relaxing chosen-ciphertext security. In *CRYPTO'03*, volume 2729 of *Lecture Notes in Computer Science*, pages 565–582. Springer, 2003.

[Clo] Cloudwatt. `https://www.cloudwatt.com/`.

[CN07] Ling Cheung and Calvin C. Newport. Provably secure ciphertext policy abe. *IACR Cryptology ePrint Archive*, 2007 :183, 2007.

[Coc01] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *IMA Int. Conf.'01*, volume 2260 of *Lecture Notes in Computer Science*, pages 360–363. Springer, 2001.

[Cor00] Jean-Sébastien Coron. On the exact security of full domain hash. In *CRYPTO'00*, volume 1880 of *Lecture Notes in Computer Science*, pages 229–235. Springer, 2000.

[CS02] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. `http://shoup.net/papers/cca2.pdf`, 2002.

[CS04] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM J. Comput.*, 33(1) :167–226, January 2004.

[CT07] Cheng-Kang Chu and Wen-Guey Tzeng. Identity-based proxy re-encryption without random oracles. In *ISC'07*, volume 4779 of *Lecture Notes in Computer Science*, pages 189–202. Springer, 2007.

[CWC+09] Cheng-Kang Chu, Jian Weng, Sherman S. M. Chow, Jianying Zhou, and Robert H. Deng. Conditional proxy broadcast re-encryption. In *ACISP'09*, volume 5594 of *Lecture Notes in Computer Science*, pages 327–342. Springer, 2009.

[CWYD10] Sherman S. M. Chow, Jian Weng, Yanjiang Yang, and Robert H. Deng. Efficient unidirectional proxy re-encryption. In *AFRICACRYPT'10*, volume 6055 of *Lecture Notes in Computer Science*, pages 316–332. Springer, 2010.

[CZL11] Min-Rong Chen, Xi Zhang, and Xia Li. Comments on Shao-Cao's unidirectional proxy re-encryption scheme from PKC 2009. *J. Inf. Sci. Eng.*, 27(3) :1153–1158, 2011.

[DAB+02] John R. Douceur, Atul Adya, William J. Bolosky, Dan Simon, and Marvin Theimer. Reclaiming space from duplicate files in a serverless distributed file system. In *ICDCS*, pages 617–624, 2002.

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *STOC'91*, pages 542–552. ACM, 1991.

[Del07] Cécile Delerablée. Identity-based broadcast encryption with constant size ciphertexts and private keys. In *ASIACRYPT'07*, volume 4833 of *Lecture Notes in Computer Science*, pages 200–215. Springer, 2007.

[Des87] Yvo Desmedt. Society and group oriented cryptography : A new concept. In *CRYPTO'87*, volume 293 of *Lecture Notes in Computer Science*, pages 120–127. Springer, 1987.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6) :644–654, 1976.

[DJ11] Julien Devigne and Marc Joye. Binary huff curves. In *CT-RSA'11*, volume 6558 of *Lecture Notes in Computer Science*, pages 340–355. Springer, 2011.

[DP06] Cécile Delerablée and David Pointcheval. Dynamic fully anonymous short group signatures. In *VIETCRYPT'06*, volume 4341 of *Lecture Notes in Computer Science*, pages 193–210. Springer, 2006.

[Dro] Dropbox. `https://www.dropbox.com/`.

[DWLC08] Robert H. Deng, Jian Weng, Shengli Liu, and Kefei Chen. Chosen-ciphertext secure proxy re-encryption without pairings. In *CANS'08*, volume 5339 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2008.

[Fac] Facebook. `https://www.facebook.com/`.

[Fis05] Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In *CRYPTO'05*, volume 3621 of *Lecture Notes in Computer Science*, pages 152–168. Springer, 2005.

[FN93] Amos Fiat and Moni Naor. Broadcast encryption. In *CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 480–491. Springer, 1993.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer, 1999.

[FP01] Pierre-Alain Fouque and David Pointcheval. Threshold cryptosystems secure against chosen-ciphertext attacks. In *ASIACRYPT'01*, volume 2248 of *Lecture Notes in Computer Science*, pages 351–368. Springer, 2001.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself : Practical solutions to identification and signature problems. In *CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[FSGW11] Liming Fang, Willy Susilo, Chunpeng Ge, and Jiandong Wang. Interactive conditional proxy re-encryption with fine grain policy. *Journal of Systems and Software*, 84(12) :2293–2302, 2011.

[FSW09] Liming Fang, Willy Susilo, and Jiandong Wang. Anonymous conditional proxy re-encryption without random oracle. In *ProvSec'09*, volume 5848 of *Lecture Notes in Computer Science*, pages 47–60. Springer, 2009.

[GA07] Matthew Green and Giuseppe Ateniese. Identity-based proxy re-encryption. In *ACNS'07*, volume 4521 of *Lecture Notes in Computer Science*, pages 288–306. Springer, 2007.

[Gam84] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer, 1984.

[Gen09] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. `crypto.stanford.edu/craig`.

[GGH13] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT'13*, volume 7881 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2013.

[GL05] Marc Girault and David Lefranc. Server-aided verification : Theory and practice. In *ASIACRYPT'05*, volume 3788 of *Lecture Notes in Computer Science*, pages 605–623. Springer, 2005.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *STOC'82*, pages 365–377. ACM, 1982.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2) :270–299, 1984.

[Gol04] Oded Goldreich. Foundations of cryptography : Basic applications. *Cambridge University Press*, 2, 2004.

[HKK+12] Goichiro Hanaoka, Yutaka Kawai, Noboru Kunihiro, Takahiro Matsuda, Jian Weng, Rui Zhang, and Yunlei Zhao. Generic construction of chosen ciphertext secure proxy re-encryption. In *CT-RSA'12*, volume 7178 of *Lecture Notes in Computer Science*, pages 349–364. Springer, 2012.

[Huf48] Gerald B. Huff. Diophantine problems in geometry and elliptic ternary forms. *Duke Math. J.*, 15 :443–453, 1948.

[Icl] iCloud. `https://www.icloud.com/`.

[ID03] Anca-Andreea Ivan and Yevgeniy Dodis. Proxy cryptography revisited. In *NDSS'03*. The Internet Society, 2003.

[Jou00] Antoine Joux. A one round protocol for tripartite Diffie-Hellman. In *ANTS'10*, volume 1838 of *Lecture Notes in Computer Science*, pages 385–394. Springer, 2000.

[Jou04] Antoine Joux. A one round protocol for tripartite Diffie-Hellman. *J. Cryptology*, 17(4) :263–276, 2004.

[JSJL10] Xiaoqi Jia, Jun Shao, Jiwu Jing, and Peng Liu. CCA-secure type-based proxy re-encryption with invisible proxy. In *CIT'10*, pages 1299–1305, 2010.

[JTV10] Marc Joye, Mehdi Tibouchi, and Damien Vergnaud. Huff's model for elliptic curves. In *ANTS'10*, volume 6197 of *Lecture Notes in Computer Science*, pages 234–250. Springer, 2010.

[KJP06] Sébastien Kunz-Jacques and David Pointcheval. About the security of MTI/C0 and MQV. In *SCN'06*, volume 4116 of *Lecture Notes in Computer Science*, pages 156–172. Springer, 2006.

[KL07] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. Chapman & Hall/CRC Press, 2007.

[LCLS09] Xiaohui Liang, Zhenfu Cao, Huang Lin, and Jun Shao. Attribute based proxy re-encryption with delegating capabilities. In *ASIACCS'09*, volume 2951, pages 276–286, 2009.

[LeMa] LeMonde.fr du 04 août 2011. Le marquage des photos sur Facebook déclenche une polémique. `http://www.lemonde.fr/actualite-medias/article/2011/08/04/le-marquage-des-photos-sur-facebook-declenche-une-polemique_1556051_3236.html`.

[LeMb] LeMonde.fr du 19 février 2009. Vie privée sur internet : la polémique Facebook. `http://www.lemonde.fr/technologies/article/2009/02/19/vie-privee-sur-internet-la-polemique-facebook_1157484_651865.html`.

[LHS+13] Kaitai Liang, Qiong Huang, Roman Schlegel, Duncan S. Wong, and Chunming Tang. A conditional proxy broadcast re-encryption scheme supporting timed-release. In *ISPEC'13*, volume 7863 of *Lecture Notes in Computer Science*, pages 132–146. Springer, 2013.

[LLT+12] Kaitai Liang, Zhen Liu, Xiao Tan, Duncan S. Wong, and Chunming Tang. A CCA-secure identity-based conditional proxy re-encryption without random oracles. In *ICISC'12*, volume 7839 of *Lecture Notes in Computer Science*, pages 231–246. Springer, 2012.

[LPS12] Rodolphe Lampe, Jacques Patarin, and Yannick Seurin. An asymptotically tight security analysis of the iterated Even-Mansour cipher. In *ASIACRYPT'12*, volume 7658 of *Lecture Notes in Computer Science*, pages 278–295. Springer, 2012.

[LSC11] Song Luo, Qingni Shen, and Zhong Chen. Fully secure unidirectional identity-based proxy re-encryption. In *ICISC'11*, volume 7259 of *Lecture Notes in Computer Science*, pages 109–126. Springer, 2011.

[LV08a] Benoît Libert and Damien Vergnaud. Tracing malicious proxies in proxy re-encryption. In *Pairing'08*, volume 5209 of *Lecture Notes in Computer Science*, pages 332–353. Springer, 2008.

[LV08b] Benoît Libert and Damien Vergnaud. Unidirectional chosen-ciphertext secure proxy re-encryption. In *Public Key Cryptography'08*, volume 4939 of *Lecture Notes in Computer Science*, pages 360–379. Springer, 2008.

[LW10] Allison B. Lewko and Brent Waters. New techniques for dual system encryption and fully secure hibe with short ciphertexts. In *TCC'10*, volume 5978 of *Lecture Notes in Computer Science*, pages 455–479. Springer, 2010.

[LZD+10] Junzuo Lai, Wen Tao Zhu, Robert H. Deng, Shengli Liu, and Weidong Kou. New constructions for identity-based unidirectional proxy re-encryption. *J. Comput. Sci. Technol.*, 25(4) :793–806, 2010.

[MA09a] Chunbo Ma and Jun Ao. Group-based proxy re-encryption scheme. In *ICIC (1)'09*, volume 5754 of *Lecture Notes in Computer Science*, pages 1025–1034. Springer, 2009.

[MA09b] Chunbo Ma and Jun Ao. Group-based proxy re-encryption scheme secure against chosen ciphertext attack. *I. J. Network Security*, 8(3) :266–270, 2009.

[Mat07] Toshihiko Matsuo. Proxy re-encryption systems for identity-based encryption. In *Pairing'07*, volume 4575 of *Lecture Notes in Computer Science*, pages 247–267. Springer, 2007.

[Meg] Mega. `https://mega.co.nz/`.

[MNT10] Toshihide Matsuda, Ryo Nishimaki, and Keisuke Tanaka. CCA proxy re-encryption without bilinear maps in the standard model. In *Public Key Cryptography'10*, volume 6056 of *Lecture Notes in Computer Science*, pages 261–278. Springer, 2010.

[MVO96] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1996.

[MW96] Ueli M. Maurer and Stefan Wolf. Diffie-Hellman oracles. In *CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 268–282. Springer, 1996.

[NIS95] NIST. Secure hash standard. *Federal Information Processing Standards, Publication FIPS-180-1*, 1995.

[NIS02] NIST. Secure hash standard. *Federal Information Processing Standards, Publication FIPS-180-2*, 2002.

[Num] Numergy. `https://www.numergy.com/`.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC'90*, pages 427–437. ACM, 1990.

[PA13] B. R. Purushothama and B. B. Amberker. On the security of group-based proxy re-encryption scheme. *IACR Cryptology ePrint Archive*, 2013 :456, 2013.

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *J. Cryptology*, 13(3) :361–396, 2000.

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In *STOC'08*, pages 187–196. ACM, 2008.

[Riv92] Ronald Linn Rivest. The MD5 message-digest algorithm. *Request for Comments (RFC 1321), Network Group*, 1992.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2) :120–126, 1978.

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *FOCS'99*, pages 543–553. IEEE Computer Society, 1999.

[SC09] Jun Shao and Zhenfu Cao. CCA-secure proxy re-encryption without pairings. In *Public Key Cryptography'09*, volume 5443 of *Lecture Notes in Computer Science*, pages 357–376. Springer, 2009.

[Sch89] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer, 1989.

[SCL10] Jun Shao, Zhenfu Cao, and Peng Liu. CCA-secure PRE scheme without random oracles. *IACR Cryptology ePrint Archive*, 2010 :112, 2010.

[SH12] Jin Sun and Yupu Hu. CCA-secure unidirectional proxy broadcast re-encryption in the standard model. *Journal of Computational Information Systems (JCIS)*, 8(14) :5909–5916, 7 2012.

[Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, 1984.

[SLCW11] Jun Shao, Peng Liu, Zhenfu Cao, and Guiyi Wei. Multi-use unidirectional proxy re-encryption. In *ICC'11*, pages 1–5. IEEE, 2011.

[SLZ12] Jun Shao, Peng Liu, and Yuan Zhou. Achieving key privacy without losing CCA security in proxy re-encryption. *Journal of Systems and Software*, 85(3) :655–665, 2012.

[Sub] Subversion. `http://subversion.apache.org/`.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT'05*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473. Springer, 2005.

[SWLX11] Jun Shao, Guiyi Wei, Yun Ling, and Mande Xie. Identity-based conditional proxy re-encryption. In *ICC'11*, pages 1–5. IEE, 2011.

[Tan08] Qiang Tang. Type-based proxy re-encryption and its construction. In *INDOCRYPT'08*, volume 5365 of *Lecture Notes in Computer Science*, pages 130–144. Springer, 2008.

[THJ08] Qiang Tang, Pieter H. Hartel, and Willem Jonker. Inter-domain identity-based proxy re-encryption. In *Inscrypt'08*, volume 5487 of *Lecture Notes in Computer Science*, pages 332–347. Springer, 2008.

[VSRR11] S.Sree Vivek, S. Sharmila Deva Selvi, V. Radhakishan, and C. Pandu Rangan. Conditional proxy re-encryption - a more efficient construction. In *CNSA*, pages 502–512, 2011.

[Wat05] Brent Waters. Efficient identity-based encryption without random oracles. In *EUROCRYPT'05*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer, 2005.

[Wat11] Brent Waters. Ciphertext-policy attribute-based encryption : An expressive, efficient, and provably secure realization. In *Public Key Cryptography'11*, volume 6571 of *Lecture Notes in Computer Science*, pages 53–70. Springer, 2011.

[WC09] Hongbing Wang and Zhenfu Cao. A fully secure unidirectional and multi-use proxy re-encryption scheme. In *ACM CCS*, Poster, Session, 2009.

[WCW10] Hongbing Wang, Zhenfu Cao, and Licheng Wang. Multi-use and unidirectional identity-based proxy re-encryption schemes. *Inf. Sci.*, 180(20) :4042–4059, 2010.

[WCY+10] Jian Weng, Min-Rong Chen, Yanjiang Yang, Robert H. Deng, Kefei Chen, and Feng Bao. CCA-secure unidirectional proxy re-encryption in the adaptive corruption model without random oracles. *SCIENCE CHINA Information Sciences*, 53(3) :593–606, 2010.

[WDD+09] Jian Weng, Robert H. Deng, Xuhua Ding, Cheng-Kang Chu, and Junzuo Lai. Conditional proxy re-encryption secure against chosen-ciphertext attack. In *ASIACCS'09*, pages 322–332. ACM, 2009.

[Wil95] Andrew Wiles. Modular elliptic curves and fermat's last theorem. 1995. `math.stanford.edu/~lekheng/flt/wiles.pdf`.

[WWMO10] Lihua Wang, Licheng Wang, Masahiro Mambo, and Eiji Okamoto. New identity-based proxy re-encryption schemes to prevent collusion attacks. In *Pairing'10*, volume 6487 of *Lecture Notes in Computer Science*, pages 327–346. Springer, 2010.

[WWZ10] Xu An Wang, Ping Wei, and Minqing Zhang. New CCA-secure proxy re-encryption scheme without random oracles. In *CIS'10*, pages 377–381, 2010.

[WY10] Xu An Wang and Xiaoyuan Yang. On the insecurity of an identity based proxy re-encryption scheme. *Fundam. Inform.*, 98(2-3) :277–281, 2010.

[WYT+09] Jian Weng, Yanjiang Yang, Qiang Tang, Robert H. Deng, and Feng Bao. Efficient conditional proxy re-encryption with chosen-ciphertext security. In *ISC'09*, volume 5735 of *Lecture Notes in Computer Science*, pages 151–166. Springer, 2009.

[WZH11] Jian Weng, Yunlei Zhao, and Goichiro Hanaoka. On the security of a bidirectional proxy re-encryption scheme from PKC 2010. In *Public Key Cryptography'11*, volume 6571 of *Lecture Notes in Computer Science*, pages 284–295. Springer, 2011.

[ZW12a] Jindan Zhang and Xu An Wang. On the security of a multi-use CCA-secure proxy re-encryption scheme. In *INCoS'12*, pages 571–576. IEEE, 2012.

[ZW12b] Jindan Zhang and Xu An Wang. Security analysis of a multi-use identity based CCA-secure proxy re-encryption scheme. In *INCoS'12*, pages 581–586. IEEE, 2012.

Résumé

La protection de la vie privée est un des enjeux majeurs de la société moderne dans laquelle Internet est omnipotent. Dans cette thèse, nous étudions des techniques permettant de réaliser un stockage dématérialisé qui préserve la confidentialité des informations. Nous nous intéressons ainsi à protéger les données stockées tout en permettant à leur propriétaire de les partager avec les personnes de son choix. Le serveur de re-chiffrement, une des primitives proposées par la cryptographie, est la solution que nous décidons de retenir.

Tout d'abord, nous donnons une définition d'un système de serveur de re-chiffrement qui regroupe tous les modèles classiques existants. Nous décrivons également les caractéristiques habituelles que peut présenter cette primitive ainsi que son modèle de sécurité. Ensuite, nous nous concentrons plus particulièrement sur certains schémas spécifiques afin d'en améliorer la sécurité. Nous présentons pour cela une méthode qui transforme un schéma sûr contre une attaque à chiffré choisi rejouable en un schéma sûr contre une attaque à chiffré choisi. Nous étudions aussi les schémas fondés sur le chiffrement Hash ElGamal et proposons d'y apporter des modifications afin qu'ils atteignent une meilleure sécurité. Pour terminer et dans le but d'obtenir le stockage le plus fonctionnel possible, nous proposons deux nouveaux modèles. Le premier, que nous appelons serveur de re-chiffrement combiné, permet d'obtenir une gestion dynamique des droits d'accès. Le second, que nous appelons serveur de re-chiffrement sélectif, permet d'obtenir une gestion des droits d'accès plus fine que celle offerte par le serveur de re-chiffrement conditionnel.

Mots-clés : cryptographie à clé publique, chiffrement (informatique), stockage en ligne (informatique), protection de l'information (informatique).

Abstract

Privacy is one of the main issues of our modern day society in which the Internet is omnipotent. In this thesis, we study some technics allowing to realise a privacy-preserving cloud storage. In this way, we focus to protect stored data while allowing their owner to share them with people of his choice. Proxy re-encryption, one of the primitives offered by cryptography, is the solution we decide to consider.

First, we give a definition of a proxy re-encryption system unifying all existing conventional models. We also describe usual characteristics that this primitive may present and we provide its security model. Then, we focus more precisely on some specific schemes in order to improve their security. In this meaning, we expose a method which turns a secure scheme against a replayable chosen ciphertext attack into a secure scheme against a chosen ciphertext attack. We study schemes based on the Hash ElGamal encryption too and propose some modifications in order to reach a better security. Finally and in order to obtain the most functional cloud storage, we propose two new models. The first one, that we call combined proxy re-encryption, offers dynamic right access. The second one, that we call selective proxy re-encryption, enables a more fine-grained access right control than the one offered by the conditional proxy re-encryption.

Keywords : public key cryptography, data encryption (computer science), data protection.