Contributions au proxy de re-chiffrement et à la délégation d'authentification

Anass Sbai

UNIVERSITÉ de Picardie Jules Verne

THÈSE de Doctorat

Spécialité "Informatique"

présentée à **l'École doctoral en Sciences Technologie et Santé**

de l'Université de Picardie Jules Verne

par

Anass SBAI

pour obtenir le grade de docteur de l'Université de Picardie Jules Verne

Contributions au proxy de re-chiffrement et à la délégation d'authentification

Soutenue le 05 juillet 2021 , après avis des rapporteurs, devant le jury d'examen :

Michaël Krajecki,	Professeur des Universités, DGA/URCA, Reims	Président
Mostafa AZIZI ,	Professeur des Universités, ESTO/UMF, Oujda	Rapporteur
Luiz Angelo Steffenel,	Maître de conférences HDR, LICIIS/URCA, Reims	Rapporteur
Laurent-Stéphane Didier,	Professeur des Universités, IMATH/USTV, Toulon	Examinateur
Sorina Ionica ,	Maître de conférences, MIS/UPJV, Amiens	Examinateur
Nesrine Kaaniche ,	Maître de conférences, SAMOVAR/Télécom SudParis	Examinateur
Gilles Dequen ,	Professeur des Universités, MIS/UPJV, Amiens	Directeur de thèse
Cyril DROCOURT ,	Maître de conférences, MIS/UPJV, Amiens	Co-directeur de thèse

Remerciement

Avant tout, je voudrais remercier "ALLAH" pour m'avoir donné assez de courage, de force, la bonne santé et le bien-être pour développer et présenter cet humble travail.

Je suis également très reconnaissant envers mes chers parents, mon frère et ma soeur qui m'ont apporté leur soutien tout au long de cette aventure.

Un travail de doctorat ne se construit pas seulement à partir des théories, des équations et des tests. Au cours des années de cette thèse, la collaboration et l'ambiance amicale avec diverses personnes ne m'ont pas seulement apporté des connaissances scientifiques, mais aussi des contributions personnelles. C'est pourquoi je tiens à remercier toutes les personnes qui m'ont permis d'arriver à la fin de ce cycle doctoral.

Je tiens à exprimer ma grande reconnaissance à Gilles Dequen pour l'accueil qu'il m'a réservé dans son laboratoire et pour m'avoir initié à la recherche dans le domaine de la cryptographie. Il est et il sera pour moi l'exemple de rigueur et de droiture dans l'exercice de notre profession.

Je remercie vivement Cyril Drocourt pour les conseils et le soutien exceptionnels qu'il m'a apporté. Ses encouragements, et son temps généreusement consacré à mes recherches ont été pour moi un atout majeur dans l'accomplissement de mes travaux. Qu'il trouve ici l'expression de ma profonde gratitude.

Je voudrais adresser mes vifs remerciement à Mme Sorina Ionica pour les différents échanges que nous avons eu sur mes travaux de recherche que ce soit lors des réunions d'équipe GOC ou en dehors de ces réunions et aussi pour avoir accepté de faire partie des membre de jury.

J'aimerais remercier les rapporteurs M. Azizi Mostafa et M. Steffenel Luiz Angelo pour avoir accepté d'évaluer cette thèse et pour leurs précieuses remarques et discussions.

Je tiens également à remercier Mme. Nesrine Kaaniche, M. Michaël Krajecki et M. Laurent-Stéphane Didier pour avoir accepté de faire partie des membres de jury et d'avoir pris le temps de lire et d'évaluer ce travail.

Mes remerciements vont aussi à toutes à tous les partenaires du projet VertPom en particulier M. Humberto Henao-Fernandez du laboratoire LTI et à Mme. Laeticia Kergozou de CIAC-IT.

Je remercie aussi l'ADEME pour le soutien financier, qui m'a donné l'opportunité de me concentrer pleinement sur la conduite d'une recherche de haute qualité.

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

La cryptologie a considérablement évoluée, notamment durant les trois dernières décennies. Initialement, la cryptologie était toujours associée à l'armée du fait de son importance en temps de guerre. Par exemple, pendant la seconde guerre mondiale, les cryptographes allemands avaient ainsi conçu une machine de chiffrement qui joua un rôle crucial. Connu sous le nom d'Enigma, cet ensemble de machines était un atout stratégique inestimable pour l'armée allemande, leur permettant de communiquer en toute sécurité. Du côté des alliés, Turing mit au point en 1940 les plans de la première machine permettant de casser la méthode de chiffrement des Allemands. La cryptanalyse d'Enigma a donné alors un grand avantage et a contribué au changement de balance des forces.

Depuis lors, la cryptologie a été confrontée à un statut restrictif au sein des agences gouvernementales. En France par exemple, l'usage civile des méthodes de chiffrement de bout en bout était strictement interdit jusqu'en 1996. Avec l'arrivée d'internet au début des années 70, les besoins du commerce en matière de confidentialité sur les canaux de communication électronique se sont multipliés. L'étude publique de la cryptologie pour satisfaire ces besoins fait que la cryptologie a finalement pu sortir de ses cloîtres et devenir une science. C'est ainsi que son champ d'application s'est élargi pour couvrir actuellement de nombreuses applications (cartes bancaires, commerce électronique, objets connectés...).

L'évolution de la cryptologie, engendrée en partie par la compétition économique et militaire internationale, a été possible grâce, d'une part, au progrès théorique concrétisé par l'élaboration d'outils mathématiques très puissants structurant nos notions de calcul, de preuve et de croyance, et d'autre part, au progrès technologique concrétisé par la mise au point des systèmes électroniques et informatiques avancés, permettant la mise en œuvre des nouveaux outils théoriques à des coûts très compétitifs. Par la suite, une multitude de scénarios de la vie quotidienne ont pu être réalisés d'un point de vue numérique, au travers de canaux de communication, au-delà de l'échelle des interactions humaines.

L'internet des objets (IdO ou en anglais Internet of Things IoT) ainsi que le "cloud-computing" sont devenus alors des technologies clés pour notre ère. Ils servent en effet à dématérialiser la réception, le stockage, le traitement et l'envoi de données. Ils permettent par ailleurs et surtout, de diminuer les coûts en éliminant les contraintes de transport, d'achat et d'installation d'infrastructures. L'IdO est défini par Perera et collab. [2015] comme étant un réseau dans lequel un nombre massif d'objets, de capteurs ou de dispositifs sont connectés pour fournir des services à valeur ajoutée. L'IdO permet ainsi d'améliorer le renseignement, de promouvoir l'interaction entre l'homme et l'environnement mais aussi d'améliorer la fiabilité, l'efficacité énergétique et la consommation des ressources. Nous estimons que, d'ici 2025, 75 milliards d'appareils seront connectés, générant des données volumineuses pour l'analyse et l'extraction de connaissances.

Dans le domaine de la santé par exemple, la pandémie actuelle de Covid-19 impose de nouvelles exigences et crée le besoin de solutions permettant de fournir des soins à distance, loin des établissements hospitaliers. L'IdO permet de suivre les patients à distance et d'améliorer les soins personnalisés et préventifs. Plusieurs objets existent sur le marché actuel tels que les bio-capteurs portables, permettant de suivre l'activité et le rythme cardiaque des patients, les systèmes automatisés d'administration d'insuline (AID) pour les personnes atteintes de diabète... Les données

collectées à travers ces dispositifs doivent être stockées et partagées avec les différents acteurs concernés à savoir le patient, ses médecins et infirmiers.

Toutefois, les données ne sont pas seulement volumineuses, elles comprennent également des données sensibles, et notamment des informations personnelles en fonction du type d'application et de la source des données. Par conséquent, ces données doivent être gérées avec soin pour éviter toute violation de la vie privée des utilisateurs. De son côté, le cloud doit être conçu pour stocker, traiter et analyser les données issues des objets connectés afin de découvrir de nouvelles connaissances ou de prendre des décisions cruciales. Comme il s'agit de technologies récemment étudiées, elles présentent cependant de nombreuses lacunes en matière de sécurité et surtout de respect de la vie privée.

L'un des plus grands défis à relever aujourd'hui est donc de pouvoir collecter, stocker, partager et traiter les données de manière sécurisée. Pour ce faire, des réponses doivent être apportées aux questions relatives à la vie privée, l'authenticité, la confiance et comment ces notions sont manipulées. Répondre à ces questions nous amène à étudier, utiliser et concevoir de nouveaux protocoles cryptographiques.

Dans le cadre du projet VertPom, l'objectif principal est de créer de la richesse environnementale et sociétale à travers une ville intelligente. Cette dernière repose sur son réseau intelligent connu aussi sous le nom de SmartGrid. Il s'agit d'un réseau de distribution utilisant les technologies de l'information, de la communication et les réseaux électriques afin d'optimiser la production et la livraison d'électricité. Différents capteurs ainsi que des compteurs intelligents entrent en jeu pour collecter les informations nécessaires concernant l'état du réseau électrique ainsi que le niveau de consommation et de production en temps réel. Au niveau du SmartGrid, les données collectées doivent être partagées entre les différents acteurs du réseau énergétique. Cependant, les données de consommation sont considérées comme des données sensibles. En effet, ces données peuvent révéler des informations pointues sur la vie quotidienne des clients. Elles peuvent aussi être une source d'attaque destructive visant des centrales de production d'énergie. Enfin, la confidentialité et l'intégrité des données doivent être assurées à toutes les étapes et en conformité avec les règles du RGPD et de la loi internet et liberté.

C'est dans ce contexte, et pour répondre aux besoins exprimés dans ce projet, que la présente thèse a été rédigée.

Dans le premier chapitre, il est à ce titre procédé à la présentation des différents outils techniques nécessaires à l'élaboration de nos outils de chiffrement. Des définitions sont fournies afin d'aider le lecteur à comprendre les chapitres suivants. Une première partie des définitions concerne les différents outils cryptographiques, tandis que la seconde partie est relative aux notions de sécurité des systèmes de chiffrement et de preuve de robustesse de ces derniers.

Dans le deuxième chapitre, sont présentées certaines méthodes permettant de répondre à la problématique évoquée à la fin du chapitre précédent. Parmi les différentes solutions existantes, nous présentons laquelle est la mieux adaptée à notre contexte, à savoir les Proxy de Rechiffrement (PRE). Une étude est menée sur les différents algorithmes disponibles dans la littérature. Un choix minutieux de l'algorithme est ensuite proposé, prenant en compte l'efficacité ainsi que la robustesse du système. Sont finalement présentés, l'implémentation, la technologie utilisée ainsi que les résultats obtenus.

Dans le troisième chapitre, nous nous concentrons d'avantage sur la sécurité des PRE. En effet, la notion de sécurité diffère selon le type du crypto-système étudié. La construction d'un PRE repose principalement sur un bloc ayant des propriétés spécifiques. Une section y est dédiée afin de présenter les différentes méthodes pour sa conception. La robustesse de certains algorithmes est ensuite analysée afin de montrer leur vulnérabilité face à certaines attaques. Enfin, une nouvelle construction est présentée, sa sécurité est prouvée sur la base du modèle de sécurité défini en début de chapitre.

Le quatrième chapitre concerne finalement les méthodes de délégation d'authentification.

Nous commençons par rappeler les principales méthodes existantes, lesquels possèdent certaines limites que nous essayons de contourner. Le premier protocole d'authentification à base de PRE est proposé et prouvé sécurisé en utilisant un modèle de logique à base de croyance, appelé logique BAN. Le modèle existe déjà mais nécessite certaines adaptations afin de pouvoir intégrer les différentes fonctions d'un PRE dans la preuve. La méthode de base ainsi que les modifications apportées sont présentées et détaillées.

Contexte et problématique

Le travail de cette thèse s'inscrit dans le cadre du projet VERTPOM pour « Véritable énERgie du Territoire POsitif et Modulaire » BORONAT [2017], qui consiste à créer une entité nommée Banque de l'énergie, qui permettra de suivre en temps réel la consommation et la production d'énergie afin d'en optimiser les flux.

Le projet part du constat que les demandes en termes d'énergie continuent de croître, mais que néanmoins, les anciennes sources de production (nucléaire, fossile...) sont moins appréciées et ont tendance à être abandonnées en raison de la pollution occasionnée. L'utilisation des sources d'énergie renouvelables est devenue privilégiée et primordiale mais présente de grands défis en terme d'optimisation.

Le marché de l'énergie actuel regroupe plusieurs acteurs. Comme illustré dans la figure 1, nous trouvons au début de la chaîne, les producteurs de l'énergie (par exemple : EDF). Il s'agit des entreprises exploitant des centrales nucléaires et thermiques, des gisements de gaz ou des producteurs d'énergies renouvelable. Puis arrivent les gestionnaires du réseau de transport, qui gèrent les grandes infrastructures de répartition d'énergie. En France, nous trouvons RTE qui s'occupe du réseau public de transport d'électricité haute tension tandis que le transport de gaz à haute tension est assuré par GRTgaz et Teréga. Ensuite, il y a les gestionnaires de réseaux de distribution qui assurent la construction, l'exploitation et l'entretien du réseau de distribution d'énergie (par exemple : Enedis, GRDF...). Enfin, nous avons les fournisseurs d'énergie qui ont pour rôle de commercialiser l'énergie qu'ils produisent ou qu'ils achètent. Ils gèrent ainsi les contrats avec les consommateurs finaux et s'occupent de la facturation (par exemple : ENGIE, Total direct énergie, Energies du Santerre...). Il existe aussi les ELD (entreprise locale de distribution), qui peuvent être en même temps fournisseur et distributeur d'énergie et desservent 5% du territoire français (par exemple : Gazelec). Sur les 95 % restant, c'est GRDF et Enedis qui assurent la distribution d'énergie.

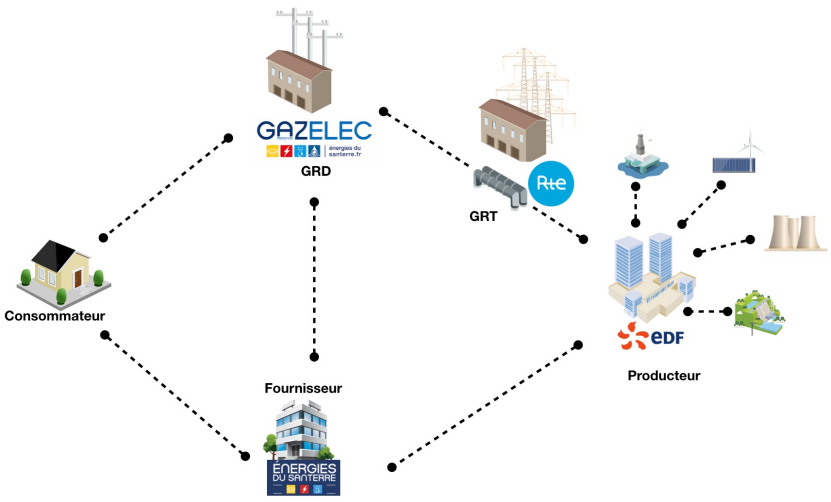


FIGURE 1 – Acteurs du marché de l'énergie.

Une gestion très précise de la production et de la consommation doit être adaptée, en utilisant

de nombreux capteurs, en commençant par le système de comptage intelligent. Cela implique l'installation de toute une infrastructure et de réseaux de communication incluant ces compteurs intelligents. Dans le cas du réseau géré par Enedis, il s'agit des compteurs nommés "Linky" qui sont déployés ces dernières années chez les particuliers. Cependant les ELD doivent mettre en place leurs propres technologies, et déployer leurs propres compteurs. Dans un premier temps, le projet VERTPOM se focalise sur l'étude des flux d'un ELD en particulier nommé Gazelec, qui déploie depuis quelques années des compteurs intelligents nommés Ibox.

La figure 2 présente de manière simplifiée comment les systèmes de comptage intelligents fonctionnent.

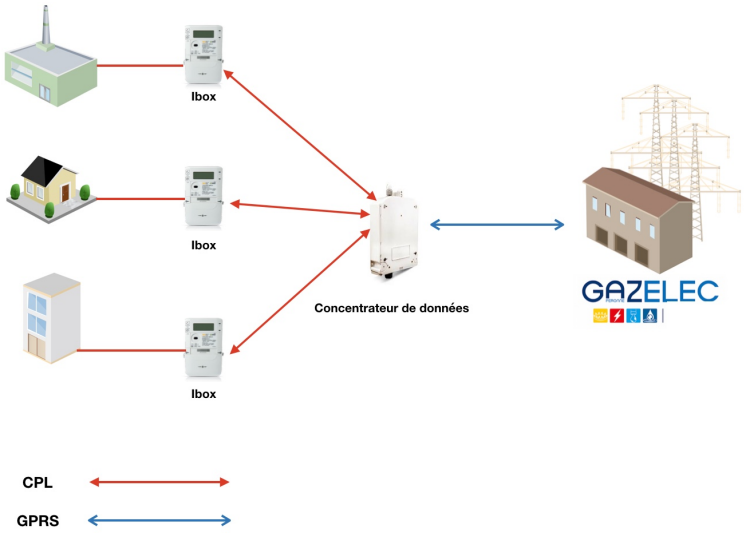


FIGURE 2 – Systèmes de comptage intelligent.

Dans cet exemple, nous avons deux entités principales qui sont : les consommateurs d'énergie, et le gestionnaire du réseaux de distribution. Nous avons aussi deux objets primordiaux qui sont les compteurs intelligents et les concentrateurs de données. Le premier est installé chez le client afin de mesurer la quantité d'énergie consommée puis la transmet au concentrateur de données via CPL (Courant Porteur de Ligne). Il s'agit d'une technologie de communication qui permet d'envoyer des données via les câbles électriques existants. Le concentrateur de données quant à lui est situé dans le transformateur du quartier et a pour but de centraliser les données issues des compteurs intelligents et de les transmettre à son tour au centre de supervision du gestionnaire du réseau de distribution. Cette transmission se fait principalement via un réseau GPRS.

Ceci donne naissance au concept de SmartGrid. En effet, il s'agit d'un réseau de transport et de distribution d'énergie accompagné et géré par des outils de surveillance et de télécommunication. Il assure ainsi un échange, en temps réel, d'énergie et d'information entre différents acteurs, depuis le site de production jusqu'aux utilisateurs commerciaux, industriels et résidentiels.

Néanmoins, plusieurs défis restent à relever, essentiellement au niveau de la sécurité dans la mesure où elle est relative à plusieurs objets de différentes conceptions, et dont le coût de consommation de ressources est faible. Il s'agit d'une préoccupation de sécurité nationale, ce qui requiert une grande attention que ce soit pour assurer la confidentialité des données de consommation, l'intégrité pour éviter la fraude ou, surtout, pour être à l'abri des attaques visant à déstabiliser les systèmes de production. En effet, une attaque telle que STUXNET MATROSOV et collab. [2010] a ciblée précisément les données liées à un type spécifique de logiciel nommé SCADA. Plus récemment, RONEN et collab. [2017] ont exploité une vulnérabilité dans le protocole ZLL, permettant d'insérer un ver qui se propage rapidement et de manière invisible sur le réseau. Il permet à l'attaquant d'allumer ou d'éteindre toutes les lumières de la ville. Une telle réaction en chaîne sur une grande ville pourrait faire trembler une centrale nucléaire.

Dans un autre contexte, BRINKHAUS et collab. [2011] ont montré que les compteurs intelligents sont capables de devenir des dispositifs de surveillance si un attaquant peut accéder aux courbes de charge stockées ou transmises par les compteurs intelligents. Ces données, peuvent être utilisées pour identifier tous les appareils connectés au réseau électrique tels que le réfrigérateur, micro-ondes, télévision etc. Ainsi, en réalisant des prédictions sur l'énergie consommée par les différents appareils, il est possible de chercher les corrélations entre les prédictions et les données réelles, ce qui peut même se traduire dans l'extrême à deviner quel film a été visionné.

Dans le cadre du projet VERTPOM illustré dans la figure 3, nous nous occupons de la gestion sécurisée de grandes quantités de données. Ce dernier, actuellement en cours de développement, consiste à créer la banque d'énergie (BE), qui est un outil dédié à la gestion énergétique du territoire. Elle vise à maintenir un équilibre optimisé entre l'énergie disponible de la production (énergie conventionnelle et renouvelable) et l'utilisation (consommation et pertes), en relation avec les moyens de stockage de l'énergie BORONAT [2017]. À cette fin, la banque de l'énergie utilise des algorithmes de prévision et des simulations de la production, de la consommation et des pertes d'énergie sur les différents systèmes de distribution. Ainsi, les réseaux d'énergie doivent être plus réactifs, plus flexibles, et donc favoriser les interactions entre les acteurs du marché. Ces objectifs sont en partie atteints par la collecte de données sur les réseaux grâce à des capteurs et des dispositifs télécommandés. Comme le montre la figure 1, les données nécessaires au bon fonctionnement de la banque de l'énergie sont renvoyées par les compteurs intelligents aux serveurs du gestionnaire de réseau via les concentrateurs de données. Certaines données gérées par les fournisseurs d'énergie (EP : Energy Provider) et d'autres informations provenant de la rue (capteurs de lumière etc...) sont également envoyées à la BE.

L'objectif de notre projet est de mettre en place les mécanismes nécessaires, afin de garantir un haut niveau sécurité pour l'ensemble des briques logicielles qui composent l'outil VERTPOM. Plusieurs données sont remontées depuis les fournisseurs d'énergie (FE) et les gestionnaires des réseaux de distribution (GRD), issues de capteurs de plus en plus nombreux. Ces dernières doivent être conservées pendant un laps de temps relativement long, imposé par le cadre juridique de conservation des données, mais surtout afin de pouvoir y appliquer des traitements à long terme. Enfin, la confidentialité des données doit être assurée à toutes les étapes et en conformité aux règles du RGPD et de la CNIL. En effet, afin d'être en conformité avec la loi informatique et liberté, le client reste le seul propriétaire de ses données de consommations, et seul un consentement avec ce dernier permet de relever sa courbe de charge.

Afin de résoudre ce problème, nous nous sommes intéressé à la problématique de partage des données chiffrées. Les données seront communiquées et stockées chiffrées, et seul le consentement d'un propriétaire de donnée peut donner lieu à un droit d'accès sur la donnée en question.

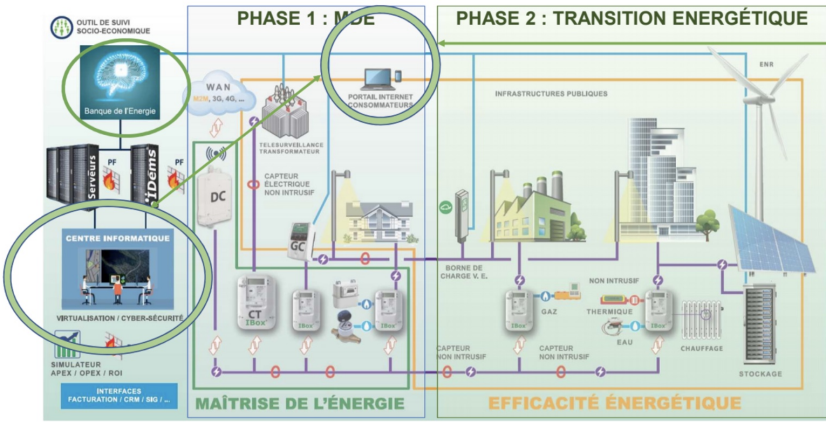


FIGURE 3 – Modèle architecturale de la banque de l'énergie.

« Projet soutenu dans le cadre du Programme d'investissements d'avenir (PIA) opéré par l'ADEME »

Chapitre 1

Introduction

Sommaire

1.1 Cryptographie

Nous vivons dans une ère où l'information est devenue accessible depuis n'importe quel point sur terre. La dématérialisation massive des informations rend son échange plus facile et moins coûteux, mais ceci ne vient pas sans risque. La cryptographie est devenue alors un outil omniprésent et indispensable dans notre quotidien. Elle nous fournit l'ensemble des méthodes permettant d'assurer la confidentialité, l'intégrité et l'authenticité des données.

La confidentialité empêche l'accès aux informations secrètes échangées entre un émetteur et un récepteur. L'intégrité permet de vérifier si le message n'a pas subi d'altération lors de son transfert. L'authenticité quant à elle garantie l'identité de l'émetteur du message.

Alors que la cryptographie consiste à étudier et concevoir des mécanismes pour protéger les informations, le terme cryptanalyse fait lui référence à l'étude et l'analyse des procédés de chiffrements afin de les affaiblir ou les casser dans un but d'amélioration continue. Ces deux composantes constituent alors la science du secret "cryptologie".

Cette science fut utilisée depuis l'antiquité. L'une des premières méthodes de chiffrement connu tient son nom à l'empereur romain César. Par précaution, il communiquait ses courriers chiffrés en décalant de manière cyclique l'alphabet avec un nombre de pas déterminé. Ce nombre de pas représente la clé secrète. Si par exemple la clé avait comme valeur 8, alors : "MONMESSAGE" devient "VXWVNBBJPN". C'est ce qu'on appelle un chiffrement par substitution monoalphabétique. Le décryptage du chiffrement de César est presque trivial puisqu'il n'y a que 26 clés possibles à vérifier (recherche exhaustive de clés).

Plus tard Al-Kindi a proposé une méthode permettant de casser ce type de système de manière plus élégante. Il s'agit de l'analyse de fréquence. En effet, chaque langue contient un certain nombre de lettre qui n'apparaissent pas toutes avec la même fréquence. En français par exemple, la fréquence d'apparition de la lettre E est la plus élevée et sa valeur varie entre 12% et 18% suivie par la lettre S puis A. Si nous procédons par analyse de fréquence pour décrypter le chiffré de l'exemple précédent "VXWVNBBJPN", sachant que la langue du texte en question est le français, les premières tentatives seront de vérifier si l'une des lettres qui se répètent le plus à savoir V et B correspondent à la lettre E, S ou A . Les clés à tester seront alors 17, 23, 3, 9, 21, 1. Arrivé à la 4ème tentative nous retrouvons le message en clair, ce qui réduit l'ensemble de recherche considérablement.

- Si E est substitué par V, le message en clair résultant sera : "EGFEWKKSYW"
- Si E est substitué par B, le message en clair résultant sera : "YAZYQEEMSQ "
- Si S est substitué par V, le message en clair résultant sera : "SUTSKYYGMK"
- Si S est substitué par B, le message en clair résultant sera : "MONMESSAGE"

Au fur et à mesure que la cryptanalyse évolue, les méthodes de chiffrement s'améliorent de manière à contrer les différentes attaques connues. On peut citer à titre d'exemple le chiffrement de Vigenère qui améliora le chiffrement de césar en proposant un système de chiffrement par substitution poly-alphabétique. Sa méthode résiste alors à l'analyse de fréquence. Trois siècles plus tard, Kasiski arrive à casser le système de chiffrement de Vigenère, en combinant l'analyse de fréquence avec une astuce permettant de deviner la taille de la clé.

Les bases de la cryptologie moderne commencent à se formaliser avec les travaux d'Auguste Kerckhoffs, dont l'une des règles principales est que la sécurité d'un système de chiffrement ne doit pas dépendre du secret de la méthode utilisée. Mais c'est aux articles de Claude Shannon (SHANNON [1948], SHANNON [1949]) que l'on doit le progrès de la cryptographie moderne qui présentent les bases des théories utilisées aujourd'hui.

1.2 Outils techniques

1.2.1 Rappels mathématiques

Groupe

Soit $\mathbb{G}$ un ensemble muni d'une loi de composition interne '$*$' . On dit que $(\mathbb{G},*)$ est un groupe si les trois conditions suivantes sont vérifiées :

- La loi '$*$' est associative :

$$\forall x,y,z\in\mathbb{G}, x*(y*z)=(x*y)*z$$

- La loi '$*$' admet un élément neutre :

$$\exists e\in\mathbb{G} \text{ tel que } \forall x\in\mathbb{G}, x*e=e*x=x$$

- Tout élément x de $\mathbb{G}$ admet un élément symétrique pour la loi '$*$' :

$$\forall x\in\mathbb{G}, \exists x'\in\mathbb{G}, \text{ tel que } x*x'=x'*x=e$$

Si la loi '$*$' est commutative alors $(\mathbb{G},*)$ est un groupe commutatif (appelé aussi groupe abélien) e.g :

$$\forall x,y\in\mathbb{G}, x*y=y*x$$

Si le cardinal de $\mathbb{G}$ est fini alors $(\mathbb{G},*)$ est un groupe fini. Dans ce cas le nombre d'éléments de $\mathbb{G}$ est appelé l'ordre de $(\mathbb{G},*)$.

Soit g un élément de $\mathbb{G}$. On dit que g est un générateur de $(\mathbb{G},*)$ si :

$$\forall y\in\mathbb{G}, \exists n\in\mathbb{N} \,/\, y=g^n$$

Dans ce cas $(\mathbb{G},*)$ est dit monogène. Si de plus $(\mathbb{G},*)$ est d'ordre fini, alors $(\mathbb{G},*)$ est un groupe cyclique.

Anneaux :

Soit $\mathbb{G}$ un ensemble muni de deux lois de composition interne '$+$' et '$*$'. On dit que $(\mathbb{G},+,*)$ est un anneau si :

- $(\mathbb{G},+)$ est un groupe commutatif d'élément neutre $0_{\mathbb{G}}$.

- La loi '$*$' admet un élément neutre différent $1_{\mathbb{G}}$

- La loi '$*$' est associative et distributive à gauche et à droite par rapport à '$+$' :

$$\forall x,y,z\in\mathbb{G}, \quad x*(y+z)=x*y+x*z \quad \text{et} \quad (x+y)*z=x*z+y*z$$

- Si la loi '$*$' est commutative alors $(\mathbb{G},+,*)$ est dit anneau commutatif ou abélien.

Corps :

- Un corps $\mathbb{K}$ est un anneau commutatif dans lequel tout élément non nul est inversible.

$$\forall x\in\mathbb{K}\setminus\{0\}, \exists x'\in\mathbb{K}, \text{ tel que } x*x'=x'*x=e$$

Courbe elliptique :

Une courbe elliptique E définie sur un corps $\mathbb{K}$ est représentée par une équation cubique à laquelle on rajoute toujours un point supplémentaire O = [0,1,0] à l'infini qui n'est autre que l'élément neutre de $\mathbb{K}$:

$$E : Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3 \text{ avec } a_1, \ldots, a_6 \in \mathbb{K}$$

Pour faciliter la notation, nous écrivons généralement l'équation sous la forme d'une équation de Weierstrass en utilisant des coordonnées non-homogènes $x = X/Z$ et $y = Y/Z$:

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6,$$

La représentation courte des équations de Weierstrass issue des simplifications sont les suivantes :

- Si $car(\mathbb{K}) \neq \{2,3\}$ alors, $E : y^2 = x^3 + a_4x + a_6$

- Si $car(\mathbb{K}) = 2$ alors, $E : y^2 + xy = x^3 + a_2x^2 + a_6$ est appelée courbe de Koblitz.

- Si $car(\mathbb{K}) \neq 2$, $A \in \mathbb{K}\backslash\{-2,2\}$ et $B \in \mathbb{K}\backslash\{0\}$ alors, $E : By^2 = x^3 + Ax^2 + x$ est appelée courbe de Montgomery.

Couplage :

Soit $\mathbb{G}_1, \mathbb{G}_2$ et $\mathbb{G}_T$ trois groupes cycliques d'ordre q. Un couplage $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ est une application bilinéaire non-dégénéré.

Bilinéaire : $\forall (a,b) \in \mathbb{Z}^2$, $\forall (g_1, g_2) \in \mathbb{G}_1 \times \mathbb{G}_2$, $e({g_1}^a, {g_2}^b) = e({g_1}^b, {g_2}^a) = e(g_1, g_2)^{ab}$

Non-dégénéré : $\forall (g_1, g_2) \neq (1,1)$, alors $e(g_1, g_2) \neq 1$

1.2.2 Complexité

La complexité joue un rôle majeur en cryptographie. Elle permet d'étudier la quantité de ressources nécessaires aux algorithmes pour résoudre des problèmes de calcul, ceci afin de pouvoir distinguer entre les problèmes faciles "qui peuvent être résolus par des algorithmes efficaces" et les problèmes difficiles qui sont insolubles. La ressource cruciale prise en compte est le temps d'exécution mais d'autres ressources, tel que le stockage, sont aussi étudiées dans certains cas. Nous utilisons comme modèle de calcul la machine de Turing : le temps d'exécution est mesuré en fonction de la taille de l'entrée. Une addition peut ainsi être résolue en un temps d'ordre n noté $O(n)$ où n est la taille de l'entrée. Tant que l'ordre de complexité d'un problème est de $O(n^k)$ avec k une constante, on dit qu'il appartient à la classe P. Il s'agit de la classe regroupant tous les problèmes ayant des algorithmes qui s'exécutent en un temps polynomial. Si nous prenons pour exemple le problème de factorisation d'entiers naturels, le plus simple algorithme pour résoudre ce problème procède par divisions successives. Sa complexité est d'ordre $2^{n/2}$ pour un entier de taille de $n - bit$ ce qui prend un temps d'exécution exponentiel au lieu d'un temps polynomial en fonction de n. Cependant, en ayant un certains nombres d'entiers nous pouvons vérifier efficacement si ces derniers sont des facteurs ou non de notre élément d'entrée. On dis alors que le problème de factorisation est dans la classe NP.

Il existe plusieurs algorithmes qui améliorent le temps d'exécution à $2^{n^{1/3}}$ tel que (HIARY [2016]). La plus part de ces algorithmes sont déterministes, c'est à dire que pour un problème donné, les résultats ainsi que les étapes d'exécution de l'algorithme sont toujours les mêmes pour la même donnée en entrée.

Cependant, nous avons besoin d'une plus grande variété d'algorithmes à notre disposition, à savoir des algorithmes qui incluent l'élément du hasard. En effet, rien n'empêche un attaquant d'essayer de deviner des informations de manière aléatoire. En cryptographie, les attaquants sont alors souvent modélisés sous forme d'un PPTA (Probabilistic Polynomial Time Algorithm). Il s'agit d'algorithmes qui utilisent pendant le calcul une source de valeurs aléatoires et retourne un résultat dans un temps polynomial en fonction de la taille d'entrée. L'efficacité d'un tel algorithme dépend moins de sa rapidité mais plutôt de la fréquence de succès à résoudre un problème. Un PPTA résout un problème de calcul donné si, pour chaque entrée, l'algorithme fournit la bonne réponse avec une probabilité non-négligeable. La probabilité d'erreur d'un tel algorithme peut être réduite en exécutant l'algorithme plusieurs fois.

Cependant, en cryptographie on cherche à construire des crypto-systèmes basé sur des problèmes calculatoires qu'un attaquant ne peut pas résoudre en un temps polynomial. C'est à dire que pour résoudre le problème donné, le PPTA représentant l'attaquant doit avoir une probabilité de succès égale à : $1/2+\epsilon$ où ϵ est une valeur négligeable.

De façon informelle, on définit ϵ comme étant une fonction négligeable si sa complexité est de l'ordre de $O(n^{-k})$ pour tout $k \geq 0$ où n dépend de la taille d'entrée de l'algorithme, ainsi ϵ décroît plus vite que l'inverse du polynôme. À travers cette définition on peut avoir les deux propriétés suivantes :

- Si ϵ_1 et ϵ_2 sont deux valeurs négligeables alors $\epsilon_1+\epsilon_2$ est négligeable.

- Si ϵ est négligeable et p est un polynôme alors $\epsilon'(n) = p(n).\epsilon(n)$ est négligeable.

1.2.3 Hypothèses cryptographiques

Problème du logarithme discret (DLP : Discrete Logarithm Problem) :

Soit $\mathbb{G}$ un groupe cyclique d'ordre n et g un générateur de $\mathbb{G}$. Etant donné un élément $y \in \mathbb{G}$, le problème du logarithme discret est de retrouver x tel que $g^x = y$.

La difficulté de calcul du logarithme discret dépend du groupe utilisé. Le meilleur algorithme générique pour résoudre le DLP dans un groupe d'ordre premier N est la méthode *rho* de Pollard qui a un temps d'exécution en $O(\sqrt{N})$.

Problème de calcul de Diffie-Hellman (CDHP : Computational Diffie Hellman Problem) :

Soit $\mathbb{G}$ un groupe cyclique d'ordre n et g un générateur de $\mathbb{G}$. Etant donné un élément $g^x, g^y \in \mathbb{G}$, le problème de Calcul de Diffie-Hellman (CDH) consiste à calculer g^{xy}. La seule façon connue (théorique et pratique) de résoudre le CDHP est de résoudre le DLP associé.

L'hypothèse CDH stipule que tout PPTA $\mathscr{A}$ a un avantage négligeable pour résoudre le problème CDH en $\mathbb{G}$.

Problème de décision de Diffie-Hellman (DDHP : Decisional Diffie Hellman Problem) :

Soit $\mathbb{G}$ un groupe cyclique d'ordre n et g un générateur de $\mathbb{G}$. Etant donnée trois éléments $a, b, c \in \mathbb{G}$ le problème de Décision de Diffie-Hellman (DDH) est de décider s'il existe des entiers x, y tel que $a = g^x, b = g^y$ et $c = g^{xy}$.

L'hypothèse DDH stipule que pour tout PPTA $\mathscr{A}$, les deux distributions suivantes ne peuvent être distinguées :

— $\mathbb{G}, g, g^x, g^y, g^{xy}$

— $\mathbb{G}, g, g^x, g^y, g^z$

où g est un générateur de $\mathbb{G}$ et x, y, z sont choisis uniformément et aléatoirement de $\{1, ..., |\mathbb{G}|\}$.

1.3 Cryptographie symétrique

1.3.1 Définition

La cryptographie symétrique repose sur l'existence d'une clé secrète partagée entre deux entités échangeant des informations. Cette clé sert en même temps à chiffrer et déchiffrer les messages comme le montre la figure 1.1. Plusieurs primitives se basent sur ce principe, comme par exemple le chiffrement par bloc (DES, AES, CLEFIA, Serpent ...) et le chiffrement par flux (RC4, Salsa, Trivium ...). Le chiffrement par flux est surtout utilisé lorsqu'il s'agit de chiffrement de donnée de taille aléatoire. Nous allons voir qu'on peut utiliser le chiffrement par blocs dans un mode qui nous permet d'avoir un chiffrement par flux. Ce qui a poussé les chercheurs à essayer de concevoir des chiffrement par flux qui soit plus rapides que le chiffrement par bloc mais se retrouve généralement avec des systèmes plus faibles (en raison des exigences de vitesse).

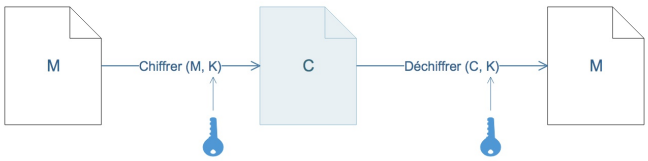


FIGURE 1.1 – Chiffrement à clé secrète

1.3.2 Chiffrement par bloc

DES est le premier chiffrement par bloc standardisé dont la structure est rendue accessible au public et qui est créé par une entreprise privée au lieu d'une institution gouvernementale. En effet, DES était le résultat d'une collaboration entre IBM et la NSA. Il a été dérivé du crypto-système Lucifer d'IBM SORKIN [1984] dont le sujet était controversé à l'époque, puisque les tailles des clés ont été réduites de 128-bits à 56-bits. Les deux crypto-systèmes reposent sur le principe de Shannon à savoir permutation et substitution (diffusion et confusion) via un réseau Feistel de 16 tours (voir la figure 1.2). Plus tard, il s'est avéré que DES est plus résistant que Lucifer BEN-AROYA et BIHAM [1996]. Néanmoins, l'évolution en terme de puissance de calcul a suscité le remplacement de DES vu que l'espace des clés devenait très réduit pour une attaque par force brute.

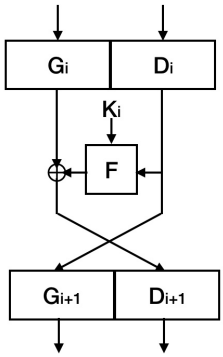


FIGURE 1.2 – Un tour du réseau de Feistel FEISTEL [1974].

En 1997, le NIST annonce le lancement de la compétition AES (Advanced Encryption Standard). Trois ans plus tard, l'algorithme Rijndael a été retenu parmi les cinq finalistes et devient l'AES. L'algorithme prend en entrée un bloc de texte de taille 128-bits et peut être utilisé avec des clés de chiffrement de taille 128, 192 ou 256-bits. En plus de sa robustesse contre les différentes attaques connues, il est très performant.

Cependant, un algorithme de chiffrement par bloc est fondamentalement déterministe : à partir d'un bloc de données de taille fixe et d'une clé secrète, on produit toujours le même chiffré de la même taille. Ceci peut donner des indices à un attaquant passif sur les données transmises. L'emploi d'un mode opératoire est alors recommandé et permet surtout de chiffrer des messages de tailles quelconques.

1.3.3 Modes opératoires

Il existe plusieurs modes d'opération. Pour une utilisation dans des protocoles cryptographiques, le mode d'opération doit rompre le caractère déterministe de l'algorithme de chiffrement en y introduisant une valeur aléatoire. L'utilisation d'un mode opératoire adéquat permet de transformer un chiffrement par bloc en un chiffrement par flux. Le mode CBC introduit un vecteur d'initialisation unique pour chaque message et fait en sorte que les blocs soient chaînés lors du chiffrement, ce qui permet d'avoir pour deux messages clairs identiques deux chiffrés différents. Parmi ses inconvénients, il faut noter la nécessité d'un remplissage, c'est pourquoi il ne peut être considéré pour une utilisation en tant que chiffrement par flux. De plus, le chiffrement et déchiffrement qui ne peuvent pas être parallélisé.

Il existe un autre mode opératoire qui permet d'assurer le non déterminisme du chiffrement par bloc : il s'agit du mode CTR (Counter mode). Ce mode ne nécessite aucune méthode de remplissage. Il est basé sur l'utilisation d'un compteur initialisé par une valeur aléatoire et incrémenté pour chaque nouveau bloc. Le principe est de chiffrer la valeur du compteur en utilisant la même clé. Le résultat est une valeur aléatoire différente à chaque nouveau bloc qui sera utilisée comme un masque jetable.

Malgré l'avantage de ces deux derniers modes, ils restent vulnérables à certaines attaques. En pratique, le mode opératoire CBC est plutôt utilisé pour assurer l'authenticité du message chiffré en calculant un MAC (Message Authenticating Code). Le code d'authentification correspond au dernier bloc chiffré en utilisant un chiffrement par bloc avec le mode opératoire CBC. Cette méthode s'appelle CBC-MAC. En combinant cette méthode avec un système de chiffrement par bloc utilisant le mode opératoire CTR, nous obtenons un des modes opératoires les plus utilisés : le CCM (Counter CBC Mac) illustré dans la figure 1.3. Le GCM (Galois Counter Mode) est un autre mode qui permet le chiffrement de message de taille non fixe et d'assurer le non déterminisme et l'authenticité du message chiffré.

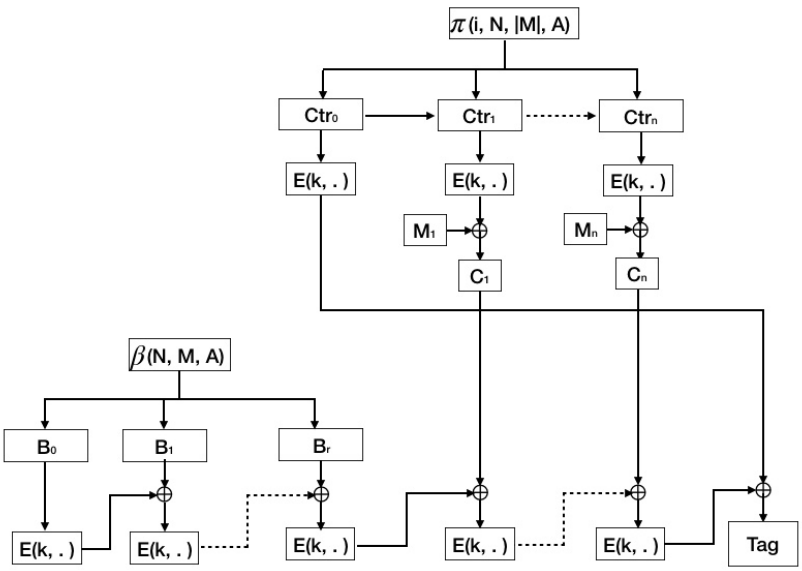


FIGURE 1.3 – Structure du mode opératoire CCM.

Lors de l'utilisation de mécanismes cryptographiques symétriques, que ce soit le chiffrement par bloc ou par flux, il est recommandé d'utiliser des clés de taille 128-bits au minimum. L'avantage des systèmes de chiffrement symétrique est dû à leur rapidité.

1.4 Cryptographie asymétrique

1.4.1 Définition

La cryptographie asymétrique, aussi appelée cryptographie à clé publique, repose sur l'utilisation d'une paire de clés, l'une est publique et l'autre privée. Ces deux clés sont généralement créées simultanément et sont liées par des fonctions mathématiques dites à sens unique avec impossibilité de déduire la clé privée à partir de la clé publique.

Une fonction f est dite à sens unique si :

- Étant donné x, il est facile de calculer $f(x)$.

- Étant donné y, il est difficile de trouver x tel que $f(x) = y$.

Exemples de fonction à sens unique : Le produit de deux grands nombres premiers (L'inverse de cette fonction, nécessite la factorisation qui est connu comme étant un problème difficile), l'exposant modulaire (L'inverse de cette fonction correspond au problème de calcul du logarithme discret) et la somme des sous-ensembles (l'inverse de cette fonction est connu comme étant un problème NP-complet).

La sécurité de tels systèmes repose alors sur les problèmes calculatoires sous-jacents. Voici quelques exemples de crypto-systèmes asymétriques :

— Diffie-Hellman DIFFIE et HELLMAN [1976] : logarithme discret

— RSA RIVEST et collab. [1978] : factorisation de grands entiers

— ElGamal ELGAMAL [1985] : logarithme discret

1.4.2 Protocole d'échange de clé

La cryptographie asymétrique répond à une problématique majeure de la cryptographie symétrique qui est la transmission sécurisé de la clé secrète. Diffie-Hellman (DH) DIFFIE et HELLMAN [1976] fut le premier algorithme de cryptographie asymétrique. Il s'agit d'un protocole d'échange de clé qui permet d'établir une clé symétrique entre deux entités. Ce dernier repose sur la difficulté du problème du logarithme discret. En effet, chaque entité détient une paire de clé publique/privée. Afin de procéder à un échange de clé, les deux entités doivent envoyer mutuellement leur clé publique, puis calculent chacun de son côté la clé secrète en utilisant sa propre clé privée. La figure 1.4 présente le déroulement du protocole DH, où g est un générateur d'un groupe $\mathbb{G}$ dont le DLP est difficile.

FIGURE 1.4 – Échange de clé de Diffie-Hellman

Bien qu'il soit sécurisé contre les écoutes passives du réseau, ce protocole est vulnérable face aux attaques de l'homme du milieu. La figure 1.5 illustre une attaque d'homme du milieu active dont le principe est d'usurper l'identité des deux entités et échanger de fausses clé avec ces derniers pour récupérer les informations échangés.

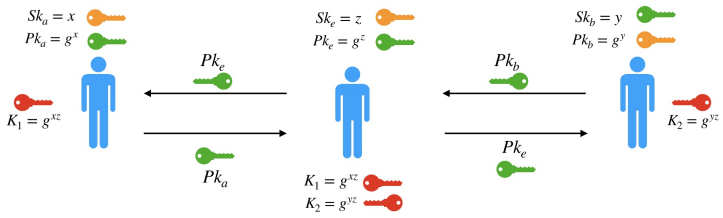


FIGURE 1.5 – Attaque d'homme du milieu

Afin d'éviter ce type d'attaque, l'une des solutions suggérées par les chercheurs est l'utilisation des signatures digitales. Cela permet d'assurer l'authenticité des clés et de vérifier l'identité de chaque partie. RSA est l'un des systèmes de signature et de chiffrement les plus connus et les plus largement répandus.

1.4.3 Chiffrement à clé publique :

La figure 1.6 illustre le fonctionnement d'un chiffrement à clé publique où la clé publique sert à chiffrer le message et la clé privée est utilisée pour déchiffrer.

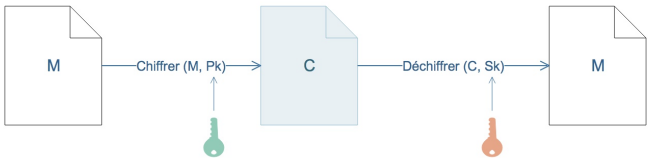


FIGURE 1.6 – Chiffrement à clé publique

Plusieurs méthodes de chiffrement asymétrique existent. Le premier chiffrement à clé publique connu a été proposé par MERKLE et HELLMAN [1978]. Il repose sur le problème de la somme des sous-ensembles. Cependant, il a été prouvé vulnérable par SHAMIR [1982]. Ce dernier, en collaboration avec Adleman et Rivest avait proposé une construction basée sur le problème de factorisation RIVEST et collab. [1978]. Leur crypto-système est connu sous le nom de RSA, et considéré comme étant l'un des crypto-systèmes à clé publique les plus répandus dans le monde.

Chiffrement RSA :

Le système repose sur la difficulté de factoriser un entier n en deux entiers premiers p et q. Cependant, la taille de n conditionne la vitesse du chiffrement / déchiffrement. En effet, le NIST BARKER et collab. [2014] recommande l'utilisation d'un ordre de groupe de taille 2048-bit au minimum.

■ Génération des clés :

- Choisir deux entiers premiers p et q (différents, grands, aléatoires).
- Calculer $n = p * q$
- Calculer $z = (p-1)(q-1)$

- Choisir un entier e premier avec z ($pgcd(e,z) = 1$)
- Choisir un entier d tel que : $e * d = 1(mod z)$

  La paire de clé publique et privée est : $pk = (e,n)$ et $sk = (d,n)$

■ Chiffrement :

- $C = M^e mod(n)$

■ Déchiffrement :

- $M = C^d mod(n)$.

Chiffrement ElGamal :

ElGamal ElGamal [1985] est aussi un algorithme de chiffrement à clé publique. Ce dernier repose sur le problème du logarithme discret et contrairement à RSA il n'a pas été breveté. Néanmoins, il n'est pas utilisé autant que RSA car il est un peu plus couteux en terme de calcul. Cependant, il commence à gagner en popularité avec l'arrivée des courbes elliptiques lui permettant d'augmenter ses performances en temps d'exécution et en taille de clé.

■ Génération des clés :

- Choisir un groupe cyclique $\mathbb{G}$ d'ordre q dans lequel le problème DDH est difficile et un générateur de groupe g.
- Choisir aléatoirement $x \xleftarrow{\$} \mathbb{Z}_q$
- Calculer $y = g^x$

La clé publique est alors $pk = (\mathbb{G}, q, g, y)$ et la clé privée est l'élément $sk = x$.

■ Chiffrement :

Pour chiffrer un message m avec la clé publique pk : $(\mathbb{G}, q, g, y)$ il faut :

- Choisir aléatoirement $r \xleftarrow{\$} \mathbb{Z}_q$
- Calculer $u = g^r$ et $v = m \times y^r$

Le message chiffré est composé des deux éléments calculé ci-dessus $C = (u, v)$

■ Déchiffrement :

Pour déchiffrer un message chiffré $C = (u, v)$, on doit disposer de la clé privée $sk = x$ et ainsi :

- Calculer $\frac{v}{u^x} = \frac{m \times y^r}{g^{r \times x}} = \frac{m \times \cancel{g^{r \times x}}}{\cancel{g^{r \times x}}} = m$

En plus du chiffrement, les algorithmes de cryptographie asymétrique sont utilisés pour résoudre plusieurs autres besoins comme la signature numérique.

1.4.4 Signature numérique :

La signature numérique a le même rôle qu'une signature physique, elle permet de prouver l'identité de l'entité qui a signé le message de façon à ce qu'elle soit vérifiable publiquement. En général, une entité possède une clé privée et une clé publique correspondante, liée à l'identité de l'entité . L'entité génère une chaîne de caractères appelée signature, qui dépend du message à signer et de sa clé privée. L'entité est alors reconnue comme étant l'émettrice d'un message si le message qu'elle a signé peut être vérifié par n'importe qui en utilisant la clé publique de l'entité, le message et la signature.

Un système de signature numérique se compose de trois algorithmes, à savoir l'algorithme de génération de clé, l'algorithme de signature et l'algorithme de vérification. La sécurité de la

signature numérique doit répondre à l'exigence suivante : aucun adversaire, sans la connaissance de la clé privée, ne peut générer un message et une signature qui seront validés par l'algorithme de vérification. Grâce à cette propriété, le système de signature numérique atteint la propriété de non-répudiation, c'est-à-dire qu'un signataire ne peut pas nier ultérieurement le fait d'avoir signé un message.

1.4.5 Chiffrement hybride :

L'inconvénient de la cryptographie à clé publique réside dans sa vitesse de traitement. Elle est souvent utilisée uniquement pour l'échange des clés. Une clé secrète qui est utilisée pour chiffrer un message au moyen d'un algorithme de chiffrement symétrique est chiffrée au moyen d'un algorithme de chiffrement à clé publique, puis jointe au message chiffré. Le destinataire déchiffre d'abord la clé secrète en utilisant sa propre clé de déchiffrement, puis il déchiffre le message en utilisant la clé secrète ainsi couverte. Cette méthode fait partie des systèmes de chiffrement hybrides. Elle comporte deux étapes, la première est le mécanisme d'encapsulation de clé et la deuxième est le mécanisme d'encapsulation de données FUJISAKI et OKAMOTO [1999].

$$\text{KEM}\|\text{DEM} = (\xi_{\text{PK}}^{asym}(\text{K})\|\xi_{\text{K}}^{sym}(\text{M}))$$

Un des avantages des systèmes de chiffrement hybrides est qu'un texte en clair peut être de n'importe quelle longueur et de n'importe quelle chaîne de bits. En prenant par exemple le système de chiffrement ElGamal, si le groupe utilisé est défini sur une courbe elliptique, alors le message en clair doit être un point de courbe elliptique et donc de taille et de forme bien précise, il est alors indispensable d'utiliser une fonction de mapping. L'utilisation des courbes elliptiques en cryptographie asymétrique permet d'avoir des clés relativement courtes, par exemple 256-bits pour un même niveau de sécurité équivalent à 2048-bits pour une clé RSA. Cela dit, les algorithmes utilisés restent beaucoup plus lents si on les compare au crypto-système symétrique.

1.5 Cryptographie fonctionnelle :

La cryptographie fonctionnelle prend en charge des clés secrètes restreintes qui permettent à leurs détenteurs de ne signer ou chiffrer que les messages satisfaisant une certaine fonction. Le chiffrement fonctionnel commence à se formaliser avec les travaux de BONEH et collab. [2011], mais certains systèmes étaient déjà existants et ce depuis 2005 avec les travaux de SAHAI et WATERS [2005]. Concernant les systèmes de signature fonctionnels, le principe est apparu pour la première fois dans BOYLE et collab. [2014].

1.5.1 Cryptographie à base d'identité :

La cryptographie à base d'identité s'apparente aux crypto-systèmes à clé publique. L'une des différences est que dans les systèmes de chiffrement et de signature à base d'identité, la clé publique n'est pas une valeur choisie aléatoirement, mais comporte l'identité de l'utilisateur. Le principe introduit par SHAMIR [1984] nécessite l'existence d'un centre de confiance qui a pour seule tâche de délivrer une carte à puce à chaque utilisateur qui rejoint pour la première fois le réseau. Cette dernière contient les informations nécessaires pour chiffrer et signer les messages. Le système est idéal comme le décrit l'auteur pour un groupe fermé. Si tous les utilisateurs ont reçu leurs cartes à puces, le centre de confiance peut fermer mais le réseau continue de fonctionner.

L'un de ses avantages est qu'il n'est plus nécessaire de lier une clé publique au nom de son propriétaire puisque l'identité est devenue la clé publique (adresse électronique, numéro de téléphone ...). Cela simplifie également la gestion des clés puisque les clés publiques sont mémorisables par l'homme. Seule la méthode de signature a été proposée par SHAMIR [1984] et plusieurs

solutions pratiques pour les signatures basées sur l'identité (IBS) ont été élaborées depuis. Cependant, la création d'un système de chiffrement à base d'identité (IBE) est restée un problème ouvert jusqu'en 2001 où Boneh, Franklin et Cocks ont proposé le premier IBE BONEH et FRANKLIN [2001].

1.5.2 Cryptographie à base d'attribut :

Dans un système ABE, les identités sont remplacées par un ensemble d'attributs $\mathbb{A}$. Généralement, les schémas de chiffrement à base d'attributs sont classés en deux catégories :

CP-ABE (Ciphertext policy) : la clé privée d'un utilisateur est associée à un ensemble d'attributs et un texte chiffré spécifie une structure d'accès à un univers défini d'attributs dans le système. Un utilisateur sera en mesure de déchiffrer un message chiffré, si et seulement si ses attributs satisfont la structure d'accès du message chiffré correspondant.

KP-ABE (Key policy) : dans ce cas de figure, la structure d'accès est encodée dans la clé secrète de l'utilisateur. Le message chiffré est alors calculé en fonction d'un ensemble d'attributs. Ainsi, seuls les utilisateurs dont la structure d'accès liée à leur clé privée est satisfaite par l'ensemble d'attributs sont en mesure de déchiffrer le message chiffré.

Un ABE repose toujours sur une structure d'accès $\boldsymbol{ST}$ qui représente la politique d'accès. Elle est soit liée aux clés secrètes distribuées par une entité de confiance ou prises en entrée lors du calcul du message chiffré. En effet, il existe deux types de structures d'accès :

— Structure d'accès monotone : Si $\mathbb{A}$ est un ensemble d'attributs qui satisfait une structure d'accès $\boldsymbol{ST}$, $\forall \mathbb{A}'$ tel que $\mathbb{A} \subset \mathbb{A}'$ alors $\mathbb{A}'$ satisfait la structure $\boldsymbol{ST}$. Par exemple, soit $\boldsymbol{ST} = a_0 \cap a_1$, alors $\mathbb{A} = \{a_0, a_1\}$ et $\mathbb{A}' = \{a_0, a_1, a_2\}$ satisfait $\boldsymbol{ST}$.

— Structure d'accès non-monotone : $\exists \mathbb{A}'$ tel que $\mathbb{A} \subset \mathbb{A}'$ et $\mathbb{A}'$ ne satisfait pas $\boldsymbol{ST}$. Par exemple, soit $\boldsymbol{ST} = a_0 \cap \neg a_2$. En reprenant l'exemple précédent où $\mathbb{A} = \{a_0, a_1\}$ et $\mathbb{A}' = \{a_0, a_1, a_2\}$, seul $\mathbb{A}$ satisfait $\boldsymbol{ST}$.

Au-delà des ABE, ce schéma n'est qu'un type appartenant au concept général des systèmes de chiffrement dits fonctionnels. Il inclut les IBE, les ABE ainsi que de nombreux autres concepts tels que le chiffrement avec recherche de mots-clés.

1.6 Autres primitives cryptographiques

1.6.1 Fonction de hachage :

C'est une fonction cryptographique principalement utilisée pour vérifier l'intégrité et l'authenticité des données. Elle prend en entrée un message de taille arbitraire pour en ressortir une empreinte de taille fixe. On peut classer ces fonctions de hachage en deux catégories. Ceux utilisant des clés secrètes MAC (Message authentication code) comme CMAC, HMAC ... et ceux qui n'utilisent pas de clé MDC (Manipulation Detection Code) par exemple MD-5, SHA-3 ... Dans la littérature actuelle, les fonctions de hachage sont plutôt identifiées à cette deuxième catégorie.

Deux propriétés sont importantes pour la construction des MDC, soit H une fonction de hachage : $\mathrm{H} : \{0,1\}^* \rightarrow \{0,1\}^n$.

— Sens unique : H est dite à sens unique si : étant donné y un élément de l'ensemble des images de H , il est difficile de retrouver x tel que $\mathrm{H}(x) = \mathrm{Y}$. Cette propriété est aussi appelée résistance à la pré-image.

— Résistance aux collisions : H est dite résistante aux collisions si : il est difficile de trouver x et x' avec $x' \neq x$ tel que $\mathrm{H}(x) = \mathrm{H}(x')$;

Concernant la première classe des fonctions de hachage c'est-à-dire MAC, la fonction H est définie de $\{0,1\}^*, \{0,1\}^l \rightarrow \{0,1\}^n$. Elle prend alors en entrée deux éléments, un message de taille

arbitraire x et une clé de taille fixe k et le résultat de $H(x,k)$ est de taille fixe. Cette fonction doit satisfaire la condition suivante :

Étant donné H et un grand nombre de paires $\{x_i, H(x_i,k)\}$, il est difficile de retrouver k et/ou $H(x,k)$ pour tout $x \neq x_i$. Ceci implique que la fonction H est à sens unique et résistance aux collisions.

Il existe une troisième propriété utilisée pour l'évaluation de la sécurité des fonctions de hachage qui est :

— la résistance à la seconde pré-image : étant donné un message $x \in \{0,1\}^*$ et son empreinte $H(x) = y \in \{0,1\}^n$, il est difficile de trouver $x' \in \{0,1\}^*$ tel que $H(x) = H(x')$ avec $x' \neq x$. Cette propriété implique la résistance aux collisions.

Pour une utilisation au-delà de 2020, la taille minimale des empreintes générées par une fonction de hachage est de 256-bits. La meilleure attaque connue permettant de trouver des collisions doit nécessiter le calcul de $2^{n/2}$ empreintes, où n désigne la taille en bits des empreintes.

1.6.2 Preuve de connaissance à divulgation nulle :

Ce concept a été introduit par GOLDREICH et collab. [1986]. Il désigne un protocole entre deux entités 'prouveur' et 'vérificateur'. Ce type de protocole permet au 'prouveur' de créer une preuve sur une certaine déclaration et au 'vérificateur' de vérifier sa validité. Une déclaration est de la forme $x \in L$, où x est un mot et L une langue formelle. En exécutant un protocole de connaissance à divulgation nulle par un prouveur honnête, le vérificateur ne devrait rien apprendre au-delà de la validité de la déclaration. En effet, tout ce que le vérificateur "voit" lorsqu'il interagit avec l'auteur de la preuve au moyen du protocole de connaissance zéro peut être efficacement simulé par l'auteur de la preuve lui-même. Il est essentiel de noter que la condition de connaissance zéro doit être remplie même si le vérificateur s'écarte du protocole de manière arbitraire. Les premières constructions étaient des preuves interactives où le prouveur et le vérificateur interagissent à maintes reprises avant la fin du protocole. Il existe certaines méthodes permettant de transformer les preuves interactives à divulgation nulle en signatures tel que l'heuristique de FIAT et SHAMIR [1986]. Un exemple de preuve interactive de connaissance à divulgation nulle est le protocole SCHNORR [1989]. Il s'agit d'un protocole dit sigma 'Σ' puisqu'il nécessite trois interactions (engagement, challenge et la réponse) :

— Soit un groupe cyclique $\mathbb{G}$ d'ordre q dans lequel le problème CDH est difficile et un générateur de groupe g.

— La clé privée du prouveur est : $y = g^x$ où $x \xleftarrow{\$} \mathbb{Z}_q$. Le protocole permettant de prouver la connaissance de x sur l'entrée commune y se déroule comme suit :

— Le prouveur choisi aléatoirement $r \xleftarrow{\$} \mathbb{Z}_q$.

— Calcule $a = g^r$.

— Envoie a au vérificateur (engagement).

— Le vérificateur choisi aléatoirement un challenge $c \in \{0,1\}$ et l'envoie au prouveur (challenge).

— Le prouveur calcule $s = r + cx \ mod \ q$ et l'envoie au vérificateur (réponse).

— Le vérificateur vérifie que $g^s = ay^c$.

Une preuve interactive doit satisfaire deux conditions en plus de la propriété connaissance à divulgation nulle (zero-knowledge) : la consistance et la robustesse.

Consistance : l'exécution du protocole entre l'auteur de la preuve et le vérificateur devrait toujours aboutir à l'acceptation de la preuve par le vérificateur, si $x \in L$.

Robustesse : l'exécution du protocole entre le prouveur et le vérificateur devrait toujours entraîner le rejet de la preuve par le vérificateur, si $x \notin L$, sachant que l'auteur de la preuve n'est pas tenu de suivre le protocole, c'est-à-dire qu'il peut se comporter de manière arbitraire.

En effet, ces deux propriétés, consistance et robustesse, protègent l'intérêt du vérificateur tandis que la propriété de zero-knowledge protège l'intérêt du prouveur. En effet, le prouveur est capable de convaincre le vérificateur de la validité d'une déclaration donnée, sans en divulguer la moindre information hormis la validité de la déclaration.

1.7 Définitions, objectifs et modèles de sécurité

1.7.1 Définitions de la notion de sécurité

L'une des premières définitions de sécurité des crypto-systèmes a été donnée par SHANNON [1949] connu sous le nom de sécurité parfaite. « Étant donné le chiffré d'un certain message, aucune information sur le texte en clair ne sera révélée même si l'adversaire a une puissance de calcul illimitée ». Par contre, il est très difficile voire impossible de créer de tels systèmes pratiques et de prouver leurs sécurités parfaites. GOLDWASSER et MICALI [1984] établissent alors une définition de sécurité parallèle à celle de Shannon en se basant sur la complexité de calcul appelée sécurité sémantique. « Étant donné le chiffré d'un certain message, aucune information sur le texte en clair ne sera révélée de manière réalisable ». Ainsi, un adversaire ne dispose que de ressources limitées et est modélisé sous forme d'un PPTA.

Il est alors dit qu'un crypto-système atteint la sécurité sémantique s'il n'existe aucun PPTA qui peut révéler une information partielle sur le message avec une probabilité non négligeable supérieure à celle de tous les autres PPTA qui n'ont accès qu'à la longueur du message (et non au cryptogramme).

La sécurité sémantique reste insuffisante car elle ne considère que les adversaires passifs. Or, plusieurs systèmes considérés comme sûrs du point de vue sémantique sont vulnérables lorsqu'ils sont face à un adversaire ayant le pouvoir de déchiffrer certains messages. BLEICHENBACHER [1998] a démontré qu'il était possible de mener en pratique ce genre d'attaque.

Il faut alors définir quel genre d'attaquant nous allons affronter. Plus précisément quels sont les moyens à disposition de ce dernier (types d'attaques) ainsi que ses objectifs. Plus le pouvoir de l'attaquant est élevée et ses objectifs moins difficile à atteindre, plus le niveau de sécurité augmente comme illustré dans la figure 1.7.

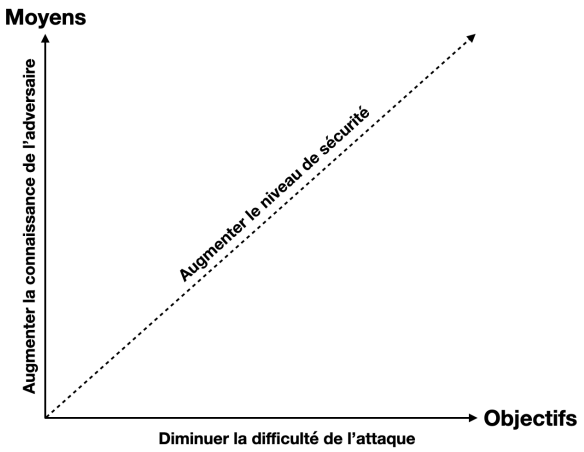


FIGURE 1.7 – Niveau de sécurité

1.7.2 Objectifs de sécurité

Les objectifs de sécurité permettent de décrire les buts de l'adversaire auxquels le crypto-système devrait résister. Ils sont généralement modélisés sous forme de jeux entre un adversaire et un challengeur. À savoir que le challengeur, dans le cas des crypto-systèmes asymétriques, génère une paire de clé (pk, sk) et envoie la clé publique pk à l'adversaire. C, C$'$ correspondent respectivement aux messages chiffrés de m, m'. C$_b$ est le message chiffré correspondant à m_0 ou m_1.

Voici les objectifs de sécurité les plus connus :

— Incassabilité (TB. Total Break) : il devrait être impossible pour un attaquant de retrouver la clé de déchiffrement à partir des clés publiques et/ou des messages chiffrés.

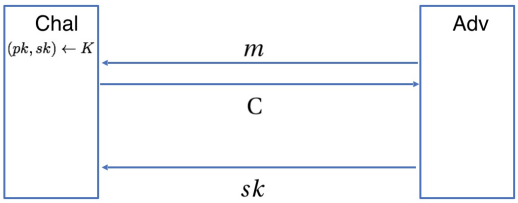


FIGURE 1.8 – Total break

— Sens-unique (SU ou OW. One-wayness) : ayant un message chiffré C avec une clé publique donnée Pk, il devrait être impossible d'inverser la fonction de chiffrement et récupérer le message en clair correspondant.

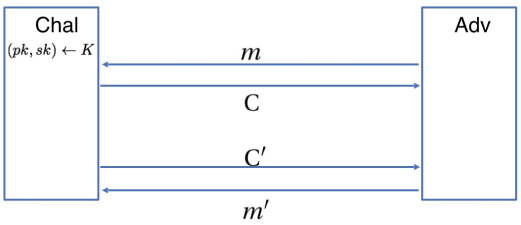


FIGURE 1.9 – Sens unique

— Indistingabilité (IND. indistinguishability) : étant donné deux messages de même longueur et le chiffré d'un des deux messages, l'attaquant doit deviner lequel des deux messages a été chiffré. Cette notion implique directement la sécurité sémantique du fait que si un adversaire est capable de différencier lequel des deux messages a été chiffré alors il a connaissance d'au moins un bit du message en clair.

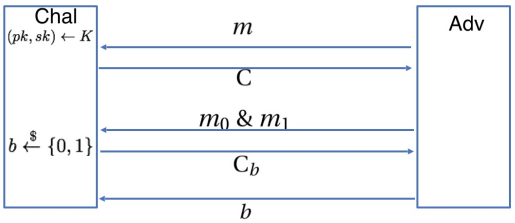


FIGURE 1.10 – Indistangabilité

— Non-malléabilité (NM. Non-Malleability) : il devrait être impossible de transformer un texte chiffré en un autre texte chiffré, de telle sorte que les textes en clair correspondants soient liés.

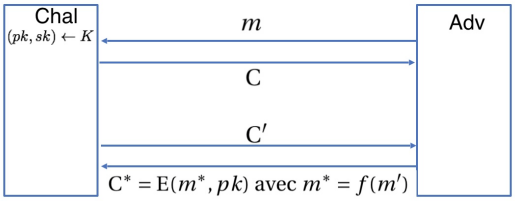


FIGURE 1.11 – Non malléabilité

1.7.3 Types d'attaques

Il est possible de définir les objectifs précités sur plusieurs types d'attaques, selon le cryptosystème que nous souhaitons étudier. Les types d'attaques décrivent les différents moyens qu'un adversaire pourrait avoir pour attaquer le schéma : COA / KPA / CPA / CCA-1 / CCA-2 .

— **Attaque à texte chiffré seulement** (COA. Cipher-text only attack) : dans ce cas de figure, nous supposons que l'attaquant est en mode passif. Il a accès à un ensemble de messages chiffrés mais pas les messages en clair correspondants. Il peut alors s'appuyer sur des méthodes telle que l'analyse de fréquence dans le but de retrouver des redondances lui permettant de deviner quelques caractères. DES par exemple, ainsi que tout autre système de chiffrement dont l'espace de clé est trop petit est vulnérable au COA en menant une attaque par force brute. Nous essayons toutes les clés possibles en ayant accès à rien d'autre qu'au texte chiffré.

— **Attaque à texte clair connu** (KPA. Known plaintext attack) :
toujours en mode passif, ici l'attaquant dispose d'un ensemble de paires de messages en clair et de leurs chiffrés qui les utilise pour obtenir la clé secrète. Lors de la deuxième guerre mondiale, Enigma a été cassée grâce à cette attaque. Les rapports journaliers envoyés par les allemands étaient chiffrés mais quelques mots pouvaient être devinés comme 'nothing to report' ou 'Heil Hitler', ce qui permit à Alan Turing de construire des ensembles de contraintes capables d'éliminer des familles entières de combinaisons de clés possibles.

— **Attaque à texte clair choisi** (CPA. Chosen plaintext attack) :
celle-ci est moins pratique et plus difficile à mener que les deux dernières attaques. Dans ce scénario, l'attaquant a accès à un oracle de chiffrement où il peut choisir n'importe quel message et récupérer le chiffré correspondant. Si le chiffrement est vulnérable à une attaque à texte clair connu, il est automatiquement vulnérable à une attaque à texte clair choisi, mais pas nécessairement l'inverse. La cryptanalyse différentielle est un exemple typique d'attaque CPA. La méthode de base consiste à observer la différence entre les deux messages chiffrés en fonction de la différence entre les messages en clair correspondants.

— **Attaque à texte chiffré choisi** (CCA-1. Chosen cipher-text attack) : c'est un scénario dans lequel l'attaquant a la possibilité de choisir un ensemble de messages chiffrés et peut les déchiffrer en ayant accès à un oracle de déchiffrement. L'attaque par chiffrement choisi est un scénario très important pour les systèmes de chiffrements à clé publique. En effet, les messages en clair ainsi que les chiffrés correspondants sont toujours à la disposition de l'attaquant en raison de la clé de chiffrement qui est publique. Le chiffrement basique du RSA par exemple est vulnérable à ce type d'attaque parce qu'il est déterministe. Un autre système vulnérable aux attaques CCA-1 est le mode d'opération CBC en utilisant le padding PKCS7.

— **Attaque adaptative à texte chiffré choisi** (CCA-2, Adaptative chosen cipher-text attack) : une attaque adaptative à texte chiffré choisi est un scénario d'attaques à texte chiffré choisi dans lequel l'attaquant a la possibilité de choisir les messages chiffrés envoyés à l'oracle de

déchiffrement en fonction des précédentes requêtes. L'attaque est censée être moins réaliste puisque nous donnons à l'attaquant plus de puissance par rapport à l'attaque de base par texte chiffré choisi. Cependant, l'attaque peut être très pratique dans le cadre d'un système de chiffrement à clé publique. Comme évoqué précédemment le RSA basique est vulnérable à une attaque à texte chiffré choisi. Il existe alors certaines implémentations de chiffrement non déterministe à base du RSA qui sont sécurisées contre les attaques à texte chiffré choisi. Néanmoins, BLEICHENBACHER [1998] a montré que la version RSA PKCS#1 v1.5 utilisée avec SSL à l'époque était vulnérable à une attaque adaptative à texte chiffré choisi.

1.7.4 Niveaux de sécurité

Les systèmes de chiffrement répondent généralement à de nombreux objectifs et résistent à de multiples types d'attaques. Les différentes notions de sécurité sont alors liées entre elles. Ce qui nous permet d'évaluer la robustesse d'un système de chiffrement est de pouvoir la comparer par rapport à d'autres schémas. La figure 1.8 illustre les implications entre les différentes notions de sécurité ainsi que leur classification selon le nombre de notions qu'ils impliquent. Une flèche à sens unique représente une implication tandis qu'une flèche à double sens représente une équivalence. Concernant leur classification, plus la couleur est claire moins le niveau est élevé. Nous ne considérons que les 3 derniers objectifs et types d'attaque c'est-à-dire {SU, IND, NM} et {CPA, CCA-1, CCA-2}, les autres notions sont élémentaires.

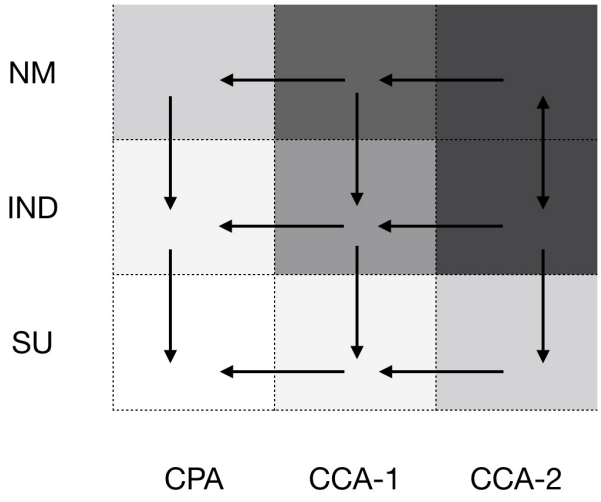
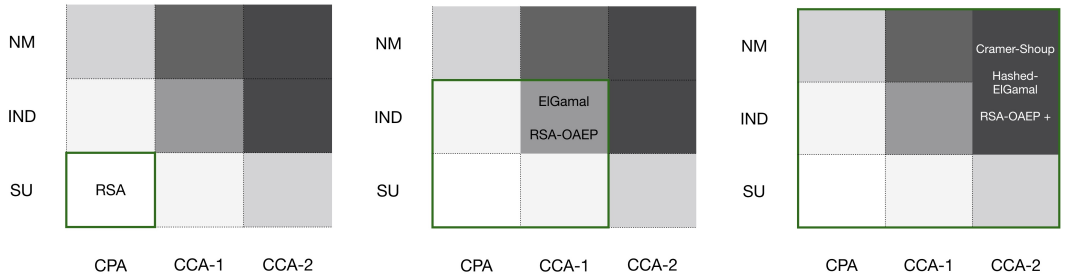


FIGURE 1.12 – Relations entre les différents niveaux de sécurité

Il est évident que CCA-2 $\Longrightarrow$ CCA-1 $\Longrightarrow$ CPA puisque l'attaquant gagne en puissance en recevant plus d'outils et d'informations par rapport aux cas qui les précédent. De même, NM $\Longrightarrow$ IND $\Longrightarrow$ SU. Dans le sens où la notion du SU n'exclut pas la possibilité de récupérer une partie du texte en clair, tandis que pour la notion IND, l'adversaire n'est pas capable de deviner un seul bit du texte en clair. Dans WATANABE et collab. [2003], les auteurs démontrent que la non-malléabilité implique l'indistingabilité et qu'en particulier NM-CCA-2 est équivalente à IND-CCA-2. Si le schéma est sécurisé contre les attaques NM-CCA-2 ou IND-CCA-2, il est alors implicitement sécurisé contre les autres attaques citées précédemment.

Pour une utilisation réelle d'un crypto-système asymétrique, le niveau requis est dans l'idéal le NM-CCA-2 ou à défaut le IND-CCA-2. La figure 1.13 présente la classification des différents systèmes de chiffrement asymétrique les plus courants selon leurs niveaux de sécurité. Le chiffrement basique du RSA par exemple est SU-CPA mais pas IND-CPA puisqu'il est déterministe. Tandis qu'en combinant RSA avec le système de padding OAEP, nous introduisons une couche de randomisation qui permet d'obtenir le niveau IND-CCA-1. ElGamal à son tour est IND-CCA-1 et ainsi SU-CCA-1, IND-CPA et SU-CPA.

En pratique, les méthodes de chiffrement asymétrique qui doivent être utilisées sont les schémas prouvés sécurisés contre les attaques IND-CCA-2 ou NM-CCA-2. Dans SCHNORR et JAKOBSSON [2000], les auteurs ont proposé le schéma "Hashed ElGamal" qui se base sur la construction du système d'ElGamal en combinant ce dernier avec le système de signature de Shnorr SCHNORR [1989]. Ceci a permis aux auteurs d'arriver à une construction IND-CCA-2.



(a) SU-CPA (b) IND-CCA-1 (c) NM-CCA-2

FIGURE 1.13 – Exemples des crypto-systèmes et les niveaux de sécurité qu'ils atteignent.

1.7.5 Modèles de sécurité

Nous avons vu précédemment la définition de sécurité donnée par SHANNON [1949]. En effet, la sécurité parfaite est difficile à atteindre avec des crypto-systèmes efficaces. Le "one time pad" par exemple répond à cette définition mais reste inefficace (le message et la clé doivent être de la même taille; la clé de chiffrement doit être généré aléatoirement à chaque fois que nous voudrions chiffrer un message). Néanmoins, la sécurité d'un tel système reste prouvable. Pour ce faire, il faut se baser sur un modèle de preuve de sécurité dite inconditionnelle, où les adversaires sont très puissants et considérés comme étant illimités en capacité de calcul. Cependant, les différents crypto-systèmes existants et utilisables en pratique reposent sur deux types de modèles. Un modèle idéalisé où certaines primitives sont considérées dans la preuve comme étant parfaite, par exemple le modèle du chiffrement idéal et le modèle des oracles aléatoires. Le deuxième type est connu sous le nom du modèle standard.

— **Modèle standard :** la cryptographie moderne suit l'approche de la sécurité prouvable. Cela signifie que lorsque nous créons un système de chiffrement, nous devons pouvoir prouver que la construction atteint effectivement la propriété de sécurité demandée. Dans le modèle standard nous nous basons impérativement sur des hypothèses mathématiques bien définies. Une fois que nous avons énoncé les hypothèses concrètes, nous montrons que le fait de casser la sécurité implique que notre système enfreint une ou plusieurs de ces hypothèses. Il s'agit généralement d'une réduction montrant que tout adversaire qui brise la construction peut être utilisé pour briser efficacement l'hypothèse. Parmi les crypto-systèmes qui sont prouvés sécurisés contre les attaques IND-CCA-2 dans le modèle standard, CRAMER et SHOUP [1998] est considéré comme étant le plus efficace.

— **Modèle des oracles aléatoires :** ce modèle est introduit par BELLARE et ROGAWAY [1993]. Ici nous supposons l'existence d'une fonction parfaitement aléatoire à laquelle toutes les parties impliquées (honnêtes ou malveillantes) ont accès. L'oracle est censé recevoir une donnée en entrée et retourne une valeur uniforme et aléatoire si cette donnée n'a jamais été soumise à l'oracle et enregistre les valeurs d'entrée/sortie correspondantes. Si la donnée reçue a été déjà soumise, il renvoie la valeur de sortie correspondante. En réalité une telle fonction n'existe pas, les oracles aléatoires sont alors instanciés avec des fonctions de hachage en supposant heuristiquement qu'elle se comporte comme un oracle. Ce modèle est un outil qui permet de donner une preuve de sécurité à certains crypto-systèmes comme (Hashed ElGamal, RSA-OAEP...), en modélisant la fonction de hachage en oracle aléatoire de

manière à ce que nous puissions exprimer les efforts d'attaques par le nombre d'invocations de l'oracle.

L'utilisation des fonctions de hachage n'implique pas impérativement l'utilisation du modèle des oracles aléatoires. En effet, si le crypto-système étudié utilise une fonction de hachage et que la preuve de ce dernier repose sur des hypothèses telles que la résistance aux collisions, la résistance à la pré-image etc ... sans avoir besoin qu'elle se comporte comme une fonction aléatoire, la preuve est alors considérée comme étant dans le modèle standard. Ce qui est le cas pour l'algorithme de CRAMER et SHOUP [1998].

Chiffrement de Cramer-Shoup :

- Génération des clés :
  - Choisir un groupe cyclique $\mathbb{G}$ d'ordre q dans lequel le problème DDH est difficile, deux générateur de groupe g_1 g_2 et H_1 une fonction de hachage universelle à sens unique.
  - Choisir $Sk = (x_i : i \in \{1,2,...,5\})$ où $x_i \xleftarrow{\$} \mathbb{Z}_q$
  - Calculer $Pk = (q, g_1, g_2, c, d, h, H_1)$ tel que $c = g_1^{x_1} \times g_2^{x_2}$, $d = g_1^{x_3} \times g_2^{x_4}$,$h = g_1^{x_5}$

  (Sk, Pk) est la paire de clés privée/publique associée à l'utilisateur.
- Chiffrement (m, Pk) :
  - Choisir $r \xleftarrow{\$} \mathbb{Z}_q$
  - Calculer $u_1 = g_1^r$, $u_2 = g_2^r$
  - Calculer $e = h^r \times m$, $\alpha = H_1(u_1, u_2, e)$
  - Calculer $v = c^r \times d^{r\times\alpha}$
  - Retourner $C = (u_1, u_2, e, v)$
- Déchiffrement (Sk , C) :
  - Vérifier si $v = u_1^{(x_1+x_3\times\alpha)} \times u_2^{(x_2+x_4\times\alpha)}$
  - Calculer $m = \frac{e}{u_1^{x_5}}$

1.8 Preuve de sécurité

Maintenant que les différents objectifs, types et modèles d'attaques ont été définis, nous pouvons formaliser une certaine méthodologie permettant de prouver la sécurité du système. Il existe plusieurs méthodologies, les plus connues et utilisées sont les preuves à base de jeux et les preuves à base de simulations. Pour la première catégorie de preuves, chaque objectif-type d'attaque correspond à un jeu bien défini, comme le montre la figure 1.8. Dans certains cas, un seul jeu n'est pas suffisant pour reproduire toutes les attaques possibles qui peuvent être menées par un adversaire. Pour parvenir à une preuve complète, il faut alors définir plusieurs jeux afin de prendre en compte toutes les manipulations possibles de l'adversaire. L'avantage des preuves à base de jeux est que la preuve est plus intuitive et plus simple à écrire. Mais ce n'est pas le cas lorsqu'il s'agit de systèmes plus complexes où plusieurs jeux interviennent.

Pour les preuves à base de simulation, l'idée vient de GOLDREICH et collab. [1986], où la sécurité sémantique a été formalisée par le paradigme idéal-réel. Le point de départ est qu'un adversaire ne doit apprendre aucune information sur le message en clair en ayant à disposition le message chiffré correspondant. Ceci a été reformulé comme suit : il est possible d'apprendre les mêmes informations sur le message en clair, que ce soit avec ou sans la connaissance du message chiffré. Ce qui veut dire que le message chiffré n'apporte aucune information en plus sur le message en clair et donc retour au point de départ, mais avec une définition plus distincte.

Cette définition nous permet de simuler l'existence de deux mondes. Le monde réel qui communique avec l'adversaire en suivant les fonctions réelles du crypto-système et qui retourne des vrais messages chiffrés. Le second est un monde idéalisé où les fonctions retournent des informations qui sont indépendantes du message en clair, sous condition que les informations retournées possèdent la même distribution. Dans ce cas, l'adversaire devra créer un PPTA qui permet de distinguer entre les deux mondes. Si l'adversaire n'y arrive pas, alors la méthode de chiffrement en question se comporte presque de la même manière que dans le monde idéal où le message chiffré ne dépend pas du message en clair.

En général nous commençons une preuve de sécurité par la définition des objectifs de sécurité, des types d'attaques et du modèle lié au crypto-système en question, puis nous énonçons les hypothèses sur lesquelles reposera la sécurité de notre schéma.

1.8.1 Preuve de sécurité d'ElGamal IND-CPA

Voici un exemple de preuve CPA sur le chiffrement d'ElGamal. Pour prouver la sécurité sémantique de ce dernier, nous allons nous baser sur l'indistinguabilité. On sait grâce à GOLDWASSER et MICALI [1984] que le niveau IND-CPA est équivalent à la sécurité sémantique.

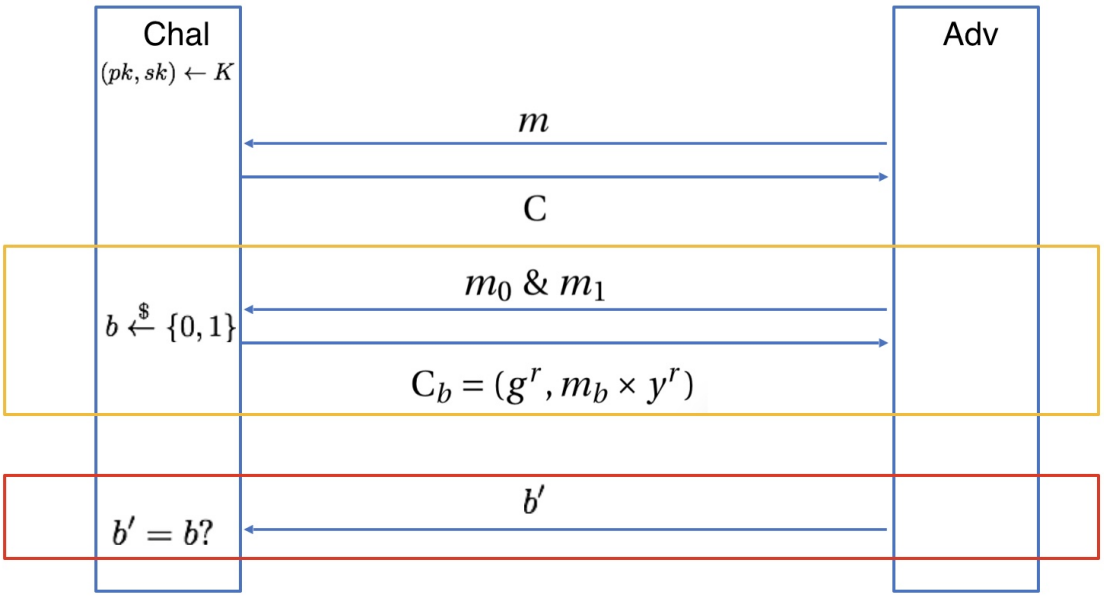


FIGURE 1.14 – Jeu IND-CPA

Preuve de sécurité à base de jeu :

Hypothèse *: La sécurité sémantique du système de chiffrement d'ElGamal est équivalente au problème DDH.*

Theorème *: S'il existe un adversaire PPTA "$\mathscr{A}$" capable de casser le chiffrement d'ElGamal, alors il peut être utilisé afin de construire un adversaire PPTA "$\mathscr{A}^*$" capable de résoudre le problème DDH.*

Nous allons suivre un raisonnement par absurde et supposer qu'un adversaire $\mathscr{A}$ est capable de gagner le jeu IND-CPA illustré dans la figure 1.4. Cela implique qu'étant donné $(q, g, y, u, v_b, m_0, m_1)$, $\mathscr{A}$ retourne avec succès la valeur du bit b.

Le message chiffré du défi retourné par le challengeur est $(g^r, y^r \times m_b)$ et la clé publique correspondante est g^x.

Notons qu'un adversaire peut calculer $c' = v_b \times m_0^{-1}$.

Si $b = 0$ alors $c' = y^r = g^{r \times x}$.

Connaissant $pk = g^x$ et $u = g^r$ l'adversaire $\mathscr{A}$ se retrouve face au problème DDH. En effet, pour savoir si $b = 0$ ou $b = 1$, il doit décider si $g^x, g^r, g^{r \times x}$ est un tuples DDH ou pas.

Puisque $\mathscr{A}$ est supposé gagner le jeu et donc connaitre la valeur du bit b, alors il peut construire un adversaire $\mathscr{A}^*$ permettant de répondre au problème DDH avec $Pr(\mathscr{A}^*(g^x, g^r, g^{r \times x}) = 1)$.

Nous pouvons donc conclure que si un attaquant n'est pas capable de résoudre le problème DDH, alors l'indistingabilité du chiffrement ElGamal est vérifiée. Ainsi le système est sémantiquement sécurisé.

1.8.2 Preuve de sécurité d'ElGamal IND-CCA-2

Voici un exemple d'attaque CCA-2 sur le chiffrement d'ElGamal. Pour prouver la sécurité de ce dernier, nous allons nous baser sur le modèle d'indistingabilité.

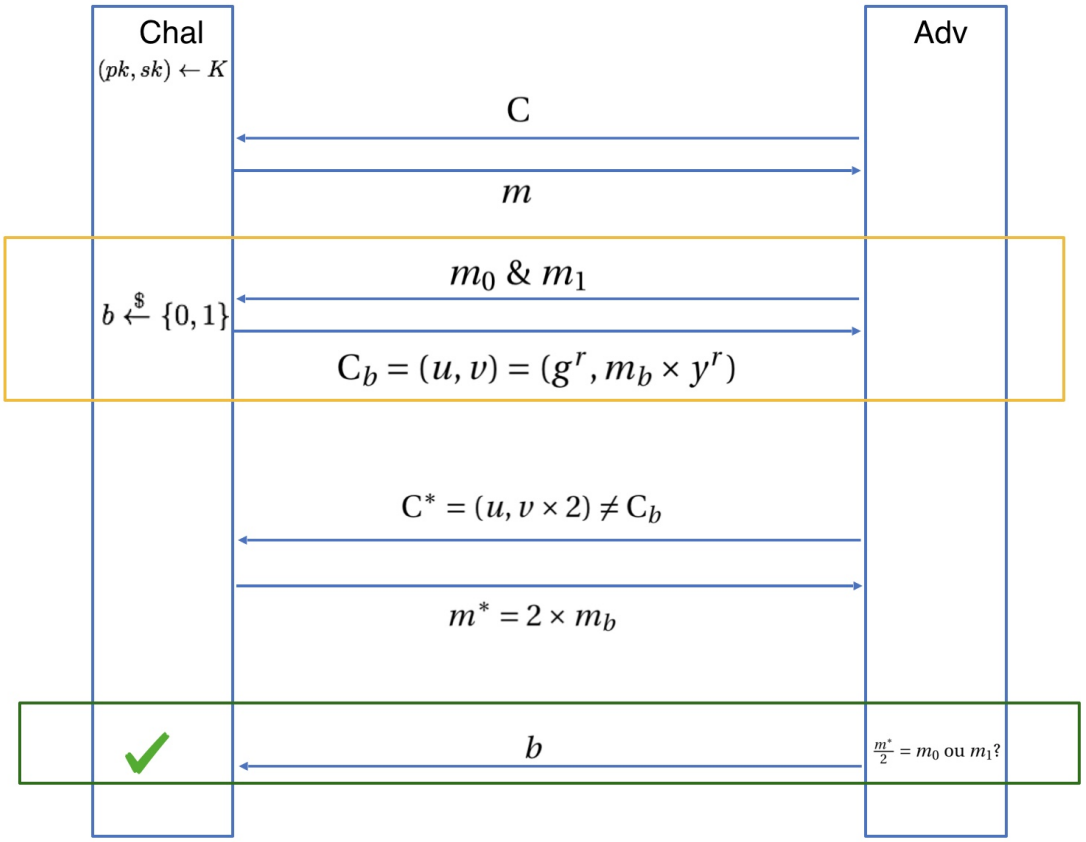


FIGURE 1.15 – Jeu IND-CCA-2

Les interactions illustrées à la figure 1.5 montrent qu'un adversaire $\mathscr{A}$ peut toujours gagner le jeu IND-CCA-2 lorsqu'il s'agit d'utiliser le chiffrement d'ElGamal. Ceci est dû à sa malléabilité puisque le système est homomorphe multiplicativement. En effet, étant donné un chiffré $C = (u, v)$ pour le message m_b, nous pouvons construire le chiffré $C' = (u, k \times v)$ qui sera valide pour le challengeur et sera déchiffré. Le déchiffrement correspondra au message $k \times m_b$. Il suffit alors de calculer $\frac{\not{k} \times m_b}{\not{k}}$ afin de retrouver le message qui a été chiffré et ainsi la valeur de b.

$C^* = E(m^*, pk)$ avec $m^* = f(m')$ Nous pouvons alors conclure que le chiffrement ElGamal n'atteint pas le niveau de sécurité IND-CCA-2.

Chapitre 2

Proxy de re-chiffrement au service du SmartGrid

Sommaire

2.1 Introduction

À ce jour, le stockage dans le cloud reste restrictif en termes de confidentialité. En effet, les fournisseurs les plus connus (par exemple GoogleDrive, Dropbox, iCloud, pCloud, OVH ...) n'assurent pas la confidentialité totale des données de leurs clients : soit les données sont chiffrées par une clé connue par le fournisseur de stockage cloud CSP (Cloud Service Provider), soit elles sont stockées en clair. Ces CSP sont censés être considérés comme étant des entités honnêtes mais curieuses. En réalité, ils sont considérés comme des entités entièrement fiables. Afin d'assurer la confidentialité vis à vis du CSP, les données doivent être stockées chiffrées et accessibles uniquement par les personnes autorisées, ce qui se traduit par une délégation de contrôle d'accès sur des données chiffrées par les propriétaires des données. La problématique de partage de données est par conséquent résolue. La question demeure cependant de la sécurisation de la délégation de contrôle d'accès sur les données chiffrées.

Une solution naïve au problème de délégation d'accès sécurisé consisterait à utiliser des techniques de chiffrement classiques (symétriques ou asymétriques) et à partager la clé de déchiffrement avec les entités désignées par le propriétaire des données.

Concernant le chiffrement symétrique, il ne peut être utilisé seul puisque la même clé est utilisée pour le chiffrement et pour le déchiffrement. Ceci implique que cette même clé est prépartagée entre le propriétaire de données et les différentes entités auxquelles nous voulons donner le droit d'accès.

Pour la cryptographie à clé publique, il existe plusieurs solutions :

— Tous les destinataires connaissent la clé privée liée à la clé publique de chiffrement, le problème est alors identique au chiffrement symétrique,

— Chaque donnée est chiffrée avec la clé publique du destinataire. Le problème est que les utilisateurs devant avoir le droit d'accès aux informations chiffrées ne sont pas nécessairement connus à l'avance.

Par conséquent, la seule possibilité est que ces données soient chiffrées sous une clé publique contrôlée par le propriétaire des données, ce qui implique que le propriétaire des données doit déchiffrer les données et les chiffrer ensuite avec la clé de l'utilisateur en question. Néanmoins, cette solution de déchiffrement et de chiffrement exige que le propriétaire des données soit disponible et en ligne pour re-chiffrer les données si nécessaire, ce qui interdit une gestion asynchrone du processus. Elle est donc extrêmement inefficace. Le problème devient de plus en plus complexe lorsque nous considérons de multiples données et diverses entités.

La meilleure solution serait de permettre à plusieurs personnes de déchiffrer les données sans avoir à passer par des étapes de déchiffrement / chiffrement ou sans partager sa clé privée ou secrète.

Le problème de partage de données a été résolu par diverses solutions dans la littérature, à commencer par le chiffrement de diffusion (broadcast encryption) conçu par FIAT et NAOR [1993]. Dans ce cas, chaque utilisateur peut accéder aux données indépendamment des autres. Cela nécessite au moment du chiffrement la connaissance de qui aura le privilège d'accéder à ces données. Une autre approche similaire, introduite par SAHAI et WATERS [2005], est le chiffrement basé sur les attributs (ABE). Inspiré par le travail de D Boneh sur les schémas de chiffrement basé sur l'identité (IBE), leur idée était de créer un nouveau type de système IBE BONEH et FRANKLIN [2001] pour combiner le chiffrement et le contrôle d'accès. Dans ce cas, les privilèges d'accès ne sont pas adressés à un ensemble d'utilisateurs mais uniquement aux utilisateurs avec un nombre spécifique d'attributs. Malheureusement, cela ne permet pas un partage sélectif. Comme alternative à ces solutions, nous choisissons d'utiliser le proxy de re-chiffrement (PRE). Grâce à ce cryptosystème, nous pourrons transférer les droits de déchiffrement à des entités spécifiques.

Dans le cadre du projet VertPom, nous avons proposé l'utilisation des PRE comme une solution de sécurité en tant que services. Elle permet aux gestionnaires d'énergie de stocker les don-

nées de consommations de manière chiffrées. Ceci préservera la vie privée des utilisateurs, tout en leur permettant de donner leur accord pour partager ces données à d'autres entités telles que la banque de l'énergie.

Nous considérons important de donner la définition et l'état de l'art du PRE afin que le lecteur puisse mieux comprendre notre choix du PRE ainsi que le reste du manuscrit. Nous présentons aussi dans ce chapitre, l'architecture choisie pour notre solution ainsi que l'implémentation du PRE. Nous comparons les performances en termes de temps et de stockages entre les différentes instanciations.

2.2 Proxy de re-chiffrement

2.2.1 Définition

Le proxy de re-chiffrement (PRE) est un crypto-système qui permet de transformer des messages chiffrés d'une entité A (délégataire) en messages déchiffrables par une entité B (délégué). Faisant partie des systèmes de chiffrement asymétrique, les messages sont chiffrés à l'aide d'une clé publique et sont déchiffrables par la clé privée correspondante. Cependant, le re-chiffrement nécessite l'utilisation d'une clé créée par le délégataire. Elle permet de transformer les messages chiffrés sans apprendre aucune information sur le message en clair (Figure 2.1).

Le PRE peut être considéré comme un type spécial de PKE [1], où les utilisateurs disposent également d'une paire de clés publique et privée. Par conséquent, toute personne ayant connaissance d'une clé publique est capable de produire des messages chiffrés destinés aux propriétaires de la clé privée correspondante; inversement, ces messages chiffrés ne peuvent être déchiffrés qu'à l'aide de la clé privée correspondante. La particularité est que les messages peuvent être re-chiffrés afin d'être déchiffrés par une clé privée différente de celle prévue à l'origine.

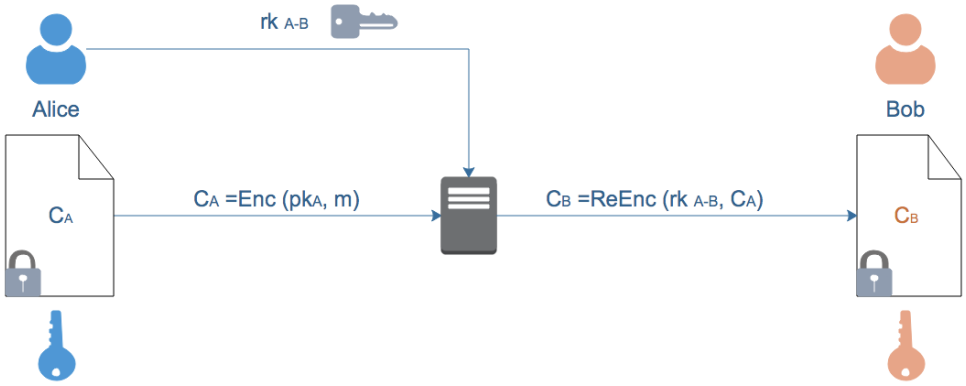


FIGURE 2.1 – Proxy de re-chiffrement

Nous retrouvons alors les trois acteurs principaux :

- Le délégataire (Alice) : noté "a" est l'entité propriétaire de la donnée qui délègue les droits de déchiffrement à une autre entité, à savoir le délégué, en créant une clé de re-chiffrement envoyée au proxy de re-chiffrement.
- Le délégué (Bob) : noté "b" est l'entité à qui nous avons accordé les droits de déchiffrement des messages chiffrés qui ne lui sont pas destinés à l'origine. Ces messages chiffrés doivent être re-chiffrés par la clé de re-chiffrement qui fait objet d'une autorisation d'accès pour le délégué.

1. PKE (Public Key Encryption) : chiffrement à clé publique. Cependant, il existe certaines propositions de PRE à base de chiffrement symétrique que nous discutons dans la section suivante.

- Le proxy : cette entité gère le processus de re-chiffrement des messages chiffrés, destinés à l'origine au délégataire, en des messages déchiffrables par le délégué. La clé de re-chiffrement utilisée par ce dernier lors du processus de transformation ne doit permettre aucune divulgation d'information au proxy.

En général, un schéma de PRE peut être défini comme un tuple ζ : {$Setup$, $KeyGen$, $ReKeyGen$, ξ^{asym}, $ReEnc$, Dec}. Où :

— $Setup(1^k) = params$: prend en entrée un paramètre de sécurité k et génère les paramètres du système qui définissent en général la longueur recommandée des messages et des clés.

— $KeyGen(params) = (Pk, Sk)$: est la fonction qui génère la paire de clé publique/privée respectivement Pk et Sk.

— $RekeyGen(Sk_a, Pk_b) = Rk_{a\to b}$: dans le cas du PRE unidirectionnel (définit dans 2.2.3), il faut entrer la clé privée de a et la clé publique de b pour générer la clé de re-chiffrement.

— $\xi^{asym}_{Pk_a}(M) = C_a$: est la fonction de chiffrement. Elle prend en entrée un message en clair M et une clé publique Pk .

— $ReEnc(C_a, Rk_{a\to b}) = C_b$: est la fonction de re-chiffrement. Elle prend en entrée un message chiffré ainsi qu'une clé de re-chiffrement.

— $Dec(C, Sk) = M$: est la fonction de déchiffrement. Elle prend en entrée un message chiffré ou re-chiffré et retourne le message en clair correspondant.

Dans certains cas, nous pouvons trouver deux autres fonctions utilisées pour le chiffrement et le déchiffrement, de telle sorte que le message chiffré créé par cette fonction de chiffrement (appelé aussi chiffré non transformable) ne peut pas être re-chiffré et que seul le propriétaire de la clé privée peut le déchiffrer.

2.2.2 Exemple BBS

Apparu pour la première fois en 1998, le premier PRE a été conçu par BBS (Blaze, Bleum et Strauss) BLAZE et collab. [1998] sur la base du système de chiffrement asymétrique ElGamal. Les auteurs montrent qu'il est possible d'incorporer une clé de substitution pour re-chiffrer un message déjà chiffré sans le compromettre. La proposition de BBS est une solution très élégante et elle permet de garder les mêmes algorithmes de chiffrement et de déchiffrement qu'ElGamal. Elle permet aussi de déchiffrer les messages chiffrés ou re-chiffrés avec la même fonction de déchiffrement. Ceci est dû au processus de re-chiffrement. En effet, ce dernier transforme les messages chiffrés d'Alice avec ElGamal en gardant la même distribution pour les messages re-chiffrés destinés à Bob. Ainsi, nous ne pouvons pas distinguer entre un message chiffré ou re-chiffré.

■ $Setup$:

— Choisir un groupe cyclique $\mathbb{G}$ d'ordre q dans lequel le problème DDH est difficile et un générateur de groupe g.

— Retourner $params = (\mathbb{G}, q, g)$

■ $KeyGen(params)$:

— Choisir aléatoirement $Sk \xleftarrow{\$} \mathbb{Z}_q$

— Calculer $Pk = g^{Sk}$

La paire de clé publique privée est respectivement Pk, Sk .

- ξ^{asym} :
  Pour chiffrer un message m avec la clé publique $\mathrm{P}k_a : (\mathbb{G}, q, g, y)$ il faut :
  — Choisir aléatoirement $r \xleftarrow{\$} \mathbb{Z}_q$
  — Calculer $u = \mathrm{P}k_a{}^r$ et $v = m \times g^r$
  Le message chiffré est composé des deux éléments calculés ci-dessus $\mathrm{C}_a = (u_a, v_a)$
- $\mathrm{R}ekey\mathrm{G}en$:
  La génération des clés de re-chiffrement nécessite l'utilisation des deux clés privées du délégataire et du délégué, i.e. $\mathrm{S}k_a$, $\mathrm{S}k_b$ et procède comme suit :
  — Calculer $rk_{a\to b} = \frac{\mathrm{S}k_b}{\mathrm{S}k_a}$
- $\mathrm{R}e\mathrm{E}nc$:
  Pour re-chiffrer un message C_a avec la clé de re-chiffrement rk il faut :
  — Calculer $u_b = u_a{}^{rk_{a\to b}}$ et $v_a = v_b$
  Le message re-chiffré est $\mathrm{C}_b = (u_b, v_b)$
- Déchiffrement :
  Pour déchiffrer un message chiffré ou re-chiffré $\mathrm{C} = (u, v)$, on doit disposer de la clé privée $\mathrm{S}k = x$ et ainsi :
  — Calculer $\frac{v}{u^{\frac{1}{\mathrm{S}k}}} = \frac{m \times g^r}{\mathrm{P}k^{r \times \frac{1}{\mathrm{S}k}}} = \frac{m \times \cancel{g^r}}{\cancel{g^{r \times x \times \frac{1}{x}}}} = m$

L'un des inconvénients de leur méthode est que le système est bidirectionnel : si un utilisateur A délègue les droits de déchiffrement à un utilisateur B, ce dernier délègue par conséquent les droits de déchiffrement à A, principalement à cause des clés de re-chiffrement qui nécessitent l'utilisation des deux clés secrètes du délégué et du délégataire. En calculant l'inverse multiplicative de la clé de re-chiffrement : $rk_{a\to b}{}^{-1} = \frac{\mathrm{S}k_b}{\mathrm{S}k_a}{}^{-1} = \frac{\mathrm{S}k_a}{\mathrm{S}k_b} = rk_{b\to a}$, nous retrouvons alors la clé de re-chiffrement qui permet de déléguer les droits de déchiffrement de Bob vers Alice. Depuis cet article, de nombreux travaux sur le sujet ont été réalisés. La section suivante présente les différentes propriétés qu'un PRE pourrait avoir en commençant par la directionnalité, qui était la première propriété a être définie formellement.

2.2.3 Propriétés d'un PRE

Ivan et Dodis [2003] formalisent la conception des proxy de re-chiffrement en classant ces systèmes en deux types : unidirectionnel et bidirectionnel. Ensuite, ils présentent des modèles génériques pour les fonctions de proxy unidirectionnel, bidirectionnel de chiffrement et de signature. En se basant sur ces modèles, ils proposent des fonctions de proxy pour différentes méthodes de chiffrement (RSA , ElGamal, IBE ...). Cependant, leur système exigeait qu'Alice et Bob partagent un secret. Bien qu'il soit clair qu'une telle phase de pré-partage peut-être accomplie de manière sûre, elle est indésirable. En effet, il se peut qu'Alice et Bob n'aient aucune relation préalable et que la communication bidirectionnelle soit impossible.

Ateniese et collab. [2006] donne une définition plus formelle du PRE et définit concrètement ces propriétés telles que :

— ***Directionnalité*** : elle représente la direction de la délégation et dépend principalement de la manière dont les clés de re-chiffrement ont été construites. Elle peut être soit uni-directionnelle, soit bi-directionnelle. Dans le premier cas, la délégation des droits de déchiffrement d'Alice à Bob ne permet pas à Alice de déchiffrer les messages chiffrés de Bob, ainsi les droits de déchiffrement sont délégués seulement du délégataire au délégué. Dans le cas contraire, si le PRE est bi-directionnel alors la délégation des droit de déchiffrement d'Alice à Bob per-

met à Alice de déchiffrer les messages destinés à Bob. Les droits de déchiffrement sont donc délégués dans les deux sens en même temps. Toutefois, cette situation n'est pas courante, mais peut être souhaitable pour les scénarios où la relation de confiance entre les délégués et les délégataires est symétrique.

— ***Interactivité*** : cette propriété dépend aussi de la manière dont les clés de re-chiffrement ont été construites. Dans l'exemple présenté précédemment, la clé de re-chiffrement a été créée à l'aide des deux clés privées du délégataire et du délégué. Dans ce cas, le PRE résultant est dit "interactif", dans le sens où l'interaction entre le délégataire et le délégué est obligatoire afin d'échanger la clé privée pour la création de la clé de re-chiffrement. Un PRE interactif est souvent indésirable, sauf dans le cas où l'acceptation de la délégation par le délégué est nécessaire. Si, par contre, la clé de re-chiffrement peut être générée par Alice sans interagir avec Bob, et donc en utilisant uniquement la clé publique de Bob, le PRE est considéré comme non-interactif.

— ***Transparence*** : cette propriété est aussi connue sous le nom d'invisibilité, dans le sens où le délégataire et le délégué ne savent pas si le proxy est actif, s'il a effectué une opération, s'il a apporté des modifications ou même s'il existe. En particulier, le délégué ne peut pas faire la distinction entre un message chiffré et un message re-chiffré.

— ***Key-optimal*** : certaines constructions ont pour résultat un PRE unidirectionnel au prix de l'augmentation de la taille de la clé secrète du délégué. Elle croît de manière linéaire avec le nombre de délégations que Bob accepte. Un PRE est dit "key-optimal" si la taille du secret, c'est-à-dire la clé secrète de Bob, doit rester inchangée, quel que soit le nombre de délégations qu'il accepte.

— ***Accès original*** : cette propriété permet au délégataire de déchiffrer tout message re-chiffré dont il était le propriétaire à l'origine.

— ***Résistant aux collusions*** : cette propriété traduit la sécurité de la clé secrète du délégataire contre les attaques de collusion entre le délégué et le proxy. Si le proxy et Bob sont complices, alors ils ne devraient pas obtenir la clé secrète d'Alice. Cette propriété est intéressante dans la mesure où le secret d'Alice ne sera jamais divulgué, mais surtout parce qu'Alice peut garder les droits de signature avec la même clé publique.

— ***Transitivité*** : le PRE est dit transitif s'il peut tout seul re-déléguer les droits de déchiffrement. Il est nécessaire qu'un PRE soit non-transitif afin de s'assurer que les droits de déchiffrement n'ont pas été délégués à d'autres entités, c'est-à-dire qu'à partir de $\mathrm{R}k_{a\to b}$ et $\mathrm{R}k_{b\to c}$ le proxy ne doit pas pouvoir calculer $\mathrm{R}k_{a\to c}$.

— ***Transférabilité*** : cette propriété combine la transitivité et la résistance aux collusions, dans le sens où le complot entre le proxy et le délégué ne permet pas la re-délégation des droits de déchiffrement. Le proxy et les délégués ne peuvent ainsi pas redéfinir les droits de déchiffrement. (En utilisant $\mathrm{R}k_{a\to b}$, $\mathrm{P}k_c$ et $\mathrm{S}k_b$ il n'est pas possible de calculer $\mathrm{R}k_{a\to c}$)

— ***Temporalité*** : certaines constructions prennent en considération la dimension temporelle, de telle sorte que la délégation des droits de déchiffrement n'est valable que pour une période de temps déterminée. Ainsi, le délégué ne peut déchiffrer les messages reçus du délégataire qu'à un certain moment ou pendant une certaine durée.

Ces propriétés s'avèrent très intéressantes, et certaines dépendent du scénario d'utilisation des PRE. Jusqu'à présent aucun PRE ne permet d'avoir toutes ces propriétés à la fois. Nous allons citer les constructions les plus importantes dans la section suivante et voir quelles sont leurs propriétés.

2.2.4 PRE dans la littérature

Une fois le concept formalisé, ATENIESE et collab. [2006] proposent alors les premières constructions de PRE qui répondent à des propriétés telles que la temporalité. Leur PRE permet de limiter la

validité des clés de re-chiffrement à une période de temps spécifique. Cependant, ils introduisent un tiers de confiance qui diffuse une valeur aléatoire associée à chaque période de temps. Afin d'éviter les coûts élevés de gestion des certificats pour le chiffrement traditionnel à clé publique, GREEN et ATENIESE [2007] proposent le premier système fonctionnel d'un PRE à base d'identité reposant sur la construction de BONEH et FRANKLIN [2001], le schéma en résultant est le premier PRE "multi-use". L'inconvénient majeur est la taille des messages chiffrés qui croît à chaque re-chiffrement. Plusieurs travaux se sont concentrés sur la construction de proxy de re-chiffrement à base d'identité et ses dérivées, comme les systèmes à base d'attribut (ABE). Ce dernier permet d'intégrer un système de contrôle d'accès avec le système de re-chiffrement.

La combinaison entre ABE et PRE, appelée ABPRE, résulte en un crypto-système qui permet de transformer un message chiffré C_1 associé à une structure d'accès ST_1 en un message chiffré C_2 associé à une deuxième politique d'accès ST_2.

Le chiffrement avec recherche de mots clés a été combiné à son tour avec la construction d'un PRE pour la première fois par SHAO et collab. [2010] où ils proposent un PRES (Proxy Re-Encryption with keyword Search) multi-use, transparent et bidirectionnel. Ce type de système s'avère très utile pour les systèmes de messagerie puisqu'il permet de vérifier l'existence d'un mot clé par un proxy en utilisant une porte dérobée avant de le re-chiffrer.

Dans le cas des PRE à base de chiffrement symétrique, certaines solutions ont été proposées. La méthode naïve de déchiffrer puis chiffrer résout le problème que ce soit pour le chiffrement asymétrique ou symétrique. Cependant, tout l'intérêt du PRE est d'éviter le déchiffrement puis le chiffrement avec une nouvelle clé mais plutôt de permettre directement la transformation du message chiffré.

Prenons par exemple le chiffrement de Vernam comme système de chiffrement symétrique. Soit M un message en clair, K_a une clé tel que ces deux éléments sont de même taille et $C_a = M \oplus K_a$. Pour transformer C_a en un message déchiffrable par une clé K_b, nous pouvons créer une clé de re-chiffrement $rk = K_a \oplus K_b$.

Le re-chiffrement sera alors une simple opération xor du message chiffré C_a avec la clé de re-chiffrement rk, ainsi :

$$C_b = C_a \oplus rk$$

$$C_b = (M \oplus \cancel{K_a}) \oplus (\cancel{K_a} \oplus K_b)$$

$$C_b = M \oplus K_b$$

Nous avons alors réussi à re-chiffrer notre message chiffré tout en évitant la méthode naïve, à savoir déchiffrer puis chiffrer. Néanmoins, la taille des clés est égale à la taille du message chiffré, que ce soit pour le chiffrement ou le déchiffrement. Nous pouvons voir aussi que le proxy seul ne peut pas retrouver les clés de chiffrement, mais sa collusion avec un utilisateur permet de retrouver la clé secrète du délégataire.

COOL et KEROMYTIS [2005] a montré que toute méthode de chiffrement symétrique E permettant de trouver $E(M, K_3) = E(E(M, K_1), K_2)$ présente une vulnérabilité. En termes de sécurité, une attaque à texte clair connu par brute force nécessitera $2^{\frac{n}{2}}$ au lieu de 2^n avec n la taille de la clé. Il propose alors une alternative mais qui repose aussi sur un double chiffrement qui reste presque similaire en termes de calcul qu'une méthode naïve. Il en est de même pour la solution proposée dans HIROSE [2010] qui permet de re-chiffrer des messages chiffrés à l'aide d'une fonction de chiffrement symétrique, combinée au mode opératoire GCM.

Les seules solutions SYALIM et collab. [2011] SAKURAI et collab. [2017] proposées par la suite qui permettent d'avoir de meilleures performances sont basées sur la méthode AONT (All Or Nothing Transform) proposée par RIVEST [1997] et utilisée dans OAEP. Ces solutions reposent sur un système de permutation à base de clés qui nécessite au total quatre clés secrètes. Néanmoins les PRE en résultant ne peuvent pas répondre aux différentes propriétés citées précédemment.

Dans le cadre de notre projet, nous nous sommes plutôt intéressés aux PRE traditionnels à

	basé sur	type	unidirectionnel	multi-use	key-private	non-transitive	key-optimal	non-interactive	résistant aux collusions	CCA
BLAZE et collab. [1998]	*ElGamal*	PKE	X	✓	X	X	✓	X	X	X
IVAN et DODIS [2003]	*ElGamal*; RSA;	IBE; PKE; *Sym*	✓	X	X	✓	✓	✓	X	X
ATENIESE et collab. [2006]	*Couplage*	PKE	X	X	✓	✓	✓	✓	X	X
GREEN et ATENIESE [2007]	*Couplage*	IBE	✓	X	X	✓	✓	✓	X	X
DENG et collab. [2008a]	*ElGamal*	PKE	X	X	X	X	✓	X	X	✓
LIBERT et VERGNAUD [2008]	*Couplage*	PKE	✓	X	X	✓	✓	✓	✓	X
ATENIESE et collab. [2009]	*Couplage*	PKE	✓	X	✓	✓	✓	✓	✓	X
CHOW et collab. [2010]	*ElGamal*	PKE	✓	X	✓	✓	✓	✓	✓	X
XAGAWA et TANAKA [2010]	LWE	PKE	X	X	X	✓	X	X	X	X
KIRSHANOVA [2014]	LWE	PKE	✓	X	X	✓	X	X	✓	X
NUÑEZ et collab. [2015]	NTRU	PKE	X	✓	X	✓	X	X	X	X
SELVI et collab. [2017]	*ElGamal*	PKE	✓	X	✓	✓	✓	✓	✓	✓

TABLEAU 2.1 – PRE dans la littérature

base de chiffrement asymétrique qui permettent de déléguer les droits de déchiffrement de manière sélective et unidirectionnelle en mettant l'accent sur leurs niveaux de sécurité et leurs performances.

Le tableau 2.1 présente différentes constructions et permet une comparaison par rapport au niveau de sécurité et aux propriétés de chaque PRE. Les premiers travaux IVAN et DODIS [2003] ATENIESE et collab. [2006] se concentrent plus sur la création d'un PRE qui est unidirectionnel. Ces propositions assurent une sécurité contre les attaques IND-CPA et ne sont pas résistantes aux collusions. CANETTI et HOHENBERGER [2007] proposent alors le premier schéma de PRE sécurisé contre les attaques IND-CCA où ils prouvent la sécurité de leur schéma en se basant sur la composabilité universelle CANETTI [2001]. Leur construction est bidirectionnelle, elle utilise le couplage et est prouvée sécurisée dans le modèle des oracles aléatoires. Les auteurs affirment qu'il est néanmoins recommandé de disposer d'un système dont la sécurité est prouvée dans le modèle standard et pour des raisons de performance, d'éviter d'utiliser le couplage, que ce soit pour des systèmes unis ou bidirectionnels.

Le problème ouvert présenté par Canetti et al. a été abordé par les auteurs de DENG et collab. [2008a] . Les auteurs arrivent en effet à résoudre une partie de ce problème ouvert qui concerne la construction d'un PRE sécurisé contre les attaques CCA sans utiliser le couplage. Le coût de calcul diminue et leur construction permet aussi de diminuer la taille du secret au niveau du re-chiffrement. Leur PRE est bidirectionnel et sa sécurité repose sur le CDH, sa construction se base principalement sur ElGamal et la signature de Schnorr, qui implique l'utilisation des oracles aléatoires. La même année, LIBERT et VERGNAUD [2008] les auteurs présentent un système PRE unidirectionnel inspiré par ATENIESE et collab. [2006] et réussissent à incorporer la propriété de temporalité à leurs schémas. Ils montrent comment le schéma temporel peut être étendu pour permettre des plages temporelles, au lieu de périodes uniques.

ATENIESE et collab. [2009] met le point sur une nouvelle propriété et formalise la notion de confidentialité des clés (key privacy). Ceci signifie qu'en utilisant la clé de re-chiffrement, nous ne pouvons pas récupérer l'identité du délégué et du délégant. Il montre pourquoi les systèmes précédents n'avaient pas cette propriété et propose un nouveau système de re-chiffrement considéré comme le premier PRE unidirectionnel à clé privée. Leur construction est à usage unique et

prouvée sécurisée contre les attaques CPA.

CHOW et collab. [2010] a démontré la possibilité de mener une attaque CCA sur le système de SHAO et CAO [2009] et montre comment résoudre le problème. En se basant sur le PRE de DENG et collab. [2008a], CHOW et collab. [2010] ont proposé leur propre système sans utiliser le couplage et en s'appuyant uniquement sur El-Gamal et la signature de Schnorr. SELVI et collab. [2017] trouvent une faille dans la preuve de sécurité de la construction de Chow et proposent de la corriger. Le système est unidirectionnel IND-CCA sécurisé dans le modèle de l'oracle aléatoire.

A partir de 2010, les chercheurs se sont davantage concentrés sur la création d'un PRE qui atteint une sécurité post-quantique. La majorité des systèmes proposés repose sur les réseaux euclidiens et plus précisément sur le problème d'apprentissage avec erreurs (LWE) tels que XAGAWA et TANAKA [2010], AONO et collab. [2013], KIRSHANOVA [2014]. Plus tard, NUÑEZ et collab. [2015] propose le premier PRE à base du crypto-système NTRU HOFFSTEIN et collab. [1998] qui repose sur le problème du plus court vecteur. Les opérations effectuées étant simplement des additions et des multiplications de polynômes, elles peuvent être calculées très efficacement et peuvent même être parallélisées. Cependant, les différentes propositions restent inefficaces pour une utilisation à grande échelle, surtout à cause des tailles des clés.

Jusqu'à maintenant, nous avons présenté les PRE proposés dans la littérature comme étant sécurisée contre les attaques IND-CPA ou IND-CCA. Néanmoins, il existe une certaine différence entre la sécurité des crypto-systèmes à clés publiques et la sécurité des PRE que nous présentons en détail dans le chapitre suivant.

Dans un premier temps, nous nous sommes intéressés à l'implémentation du PRE dans le cadre du projet VertPom. Ceci permettra le partage de données chiffrées principalement entre les fournisseurs d'énergie et la banque de l'énergie. En se basant sur le test de performance présentés dans le tableau 2.1 ainsi que sur les différentes études menées dans QIN et collab. [2016] et NUÑEZ et collab. [2017], nous avons conclu que l'algorithme de CHOW et collab. [2010] est l'un des PRE les plus efficaces que ce soit en termes de calcul ou en termes de taille de clé. La section suivante présente l'architecture utilisée ainsi que les détails d'implémentation.

2.3 PREaaS

Le cloud computing est devenu une technologie importante et omniprésente qui permet de fournir différentes ressources pour héberger les données et/ou effectuer des calculs. Il peut être classé en différents modèles : Software as a Service (SaaS) comme Flikr, Platform as a Service (PaaS) comme Google App Engine et Infrastructure as a Service (IaaS) comme Amazon AWS EC2. Les infrastructures, les plateformes ainsi que les applications sont alors les principaux services fournis par les fournisseurs de service cloud (CSP : Cloud Service Provider). Nous trouvons aussi d'autres services tels que Amazon AWS S3 qui fournit le stockage en tant que service. Toutefois, plusieurs questions se posent par rapport à la sécurité fournie par les CSP. En effet, les fournisseurs les plus connus n'assurent pas la confidentialité totale des données de leurs clients. Généralement, il est de la responsabilité du client de protéger ses données.

C'est dans ce sens que plusieurs travaux visant à protéger la vie privée ont émergé. Nous trouvons par exemple CryptDB POPA et collab. [2011], ESPRESSO KANG et collab. [2014] ou ES4AM HASAN et MOUFTAH [2015]. Ces solutions fournissent des services de chiffrement afin de maintenir la confidentialité. Néanmoins, chaque solution a ses limites. Par exemple, CryptDB ne peut pas être utilisé pour des bases de données ou des systèmes de fichiers sans SQL et la complexité de la gestion des clés augmente avec le nombre d'utilisateurs, ESPRESSO utilise uniquement le chiffrement symétrique et nécessite de faire confiance à un tiers ou aux CSP pour assurer le chiffrement et la gestion des clés. Plus proche de notre contexte, ES4AM fournit un système de chiffrement pour les infrastructures de mesure avancée des smart-grid (AMI : Advanced Metering Infrastructure) basé sur le chiffrement symétrique. Dans ce cas, la confiance reste toujours une préoccupation et

aucune de ces solutions ne permet le partage de données.

2.3.1 Partage de données et smartgrid

Depuis l'apparitions des proxy de re-chiffrement, rares sont les applications qui les utilisent. Nous pouvons tout de même citer Skycryptor JIVANYAN et collab. [2015] et Nucypher EGOROV et collab. [2018]. Skycryptor est utilisé comme une solution de sécurité en tant que service pour le partage de fichiers. Fondamentalement, la solution proposée utilise une clé symétrique unique pour chaque fichier à chiffrer avec AES puis chiffre la clé secrète avec la clé publique asymétrique de l'utilisateur générée grâce à l'algorithme du PRE. La solution est un logiciel dédié machine et est désormais commercialisée sous le nom de BeSafe. La machine de chaque utilisateur possède sa propre paire de clés et le re-chiffrement est utilisé pour partager des fichiers entre différents appareils ou utilisateurs. Mais surtout, les utilisateurs doivent installer le logiciel BeSafe et l'utiliser pour chiffrer les données.

La solution pourrait être adaptée à notre problématique et utilisée comme solution mais beaucoup de changements restent à faire. Au lieu de cela, nous proposons le PREaaS qui ne nécessite aucun client lourd et qui est plus flexible, modulaire et transparent.

Si nous examinons le système global, il y a trois acteurs principaux en plus du fournisseur de stockage cloud [2] comme illustré dans la Figure 2.2 (selon NUÑEZ et collab. [2017]) :

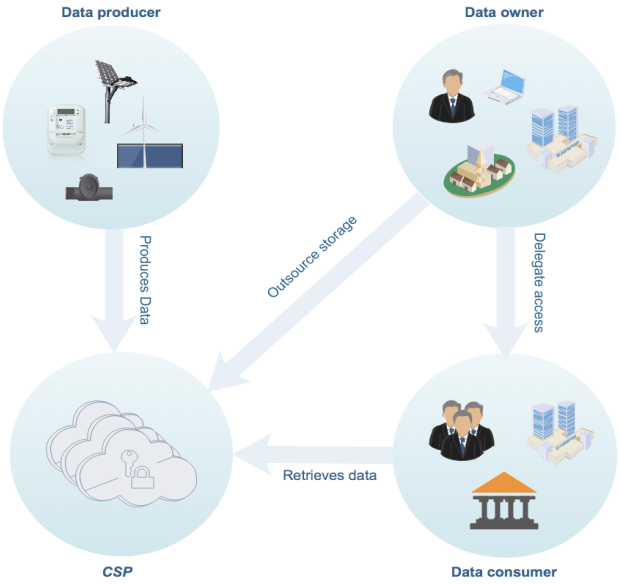


FIGURE 2.2 – Principaux acteurs dans un scénario de partage de données

— *Producteur de données (DP : Data Producers)* : dans notre cas, il contient tous les capteurs, compteurs et appareils générant des données qui doivent être chiffrées avant d'être stockées.

— *Propriétaire de données (DO : Data Owners)* : entité qui a le droit d'accéder à ses données et aussi ceux qui peuvent accorder l'accès aux données.

— *Consommateur de données (DC :Data Consumers)* : accès, avec l'autorisation des propriétaires des données, aux informations partagées par le biais du CSP. Il s'agit principalement la BE (Banque de l'Energie), mais aussi d'autres entités comme le NEM (Network Energy Manager).

2. Le CSP dans notre cas d'utilisation pourrait être un client NEM, une ville NEM ou un EP.

Ainsi, en reprenant la figure 2.3, nous pouvons voir que le CSP va stocker les données produites par les différents capteurs. Les données de consommation sont uniquement sous la maîtrise de l'utilisateur et ne sont en aucun cas diffusées ou transmises à une partie tierce autre que le fournisseur, sous réserve d'un consentement. Les données seront alors chiffrées avant d'être envoyées au CSP et seul le propriétaire de données pourra y avoir accès. Le fournisseur et gestionnaire d'énergie devient alors le consommateur de données. Il devra cependant récupérer le consentement du propriétaire de données afin d'accéder aux données stockées par le CSP. Ce consentement se traduira en une clé de re-chiffrement générée par le DO.

Nous supposons qu'il existe un secret partagé entre le propriétaire des données (DO) et le producteur des données (DP) qui chiffrent ces données. Notre système utilise la technique KEM/DEM $(\xi_{\mathrm{P}}^{asym}(\mathrm{K}) \| \xi_{\mathrm{K}}^{sym}(\mathrm{M}))$ où le secret partagé (par exemple, la clé secrète "K") est utilisé pour chiffrer les données avec un chiffrement symétrique, puis encapsulé dans le chiffré de la clé de session générée par le DP en utilisant le chiffrement asymétrique du PRE. Le résultat est envoyé pour être stocké dans le cloud. Lorsque les données doivent être partagées avec les consommateurs de données (DC), le DO crée une clé de re-chiffrement et la transfère au PRE. Celui-ci les re-chiffre en utilisant les clés de re-chiffrement correspondantes et les transmet au destinataire. Enfin, les données peuvent être consommées par le DC.

2.3.2 Architecture

Approche naïve :

La première approche (Figure 2.3) est de mettre le PRE comme serveur proxy du côté de la Banque de l'Énergie (BE). La BE enverra des requêtes avec sa clé publique aux différentes entités passant par le PRE pour récupérer les données nécessaires. Les entités correspondantes devraient générer des clés de re-chiffrement si le domaine appartient au propriétaire ou, à l'inverse, envoyer la clé publique de BE aux DO pour les générer, ainsi, le PRE pourrait re-chiffrer les données pour la BE. Les clés correspondantes (clé publique et sa clé de re-chiffrement) seront alors stockées pour les opérations à venir. La procédure de re-chiffrement peut toutefois être très coûteuse. De plus, outre les problèmes de gestion des clés, le fait de n'avoir qu'un seul mandataire pour le faire peut générer des retards. Il ne serait pas non plus possible de partager les données entre les EP [3] et les NEM [4].

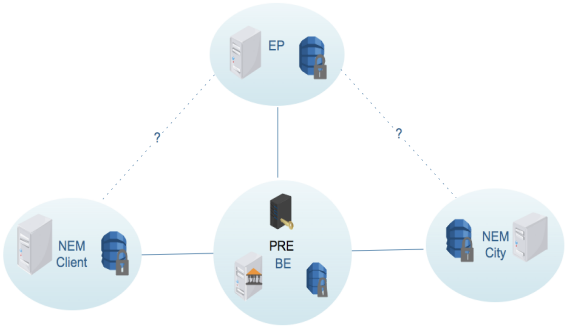


FIGURE 2.3 – Proxy Re-Encryption pour la BE

Deuxième approche :

Cette approche (Figure 2.4) consiste en l'utilisation de Proxy dans chaque cloud, laquelle pourrait être plus appropriée que la première approche. Elle permet en effet au NEM et au EP de par-

3. Energy Provider : fournisseur d'énergie
4. Network Energy Manager : gestionnaire du réseau énergétique

tager des données entre eux et de décharger la banque de tout le travail. Ainsi, chaque entité gérera ses propres clés, re-chiffrera les données nécessaires et les transférera au destinataire, comme illustré à la figure 3. Néanmoins, en raison de sa nature, le re-chiffrement prend encore beaucoup de temps. En outre, dans un cas plus général, il peut y avoir plusieurs NEM et plusieurs EP à la fois. Finalement, la situation sera plus compliquée si chaque entité est obligée d'avoir son propre PRE et de gérer un flux de données important.

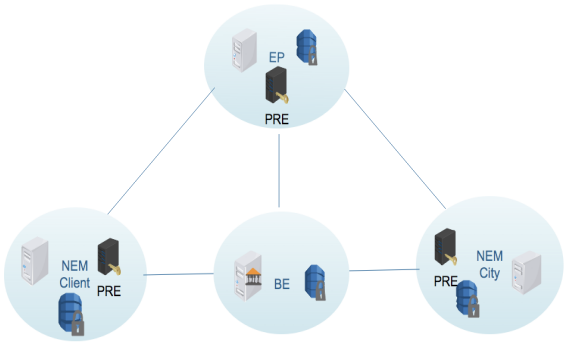


FIGURE 2.4 – Deuxième approche : PRE pour chaque entité

PRE en tant que Service :

La dernière approche (Figure 2.5) et la plus appropriée est de proposer le PRE comme un service de partage de données dans les systèmes multi-cloud. Les flux de données existent déjà et au lieu que le NEM fournisse des informations aux EP, il suffira de se référer au PREaaS plutôt qu'aux EP. Le potentiel du cloud computing prendrait en charge la consommation de temps de re-chiffrement. Notre solution peut être considérée comme faisant partie du SECaaS (Security As A Service). Notre PREaaS, a l'avantage de ne gérer que les données chiffrées et la gestion des clés ne traite que les clés publiques et les clés de re-chiffrement. Il n'affecte en aucun cas la confidentialité, même si les CSP ou les utilisateurs sont corrompus et de connivence avec le PREaaS. Ceci est garanti par un choix minutieux des algorithmes utilisés par le service.

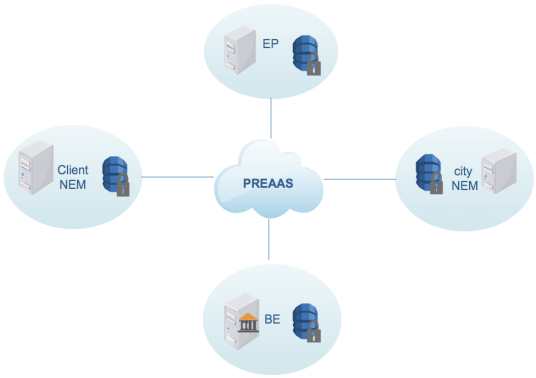


FIGURE 2.5 – PRE en tant que service

2.3.3 Flux de données

Plusieurs interactions existent entre les différentes entités. Elles concernent les différentes phases : enregistrement, authentification, stockage et le transfert de données. Une première partie est dédiée aux différents échanges entre le propriétaire de données, le CSP et le producteur de données. La deuxième partie présente la phase de partage de données entre le propriétaire et le consommateur de données à travers le CSP. La dernière partie présente le rôle du PREaaS dans

le scénario VertPom. En ce qui concerne l'authentification, une solution est présentée et détaillée dans le chapitre 4.

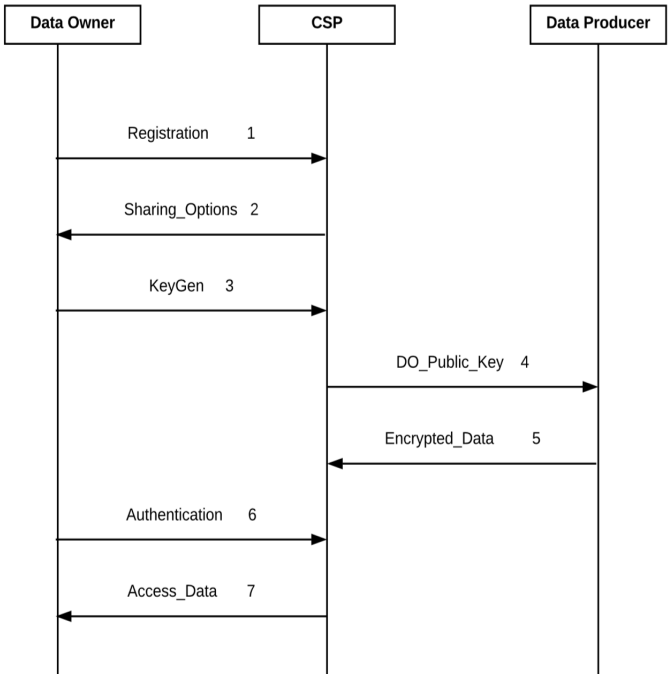


FIGURE 2.6 – Processus d'initialisation entre le DO, CSP et DP

La Figure 2.6 représente le processus d'initialisation entre un DO, un DP et un CSP :

— ① L'enregistrement du propriétaire des données auprès du CSP.

— ② Le CSP propose l'option de partage aux DOs afin d'utiliser un algorithme dédié.

— ③ Générer une paire de clés publique/privée par le DO, chiffrer la clé privée avec un schéma de chiffrement symétrique en utilisant une clé dérivée d'un mot de passe (l'utilisation de SCRYPT à cet effet est recommandée) et envoyer la clé publique et la clé privée chiffrée au CSP.

— ④ Le CSP stocke la clé privée chiffrée et transmet la clé publique.

— ⑤ Chiffrement des données avec une clé de session qui sera chiffrée avec la clé publique correspondante.

— ⑥ Authentification du DO auprès du CSP qui lui accordera l'accès en réponse aux données correspondantes.

— ⑦ Le déchiffrement de la clé de session qui sera déchiffrée par le DO avec sa clé privée.

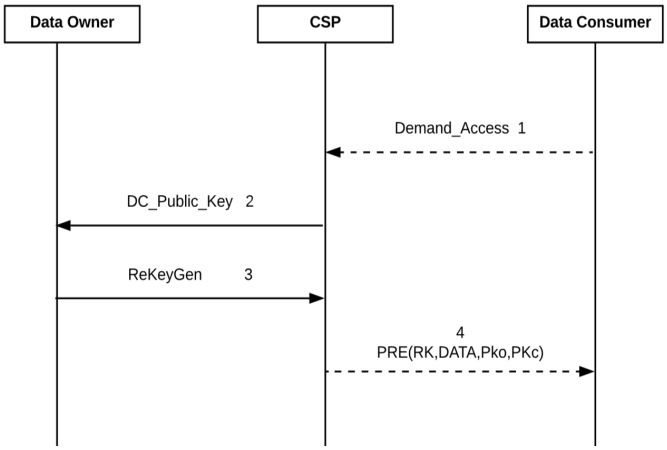


FIGURE 2.7 – Processus de partage entre le DO, CSP et DC

La figure 2.7 décrit le processus de partage entre le DO et le DC. Les flèches pointillées signifient que les demandes sont transmises indirectement en passant par les PREaaS. Ces derniers demanderont dans un premier temps l'accès aux données nécessaires. Le PREaaS transmettra la demande au DO avec la clé publique de la DC. En réponse, il crée une clé de re-chiffrement et la transmet au CSP. Celui-ci appelle le PREaaS pour qu'il re-chiffre le chiffré correspondant de la clé de session et envoie les données chiffrées nécessaires.

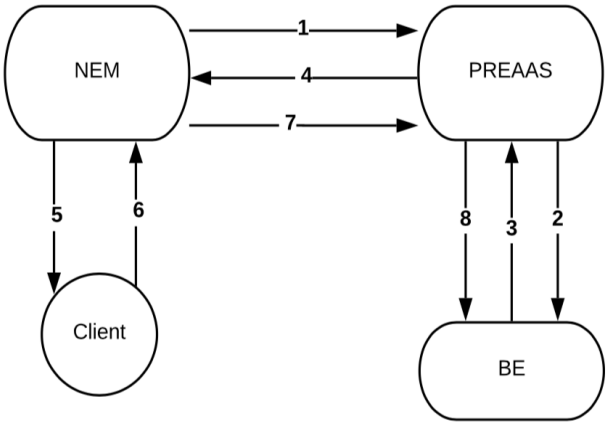


FIGURE 2.8 – Flux de données

Un exemple concret de flux de données est schématisé à la figure 2.8. Il comprend un NEM, un client, le BE et les PREaaS. Si nous prenons en considération tout le processus depuis l'enregistrement jusqu'à la consommation des données, il y aura alors 8 étapes :

— 1 : Enregistrement du NEM (qui joue le rôle de CSP) auprès de PREaaS.

— 2 : Notification à la BE qu'un nouveau NEM a été ajouté.

— 3 : Envoi d'une demande de partage pour accéder aux données nécessaires par la BE .

— 4, 5 : PREaaS reçoit et transmet la demande de partage en envoyant la clé publique de l'établissement au NEM qui l'enverra au client avec une demande d'accès.

— 6 : Le client crée et transfère la clé de re-chiffrement au NEM.

— 7 : Le NEM s'authentifie avec le PREaaS et renvoie les données chiffrées.

— 8 : Le PREaaS re-chiffre les données correspondantes avec la clé reçue et les transfère à la BE pour qu'elles soient consommées.

Dans le cas d'un partage "automatique ou prédéfini" entre le NEM et la BE, le PREaaS reçoit par défaut les nouvelles données produites par le CSP. Ceci n'implique que deux étapes (7,8) et dès qu'il y a un nouveau client, le NEM lui demande de créer une clé de re-chiffrement qui sera renvoyée au PREaaS et ajoutée à son répertoire (5,6,7,8).

2.4 Implémentation

Cette section présente l'algorithme PRE utilisé, l'environnement d'implémentation et les performances qui en résultent. L'algorithme utilisé est une variante de l'algorithme de CHOW et collab. [2010] proposée par SELVI et collab. [2017], il est choisi en raison de son efficacité qui est due au fait qu'il ne nécessite pas de couplages. Nous allons commencer par présenter l'algorithme d'origine proposé dans CHOW et collab. [2010] , puis revoir en quoi consiste les différents changements menés par les auteurs de SELVI et collab. [2017] afin de fixer la vulnérabilité existante. Les détails concernant la sécurité des PRE seront présenté dans le chapitre 3.

2.4.1 Algorithme de Chow

CHOW et collab. [2010] ont proposé la première construction d'un PRE qui est sécurisé contre les attaque CCA et résistant aux collusions sans se baser sur les applications bilinéaires. Le PRE de Chow repose principalement sur le chiffrement ElGamal et sa sécurité a été prouvée sous l'hypothèse de CDH. Nous allons rappeler dans un premier temps les différents algorithmes composant le PRE de Chow. Voici le schéma :

- $Setup(1^k)$: Choisir deux nombre premier p et q tels que $q/p-1$ où q est de taille $k-bits$. Soit g un générateur du groupe $\mathbb{G}$ tel que $\mathbb{G}$ est un sous-groupe de ${\mathbb{Z}^*}_q$ d'ordre q.
  On choisit quatre fonctions de hachage : $H_1 : \{0,1\}^{l_0} \times \{0,1\}^{l_1} \rightarrow {\mathbb{Z}^*}_q, H_2 : \mathbb{G} \rightarrow \{0,1\}^{l_0+l_1}, H_3 : \{0,1\}* \rightarrow {\mathbb{Z}^*}_q, H_4 : \mathbb{G} \rightarrow {\mathbb{Z}^*}_q$. L'espace des messages M est $\{0,1\}^{l_0}$, où $l_0 = l_1 = k$. Les paramètre du système sont : $params = (q, \mathbb{G}, g, H_1, H_2, H_3, H_4, l_0, l_1)$.

Les fonctions de hachages H_1, H_2 et H_3 sont modélisées comme étant des oracles aléatoires dans la preuve de sécurité. k est le paramètre de sécurité.

- $KeyGen(params)$:
  - Choisir $x_{a,1}, x_{a,2} \xleftarrow{\$} {\mathbb{Z}^*}_q$.
  - Calculer $Pk_{a,1} = g^{x_{a,1}}$, $Pk_{a,2} = g^{x_{a,2}}$
  - Retourner $Sk_a = (x_{a,1}, x_{a,2})$ & $Pk_a = (Pk_{a,1}, Pk_{a,2})$.

La génération de clé prend en entrée les paramètres du système et procède de la même manière qu'ElGamal. Cependant, elle génère deux éléments de clés privées et ses correspondantes publiques.

- $RekeyGen(Sk_a, Pk_a, Pk_b)$:
  - Choisir $h \xleftarrow{\$} \{0,1\}^{l_0}, \pi \xleftarrow{\$} \{0,1\}^{l_1}$.
  - Calculer $v = H_1(h, \pi), V = {Pk^v}_{b,2}$ et $W = H_2(g^v) \oplus (h||\pi)$.
  - Définir $rk = \dfrac{h}{x_{a,1} H_4(Pk_{a,2}) + x_{a,2}}$
  - Retourner $Rk_{a \rightarrow b} = (rk, V, W)$.

Nous pouvons constater que le PRE sera non-interactif en analysant les éléments d'entrée de la fonction de génération des clés de re-chiffrement. Seule la clé publique du délégué est donnée en entrée. Pour le délégataire seule sa clé privée suffit, mais cela nécessitera la clé publique de ce der-

nier à un certain moment du calcul. Ces coûts peuvent être éliminés en donnant la clé publique en entrée directement.

- $\xi^{asym}_{\mathrm{P}k_a}(m)$:
  - Choisir $u \xleftarrow{\$} \mathbb{Z}^*{}_q$, $w \xleftarrow{\$} \{0,1\}^{l_1}$.
  - Calculer $\mathrm{D} = (\mathrm{P}k_{a,1}^{\mathrm{H}_4(\mathrm{P}k_{a,2})}\mathrm{P}k_{a,2})^{u}$, $r = \mathrm{H}_1(m,w)$.
  - Calculer $\mathrm{E} = (\mathrm{P}k_{a,1}^{\mathrm{H}_4(\mathrm{P}k_{a,2})}\mathrm{P}k_{a,2})^{r}$.
  - Calculer $\mathrm{F} = \mathrm{H}_2(g^r) \oplus (m||w)$.
  - Calculer $s = u + r \times \mathrm{H}_3(\mathrm{D},\mathrm{E},\mathrm{F}) \bmod(q)$.
  - Retourner $\mathrm{C}_a = (\mathrm{D},\mathrm{E},\mathrm{F},s)$.

Le processus de chiffrement utilise presque le même principe que Hashed ElGamal, où un masque jetable est calculé, haché puis additionné avec le message en clair. Aussi la signature est calculée de la même manière en se basant sur une variante de la signature de Schnorr. Cependant, les autres éléments du message chiffré reposent sur les deux éléments de la clé publique afin de permettre la transformation du message chiffré par l'algorithme suivant :

- $\xi^{asym'}_{\mathrm{P}k_a}(m)$:
  - Choisir $h \xleftarrow{\$} \{0,1\}^{l_0}$ et $\pi \xleftarrow{\$} \{0,1\}^{l_1}$.
  - Calculer $v = \mathrm{H}_1(h,\pi)$.
  - Calculer $\mathrm{V} = \mathrm{P}k_{a,2}{}^{v}$ et $\mathrm{W} = \mathrm{H}_2(g^v) \oplus (h||\pi)$.
  - Choisir $w \xleftarrow{\$} \{0,1\}^{l_1}$.
  - Calculer $r = \mathrm{H}_1(m,w)$.
  - Calculer $\mathrm{E}' = (g^r)^h$ et $\mathrm{F} = \mathrm{H}_2(g^r) \oplus (m||w)$.
  - Retourner $\mathrm{C}_a = (\mathrm{E}',\mathrm{F},\mathrm{V},\mathrm{W})$.

Cette fonction de chiffrement permet d'avoir des messages chiffrés non-transformables. Elle utilise le même processus que celui de Hashed ElGamal avec un seul élément de la clé publique. Ainsi, la transformation ne peut pas avoir lieu.

- $\mathrm{R}e\mathrm{E}nc(\mathrm{C}_a,\mathrm{P}k_a,\mathrm{P}k_b,\mathrm{R}k_{a\to b})$:
  - Vérifier si : $\mathrm{D}.\mathrm{E}^{\mathrm{H}_3(\mathrm{D},\mathrm{E},\mathrm{F})} = (\mathrm{P}k_{a,1}^{\mathrm{H}_4(\mathrm{P}k_{a,2})}\mathrm{P}k_{a,2})^{s}$ sinon retourner $\perp$.
  - Calculer $\mathrm{E}' = \mathrm{E}^{rk}$
  - Retourner $\mathrm{C}'_b = (\mathrm{E}',\mathrm{F},\mathrm{V},\mathrm{W}) = (g^{r.h},\mathrm{H}_2(g^r) \oplus (m||w),\mathrm{P}k^{v}{}_{b,2},\mathrm{H}_2(g^v) \oplus (h||\pi))$.

La fonction de re-chiffrement prend en entrée les deux clés publiques, celle du délégué et du délégataire, le message chiffré du délégataire et la clé de re-chiffrement. Le premier traitement effectué permet de vérifier si le message chiffré est valide. Si c'est le cas, il procède au re-chiffrement. Ce processus calcule à partir d'un élément de la clé de re-chiffrement et d'un élément du message chiffré, l'élément qui permettra au délégué de calculer le masque jetable afin de pouvoir déchiffrer le message re-chiffré. Il élimine deux éléments du message chiffré, qui seront remplacés par deux éléments de la clé de re-chiffrement.

- $Dec(C_a, Pk_a, Sk_a)$:
  (Message chiffré original)
  - Vérifier si : $D.E^{H_3(D,E,F)} = (Pk_{a,1}^{H_4(Pk_{a,2})} Pk_{a,2})^{s}$ sinon retourner $\perp$.
  - Calculer $(m||w) = F \oplus H_2(E^{\frac{1}{x_{a,1} H_4(Pk_{a,2}+x_{a,2})}})$
  - Retourner m si : $E = (Pk_{a,1}^{H_4(Pk_{a,2})} Pk_{a,2})^{H_1(m||w)}$, sinon retourner $\perp$.

  (Messages re-chiffrés)
  - Calculer $(h||\pi) = W \oplus H_2(V^{\frac{1}{Sk_{a,2}}})$.
  - Calculer $(m||w) = F \oplus H_2(E'^{\frac{1}{h}})$.
  - Si : $V = Pk_{a,2}^{H_1(h,\pi)}$ et $E' = g^{(h \times H_1(m,w))}$ retourner m , sinon retourner $\perp$.

Il existe deux fonctions de déchiffrement distinctes : la première permet de déchiffrer les messages transformables originaux, tandis que la deuxième permet de déchiffrer à la fois les messages re-chiffrés ainsi que les messages non transformables. Nous pouvons remarquer alors que le PRE proposé ne peut pas être multi-hop puisque un message re-chiffré a la même distribution qu'un message non transformable, ainsi aucune transformation ne peut lui être appliqué par la suite.

2.4.2 Différence entre l'algorithme de Chow et l'algorithme de Selvi

Ici nous allons présenter les différences majeures entre les deux constructions CHOW et collab. [2010] et SELVI et collab. [2017]. La génération des clés ainsi que les clés de re-chiffrement n'ont pas été affectées. Seul le chiffrement et, par extension, le re-chiffrement et le déchiffrement ont été redéfinis comme suit :

- $Setup(1^k)$: Choisir deux nombre premier p et q tel que $q/p-1$ où q est de taille $k-bits$. Soit g un générateur du groupe $\mathbb{G}$ tel que $\mathbb{G}$ est un sous-groupe de $\mathbb{Z}^*{}_q$ d'ordre q.
  On choisit cinq fonctions de hachage : $H_1 : \{0,1\}^{l_0} \times \{0,1\}^{l_1} \rightarrow \mathbb{Z}^*{}_q, H_2 : \mathbb{G} \rightarrow \{0,1\}^{l_0+l_1}, H_3 : \{0,1\}* \rightarrow \mathbb{Z}^*{}_q, H_4 : \mathbb{G} \rightarrow \mathbb{Z}^*{}_q et H_5 : \mathbb{G}^4 \times \{0,1\}^{l_0+l_1} \rightarrow \mathbb{G}$. L'espace des messages M est $\{0,1\}^{l_0}$, où $l_0 = l_1 = k$. Les paramètre du système sont : $params = (q, \mathbb{G}, g, H_1, H_2, H_3, H_4, H_5, l_0, l_1)$.

Nous pouvons remarquer que la seule différence entre cette version et l'ancienne proposition de Chow est l'ajout d'une nouvelle fonction de hachage. Cependant, toutes les fonctions seront modélisées comme oracles aléatoires, contrairement à la première version où H_4 n'était pas considéré ainsi.

- $\xi^{asym}_{Pk_a}(m)$:
  - Choisir $u \xleftarrow{\$} \mathbb{Z}^*{}_q, w \xleftarrow{\$} \{0,1\}^{l_1}$ et calculer $r = H_1(m, w)$.
  - Calculer $D = (Pk_{a,1}^{H_4(Pk_{a,2})} Pk_{a,2})^{u}$ et $\overline{D} = H_5(Pk_{a,1}, Pk_{a,2}, D, E, F)^{u}$.
  - Calculer $E = (Pk_{a,1}^{H_4(Pk_{a,2})} Pk_{a,2})^{r}$ et $\overline{E} = H_5(Pk_{a,1}, Pk_{a,2}, D, E, F)^{r}$
  - Calculer $F = H_2(g^r) \oplus (m||w)$.
  - Calculer $s = u + r \times H_3(D, \overline{E}, F) \bmod(q)$.
  - Retourner $C_a = (D, \overline{E}, F, s)$.

Le chiffrement inclus deux nouveaux calculs qui servent à garantir l'intégrité du message chiffré en utilisant la nouvelle fonction de hachage. Néanmoins, nous remarquons que $\overline{D}$ est calculé mais pas utilisé. Ce traitement sera ignoré lors de l'instanciation.

- $ReEnc(C_a, Pk_a, Pk_b, Rk_{a\to b})$:
  - Calculer $E = ((Pk_{a,1}^{H_4(Pk_{a,2})} Pk_{a,2})^s . D^{-1})^{H_3(D,\bar{E},F)^{-1}}$.
  - Vérifier si $(Pk_{a,1}^{H_4(Pk_{a,2})} Pk_{a,2})^s = D.E^{H_3(D,\bar{E},F)}$ sinon retourner $\perp$.
  - Calculer $\overline{D} = H_5(Pk_{a,1}, Pk_{a,2}, D, E, F)^s . (\overline{E}^{(H_3(D,\bar{E},F))})^{-1}$
  - Vérifier si : $D.E^{H_3(D,E,F)} = (Pk_{a,1}^{H_4(Pk_{a,2})} Pk_{a,2})^s$ sinon retourner $\perp$.
  - Calculer $E' = E^{rk}$
  - Retourner $C'_b = (E', F, V, W) = (g^{r.h}, H_2(g^r) \oplus (m||w), Pk^v{}_{b,2}, H_2(g^v) \oplus (h||\pi))$.

La différence ici entre les deux versions est la vérification du message chiffré à travers $\overline{D}$ en plus du D. Il a été montré dans Selvi et collab. [2017] que la vérification de la validité du message chiffré via D ne suffit pas et peut être détournée par un adversaire.

$$
\begin{aligned}
E &= \left(\left(Pk_{a,1}^{H_4(Pk_{a,2})} Pk_{a,2}\right)^{(u+r\cdot H_3(E,\bar{E},F))} \cdot \left(Pk_{a,1}^{H_4(Pk_{a,2})} Pk_{a,2}\right)^{-u}\right)^{H_3(D,\bar{E},F)^{-1}} \\
&= \left(\left(Pk_{a,1}^{H_4(Pk_{a,2})} Pk_{a,2}\right)^{rH_3(D,\bar{E},F)}\right)^{H_3(D,\bar{E},F)^{-1}} \\
&= \left(Pk_{a,1}^{H_4(Pk_{a,2})} Pk_{a,2}\right)^{r} . \\
\bar{D} &= H_5\left(Pk_{a,1}, Pk_{a,2}, D, E, F\right)^s \cdot \left(\bar{E}^{H_3(D,\bar{E},F)}\right)^{-1} \\
&= H_5\left(Pk_{a,1}, Pk_{a,2}, D, E, F\right)^{(u+r\cdot H_3(D,\bar{E},F))} \cdot \left(H_5\left(Pk_{a,1}, Pk_{a,2}, D, E, F\right)^{rH_3(D,\bar{E},F)}\right)^{-1} \\
&= H_5\left(Pk_{a,1}, Pk_{a,2}, D, E, F\right)^{u} .
\end{aligned}
$$

Puisque la vulnérabilité ne concerne que le re-chiffrement. La procédure du chiffrement des messages non transformables reste la même que celle de la première version. Par conséquent le chiffrement et le déchiffrement des messages transformables doivent être modifiées. Cependant, les messages non-transformables restent résistants aux attaques CCA. Quant aux messages re-chiffrés, ils ont la même distribution que les messages non-transformables et sont déchiffrables par la même méthode. Nous présentons alors la fonction de déchiffrement des messages transformables originaux.

- $Dec(C_a, Pk_a, Sk_a)$:
  (Message chiffré original)
  - Calculer $E = ((Pk_{a,1}^{H_4(Pk_{a,2})} Pk_{a,2})^s . D^{-1})^{H_3(D,\bar{E},F)^{-1}}$.
  - Vérifier si $(Pk_{a,1}^{H_4(Pk_{a,2})} Pk_{a,2})^s = D.E^{H_3(D,\bar{E},F)}$ sinon retourner $\perp$.
  - Calculer $\overline{D} = H_5(Pk_{a,1}, Pk_{a,2}, D, E, F)^s . (\overline{E}^{(H_3(D,\bar{E},F))})^{-1}$
  - Vérifier si : $D.E^{H_3(D,E,F)} = (Pk_{a,1}^{H_4(Pk_{a,2})} Pk_{a,2})^s$ sinon retourner $\perp$.
  - Calculer $(m||w) = F \oplus H_2(E^{\frac{1}{x_{a,1} H_4(Pk_{a,2}+x_{a,2})}})$
  - Retourner m si : $\overline{E} = H_5(Pk_{a,1}, Pk_{a,2}, D, E, F)^{H_1(m||w)}$, sinon retourner $\perp$.

2.4.3 Instanciation

Grâce à cette étude Qin et collab. [2016], nous pouvons voir la comparaison en termes de calcul des schémas PRE. En ne conservant que les schémas qui sont sécurisés contre les attaques

CCA, nous pouvons conclure que l'algorithme de Chow est le plus efficace d'entre eux en terme de temps d'exécution. Dans SELVI et collab. [2017], les auteurs trouvent une faille dans la preuve de sécurité de la construction de Chow. Ils proposent de la corriger en incorporant des informations supplémentaires à l'algorithme de chiffrement existant, ainsi que des contrôles de validité du texte chiffré dans les deux algorithmes de re-chiffrement et de déchiffrement. Plus important encore, le schéma est unidirectionnel, résistant aux collusions et sécurisé contre les attaques CCA dans le modèle de l'oracle aléatoire. Deux implémentations ont été réalisées, la première consiste à utiliser un groupe générique d'une longueur d'ordre premier de 3072 bits, et la seconde à utiliser la norme NIST ECC p-256 GUERON et KRASNOV [2015] grâce à SJCL (Standford Javascript Crypto Library) STARK et collab. [2013]. Les deux correspondent au même niveau de sécurité qui est de 128 bits. Une version formelle de l'algorithme de Chow avec ECC est donnée ci-dessous :

— $Setup(1^k)$: Pour un niveau de sécurité de 128-bit, choisir un nombre premier p de taille 256-bit et une courbe elliptique $\mathbb{E}_{\mathbb{F}p}$ tel que $a = p-3$. $G = (x_G, y_G)$, un point sur la courbe, connu comme point de base ou générateur d'ordre p. L'equation de la courbe elliptique utilisé est : $y^2 = x^3 - 3x + b \bmod (p)$. (Vous pouvez trouver les valeurs exactes des paramètres de la courbe p-256 du NIST dans GUERON et KRASNOV [2015]). Choisir 5 fonctions de hachages : $H_1 : \{0,1\}^{l_0} \times \{0,1\}^{l_1} \rightarrow \mathbb{Z}^*{}_p, H_2 : \mathbb{E}_{\mathbb{F}p} \rightarrow \{0,1\}^{l_0+l_1}, H_3 : \{0,1\}* \rightarrow \mathbb{Z}^*{}_p, H_4 : \mathbb{E}_{\mathbb{F}p} \rightarrow \mathbb{Z}^*{}_p, H_5 : \mathbb{E}_{\mathbb{F}p}^4 \times \{0,1\}^{l_0+l_1} \rightarrow E_{Fp}$. L'espace des message clair M est : $\{0,1\}^{l_0}$, où $l_0 = l_1 = k$. $params = (p, \mathbb{E}_{\mathbb{F}p}, G, H_1, H_2, H_3, H_4, l_0, l_1)$.

Pour l'instanciation de la deuxième version de l'algorithme de Chow, nous avons utilisé un groupe de points de courbe elliptique dont les paramètres sont publiés et standardisés par le NIST. En ce qui concerne les fonctions de hachage, la seule particularité est de faire un mapping vers un point de courbe pour la fonction H_5. Une simple fonction de hachage ne sera pas forcément la meilleure idée pour répondre à cette problématique. Supposons que nous pouvons utiliser SHA-3 et que l'empreinte générée à travers cette fonction sera la valeur de coordonnée x. À travers cette valeur, nous pouvons utiliser l'équation de la courbe afin de calculer la valeur du coordonnée y qui se résumera au calcul d'une racine carrée. Nous aurons alors deux valeurs possibles pour y. Il faut tirer aléatoirement une valeur des deux à chaque nouvel appel à cette fonction. Ensuite, au moment de la vérification, il faut faire appel à la fonction deux fois et vérifier lequel des deux résultats correspond à notre point de courbe. Une solution envisageable, mais pas très efficace, est l'utilisation de SHA-3 pour générer l'ordonnée y. Nous savons qu'il correspond à un seul point de la courbe et qu'une valeur est possible pour l'abscisse x. Cependant, il va falloir résoudre l'équation de la courbe et se retrouver à calculer la racine cubique, ce qui est plus coûteux en termes de calcul. Il existe certaines solutions plus optimales et plus sûres permettant de faire à la fois le hachage et le mapping vers un point de courbe tel que ICART [2009] et TIBOUCHI et KIM [2017]. Cette problématique n'a cependant pas été traitée dans le cadre de nos travaux.

— $KeyGen(params)$:
  — Choisir $x_{a,1}, x_{a,2} \xleftarrow{\$} \mathbb{Z}^*{}_p$.
  — Calculer $Pk_{a,1} = x_{a,1}.G$, $Pk_{a,2} = x_{a,2}.G$
  — Retourner $Sk_a = (x_{a,1}, x_{a,2})$ & $Pk_a = (Pk_{a,1}, Pk_{a,2})$.
— $RekeyGen(Sk_a, Pk_a, Pk_b)$:
  — Choisir $h \xleftarrow{\$} \{0,1\}^{l_0}, \pi \xleftarrow{\$} \{0,1\}^{l_1}$, puis $v = H_1(h,\pi), V = v.Pk_{b,2}$ et $W = H_2(v.G) \oplus (h||\pi)$.
  — Définir $rk = \dfrac{h}{x_{a,1} H_4(Pk_{a,2}) + x_{a,2}}$
  — Retourner $Rk_{a \rightarrow b} = (rk, V, W)$.

— $\xi^{asym}_{\mathrm{P}k_a}(m)$:

  — Choisir $u \xleftarrow{\$} \mathbb{Z}^*{}_p$, $w \xleftarrow{\$} \{0,1\}^{l_1}$, puis calculer $r = \mathrm{H}_1(m,w)$.

  — Calculer $\mathrm{D} = u.[(\mathrm{H}_4(\mathrm{P}k_{b,2}).\mathrm{P}k_{a,1}) * \mathrm{P}k_{a,2}]$ et $\mathrm{E} = r.[(\mathrm{H}_4(\mathrm{P}k_{b,2}).\mathrm{P}k_{a,1}) * \mathrm{P}k_{a,2}]$.

  — Calculer $\mathrm{F} = \mathrm{H}_2(r.\mathrm{G}) \oplus (m||w)$.

  — Calculer $\overline{\mathrm{D}} = u.\mathrm{H}_5(\mathrm{P}k_{a,1}, \mathrm{P}k_{a,2}, \mathrm{D}, \mathrm{E}, \mathrm{F})$ & $\overline{\mathrm{E}} = r.\mathrm{H}_5(\mathrm{P}k_{a,1}, \mathrm{P}k_{a,2}, \mathrm{D}, \mathrm{E}, \mathrm{F})$.

  — Calculer $s = u + r \times \mathrm{H}_3(\mathrm{D}, \overline{\mathrm{E}}, \mathrm{F}) \bmod(p)$.

  — Retourner $\mathrm{C}_a = (\mathrm{D}, \overline{\mathrm{E}}, \mathrm{F}, s)$.

— $\mathrm{ReEnc}(\mathrm{C}_a, \mathrm{P}k_a, \mathrm{P}k_b, \mathrm{R}k_{a\to b})$:

  — Calculer $\mathrm{D} = s.[(\mathrm{H}_4(\mathrm{P}k_{b,2}).\mathrm{P}k_{a,1})^*\mathrm{P}k_{a,2}]^*(\mathrm{H}_3(\mathrm{E}, \overline{\mathrm{E}}, \mathrm{F}).\mathrm{E})^{-1}$

  — Calculer $\overline{\mathrm{D}} = (s.\mathrm{H}_5(\mathrm{P}k_{a,1}, \mathrm{P}k_{a,2}, \mathrm{D}, \mathrm{E}, \mathrm{F}))^*(\mathrm{H}_3(\mathrm{E}, \overline{\mathrm{E}}, \mathrm{F}).\overline{\mathrm{E}})^{-1} = u.[\mathrm{H}_5(\mathrm{P}k_{a,1}, \mathrm{P}k_{a,2}, \mathrm{D}, \mathrm{E}, \mathrm{F})]$

  — Si :
  $s.[(\mathrm{H}_4(\mathrm{P}k_{b,2}).\mathrm{P}k_{a,1})^*\mathrm{P}k_{a,2}] = \mathrm{D}^*(\mathrm{H}_3(\mathrm{E}, \overline{\mathrm{E}}, \mathrm{F}).\mathrm{E})$ (1)
  $s.[\mathrm{H}_5(\mathrm{P}k_{a,1}, \mathrm{P}k_{a,2}, \mathrm{D}, \mathrm{E}, \mathrm{F})] = \overline{\mathrm{D}}^*(\mathrm{H}_3(\mathrm{E}, \overline{\mathrm{E}}, \mathrm{F}).\mathrm{E})$ (2)

  — Calculer $\mathrm{E}' = \mathrm{E}^{rk} = (r \times h).\mathrm{G}$, sinon retourner $\perp$.

  — Retourner $\mathrm{C}'_b = (\mathrm{E}', \mathrm{F}, \mathrm{V}, \mathrm{W})$.

— $\mathrm{Dec}(\mathrm{C}_a, \mathrm{P}k_a, \mathrm{S}k_a)$:
(Message chiffré original)

  — Calculer D et $\overline{\mathrm{D}}$. Puis vérifier (1) & (2) .

  — Si ces deux conditions sont vérifiées, calculées $(m||w) = \mathrm{F} \oplus \mathrm{H}_2(\frac{1}{x_{a,1}\mathrm{H}_4(\mathrm{P}k_{a,2} + x_{a,2})}.\mathrm{E})$

  — Si : $\mathrm{E} = \mathrm{H}_1(m,w).[(\mathrm{H}_4(\mathrm{P}k_{b,2}).\mathrm{P}k_{a,1})^*\mathrm{P}k_{a,2}]$ and $\overline{\mathrm{E}} = \mathrm{H}_1(m,w).[\mathrm{H}_5(\mathrm{P}k_{a,1}, \mathrm{P}k_{a,2}, \mathrm{D}, \mathrm{E}, \mathrm{F})]$ retourner m , sinon retourner $\perp$.

(Message chiffré transformable)

  — Calculer $(h||\pi) = \mathrm{W} \oplus \mathrm{H}_2(\frac{1}{\mathrm{S}k_{a,2}}.\mathrm{V})$, puis $(m||w) = \mathrm{F} \oplus \mathrm{H}_2(\frac{1}{h}.\mathrm{E}')$.

  — Si : $\mathrm{V} = \mathrm{H}_1(h,\pi).\mathrm{P}k_{a,2}$ et $\mathrm{E}' = (h \times \mathrm{H}_1(m,w)).\mathrm{G}$ retourner m , sinon retourner $\perp$.

La différence principale entre notre instanciation et celle de l'algorithme proposé dans Selvi et collab. [2017] est notre utilisation de groupe de points de courbe elliptique qui est un groupe additif. Ainsi, toutes les exponentiations sont transformées en multiplications scalaires. Aussi, les fonctions de hachages ont dû être adaptées afin de générer des points de courbes elliptiques.

Nous avons choisi d'utiliser JavaScript comme technologie de base pour pouvoir être exécuté côté client directement par un navigateur ou même des appareils mobiles sans aucun logiciel supplémentaire, mais aussi côté serveur grâce à Nodejs, ceci en utilisant le même langage. Pour les tests, nous avons utilisé une machine ayant un processeur i7 intel core de 2,5 GHz, avec 16 Go de RAM.

Le tableau 2.2 montre les ressources consommées en termes de temps et de stockage par les différentes fonctions de l'algorithme de Chow pour les deux implémentations. Nous pouvons voir que les fonctions de chiffrement et de re-chiffrement consomment le plus par rapport à la génération des clés et au déchiffrement. En pratique, les fonctions de chiffrement et de génération de clés ne sont pas souvent appelées. La génération de clés de re-chiffrement dépend du nombre de dé-

TABLEAU 2.2 – Performance de l'algorithme de Chow en terme de temps de calcul et de stockage

Function	$\mathbb{F}_p(ms)$	$\mathbb{F}_p(bits)$	ECC(ms)	ECC$(bits)$
KeyGen	515	512 & 6144	68	256 & 1024
ReKeyGen	502	3584	48	768
Encrypt	1000	6656	95	1024
ReEncrypt	967	6656	89	1024
Decrypt	979	–	78	–

légations nécessaires, mais elle n'est pas encore contraignante en termes de temps. En revanche, la fonction de re-chiffrement est appelée pour chaque nouvel utilisateur, nouvelle délégation et clé de session modifiée. Avoir un service indépendant qui effectue le travail de re-chiffrement est allégeant (Figure 2.9).

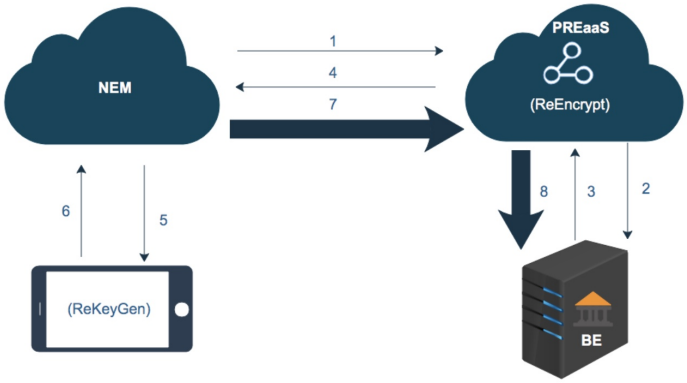


FIGURE 2.9 – Distribution des tâches entre les différents environnements

L'API qui a été développée sert de passerelle vers nos fonctionnalités de service. Cette API ne gère aucune donnée sensible et la seule préoccupation est de prévenir les attaques de DDos. Ainsi, un simple mécanisme de clé API peut être utilisé comme authentification pour identifier l'utilisateur de l'API et ses consommateurs TANG et collab. [2015]. Cependant, nous avons proposé une solution de délégation d'authentification qui permet aux différentes entités de s'authentifier auprès du PREaaS ou d'autres fournisseurs de services que nous présentons en détails dans le chapitre 4.

2.5 Conclusion

Nous nous sommes intéressés à la problématique de partage de données dans un système de stockage distribué. Ceci peut se traduire alors par la délégation de droit de déchiffrement, une méthode qui a été proposée pour la première fois par BBS en 1998 appelé Proxy de re-chiffrement (PRE).

Plusieurs travaux ont été proposés depuis et peuvent être classifiés en trois catégories : PRE à base d'homomorphisme, ceux à base de couplage et ceux à base de chiffrement standard tels que ElGamal. Les algorithmes appartenant aux deux premières catégories ont pour inconvénient une utilisation coûteuse en termes de stockage pour la première catégorie et de calcul pour la deuxième.

Nous avons proposé alors le PRE en tant que service (PREaaS) pour le partage de données dans des systèmes multi-cloud SBAI et collab. [2019]. Le potentiel du Cloud Computing prend en charge

le temps nécessaire au re-chiffrement et cette solution peut être considérée comme faisant partie de la sécurité en tant que service (SECaaS).

Notre PREaaS présente l'avantage de ne gérer que les données chiffrées et la gestion de clés, mais uniquement les clés publiques et les clés de re-chiffrement. Cela n'affecte en aucun cas la confidentialité, même si les CSP ou les utilisateurs sont corrompus et collusionnent avec le PREaaS. Ceci est garanti par un choix prudent des algorithmes utilisés par le service.

Une librairie a été créée où la nouvelle version de l'algorithme de Chow a été implémentée. Ce dernier est basé sur ElGamal et répond aux différentes propriétés dont nous avons besoin telle que l'unidirectionnalité. Nous avons choisi d'utiliser JavaScript comme technologie de base car ce code peut être exécuté directement côté client par le navigateur, par des appareils mobiles et aussi côté serveur grâce à Nodejs. Pour les tests, deux implémentations ont été effectuées : d'abord un groupe générique d'une longueur d'ordre premier de 3 072 bits et le second utilisant la courbe elliptique standardisée par le NIST. Ils correspondent tous deux au même niveau de sécurité de 128 bits. Les ressources en termes de temps consommé par les différentes fonctions de l'algorithme de Chow pour les deux implémentations montrent le grand impact de l'utilisation des courbes elliptiques sur les performances.

Quant à l'algorithme de Chow, il reste prouvé sécurisé dans le modèle des oracles aléatoires. Les preuves dans ce modèle reposent sur l'utilisation de fonctions considérées comme aléatoires qui est une solution idéaliste et reste purement théorique. En effet, jusqu'à maintenant, ces oracles aléatoires sont instanciés par des fonctions de hachage ou a priori en utilisant le HMAC.

Une grande partie des protocoles utilisant les oracles aléatoires ont pu passer de la théorie à la pratique. Malgré les différentes critiques sur le modèle des oracles aléatoires, les protocoles tels que TLS utilisent quand même des crypto-systèmes n'ayant aucune preuve dans le modèle standard tel que RSA-OAEP. Dans cette optique, notre PREaaS permet également l'utilisation de différents algorithmes PRE tels que l'algorithme de BBS avec des clés éphémères, l'algorithme d'Ateniese, ...

Le prochain chapitre concerne principalement la conception d'un PRE CCA dans le modèle standard sans l'utilisation de couplage.

Chapitre 3

Analyse et conception d'un PRE

Sommaire

3.1 Introduction

Ce chapitre aborde l'aspect sécurité des proxy de re-chiffrement (PRE). Par rapport à la définition de base de la sécurité sémantique, celle-ci ne changera pas. En effet, un message chiffré, original ou re-chiffré, ne devrait apporter aucune information en plus de ce qui est déjà connu sur le message en clair. Cependant, les méthodes utilisées pour prouver la sécurité d'un PRE ne sont pas forcement les mêmes. Nous verrons quelles sont les différences majeures entre la sécurité d'un PKE et d'un PRE ainsi que les modèles de preuve formalisés pour prouver la sécurité d'un PRE.

Ensuite, nous allons définir la vérifiabilité publique qui est une brique essentielle pour la construction des PRE sécurisés contre les attaques IND-CCA.

Nous reprenons l'algorithme de CHOW et collab. [2010] pour analyser sa construction, sa vulnérabilité ainsi que la proposition de SELVI et collab. [2017] pour y remédier. Cependant, le PRE résultant reste prouvé sécurisé dans le modèle des oracles aléatoires.

Nous soulignons que si une preuve de sécurité dans le modèle de l'oracle aléatoire a une certaine valeur, il ne s'agit encore que d'une preuve heuristique. En particulier, ces types de preuves n'excluent pas la possibilité de rompre le schéma sans rompre l'hypothèse sous-jacente CRAMER et SHOUP [1998]. Elles n'excluent pas non plus la possibilité de rompre le schéma sans trouver une sorte de faiblesse dans la fonction de hachage, comme cela a été démontré par Canetti, Goldreich et Halevi CANETTI et collab. [2004a].

Nous nous intéressons alors à la problématique posée par CANETTI et HOHENBERGER [2007] sur la construction d'un PRE unidirectionnel et sécurisé contre les attaques IND-CCA dans le modèle standard sans utiliser le couplage. La seule solution que nous trouvons dans la littérature et qui répond à la problématique posée par CANETTI et HOHENBERGER [2007] a été proposée par ZHANG et collab. [2013]. La première construction de ces derniers permet de rendre le crypto-système de Cramer-Shoup publiquement vérifiable et utilise leur résultat pour construire un PRE CCA sécurisé dans le modèle standard. Cependant, nous montrons que leur schéma est vulnérable.

Ensuite, nous proposons une nouvelle construction basée sur le crypto-système de Cramer-Shoup CRAMER et SHOUP [1998]. Nous expliquons notre construction et prouvons sa sécurité CCA.Il s'agit du premier PRE prouvé sécurisé contre les attaques CCA dans le modèle standard sans utiliser le couplage. Ce travail a été inspiré de CHOW et collab. [2010] et WANG et collab. [2009].

3.2 Sécurité des PRE

3.2.1 Définition de sécurité des PRE

Dans le chapitre 1, nous avons revu les différents objectifs, types et modèles conçus pour mesurer le niveau de sécurité des systèmes de chiffrement. Ainsi, l'utilisation de ces notions à travers un jeu de sécurité permet de modéliser l'attaque en question entre un adversaire et un challenger. L'adversaire a alors accès à plusieurs oracles détenus par le challenger qui simulent les différentes fonctions du schéma étudié. Nous trouvons généralement les oracles de génération de clés, de chiffrement et de déchiffrement.

Néanmoins, de nouvelles fonctions sont introduites pour les schémas PRE dont la fonction de re-chiffrement et celle de génération de clé de re-chiffrement. Il faut alors ajouter deux nouveaux oracles, sauf que l'oracle de re-chiffrement peut être utilisé pour simuler des déchiffrements. Puisqu'aucune restriction n'existe sur l'appel des oracles, un adversaire peut faire en sorte de demander le re-chiffrement d'un challenge en un message chiffré qu'il peut déchiffrer. L'utilisation du même principe de jeu ne peut donc pas être adapté aux PRE, à part si nous ajoutons certaines restrictions. Dans le cas contraire, gagner le jeu pour l'adversaire sera trivial. Nous introduisons alors la notion des dérivés du challenge qui sont définies par CANETTI et HOHENBERGER [2007] comme

suit :

Soit $(pk*, c*)$ le challenge, l'ensemble de ses dérivés est :

— Le challenge lui même $(pk*, c*)$.

— Si (pk_i, c_i) est dérivé de (pk_j, c_j) et (pk_j, c_j) est dérivé de $(pk*, c*)$ alors (pk_i, c_i) est dérivé de $(pk*, c*)$.

— Si l'adversaire émet une requête de re-chiffrement (pk_i, pk_j, c_i) et obtient c_j alors (pk_j, c_j) est dérivé de (pk_i, c_i).

— Si l'adversaire demande une clé de re-chiffrement (pk_i, pk_j) et le déchiffrement de $(pk_i, c_j) \in \{m_0, m_1\}$, alors (pk_j, c_j) est une dérivée de toutes les paires (pk_i, c)

Ainsi, le jeu de sécurité doit prendre en compte les dérivés du challenge et refuser toute requête les concernant. Un autre point important à prendre en compte est l'aspect multi-utilisateur des PRE. A la différence d'un système de chiffrement classique, le PRE concerne plusieurs utilisateurs à la fois et il se peut que l'attaquant soit un utilisateur ayant une paire de clés publique/privée. Au niveau du jeu de sécurité, cette clé sera forcément différente de la paire de clés utilisée lors du challenge, ce qui n'affecte pas le déroulement du jeu lorsqu'il s'agit d'un système de chiffrement classique. Mais, puisqu'il s'agit d'un système multi-utilisateurs et vu la nature du PRE, il est nécessaire d'ajouter des oracles supplémentaires. Ces oracles fournissent à l'adversaire des clés, qui peuvent être soit honnêtes soit corrompues selon que l'adversaire ignore ou non la clé secrète correspondante.

3.2.2 Niveaux d'attaques sur les PRE

Maintenant que les différences majeures ont été présentées, nous pouvons reprendre les mêmes types d'attaques définis auparavant pour étudier la robustesse des PRE mais en ajoutant les nouveaux oracles et restrictions discutées ci-dessus.

Nous avons vu que les différentes phases d'accès à l'oracle de déchiffrement définissent le type d'attaque que l'adversaire peut mener. Si nous restreignons l'accès à cet oracle à la première phase du challenge, c'est-à-dire une fois le message du challenge reçu, alors nous cherchons à prouver la robustesse du crypto-système contre les attaques CCA-1. Toutefois, l'oracle de re-chiffrement peut faire objet d'un oracle de déchiffrement. NUÑEZ et collab. [2015] définissent alors un ensemble d'attaques paramétriques, basées sur la dernière étape de l'adversaire au cours de laquelle celui-ci a accès aux oracles de déchiffrement et de re-chiffrement.

Cet ensemble paramétrique est caractérisé par une paire d'indices $i, j \in \{0, 1, 2\}$. Ainsi, $\text{CCA}_{i,j}$ désigne une attaque où l'adversaire dispose d'un oracle de déchiffrement jusqu'à la phase i et d'un oracle de re-chiffrement jusqu'à la phase j comme le montre le tableau 3.1. ${\mathcal{O}_1}^{\text{CCA}}$ et ${\mathcal{O}_2}^{\text{CCA}}$ présentent respectivement les deux phases d'accès aux différents oracles. ${\mathcal{O}_1}^{\text{CCA}}$ correspond à l'ensemble d'oracles auxquels l'adversaire aura accès avant le challenge et ${\mathcal{O}_2}^{\text{CCA}}$ l'ensemble d'oracles accessibles par l'adversaire après le challenge.

Dans ce cas une attaque CPA correspond à une attaque $\text{CCA}_{0,0}$, CCA-1 correspond à $\text{CCA}_{1,1}$ et CCA-2 correspond à $\text{CCA}_{2,2}$. Il existe néanmoins des attaques intermédiaires. Les différentes combinaisons d'oracles disponibles produisent un ensemble paramétrique d'attaques, avec 9 choix possibles. La formulation utilisée dans la définition des attaques paramétriques utilise le terme "jusqu'à la phase" pour définir la disponibilité des oracles et non "en phase". Ainsi, la possibilité d'avoir un oracle en phase 2 mais pas en phase 1 n'est pas envisageable. Quant aux différentes attaques représentées dans le tableau, certaines ne sont pas forcément appropriées au contexte PRE. En effet, le re-chiffrement est généralement effectué par une entité semi-fiable en tant que service, tandis que le déchiffrement est effectué par des utilisateurs. Nous pouvons alors considérer que l'accès à un oracle de re-chiffrement par l'adversaire est plus facile qu'un accès à un oracle de déchiffrement. Par conséquent, les attaques incluant uniquement l'accès à l'oracle de déchiffrement restreignent la définition de sécurité et ne sont pas adéquates pour évaluer le niveau de sécurité

TABLEAU 3.1 – Différents niveaux d'attaques contre les PRE

Attaques	$\mathcal{O}_1^{\text{CCA}}$	$\mathcal{O}_2^{\text{CCA}}$
$\text{CCA}_{0,0}$	$\emptyset$	$\emptyset$
$\text{CCA}_{0,1}$	$\{\mathcal{O}_{reenc}\}$	$\emptyset$
$\text{CCA}_{0,2}$	$\{\mathcal{O}_{reenc}\}$	$\{\mathcal{O}_{reenc}\}$
$\text{CCA}_{1,0}$	$\{\mathcal{O}_{dec}\}$	$\emptyset$
$\text{CCA}_{2,0}$	$\{\mathcal{O}_{dec}\}$	$\{\mathcal{O}_{dec}\}$
$\text{CCA}_{1,1}$	$\{\mathcal{O}_{reenc}, \mathcal{O}_{dec}\}$	$\emptyset$
$\text{CCA}_{1,2}$	$\{\mathcal{O}_{reenc}, \mathcal{O}_{dec}\}$	$\{\mathcal{O}_{reenc}\}$
$\text{CCA}_{2,1}$	$\{\mathcal{O}_{reenc}, \mathcal{O}_{dec}\}$	$\{\mathcal{O}_{dec}\}$
$\text{CCA}_{2,2}$	$\{\mathcal{O}_{reenc}, \mathcal{O}_{dec}\}$	$\{\mathcal{O}_{reenc}, \mathcal{O}_{dec}\}$

d'un PRE, comme par exemple le cas de $\text{CCA}_{2,0}$ où l'adversaire a accès seulement à l'oracle de déchiffrement lors de la première phase. La figure 2.3 présente les différents types d'attaques ainsi que les relations d'implications entre ces derniers prouvées dans NUÑEZ et collab. [2015].

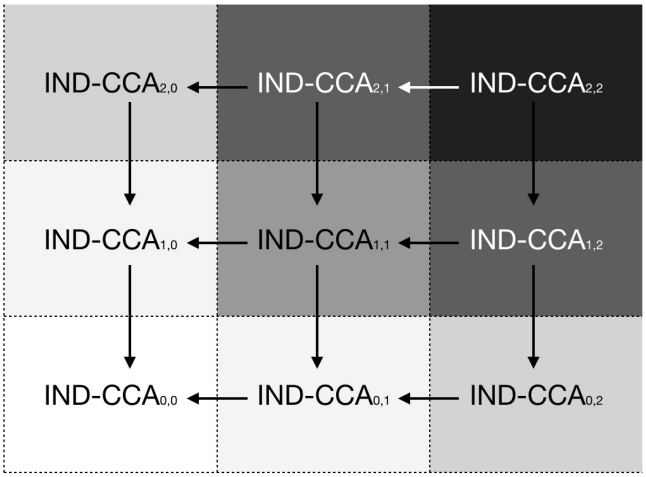


FIGURE 3.1 – Relations entre les différents niveaux de sécurité

Jusqu'à présent, le seul schéma de PRE unidirectionnel qui a été déclaré sûr contre les attaques IND – CCA sans s'appuyer ni sur le couplage, ni sur les oracles aléatoires est apparu dans ZHANG et collab. [2013]. Dans ce sens, les auteurs proposent un PVPKE (Public Verifiable Public Key Encryption) et utilisent leur résultat pour construire un schéma PRE. La prochaine section aborde la question de la vérifiabilité publique et son importance pour la construction de PRE sécurisé.

3.3 Vérifiabilité publique et PRE

3.3.1 Principe et méthode générique

La vérifiabilité publique est une propriété très utile aux systèmes de chiffrements. Elle permet de vérifier publiquement, avant le déchiffrement, la validité des messages chiffrés pour ne déchiffrer que ceux qui sont bien formés. Elle facilite ainsi la construction de crypto-systèmes tels que le chiffrement à seuil DE SANTIS et collab. [1994], les systèmes de partage de secret SHAMIR [1979] et aussi les proxy de re-chiffrement.

L'un des éléments majeurs qui a permis la création de crypto-systèmes robustes et publi-

quement vérifiables est la preuve de connaissance à divulgation nulle introduite par GOLDWASSER et collab. [1989]. En effet, les premières constructions de chiffrement asymétrique sécurisées contre les attaques CCA reposent principalement sur des preuves de connaissances à divulgation nulle non-interactive nommées NIZK (Non Interactive Zero-Knowledge).Une méthode générique permettant de passer d'un système sécurisé contre les attaques CPA vers une construction sécurisée CCA -1 a été proposée par NAOR et YUNG [1990]. L'idée est de générer deux clés de chiffrement, de calculer deux chiffrés correspondants au même message en utilisant ces clés, puis d'introduire une preuve NIZK affirmant que ces deux chiffrés correspondent effectivement au chiffrement du même message. Plus tard, Dolev, Dwork et Naor DDN et collab. [1991] ont montré comment modifier le paradigme de NY pour parvenir à une sécurité totale IND-CCA-2.

NIETO et collab. [2012] montre que nous pouvons obtenir un système de chiffrement publiquement vérifiable de deux manières différentes. La première consiste à utiliser les preuves NIZK à partir du paradigme de Naor-Yung cité auparavant. Cette transformation implique la vérifiabilité publique du message chiffré. De la même manière, la modification du paradigme de Naor-Yung par DDN et collab. [1991] offre aussi une vérifiabilité publique. La deuxième méthode consiste à utiliser la transformation de Canetti, Halevi et Katz (CHK) CANETTI et collab. [2004b]. La transformation CHK a été proposée afin de créer des systèmes de chiffrement sécurisés contre CCA à partir d'un IBE et d'une OTS (One Time Signature). Le schéma résultant est un PKE publiquement vérifiable. Ces méthodes s'avèrent très importantes pour la construction de PRE CCA sûrs mais en même temps très coûteuses. En effet, l'astuce qui permet de créer un PRE sécurisé CCA est la vérifiabilité publique des messages chiffrés. Ainsi la première étape pour le proxy sera de vérifier la validité du message chiffré avant son re-chiffrement.

3.3.2 Exemple

Reprenons l'exemple du chiffrement ElGamal et essayons de lui appliquer la méthode de Naor-Yung. Pour ce faire, nous devons :

— créer deux clés publiques indépendantes $y, y' \in \mathbb{G}$, calculer deux chiffrés (c, c') où $c = (u, v) = (g^r, m \times y^r)$ et $c' = (u', v') = (g^{r'}, m \times (y')^{r'})$ avec $r, r' \xleftarrow{\$} \mathbb{Z}_q$.

— créer une preuve NIZK affirmant que (c, c') correspond effectivement au chiffrement du même message, ce qui conduit à prouver que nous avons connaissance des valeurs r, r' tel que $u = g^r, u' = g^{r'} et \frac{v}{v'} = \frac{y^r}{(y')^{r'}}$. Nous devons alors créer la preuve $\pi \leftarrow Prove(d, w)$ où $d = (y, (u, v), y', (u', v'))$ est la déclaration à prouver et $w = (r, r')$ le témoin. Nous pouvons utiliser un protocole Σ en lui appliquant la méthode heuristique de Fiat-Shamir pour créer notre preuve π. Ainsi $\pi = (\alpha, \beta) = ((\alpha_1, \alpha_2, \alpha_3), (\beta_1, \beta_2)) = ((g^s, g^{s'}, y^s \times (y')^{s'}), (s - h \times r, s' + h \times r'))$ avec $s, s' \xleftarrow{\$} \mathbb{Z}_q$ et $h = H(d||\alpha)$ où H est considéré comme étant un oracle aléatoire.

Nous constatons alors que notre message chiffré comprend neuf éléments $(u, v, u', v', \alpha_1, \alpha_2, \alpha_3, \beta_1, \beta_2)$. Cette instanciation est en soi peu intéressante, puisque nous pouvons obtenir la sécurité CCA dans le modèle standard avec des messages chiffrés plus courts, en utilisant le chiffrement de Cramer-Shoup CRAMER et SHOUP [1998]. Sauf que ce dernier ne permet pas d'obtenir une vérifiabilité publique.

CHOW et collab. [2010] et DENG et collab. [2008b] s'appuient sur le chiffrement d'ElGamal et la signature de Schnorr avec une légère modification pour obtenir la vérifiabilité publique et ainsi créer leurs PRE. Le premier DENG et collab. [2008b] est bidirectionnel tandis que CHOW et collab. [2010] propose un PRE unidirectionnel. La même méthode est utilisée pour la construction des deux PRE et rend également le chiffrement ElGamal CCA sûr et diffère principalement dans la manière de créer les clés de re-chiffrement. Les messages chiffrés sont plus courts et ne contiennent que 4 éléments. Sauf que la signature Schnorr utilisée est une preuve NIZK obtenue aussi par la transformation de Fiat et Shamir sur le schéma d'identification Schnorr interactif. Cette transformation conduit à l'utilisation d'un oracle aléatoire dans le modèle de sécurité. Nous décrivons

dans la prochaine section la construction de PRE proposée par CHOW et collab. [2010] à base de signature de Schnorr, ainsi que la vulnérabilité trouvée par SELVI et collab. [2017].

3.4 Analyse de sécurité du PRE de Chow

Le PRE proposé par CHOW et collab. [2010] repose principalement sur la construction élaborée par DENG et collab. [2008b]. Ce dernier est un PRE bidirectionnel et repose à son tour sur une légère modification du crypto-système Hashed ElGamal ainsi que l'utilisation d'une signature de Schnorr. Nous présentons d'abord l'algorithme de base, un rappel sur la signature utilisée, puis la modification qui lui a été apporté.

3.4.1 Hashed ElGamal

Rappel du crypto-système Hashed ElGamal

- KG$en()$:
  - Soit p un grand nombre premier et g un générateur de $\mathbb{Z}_p$. Soit H_1 et H_2 deux fonctions de hachage tel que $H_1 : \{0,1\}^{l_0} \times \{0,1\}^{l_1} \rightarrow \mathbb{Z}^*{}_p, H_2 : \mathbb{Z}_p \rightarrow \{0,1\}^{l_0+l_1}$. Les paramètres du système sont $(p, g, H_1, H_2, l_0, l_1)$.
  - Choisir $x \xleftarrow{\$} \mathbb{Z}$.
  - Calculer $y = g^x$. La paire de clé publique/privée est respectivement $Pk = y$ et $Sk = x$.
- E$ncrypt(Pk, m)$:
  - Choisir $w \xleftarrow{\$} \{0,1\}^{l_1}$.
  - Calculer $r = H_1(m, w)$.
  - Calculer $E = g^r$.
  - Calculer $F = H_2(Pk^r) \oplus (m||w)$.
  - Retourner $C = (E, F)$.
- D$ecrypt(Sk, C)$:
  - Calculer $m||w = F \oplus H_2(E^{Sk})$ puis tester si $E = g^{H_1(m,w)}$
  - Si cette condition ne tient pas, retourner $\perp$ sinon retourner m

Nous pouvons remarquer que la clé publique est intégrée dans la fonction de hachage H_2 et le résultat est utilisé comme masque jetable pour calculer le deuxième élément du message chiffré. Ceci rend le système impossible à transformer en un PRE puisque le proxy ne pourra pas modifier le masque d'une clé publique Pk_a vers Pk_b, d'où l'intérêt de modifier l'algorithme de chiffrement et, par inclusion, le déchiffrement aussi.

Modification de Hashed ElGamal

- E$ncrypt(Pk, m)$:
  - Choisir $w \xleftarrow{\$} \{0,1\}^{l_1}$.
  - Calculer $r = H_1(m, w)$, $E = Pk^r$ et $F = H_2(g^r) \oplus (m||w)$.
  - Retourner $C = (E, F)$.
- D$ecrypt(Sk, C)$:
  - Calculer $m||w = F \oplus H_2(E^{\frac{1}{Sk}})$ puis tester si $E = g^{H_1(m,w)}$
  - Si cette condition ne tiens pas retourner $\perp$ sinon retourner m

La modification apportée n'altère pas la sécurité du système et permet d'avoir un masque jetable qui ne dépend plus de la clé publique. Ainsi, le premier élément du message chiffré peut être transformé de manière à ce que le masque jetable puisse être calculé sans utiliser la clé publique de départ. Cependant cette transformation à elle seule ne peut pas aboutir directement comme base pour la construction d'un PRE sécurisé contre les attaques IND-PRE-CCA. Ceci est dû principalement au fait que la validité du message chiffré ne peut pas être vérifié publiquement. En utilisant le même principe de re-chiffrement utilisé dans BBS, la clé de re-chiffrement correspond à $rk_{a\to b} = \frac{Sk_b}{Sk_a}$. Afin de re-chiffrer un message chiffré $C_a = (E, F)$, le proxy calcule $C_b = (E^{rk_{a\to b}}, F)$. En supposant que nous avons un adversaire A, ayant reçu le challenge $C^* = (g^{xr^*}, H_2(g^{r^*}) \oplus (m_\Omega || w^*))$ chiffré avec la clé publique Pk^*, ce dernier peut gagner le jeux IND-PRE-CCA de la manière suivante :

Attaque IND-PRE-CCA

— Choisir $z \xleftarrow{\$} \{0,1\}^{l_0+l_1}$.

— Calculer $C' = (E', F') = (E^*, F^* \oplus z) = (g^{xr^*}, H_2(g^{r^*}) \oplus (m_\Omega || w^*) \oplus z)$.

— Demander une clé corrompue à l'oracle de génération de clé et récupèrer la paire $(pk', sk') = (g^{x'}, x')$.

— Envoyer une requête de re-chiffrement à l'oracle pour obtenir le re-chiffrement de C' vers la nouvelle clé $C'' = (E'', F'') = (g^{x'r^*}, H_2(g^{r^*}) \oplus (m_\Omega || w^*) \oplus z)$.

■ Maintenant, l'adversaire A peut procéder au déchiffrement de C" en utilisant la clé secrète Sk'. Il peut alors calculer en premier $F'' \oplus H_2(E''^{(\frac{1}{Sk'})}) = (m_\Omega || w^*) \oplus z$. Il suffit d'éliminer z avec l'opérateur xor et vérifier à quoi correspond m_Ω pour gagner le jeux IND-CCA.

3.4.2 Signature de Schnorr

Si le Proxy pouvait vérifier la validité du message chiffré, le message soumis à l'oracle de re-chiffrement n'aurait pas pu être transformé. En effet, l'oracle de déchiffrement après avoir reçu la demande de déchiffrement du message créé par l'adversaire C" avant ou après le re-chiffrement retournera $\perp$, l'oracle de re-chiffrement permet à l'adversaire de déchiffrer le message sans passer par l'oracle de déchiffrement.

Afin de rendre Hashed ElGamal publiquement vérifiable, DENG et collab. [2008b] ont utilisé une variation de la signature de Schnorr que nous décrivons ci-dessous. L'intégration de la signature n'implique pas l'ajout d'une paire de clés privée/publique pour la signature et la vérification. En effet, l'ajout d'une nouvelle paire de clés permettra aux attaquants de soumettre aux oracles de re-chiffrement et déchiffrement des messages valides signés avec des clés différentes que de celles utilisées pour le chiffrement.

Signatur de Schnorr

— $KGen()$:

  — Soit p un grands nombre premier et g un générateur de $\mathbb{Z}_p$. Soit H une fonction de hachage tel que $H : \{0,1\}^* \to \mathbb{Z}^*{}_p$. Les paramètres du système sont (p, g, H).

  — Choisir $x \xleftarrow{\$} \mathbb{Z}$.

  — Calculer $y = g^x$. La paire de clé publique/privée est respectivement $Pk = y$ et $Sk = x$.

— $Sign(Sk, m)$:

  — Choisir $u \xleftarrow{\$} \mathbb{Z}$.

  — Calculer $D = g^u$, $e = H(m, D)$ et $s = u + Sk.e \ mod \ p$.

  — Retourner $\pi = (e, s)$.

La variation de la signature de Schnorr proposée par DENG et collab. [2008b] utilise le même algorithme pour la génération des clés et les mêmes calculs pour la génération de la signature sauf que la signature retournée est (D, s) au lieu de (e, s).

— $Verify(Pk, \pi = (e, s), m)$:
  — Calculer $D' = g^s Pk^{-e}$
  — Calculer $e' = H(m, D')$ puis tester si $e = e'$
  — Si cette condition ne tient pas, retourner $\perp$, sinon retourner 1.

3.4.3 PRE de DENG et collab. [2008b]

Les deux schémas présentés précédemment ont été combinés par DENG et collab. [2008b] afin de construire le premier PRE bidirectionnel CCA sans utiliser le couplage. Afin de générer une signature permettant de vérifier la validité du message chiffré sans incorporer une nouvelle paire de clés, les auteurs utilisent la clé publique comme générateur de groupe. Ainsi la paire de clés de signature et de vérification correspond au jeton aléatoire r et au masque jetable Pk^r. Le message chiffré avec Hashed ElGamal comporte les deux éléments (E, F). E est alors considéré comme étant la clé publique et F le message à signer. Le PRE résultant est le suivant :

- $Setup(1^k)$: Choisir deux nombre premier p et q tels que $q/p-1$ où q est de taille $k-bits$. Soit g un générateur du groupe $\mathbb{G}$ tel que $\mathbb{G}$ est un sous-groupe de $\mathbb{Z}^*{}_q$ d'ordre q.
  On choisit trois fonctions de hachage : $H_1 : \{0,1\}^{l_0} \times \{0,1\}^{l_1} \rightarrow \mathbb{Z}^*{}_q, H_2 : \mathbb{G} \rightarrow \{0,1\}^{l_0+l_1}, H_3 : \{0,1\}* \rightarrow \mathbb{Z}^*{}_q$. L'espace des messages M est $\{0,1\}^{l_0}$, où $l_0 = l_1 = k$. Les paramètres du système sont : $params = (q, \mathbb{G}, g, H_1, H_2, H_3, l_0, l_1)$.
- $KeyGen(params)$:
  — Choisir $x_a \xleftarrow{\$} \mathbb{Z}^*{}_q$.
  — Calculer $Pk_a = g^{x_a}$.
  — Retourner $Sk_a = x_a$ & $Pk_a = g^{x_a}$.
- $RekeyGen(Sk_a, Sk_a)$:
  — Définir $Rk_{a\rightarrow b} = \dfrac{x_b}{x_a}$
  — Retourner $Rk_{a\rightarrow b}$.

Les paramètres du système sont les mêmes que ceux utilisés pour Hashed ElGamal, à l'exception d'une fonction de hachage qui a été ajoutée pour calculer la signature. Quant au chiffrement, nous reprenons le même processus en ajoutant la signature qui permet de vérifier la validité du message chiffré.

- $\xi^{asym}_{Pk_a}(m)$:
  — Choisir $u \xleftarrow{\$} \mathbb{Z}^*{}_q, w \xleftarrow{\$} \{0,1\}^{l_1}$.
  — Calculer $r = H_1(m, w)$.
  — Calculer $D = Pk_a{}^u$,$E = Pk_a{}^r$.
  — Calculer $E = Pk_a{}^r$.
  — Calculer $F = H_2(g^r) \oplus (m||w)$.
  — Calculer $s = u + r \times H_3(D, E, F) \bmod(q)$.
  — Retourner $C_a = (D, E, F, s)$.

Il faut noter que la signature ne permet pas de vérifier l'identité de l'émetteur. En effet, elle permet de vérifier si l'émetteur du message chiffré connaît le jeton aléatoire utilisé pour générer le masque jetable avec lequel le message a été chiffré. Ainsi, pour chaque message chiffré, la paire de clés de signature/vérification sera différente.

- $ReEnc(C_a, Pk_a, Pk_b, Rk_{a\to b})$:
  - Vérifier si : $D.E^{H_3(D,E,F)} = Pk_a{}^s$ sinon retourner $\perp$.
  - Calculer $E' = E^{rk}$
  - Retourner $C'_b = (E', F) = (g^{r.x_b}, H_2(g^r) \oplus (m||w))$.
- $Dec(C_a, Sk_a)$:
  (Message chiffré original)
  - Vérifier si : $D.E^{H_3(D,E,F)} = (g^{x_a})^s$ sinon retourner $\perp$.
  - Calculer $(m||w) = F \oplus H_2(E^{\frac{1}{x_a}})$
  - Retourner m si : $E = (g^{x_a})^{H_1(m||w)}$, sinon retourner $\perp$.

  (Messages re-chiffrés)
  - Calculer $(m||w) = F \oplus H_2(E'^{\frac{1}{x_a}})$.
  - Si $E'=g^{(x_a \times H_1(m,w))}$ retourner m , sinon retourner $\perp$.

Ce PRE est prouvé sécurisé contre les attaques CCA dans DENG et collab. [2008b]. Quant aux propriétés de ce dernier, nous pouvons vérifier facilement qu'il n'est pas résistant aux collusions. Une deuxième version du même algorithme a été proposée par SELVI et collab. [2017] qui permet de rendre le schéma non transitif. Certaines modifications ont été apportées afin de modifier la fonction de génération des clés de re-chiffrement et par conséquent le re-chiffrement. Le schéma résultant reste cependant toujours vulnérable aux collusions.

3.4.4 De DENG et collab. [2008b] à CHOW et collab. [2010]

CHOW et collab. [2010] se base sur le PRE présenté ci-dessus afin de construire un PRE unidirectionnel qui est prouvé CCA sécurisé. La modification apportée consiste à incorporer une nouvelle paire de clés et à utiliser une nouvelle fonction de hachage. Ainsi, comme cela a été présenté dans le chapitre 2, les clés publique et privée comportent chacune deux éléments. Ces derniers permettent de créer une clé de re-chiffrement unidirectionnel.

- $KeyGen(params)$:
  - Choisir $x_{a,1}, x_{a,2} \xleftarrow{\$} \mathbb{Z}^*{}_q$.
  - Calculer $Pk_{a,1} = g^{x_{a,1}}$, $Pk_{a,2} = g^{x_{a,2}}$
  - Retourner $Sk_a = (x_{a,1}, x_{a,2})$ et $Pk_a = (Pk_{a,1}, Pk_{a,2})$.
- $RekeyGen(Sk_a, Pk_a, Pk_b)$:
  - Choisir $h \xleftarrow{\$} \{0,1\}^{l_0}, \pi \xleftarrow{\$} \{0,1\}^{l_1}$.
  - Calculer $v = H_1(h, \pi), V = Pk^v{}_{b,2}$ et $W = H_2(g^v) \oplus (h||\pi)$.
  - Définir $rk = \dfrac{h}{x_{a,1} H_4(Pk_{a,2}) + x_{a,2}}$
  - Retourner $Rk_{a\to b} = (rk, V, W)$.

La clé de re-chiffrement correspond à un message chiffré avec Hashed ElGamal en utilisant la clé publique du délégué. Le message chiffré est un jeton aléatoire h utilisé pour calculer un masque jetable qui sert à re-chiffrer et déchiffrer les messages transformables. Une collusion entre un uti-

lisateur et un proxy corrompu permet de calculer $x_{a,1}\mathrm{H}_4(\mathrm{P}k_{a,2}) + x_{a,2}$, ce qui ne révèle aucune information sur la clé secrète du délégataire.

En conséquence des modifications apportées à la génération des clés de re-chiffrements, le chiffrement est légèrement différent du PRE de DENG et collab. [2008b]. Les deux premiers éléments du message chiffré pour les messages transformables sont calculés à partir des deux éléments de la clé publique comme suit : $\mathrm{D} = (\mathrm{P}k_{a,1}^{\mathrm{H}_4(\mathrm{P}k_{b,2})}\mathrm{P}k_{a,2})^{u}$ et $\mathrm{E} = (\mathrm{P}k_{a,1}^{\mathrm{H}_4(\mathrm{P}k_{b,2})}\mathrm{P}k_{a,2})^{r}$. Ainsi, le re-chiffrement ne se fera pas directement vers la clé publique du délégué mais plutôt vers le jeton aléatoire créé par la fonction de génération des clés de re-chiffrement : $\mathrm{E}' = \mathrm{E}^{rk} = g^{r.h}$. Le délégué devrait déchiffrer le message chiffré inclus dans la clé de re-chiffrement avec sa clé secrète afin de pouvoir utiliser h pour le déchiffrement.

3.4.5 Vulnérabilité du PRE de CHOW et collab. [2010]

SELVI et collab. [2017] ont montré qu'il existe une faiblesse dans la preuve de sécurité du PRE de CHOW et collab. [2010] pour lequel la preuve est à base de simulations. Comme cela a été présenté dans le chapitre 1 le but de l'adversaire dans le cas d'une preuve de simulation est de distinguer entre la simulation et le système réel. La vulnérabilité concerne la vérification de validité des messages chiffrés transformables par le proxy. En effet, le challenger fournit les paramètres du système, puis l'adversaire soumet ses requêtes de déchiffrement / re-chiffrement aux oracles jusqu'à la fin de la phase-1 où l'adversaire reçoit le challenge.

Le challenge correspond à un message chiffré $\mathrm{C} = (\mathrm{D}, \mathrm{E}, \mathrm{F}, s)$ en utilisant une clé publique communiquée à l'adversaire (par exemple $\mathrm{P}k_a$). Maintenant l'adversaire peut exploiter la vulnérabilité en question pour distinguer la simulation du système réel et procède comme suit :

L'adversaire commence par générer un message chiffré invalide avec la fonction de chiffrement suivante en utilisant la même clé publique du challenge et le message m_0 ou m_1 :

- $f\xi_{\mathrm{P}k_a}^{asym}(m_0)$:
  - Choisir $u_f \xleftarrow{\$} \mathbb{Z}^*_q$.
  - Calculer $\mathrm{D}_f = (\mathrm{P}k_{a,1}^{\mathrm{H}_4(\mathrm{P}k_{a,2})}\mathrm{P}k_{a,2})_f^{u}$.
  - Choisir $r^* \xleftarrow{\$} \mathbb{Z}^*_q$.
  - Choisir $w_f \xleftarrow{\$} \{0,1\}^{l_1}$ et calculer $r_f = \mathrm{H}_1(m_0, w_f)$.
  - Calculer $\mathrm{E}_f = (\mathrm{P}k_{a,1}^{\mathrm{H}_4(\mathrm{P}k_{a,2})}\mathrm{P}k_{a,2})^{r^*}$.
  - Choisir $\mathrm{F}^* \xleftarrow{\$} \{0,1\}^{l_0+l_1}$
  - Calculer $s_f = u_f + r\mathrm{H}_3(\mathrm{D}_f, \mathrm{E}_f, \mathrm{F}^*)$
  - Retourner $\mathrm{C}_f = (\mathrm{D}_f, \mathrm{E}_f, \mathrm{F}, s_f)$.

Si l'adversaire soumet à l'oracle de déchiffrement le message chiffré invalide retourné par la fonction présentée ci-dessus, la simulation ainsi que le système réel vont retourner $\perp$. Cependant, l'algorithme réel de re-chiffrement va re-chiffrer le message chiffré invalide puisque la condition $\mathrm{D}.\mathrm{E}^{\mathrm{H}_3(\mathrm{D},\mathrm{E},\mathrm{F})} = (\mathrm{P}k_{a,1}^{\mathrm{H}_4(\mathrm{P}k_{a,2})}\mathrm{P}k_{a,2})^{s}$ est vérifiée. L'oracle de re-chiffrement (correspond à la simulation) vérifie la validité du message chiffré en utilisant la condition précédente, en plus, il vérifie s'il existe un tuple (m_0, w, r) tel que $(\mathrm{P}k_{a,1}^{\mathrm{H}_4(\mathrm{P}k_{a,2})}\mathrm{P}k_{a,2})^{r} = \mathrm{E}$. Toutefois, le tuple m_0, w, r n'existe pas puisque c'est r^* qui a été utilisé pour calculer E. Or, la valeur de r^* est tirée aléatoirement et ne correspond pas à $\mathrm{H}_1(m, w)$. Ainsi, l'oracle de re-chiffrement retourne $\perp$, ce qui implique que l'adversaire peut distinguer entre le système réel et la simulation. Pour remédier à cette vulnérabilité, les auteurs de SELVI et collab. [2017] ont ajouté une deuxième condition permettant de vérifier la validité du message chiffré qui prend en compte la valeur de r. La modification apportée est

présentée dans le chapitre 2. En ce qui concerne le fonctionnement des oracles dans la preuve de sécurité, il se trouvent dans l'annexe de CHOW et collab. [2010]. La prochaine section se concentre sur les PRE prouvés sécurisés dans le modèle standard.

3.5 PRE dans le modèle standard

Depuis la sortie des travaux de CANETTI et HOHENBERGER [2007], plusieurs constructions ont été réalisées afin de créer des PRE sécurisés contre les attaques CCA dans le modèle standard SHAO et CAO [2009] LIBERT et VERGNAUD [2008] ZHANG et collab. [2013]. Dans SHAO et CAO [2009], les auteurs proposent un PRE à base du crypto-système de Cramer-Shoup CRAMER et SHOUP [1998] sans vérifiabilité publique. Cette propriété étant importante pour la construction des PRE, son absence implique l'existence d'une vulnérabilité dans leur système. Quant aux travaux de LIBERT et VERGNAUD [2008] le PRE proposé reste le plus performant en termes de taille de clé et la sécurité du système est prouvée dans le modèle standard, mais repose sur une relaxation du jeu CCA appelé RCCA (Replayable-CCA). L'IND-RCCA est identique au IND-CCA, à l'exception que l'oracle de déchiffrement, à chaque fois que nous lui demandons de déchiffrer tout texte chiffré qui résulte en m_0 ou m_1, il retourne un symbole $\perp$, même si ce texte chiffré est différent du challenge C. Le dernier schéma de ZHANG et collab. [2013] a été fourni sans preuve de sécurité. Les auteurs prétendent que leur proposition est sécurisée contre les attaques IND-CCA dans le modèle standard. Nous étudions la sécurité du système dans la prochaine section et présentons notre construction.

3.5.1 Analyse du PVPKE proposé par ZHANG et collab. [2013]

Le schéma de ZHANG et collab. [2013] est basé sur le système de chiffrement de Cramer-Shoup. Une modification majeure a été apportée afin de le rendre publiquement vérifiable. Les auteurs ont opté pour l'utilisation de groupes d'ordres composites. Leur construction repose ainsi sur les deux hypothèses suivantes :

— *L'hypothèse RSA* : soit p et q deux grands nombres premiers, $\mathrm{N} = pq$, $\phi(\mathrm{N}) = (p-1)(q-1)$ et e choisi aléatoirement tel que $pgcd(e, \phi(\mathrm{N})) = 1$. Etant donné $x \in \mathbb{Z}_{\mathrm{N}}^*$, il est difficile de calculer y tel que $y^e = x \ mod \ n$.

— *L'hypothèse de décision de diffie-hellman* : soit p et q deux grands nombres premiers, $\mathrm{N} = pq$, $\phi(\mathrm{N}) = (p-1)(q-1)$ et g choisi aléatoirement $g \in \mathbb{Z}_{\mathrm{N}}^*$ tel que $g^{\phi}(\mathrm{N}) = 1 \ mod$ N. Etant donné deux éléments aléatoires $g, h \in \mathbb{Z}_{\mathrm{N}}^*$, $g^a \ mod$ N et $h^b \ mod$ N, il est difficile de décider si $a = b \ mod \ \phi(\mathrm{N})$.

En se basant sur ces deux hypothèses, les auteurs transforment le chiffrement de Cramer-Shoup en un système de chiffrement publiquement vérifiable. Le crypto-système est sécurisé contre les attaques CPA mais pas CCA comme les auteurs l'avaient prétendu. Nous passons en revue le schéma dû à ZHANG et collab. [2013], et montrons comment réaliser une attaque adaptative à texte chiffré choisi ci-dessous.

Rappel du crypto-système

— KG$en()$:

— Soit p, q , p' et q' de grands nombres premiers tels que $p = 2 \times p' + 1, q = 2 \times q' + 1$ et $N = p \times q$

— Choisir g_1, g_2 de $\mathbb{Z}_N$ tel que $g_i^{\phi(N)} \equiv 1 \bmod N$ (i = 1, 2)

— Choisir $b \xleftarrow{\$} \mathbb{Z}_{\phi(N)}$ et $x_i \xleftarrow{\$} \mathbb{Z}_{\phi(N)}$ (i = 1, 2, 3, 4, 5)

— Choisir fonction de hachage $H : \{0,1\}^* \rightarrow \mathbb{Z}_N$.

— Calculer $x'_i \equiv b \times x_i \bmod \phi(N)$ (i = 1, 2, 3, 4).

— Calculer $c = g_1^{x_1} \times g_2^{x_2}$, $d = g_1^{x_3} \times g_2^{x_4}$, $h = g_1^{x_5}$

— Retourner $Sk = (p', q', x_1, x_2, x_3, x_4, x_5)$ et $Pk = (N, g_1, g_2, H, b, x'_1, x'_2, x'_3, x'_4, c, d, h)$

— E$ncrypt(Pk, m)$:

— Choisir $r \xleftarrow{\$} \mathbb{Z}_{\mathbb{N}}$

— Calculer $u_1 = g_1^r$, $u_2 = g_2^r$.

— Calculer $e = m \times h^r \bmod N$, $\alpha = H(u_1, u_2, e)$, $v = c^r \times d^{r \times \alpha} \bmod N$

— Retourner $C = (u_1, u_2, e, v)$

— D$ecrypt(Sk, C)$:

— Calculer $\alpha = H(u_1, u_2, e)$ puis tester si $v^b = u_1^{x'_1 + \alpha \times x'_3} \times u_2^{x'_2 + \alpha \times x'_4} \bmod N$

— Si cette condition ne tient pas retourner "rejet" sinon retourner $m = \frac{e}{u_1^{x_5}}$

Dans cette section, nous démontrons que le PVPKE proposé par ZHANG et collab. [2013] n'est pas sécurisé contre les attaques CCA-2, ce qui implique que son utilisation pour concevoir le PRE n'est pas non plus sûre. Nous pouvons facilement le prouver en nous basant sur le jeu IND-CCA-2 . Rappelons que le jeu peut être vu comme deux phases, la première donne à l'attaquant l'accès à une clé publique fixe et à des oracles de déchiffrement. L'adversaire peut soumettre un grand nombre de requêtes de déchiffrement sans aucune restriction. Vient ensuite le défi qui concerne la distinction entre deux messages chiffrés créés par le challenger. Ces chiffrés correspondent à deux messages choisis par l'attaquant et chiffrés sous la même clé publique, par exemple l'attaquant envoie m_0 & m_1 et reçoit : $C^* = (u_1, u_2, e, v)$ qui correspond au chiffrement de m_i avec $i \xleftarrow{\$} \{0,1\}$. Dans la deuxième phase, l'adversaire peut soumettre des requêtes de déchiffrement à l'oracle, sauf pour le défi $C^* = (u_1, u_2, e, v)$.

L'attaque consiste à calculer un message chiffré C' non valide mais uniformément distribué de telle sorte que $C' \neq C^*$ et dont l'oracle de déchiffrement ne rejettera pas la demande puisque la vérification passera. Le texte chiffré non valide pourrait être construit de la manière suivante :

- $C' = (u_1^b, u_2^b, e^b, v' = u_1^{(x'_1 + x'_3 \times \alpha)} \times u_2^{(x'_2 + x'_4 \times \alpha)})$.
  L'oracle de déchiffrement vérifie la signature :
- $v'^b = u_1'^{(x'_1 + x'_3 \times \alpha)} \times u_2'^{(x'_2 + x'_4 \times \alpha)} = (u_1^{(x'_1 + x'_3 \times \alpha)} \times u_2^{(x'_2 + x'_4 \times \alpha)})^b$ qui est valide.
  Ainsi le résultat du déchiffrement sera :
- $m' = e' / u_1'^{x_5} = m^b \times h^{r \times b} / g_1^{x'_5 \times r} = m^b$.
  Maintenant, l'adversaire doit seulement tester si :
- $m_0^b = m^b$ or $m_1^b = m^b$ et gagner le défi.

Les autres schémas proposés dans ZHANG et collab. [2013], pourraient également être cassés par la même attaque. Prenons par exemple le PRE qu'ils ont créé à base du chiffrement publiquement vérifiable que nous venons d'analyser, le système PRE résultant n'est autre qu'une extension du PVPKE proposé avec de légères modifications. Les modifications ont été apportées au

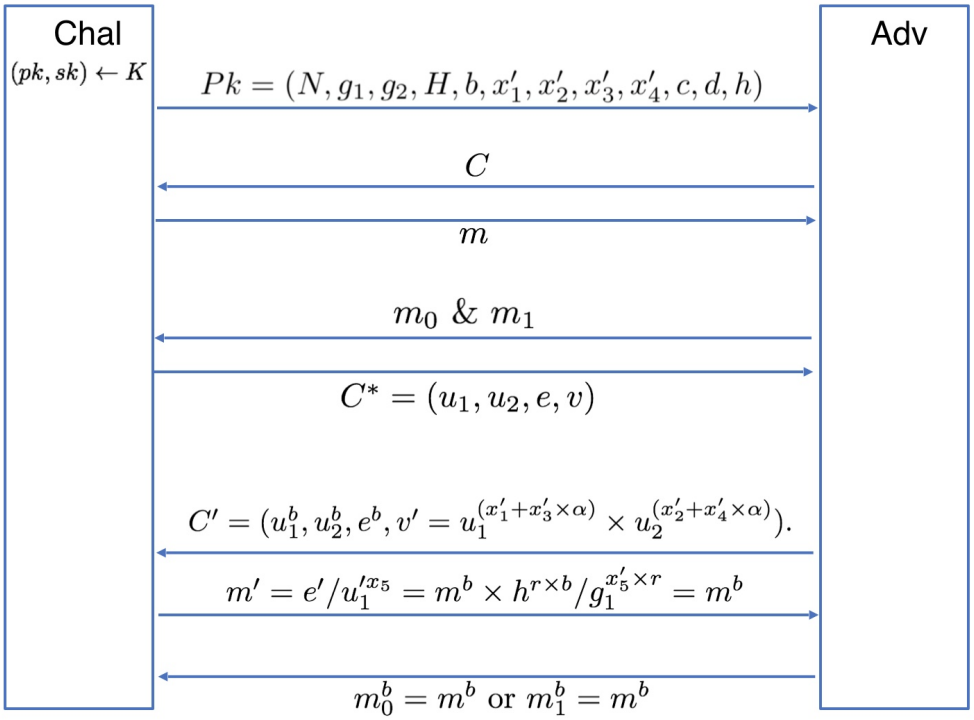


FIGURE 3.2 – jeu IND-CCA-2 pour ZHANG et collab. [2013] PRE

niveau des générations de clés, où elles ont été étendues. Une clé privée ainsi que sa publique correspondante a été ajouté, à savoir x_6 comme nouveau élément à la clé privée et $g_1^{x_6}$ pour la clé publique. Cependant le chiffrement reste quasiment identique, la seule différence étant la clé publique utilisée lors du chiffrement. Cette clé n'est autre que le hachage du produit scalaire de l'élément de la clé publique utilisée auparavant avec l'élément ajouté. Mais la signature permettant de vérifier la validité du message chiffré reste identique, elle dépend des mêmes éléments de la clé publique utilisée dans le PVPKE et du même principe de calcul. Ainsi, la vérification de validité peut être détournée de la même manière. Concernant le re-chiffrement et la génération des clés de re-chiffrement, le principe est similaire à celui utilisé par CHOW et collab. [2010] qui semble robuste. Cependant, puisque la vérification de validité des messages chiffrés est la même pour le re-chiffrement et/ou le déchiffrement, le crypto-système résultant sera faillible.

3.5.2 PRE avec paire de clés

En essayant de résoudre les problèmes ouverts présentés par CANETTI et HOHENBERGER [2007] , une nouvelle approche a été utilisée pour la création de PRE sécurisés contre les attaques CCA par WEI et collab. [2010] PURUSHOTHAMA et collab. [2013]. Nous considérons alors que le délégant, le proxy et le délégataire sont des entités ayant chacun leur propre paire de clés publique/privée. WEI et collab. [2010] étudient la possibilité et les avantages d'avoir un PRE avec sa propre paire de clés. Ils présentent deux constructions, la première est unidirectionnelle et utilise le couplage tandis que la deuxième est bidirectionnelle et repose sur le crypto-sysème de Cramer-Shoup. Le principe reste le même pour les deux constructions où chaque entité (délégant, proxy et délégataire) possède en plus de sa paire de clés de chiffrement une paire de clés pour la signature. La signature est utilisée pendant le chiffrement afin d'empêcher les adversaires de modifier le message chiffré original.

Néanmoins, le PRE bidirectionnel n'est pas entièrement sécurisé contre les attaques CCA, car aucune vérification n'est effectuée par le proxy sur la validité des messages chiffrés. Les auteurs affirment que leur système peut subir des attaques DDos, mais ne le présentent pas comme étant

une vulnérabilité contre les attaques CCA ce qui est pourtant le cas.

- Encrypt (m, pk) : Le chiffrement est en grande partie le même que dans Cramer-Shoup

$$u_1 = g_1^r, u_2 = g_2^r, e = \left[\mathrm{F}\left(h'\right)\right]_{l-l_1} \| \left(\left\lfloor \mathrm{F}\left(h'\right)\right\rfloor l_1 \oplus m\right)$$
$$\alpha = \mathrm{H}\left(u_1, u_2, e\right), v = c^r d^{r\alpha}$$

Le message chiffré est : (u_1, u_2, e, v)
- Re-Encrypt : Le proxy calcule :

$$u_1' = u_1^{\frac{z}{\left(z'+\mathrm{H}\left(pk_{proxy}\right)\right)}}, \sigma = \mathrm{sig.S}\left(sk_{\mathrm{proxy}}, \left(\mathrm{pk_{proxy}}, \mathrm{u_1'}, \mathrm{e}\right)\right)$$

Le message rechiffré est $\left(pk_{\mathrm{proxy}}, u_1', e, \sigma\right)$.

Nous remarquons que la signature publiquement vérifiable est créée au moment du re-chiffrement. Pour le texte chiffré original, seul le propriétaire de la clé privée peut vérifier sa validité. Ainsi, le proxy peut re-chiffrer n'importe quel message soumis sans aucune vérification préalable.

Contrairement à WEI et collab. [2010], au lieu d'avoir deux paires de clés distinctes pour le chiffrement et la signature, PURUSHOTHAMA et collab. [2013] proposent une nouvelle approche où le délégué et le délégataire gardent seulement leur propre paire de clés de chiffrement, tandis que le proxy utilise une paire de clés ayant la même distribution que celle du délégant et du délégataire. Leur construction fait appel à deux nouvelles fonctions ProxyPublish et Verify, utilisées pour le re-chiffrement et la vérification de validité des messages re-chiffrés. Le PRE proposé est unidirectionnel et repose sur ElGamal et la construction ATENIESE et collab. [2006].

Ces deux approches facilitent la construction de PRE sécurisé contre les attaques CCA. Au niveau des propriétés, l'unidirectionnalité peut être achevée, ainsi que la résistance aux collusions, la non-transitivité, la temporalité etc. La seule propriété que nous délaissons en utilisant cette approche est la transparence. Du fait que le proxy dispose de sa propre paire de clés publique et privée, le délégant et le délégataire auront connaissance de l'existence de la procuration.

Les PRE avec paires de clés ont pu résoudre une partie du problème ouvert de CANETTI et HOHENBERGER [2007], mais aucune des trois constructions proposées dans WEI et collab. [2010] et PURUSHOTHAMA et collab. [2013] n'arrivent à concevoir un PRE unidirectionnel qui soit CCA dans le modèle standard, sans utiliser le couplage.

3.5.3 Notre construction à base de Cramer-Shoup

Comme nous l'avons démontré, la vérifiabilité publique du PRE dans ZHANG et collab. [2013] n'est pas assurée. Afin de contourner ce problème, WEI et collab. [2010] ils considèrent le délégant, le proxy et le délégué comme des pairs, ayant leurs propres clés publiques/privées de chiffrement et leurs clés de signature/vérification. Ainsi, le proxy ne peut pas modifier le chiffré et les autres adversaires extérieurs ne peuvent pas modifier le chiffré original et le re-chiffré. L'idée d'utiliser des paires de clés au niveau du proxy nous a semblé intéressante. Dans notre cas, nous utilisons des clés publiques/privées de chiffrement qui permettent au proxy de vérifier la validité des messages chiffrés originaux et au délégué de tester la validité des messages re-chiffrés. Cette méthode peut être considérée comme contraignante en termes de transparence, mais plutôt avantageuse dans le sens où nous pouvons facilement détecter les proxies malveillants. En plus de la vérification de la validité des messages chiffrés et re-chiffrés, nous pouvons également vérifier la validité des clés de re-chiffrement, ce qui réduit les dommages causés par les attaques DDos.

Le schéma proposé repose sur la construction de Cramer-Shoup et procède comme suit :

— $Setup(1^\lambda)$: Soit $\mathbb{G}$ un groupe d'ordre premier q, de telle sorte que la taille du vecteur binaire représentant q est le paramètre de sécurité λ. Choisir des éléments aléatoires $g_1, g_2 \in \mathbb{G}$ et deux fonctions universelles de hachage à sens unique $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q$ et $H_2 : \mathbb{G}^2 \rightarrow \mathbb{Z}_q^*$. Les paramètres sont $params : (\mathbb{G}, q, g_1, g_2, H_1, H_2)$

Les éléments de groupe générés ainsi que les fonctions de hachage choisies sont les paramètres utilisés par tous les utilisateurs du système. Il est aussi possible que cela soit généré directement par les utilisateurs pour d'autres types d'implémentations par exemple lorsqu'il s'agit de systèmes pair à pair.

— $KGen(params)$: Notons que (Sk_a, Pk_a) la paire de clés privée/publique associée à l'utilisateur "a". Choisir $Sk_a = (x_i : i \in \{1,2,...,7\})$ où $x_i \xleftarrow{\$} \mathbb{Z}_q$ et établir $Pk_a = (c, d, h_1, h_2)$ tel que $c = g_1^{x_1} \times g_2^{x_2}$, $d = g_1^{x_3} \times g_2^{x_4}$,$h_1 = g_1^{x_5 + H_1(c,d) \times x_6}$, $h_2 = g_1^{x_7}$

La génération de clés prend en entrée les paramètres du système et produit une paire de clés publique/privée, la paire de clés du proxy est générée par cette même fonction.

— $RkGen(Sk_a, Pk_b, Pk_p)$:

— En entrée la clé privée d'un utilisateur "a" Sk_a et la clé publique de l'utilisateur "b" Pk_b et la clé publique du proxy Pk_p :

Choisir $j \xleftarrow{\$} \mathbb{Z}_q$, $k \xleftarrow{\$} \mathbb{Z}_q$

Calculer $rk = \frac{x_{a_5} + H_1(c,d) \times x_{a_6}}{k}$

Calculer $u_1' = g_1^j$, $u_2' = g_2^j$

Calculer $e' = h_{b1}^j \times k$, $\alpha_1' = H_2(u_1', u_2', e')$

Calculer $v' = c_b^j \times d_b^{j \times \alpha_1'}$, $\alpha_2' = H_2(\alpha_1', v')$, $v_p' = c_p^j \times d_p^{j \times \alpha_2'}$

Retourner $Rk_{a \rightarrow b} = (rk, u_1', u_2', e', v', v_p')$

La génération de clés de re-chiffrement est une combinaison entre la méthode utilisée dans Chow et collab. [2010] pour rendre l'algorithme de Cramer et Shoup [1998] unidirectionnel et le schéma de Wang et collab. [2009] avec une légère modification.

— $Encrypt(m, Pk_a)$:

— Pour un message chiffré non-transformable :

Choisir $r \xleftarrow{\$} \mathbb{Z}_q$

Calculer $u_1 = g_1^r$, $u_2 = g_2^r$

Calculer $e = h_{a2}^r \times m$, $\alpha = H_2(u_1, u_2, e)$

Calculer $v = c_a^r \times d_a^{r \times \alpha}$

Retourner $C_a = (u_1, u_2, e, v)$

— Pour un message chiffré transformable : ajouter la clé publique du proxy Pk_p en entrée :

Choisir $r \xleftarrow{\$} \mathbb{Z}_q$

Calculer $u_1 = g_1^r$, $u_2 = g_2^r$

Calculer $e = h_{a1}^r \times m$, $\alpha_1 = H_2(u_1, u_2, e)$

Calculer $v = c_a^r \times d_a^{r \times \alpha_1}$, $\alpha_2 = H_2(\alpha_1, v)$

Calculer $v_p = c_p^r \times d_p^{r \times \alpha_2}$

Retourner $C_a = (u_1, u_2, e, v, v_p)$

Le chiffrement se compose de deux méthodes : la première présente un chiffrement non transfor-

mable et garde le même algorithme que le chiffrement de Cramer-Shoup. La deuxième méthode utilise une entrée en plus qui est la clé publique du proxy. Elle permet de chiffrer le message de la même manière tout en rajoutant une signature vérifiable par le proxy.

— $ReEncrypt(Rk_{a\to b}, C_a, Pk_p, Sk_p)$:

— Entrées : la clé de re-chiffrement, un message chiffré transformable et la paire de clés privée/publique du proxy :

Vérifier si $v_p = u_1^{(x_{p_1}+x_{p_3}\times\alpha_2)} \times u_2^{(x_{p_2}+x_{p_4}\times\alpha_2)}$ & $v'_p = u_1'^{(x_{p_1}+x_{p_3}\times\alpha'_2)} \times u_2'^{(x_{p_2}+x_{p_4}\times\alpha'_2)}$

Choisir $\omega \xleftarrow{\$} \mathbb{Z}_q$

Calculer $\beta = u_1^{rk}$, $u''_1 = g_1^{\omega}$, $u''_2 = g_2^{\omega}$

Calculer $e'' = e$, $\alpha'' = H_2(u''_1, u''_2, e'', \beta)$

Calculer $v'' = c_b^{\omega} \times d_b^{\omega\times\alpha''}$

Retourner $C_b = (\beta, u'_1, u'_2, e', v', u''_1, u''_2, e'', v'')$

Si le message donné en entrée pour le re-chiffrement est un message transformable, le proxy utilise sa clé privée afin de vérifier sa validité. Une fois que la vérification est valide, le message re-chiffré est calculé. Ce dernier contient des éléments de la clé de re-chiffrement qui peuvent être omis si les clés de re-chiffrement sont stockées et que la source des messages re-chiffrés est connue par le délégué.

— $Decrypt(sk_b, C_b)$:

— Si $C_b = (u_1, u_2, e, v)$

Vérifier si $v = u_1^{(x_{b_1}+x_{b_3}\times\alpha)} \times u_2^{(x_{b_2}+x_{b_4}\times\alpha)}$

Calculer $m = \frac{e}{u_1^{x_{b_7}}}$

— Si $C_b = (u_1, u_2, e, v, v_p)$

Vérifier si $v = u_1^{(x_{b_1}+x_{b_3}\times\alpha_1)} \times u_2^{(x_{b_2}+x_{b_4}\times\alpha_1)}$

Calculer $m = \frac{e}{u_1^{(x_{b_5}+H_1(c,d)\times x_{b_6})}}$

— Si $C_b = (\beta, u'_1, u'_2, e', v', u''_1, u''_2, e'', v'')$

Vérifier si $v'' = u_1''^{(x_{b_1}+x_{b_3}\times\alpha'')} \times u_2''^{(x_{b_2}+x_{b_4}\times\alpha'')}$ & $v' = u_1'^{(x_{b_1}+x_{b_3}\times\alpha')} \times u_2'^{(x_{b_2}+x_{b_4}\times\alpha')}$

Calculer $k = \frac{e'}{u_1'^{(x_{b_5}+H_1(c,d)\times x_{b_6})}}$

Calculer $m = \frac{e''}{\beta^k}$

Avant le déchiffrement, nous vérifions la forme du message chiffré pour savoir si elle correspond à un message transformable, non-transformable ou re-chiffré. Quel que soit le cas, une vérification sur la validité du message chiffré est obligatoire avant de procéder au déchiffrement.

Exactitude et analyse de sécurité :

— L'exactitude du déchiffrement d'un message chiffré original (transformable ou non transformable) est triviale puisqu'elle est la même que dans Cramer-Shoup. L'exactitude du déchiffrement pour les messages re-chiffrés peut être vérifiée comme suit :

$$m = \frac{e''}{\beta^k} = \frac{e''}{u_1^{rk\times k}} = \frac{m\times g_1^{r\times(x_5+H_1(c,d)\times x_6)}}{g_1^{\frac{r\times(x_5+H_1(c,d)\times x_6)}{k}\times k}}$$

— Intuitivement, nous pouvons vérifier la sécurité IND-CCA de notre construction :

— Pour les messages chiffrés non-transformables, il s'agit d'un chiffrement de Cramer-Shoup CRAMER et SHOUP [1998] prouvé sécurisé contre les attaques CCA sous l'hypothèse DDH et la seconde pré-image.

— Pour les messages chiffrés transformables, le chiffrement est presque le même que celui de Cramer-Shoup. Cependant, nous calculons v_p afin que le proxy puisse vérifier la validité du texte chiffré. Dans le jeu IND CCA-2, nous donnons au challenger l'accès aux deux clés secrètes du mandataire et du délégant afin qu'il puisse vérifier la validité de v_p. Sinon, cela n'a pas d'effet sur la sécurité du système puisque v_p est calculée avec une autre clé publique. Elle sera donc linéairement indépendante de v même si nous utilisons le même jeton aléatoire.

— La génération des clés de re-chiffrement a été inspirée des travaux de CHOW et collab. [2010], où même si k est divulgué, il a été choisi au hasard pour calculer $rk = \frac{x_5 + \mathrm{H}_1(c,d) \times x_6}{k}$, donc lorsque le proxy et le délégué sont en collusion, seul $x_5 + \mathrm{H}_1(c,d) \times x_6$ peut être calculé. Cette combinaison linéaire empêche de trouver x_5 et x_6, du fait qu'il existe autant de solutions possibles que le cardinal du groupe $\mathbb{G}$ auquel x_i appartient. Ainsi, aucune information sur les clés privées n'est révélée, ce qui rend le système résistant aux collusions.

— Les messages re-chiffrés contiennent deux différents messages chiffrés de Cramer-Shoup, le premier est utilisé pour déchiffrer la clé de substitution k créée par le délégant qui est primordial pour le déchiffrement du second chiffré, comme nous l'avons vu précédemment.

3.5.4 Preuve de sécurité

Nous avons vu dans 2.2.2 que les PRE prouvés sécurisés, que ce soit dans le modèle standard ou dans le modèle des oracles aléatoires, se sont de plus en plus concrétisés après les travaux de CANETTI et HOHENBERGER [2007]. Ces derniers avaient formalisé le modèle de sécurité des schémas PRE à base du jeu CCA, où ils formulent par la suite la même définition dans le cadre de la sécurité universellement composable (UC). Nous trouvons par exemple DENG et collab. [2008b], LIBERT et VERGNAUD [2008], CHOW et collab. [2010] qui présentent leurs preuves de sécurité à base du modèle de jeu CCA proposé par CANETTI et HOHENBERGER [2007] .

Nous donnons d'abord la définition du jeu IND-CCA pour les PRE selon le modèle de Canetti CANETTI et HOHENBERGER [2007] . Nous prenons en compte les changements proposés par les auteurs, afin que cela soit adapté au PRE unidirectionnel puisque le modèle présenté était destiné au PRE bidirectionnel. Cette adaptation a été utilisée et généralisée pour la première fois par CHOW et collab. [2010] . Nous avons aussi apporté quelques modifications liées à l'ajout des paires de clés des proxy que nous discutons par la suite :

Soit λ le paramètre de sécurité. Soit $\mathscr{A}$ un oracle TM qui représente l'adversaire. Le jeu consiste en l'exécution de $\mathscr{A}$ avec les oracles suivants. Ils peuvent être invoqués plusieurs fois dans n'importe quel ordre, sous réserve des contraintes ci-dessous :

- $\mathscr{O}\mathrm{KG}en$: pour la génération d'une clé non-corrompue, l'oracle retourne $\mathrm{P}k$, où $(\mathrm{P}k, \mathrm{S}k) \leftarrow \mathrm{KG}en(params)$. Pour la génération d'une clé corrompue, l'oracle retourne $\mathrm{P}k$ et $\mathrm{S}k$ où $(\mathrm{P}k, \mathrm{S}k) \leftarrow \mathrm{KG}en(params)$.
- $\mathscr{O}\mathrm{R}k\mathrm{G}en$: à l'entrée de $\mathrm{P}k_a$, $\mathrm{P}k_b$ et $\mathrm{P}k_p$, l'algorithme de génération de clé de re-chiffrement retourne $\mathrm{R}k_{a\to b}$. Nous rejetons les requêtes de génération de clé de re-chiffrement entre une clé corrompue et une clé non-corrompue.
- $\mathscr{O}\mathrm{E}ncryption$: pour des messages chiffrés non-transformables , à l'entrée d'un message m, la sortie est $\mathrm{C} = (u_1, u_2, e, v)$. Pour un message chiffré transformable original , retourner $\mathrm{C} = (u_1, u_2, e, v, v_p)$.

- $\mathcal{O}Challenge$: cet oracle peut être appelé une seule fois. A l'entrée de, $(\mathrm{P}k^*, m_0, m_1)$, où $\mathrm{P}k^*$ est appelée la clé du challenge, l'oracle choisi un bit $b \xleftarrow{\$} \{0,1\}$ et retourne le message chiffré challenge $\mathrm{C} = \mathrm{E}nc(\mathrm{P}k, m_b)$. (Comme nous le notons plus loin, la clé de défi doit être intacte pour que $\mathscr{A}$ gagne).
- $\mathcal{O}\mathrm{ReE}ncryption$: à l'entrée de $(\mathrm{P}k_a, \mathrm{P}k_b, \mathrm{C}_a)$, Si $\mathrm{P}k_b$ est corrompu ou $a = b$ retourner $\perp$. Sinon, retourner C_b
- $\mathcal{O}\mathrm{D}ecryption$: à l'entrée de $(\mathrm{P}k, \mathrm{C})$, si $\mathrm{P}k$ n'a pas été généré avant retourne $\perp$. Sinon retourne $\mathrm{D}ecrypt(\mathrm{C}, \mathrm{S}k)$
- $\mathcal{O}\mathrm{D}ecision$: cet oracle peut être appelé une seule fois. A l'entrée de b' : si $b' = b$ et la clé de challenge pk^* est non-corrompue, alors retourne 1 sinon retourne 0.

Nous disons que $\mathscr{A}$ gagne le jeu IND-CCA avec un avantage ϵ si la probabilité que l'oracle de décision soit invoqué et produise 1, est au moins de $1/2 + \epsilon$, sur les choix aléatoires de $\mathscr{A}$ et des oracles.

Théorème 3.5.1 *Notre PRE est sécurisé contre les attaques adaptatives à textes chiffrés choisis en supposant que (1) les fonctions* H_1 *et* H_2 *sont deux fonctions universelles de hachage à sens unique, et (2) le problème de décision de Diffie-Hellman est difficile dans le groupe* $\mathbb{G}$ *.*

Nous donnons la preuve formelle de notre système sur la base des preuves WANG et collab. [2009] et CHOW et collab. [2010] comme suit :

Supposons qu'un adversaire externe $\mathscr{B}$ casse la propriété IND-CCA-2 de notre système, nous utilisons $\mathscr{B}$ afin de construire $\mathscr{A}$, pouvant distinguer si le tuple (g_1, g_2, u_1, u_2) de $\mathbb{G}$ est un tuple DDH ou pas. Les requêtes aux oracles de $\mathscr{B}$ sont traités par $\mathscr{A}$ comme suit :

— **Requête à** $\mathcal{O}\mathrm{KG}en$: si l'utilisateur a est corrompu, $\mathscr{A}$ choisi aléatoirement $\mathrm{S}k_a = (x_{a_i}) \xleftarrow{\$} \mathbb{Z}_q$ pour $(i = 1, 2, ..., 7)$, calcule $\mathrm{P}k_a = (g_1, g_2, c_a = g_1^{x_{a_1}} \times g_2^{x_{a_2}}, d_a = g_1^{x_{a_3}} \times g_2^{x_{a_4}}, h_{a_1} = g_1^{x_{a_5} + \mathrm{H}_1(c,d) \times x_{a_6}}, h_{a_2} = g_1^{x_{a_7}})$ retourne $\mathrm{S}k_a, \mathrm{P}k_a$ qui ont une distribution identique à la distribution réelle de la clé privée et publique réelle. Pour un utilisateur incorrompu b, $\mathscr{A}$ choisit aléatoirement $\mathrm{S}k_b = (x_{b_i}) \xleftarrow{\$} \mathbb{Z}_q$ pour $(i = 1, 2, ..., 9)$, calcule $\mathrm{P}k_b = (g_1, g_2, c_b = g_1^{x_{b_1}} \times g_2^{x_{b_2}}, d_b = g_1^{x_{b_3}} \times g_2^{x_{b_4}}, h_{b_1} = g_1^{x_{b_5} + \mathrm{H}_1(c,d) \times x_{b_6}} \times g_2^{x_{b_8}}, h_{a_2} = g_1^{x_{a_7}} \times g_2^{x_{b_9}})$ et retourne $\mathrm{P}k_b$. Supposant que $g_2 = g_1^w$ la sortie a une distribution identique à la distribution réelle de la clé publique réelle. Ce qui nous donne une simulation parfaite.

— **Requête à** $\mathcal{O}\mathrm{Rk}\mathrm{G}en$: à l'entrée de $\mathrm{P}k_a, \mathrm{P}k_b$ et $\mathrm{P}k_p$ si a ou b est corrompu nous rejetons la requête. Sinon $\mathscr{A}$ retourne $\mathrm{R}k_{a \to b} = (rk \xleftarrow{\$} \mathbb{Z}_{\phi(\mathrm{N})}), \mathscr{C}_{a_1}, v'_p)$ qui est indistinguable avec $\mathrm{R}k_{i \to j} = (\frac{(x_5 + \mathrm{H}_1(c,d) \times x_6)}{k}, \mathscr{C}_{i_1}, v'_p)$

— **Requête à** $\mathcal{O}\mathrm{E}ncryption$: pour un chiffrement non transformable, étant donné un message m, l'algorithme de chiffrement retourne $\mathrm{C} = (u_1, u_2, e, v) = (g_1^r, g_2^r, u_1^{x_7} \times u_2^{x_9} \times m, c^r \times d^{r \times \alpha})$ où $r \xleftarrow{\$} \mathbb{Z}_q$. C'est une simulation parfaite identique à celle du crypto-système de Cramer-Shoup. Pour un chiffrement original transformable, si les utilisateurs sont incorrompus l'algorithme de chiffrement retourne $\mathrm{C} = (u_1, u_2, e, v, v_p) = (g_1^r, g_2^r, u_1^{(x_5 + \mathrm{H}_1(c,d) \times x_6)} \times u_2^{x_8} \times m, c^r \times d^{r \times \alpha_1}, c'^r \times d'^{r \times \alpha_1})$ où $r \xleftarrow{\$} \mathbb{Z}_q$ et v_p est calculé à partir d'une clé publique aléatoire. Sinon, retourne $\perp$. C'est aussi une simulation parfaite. En se basant sur la même méthode utilisée dans CRAMER et SHOUP [1998], nous montrerons ci-après que nous ne pouvons pas construire un tuple valide (u_1, u_2, e, v, v_p) car (g_1, g_2, u_1, u_2) n'est pas un tuple DDH.

— **Requête à** $\mathcal{O}\mathrm{ReE}ncryption$: à l'entrée de $\mathrm{P}k_a, \mathrm{P}k_b, \mathrm{C}_a = (u_1, u_2, e, v, v_p)$ d'un utilisateur a à un utilisateur b, cherche dans la liste générée par $\mathcal{O}\mathrm{Rk}\mathrm{G}en$ un élément comprenant a et b. S'il n'existe pas, envoie la requête à $\mathcal{O}\mathrm{Rk}\mathrm{G}en$. Ensuite, le proxy vérifie la validité

du message chiffré en testant, si $v_p \neq u_1^{x_{p_1}} \times u_2^{x_{p_2}} \times u_1^{x_{p_3} \times \alpha_2} \times u_1^{x_{p_4} \times \alpha_2}$ retourne $\perp$. Sinon, retourne $C_j = (\beta, u_1', u_2', e', v', u_1'', u_2'', e'', v'') = (u_1^{(x_5 + H_1(c,d) \times x_6)}, u_1', u_2', e', v', u_1'', u_2'', e'', v'')$ qui inclus deux chiffrés Cramer-Shoup et ont la même distribution que pour $(u_1^{rk}, u_1', u_2', e', v', u_1'', u_2'', e'', v'')$. Ainsi, la sortie réelle et la sortie simulée sont indistinguables. Il s'agit donc également d'une simulation parfaite.

— **Requête à** $\mathcal{O}Decryption$: étant donnée un message re-chiffré $C = (\beta, u_1', u_2', e', v', u_1'', u_2'', e'', v'')$, l'algorithme de déchiffrement fonctionne comme suit. Il calcule $\alpha_1'' = H_2(u_1'', u_2'', e'', \beta)$, et vérifie si $v'' = u_1^{''(x_1 + x_3 \times \alpha_1'')} \times u_2^{''(x_2 + x_4 \times \alpha_1'')}$ & $v' = u_1^{'(x_1 + x_3 \times \alpha_1')} \times u_2^{'(x_2 + x_4 \times \alpha_1')}$. Si cette condition ne tient pas, l'algorithme de déchiffrement retourne $\perp$, sinon, calcule $k = \frac{e'}{u_1^{'(x_5 + H_1(c,d) \times x_6)}}$ et retourne $m = \frac{e''}{\beta^k}$. Dans notre simulation, à l'entrée de C_j d'un utilisateur i à j , $\mathscr{B}$ vérifie en premier la validité du message chiffré. Si c'est un message chiffré invalide retourne $\perp$, sinon calcule $k = \frac{e'}{u_1^{'(x_{j_5} + H_1(c,d) \times x_{j_6})} \times u_2^{'x_{j_8}}}$ puis retourne $m = \frac{e''}{\beta^k} = \frac{e''}{u_1^{1/(x_{j_5} + H_1(c,d) \times x_{j_6})}}$

Comme pour le chiffrement de Cramer-Shoup, si (g_1, g_2, u_1, u_2) est un tuple DDH, notre simulation de déchiffrement est un déchiffrement parfait.

Pour les messages chiffrés originaux, la preuve est la même que pour le chiffrement de Cramer-Shoup. Il y a une légère différence dans la preuve pour les messages chiffrés transformables originaux que nous expliquerons ci-dessous. Et la simulation de déchiffrement est un déchiffrement parfait.

Lemme 3.5.2 *Si (g_1, g_2, u_1, u_2) n'est pas un tuple DDH, $\mathcal{O}Decryption$ rejette tous les messages chiffrés invalides, sauf avec une probabilité négligeable.*

La preuve de ce lemme est la même que celle de CRAMER et SHOUP [1998], la seule différence est que dans la simulation de $\mathcal{O}Decryption$, pour les messages chiffrés transformables l'adversaire doit résoudre le système suivant :

$$x_1 + w \times x_2 = log_{g_1} c \bmod q$$
$$x_3 + w \times x_4 = log_{g_1} d \bmod q$$
$$x'_{p_1} + w \times x'_{p_2} = log_{g_1} c_p \bmod q$$
$$x'_{p_3} + w \times x'_{p_4} = log_{g_1} d_p \bmod q$$
$$r_1 x_1 + r_2 \alpha_1 x_3 + r_1 w x_2 + r_2 \alpha_1 w x_4 = log_{g_1} v \bmod q$$
$$r_1 x_{p_1} + r_2 \alpha_2 x_{p_3} + r_1 w x_{p_2} + r_2 \alpha_2 w x_{p_4} = log_{g_1} v_p \bmod q$$

Ils sont linéairement indépendants, ainsi notre simulation est parfaite pour l'adversaire extérieur, à moins que le proxy ne révèle sa clé privée. Si $\mathscr{A}$ peut casser notre algorithme de re-chiffrement, alors $\mathscr{B}$ peut résoudre le problème DDH dans $\mathbb{G}$, ainsi nous prouvons notre théorème.

3.6 Conclusion

Dans ce chapitre, nous avons présenté au départ comment la sécurité des PRE diffère des modèles formalisés pour les PKE. La deuxième section a été dédiée aux mécanismes permettant de concevoir un système publiquement vérifiable en expliquant pourquoi il est utile pour la conception d'un PRE. Ensuite, nous avons repris étape par étape la construction de l'algorithme de CHOW et collab. [2010] et montré sa vulnérabilité trouvée par SELVI et collab. [2017] et nous nous sommes concentrés sur les PRE sécurisés dans le modèle standard.

Nous soulignons que les systèmes de ZHANG et collab. [2013] ne sont pas sécurisés contre les attaques CCA et nous montrons comment un adversaire pourrait distinguer entre deux chiffrés dans le jeu IND-CCA-2. Nous présentons également une construction de PRE unidirectionnelle sans couplage et qui atteint la sécurité IND-CCA-2 dans le modèle standard SBAI et collab. [2020a].

Notre système repose principalement sur l'hypothèse décisionnelle de Diffie-Hellman et la résistance à la seconde pré-image des fonctions de hachage utilisées. Nous ne considérons pas l'efficacité, mais plutôt et surtout à résoudre un des deux problèmes en suspens par CANETTI et HOHENBERGER [2007]. Notre PRE a été implémenté et inclus aux différents PRE pris en charge par le PREaaS SBAI et collab. [2020b].

Chapitre 4

Délégation d'authentification

Sommaire

4.1 Introduction

Nous utilisons de plus en plus les services à distance, que ce soit sur le web ou sur d'autres plateformes. La plupart d'entre eux nécessitent que l'utilisateur crée un compte et se connecte ensuite, ce qui permet de limiter l'accès.

Il existe plusieurs méthodes d'authentification en fonction de la sensibilité des données ou du service demandé. Généralement, le couple login/mot de passe est la méthode la plus courante, mais d'autres méthodes sont utilisées telles que l'OTP (mot de passe à usage unique), les certificats ou encore la biométrie qui permet d'identifier un utilisateur grâce à ses caractéristiques physiologiques (l'iris, les empreintes digitales, etc).

Pour les utilisateurs, avoir plusieurs comptes devient contraignant car cela implique que ces derniers se souviennent de chaque mot de passe puisqu'il doit idéalement être différent pour chaque service.

Puis est apparue la méthode de l'authentification unique (SSO) qui permet aux utilisateurs d'accéder à plusieurs services en ne s'authentifiant qu'une seule fois. Par exemple, une fois connecté sur Google, nous pouvons utiliser ses services comme drive, gmail, youtube ... sans avoir besoin à se ré-authentifier.

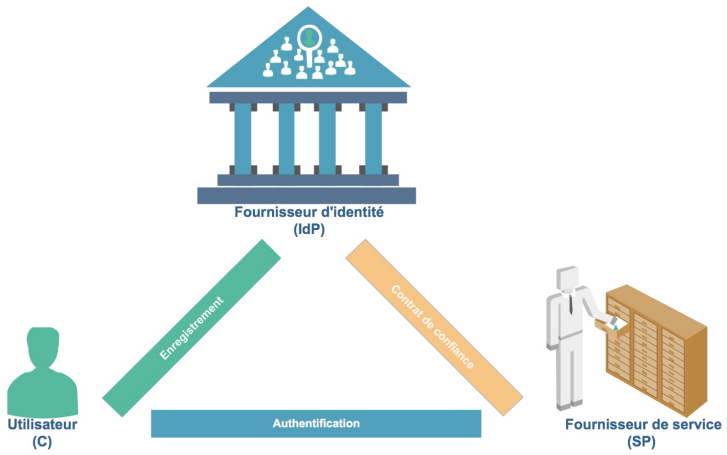


FIGURE 4.1 – Principaux acteurs

Le modèle utilisé par ce système d'authentification repose principalement sur trois acteurs (figure 3.1) : le client qui demande l'accès à une ressource ou à un service, le fournisseur d'identités (IdP) qui stocke les références des clients et le fournisseur de services (SP) qui fournit les ressources et les services sur la base des autorisations des clients communiquées par l'IdP. Dans l'exemple précédent, les fournisseurs de services cités (e.g Youtube) appartiennent à la même organisation (resp. Google). La délégation d'authentification quant à elle peut être vue comme une solution SSO dans un cadre plus générique. En effet, les différents fournisseurs de services peuvent appartenir à la même entité (IdP) ou être totalement indépendants, sous réserve qu'il existe un contrat de confiance entre l'IdP et le SP. Trois grandes approches sont alors proposées :

— **Approche centralisée** : elle est principalement utilisée pour les services qui dépendent de la même entité, laquelle gère sa propre politique de sécurité. Pour cela, il est possible d'utiliser un mécanisme d'authentification comme Kerberos.

— **Approche fédérée** : elle permet un contrôle d'accès et un SSO étendu à travers les frontières organisationnelles en répartissant le contrôle et la maintenance des différentes activités, offrant ainsi plus de commodité et d'efficacité. Elle permet à chaque entité de gérer sa propre politique de sécurité et certaines parties des données des utilisateurs peuvent être partagées avec d'autres services interactifs. Différentes solutions existent, principalement SAML, OpenID...

— **Approche coopérative** : de manière identique, elle permet à chaque partenaire de gérer son propre système d'authentification mais elle diffère de ce dernier dans le sens où les identifiants des utilisateurs ne sont pas partagées entre les différents services, par exemple le Service central d'authentification (CAS).

Les différents exemples mentionnés ci-dessus sont basés sur ce que nous considérons comme des méthodes conventionnelles, par exemple le chiffrement symétrique, comme c'est le cas pour Kerberos, ou des systèmes cryptographiques asymétriques qui utilisent nécessairement des signatures. De plus, il existe des solutions cryptographiques que nous considérons comme non conventionnelles, telles que les signatures de groupes, les accumulateurs unidirectionnels, etc, qui ne sont pas ou rarement utilisées en pratique, généralement pour des raisons d'efficacité. Toutes les solutions cryptographiques utilisées en pratique pour la signature reposent sur le modèle de l'oracle aléatoire pour des raisons d'efficacité CHAIDOS et COUTEAU [2018].

Pouvons-nous éviter ces méthodes conventionnelles et résoudre les problèmes de délégation ? À savoir :

— Authentifier les utilisateurs de manière asynchrone par l'intermédiaire des fournisseurs d'identités et sans synchroniser leurs références (identifiants des utilisateurs) avec les fournisseurs de services.

— Minimiser le nombre de communications nécessaires

— Se baser uniquement sur des primitives cryptographiques sécurisées dans le modèle standard.

Ce sont les principales questions auxquelles nous voulons répondre dans ce chapitre. Nous allons commencer par rappeler les différentes solutions les plus utilisées actuellement, puis présenter l'idée générale de notre contribution. Nous allons ainsi expliquer nos motivations pour l'utilisation d'un proxy de re-chiffrement (PRE) et définir les différentes propriétés nécessaires au PRE pour atteindre notre objectif "délégation d'authentification anonyme et asynchrone".

4.2 Solutions de délégation d'authentification :

La délégation d'authentification permet de tirer profit des relations de confiance entre différents environnements afin de faciliter et d'améliorer le processus d'authentification. Plusieurs solutions existent, parmi les plus utilisées nous trouvons :

— **Kerberos** : qui est basé sur le protocole de Needham-Schroeder à clé symétrique NEEDHAM et SCHROEDER [1978]. Il nécessite un serveur d'authentification central appelé KDC (centre de distribution de clés) pour vérifier et affirmer l'identité des clients et des serveurs. En termes de délégation, Kerberos propose deux types de délégations : sans contrainte et avec contrainte. Dans le premier cas, le serveur ou le compte d'un service auquel le droit est accordé peut se faire passer pour l'utilisateur afin de communiquer avec n'importe quel service sur n'importe quelle machine. Historiquement, c'était le seul choix possible lorsque le principe de délégation a été introduit, mais il a été complété par le principe de délégation restreinte. Il permet aux administrateurs de services de spécifier et de faire respecter les limites d'approbation des applications en limitant le champ dans lequel les services d'application peuvent agir au nom d'un utilisateur. Les administrateurs de services peuvent spécifier quels comptes de services frontaux peuvent effectuer une délégation sur leurs services principaux.

— **Shibboleth** : qui est l'une des plateformes SSO open source les plus populaires pour la gestion des identités et des accès locaux. Le framework Shibboleth est principalement basé sur SAML CANTOR et collab. [2004]. Il fournit un service SSO et une autorisation basée sur des attributs tout en préservant la vie privée des utilisateurs. Comme la plupart des autres solutions, le protocole définit deux composants fonctionnels qui sont le fournisseur de services

(SP) qui gère l'autorisation et l'IdP. Ce dernier gère et maintient l'identité des utilisateurs. Il est conçu pour fournir une identité fédérée avec l'hypothèse principale : l'IdP et le SP se font confiance au sein d'une fédération. Cette confiance est basée sur l'infrastructure à clé publique (PKI) et gérée à l'aide de certificats PKI.

— **OpenId** : qui a été créé pour l'authentification fédérée. Il est conçu comme un système de gestion d'identité centré sur l'utilisateur, où tout fournisseur d'identités peut être utilisé par les utilisateurs (à l'exception des listes blanches), lesquels peuvent même établir leur propre fournisseur. Il n'est pas nécessaire de présélectionner ou de négocier un accord avec les fournisseurs pour permettre aux utilisateurs d'utiliser tout autre compte dont ils disposent. En 2007, l'extension OpenID Attribute Exchange (OpenID AX) a été publiée HARDT et collab. [2007]. Elle définit comment stocker ou mettre à jour les informations sur les attributs du fournisseur OpenID (OP) et comment récupérer ces attributs. En 2014, OpenID Connect a été défini comme une évolution du standard OpenID et il est basé sur le protocole OAuth 2.0 permettant d'ajouter une couche d'autorisations. Il s'appuie sur les JSON Web Token (JWT) pour l'échange d'informations d'identité ou de toute autre information pertinente au fournisseur de services appelé dans le contexte OpenID Relying Party (RP).

— **Oauth** : qui introduit une couche d'autorisations qui permet aux applications d'obtenir un accès limité aux comptes des utilisateurs. Elle pourrait être interfacée avec Kerberos et d'autres solutions d'authentification décrites ci-dessus. Le système peut impliquer jusqu'à quatre acteurs : le propriétaire de la ressource (RO), l'application cliente (C), le serveur d'autorisation (AS) et le serveur de ressources (RS). En général, un client C demande l'accès aux ressources contrôlées par le RO et hébergées par le RS. Au lieu d'utiliser les informations d'identification du RO pour accéder aux ressources protégées, C obtient un jeton d'accès (une chaîne indiquant le "scope", la durée de vie et d'autres attributs d'accès spécifiques). Les jetons d'accès sont attribués à des clients tiers (C) par un AS avec l'approbation du RO. Le C utilise le jeton d'accès pour accéder aux ressources protégées hébergées par le RO.

Les différentes solutions que nous avons présentées utilisent des méthodes cryptographiques conventionnelles c'est-à-dire les chiffrements symétriques / asymétriques et les signatures. D'autres solutions sont devenues envisageables par exemple les accumulateurs que nous présenterons brièvement dans la prochaine section ou encore les PRE. Dans littérature, la seule solution utilisant les PRE pour l'authentification a été proposée par NUNEZ et collab. [2012]. Les auteurs présentent une solution pour le respect de la vie privée dans le contexte d'OpenID AX qui est basée sur les PRE. Les attributs sont stockés de manière chiffrée, ainsi l'utilisateur doit créer une clé de re-chiffrement pour chaque RP qui permettra à l'OP de re-chiffrer les attributs stockés pour le RP. La même idée pourrait être appliquée pour Shibboleth ou SAML. Il s'agit donc d'une couche supplémentaire pour garantir la confidentialité.

Dans notre cas, en plus d'assurer la confidentialité et d'authentifier les utilisateurs de façon anonyme, nous utilisons la délégation de déchiffrement comme solution principale pour la délégation d'authentification.

4.3 Contribution

La solution que nous proposons permet de :

- **Déléguer l'authentification** : l'authentification auprès d'un fournisseur d'identités permet l'accès aux fournisseurs de services.
- **Préserver l'anonymat des utilisateurs** : les identifiants des utilisateurs ne sont jamais divulgués ou partagés avec les fournisseurs de services.
- **Authentifier en mode asynchrone** : l'authentification ne requiert aucune synchronisation entre les fournisseurs de services et les fournisseurs d'identités.

Dans cette section, nous détaillerons notre solution en donnant d'abord l'idée principale, puis en définissant les différentes propriétés nécessaires au PRE utilisé et enfin en expliquant les différentes phases de notre protocole.

4.3.1 Idée générale

La solution proposée permet de déléguer l'authentification sans utiliser de signature et sans l'aide d'une couche TLS (Transport Layer Security). Il est donc possible d'authentifier l'utilisateur sur des réseaux non sécurisés sans avoir à synchroniser les informations d'identification de l'utilisateur avec les services. Notre système implique trois acteurs : l'IdP, le client et le SP. Une phase d'inscription doit être préétablie entre le client et l'IdP ainsi qu'une phase d'établissement d'une relation de confiance entre l'IdP et le SP. Ces phases consistent principalement en l'échange d'informations d'identification ainsi que de clés pré-partagées. La figure 4.2 montre les différentes interactions entre les acteurs nécessaires à la conduites de notre protocole. Les informations relatives aux utilisateurs et fournisseurs de services ainsi que les données secrètes sont présentées dans les tableaux de la sous-section 3.3.3. Le point le plus important est qu'il n'est pas nécessaire de stocker des informations relatives aux utilisateurs du côté des fournisseurs de services.

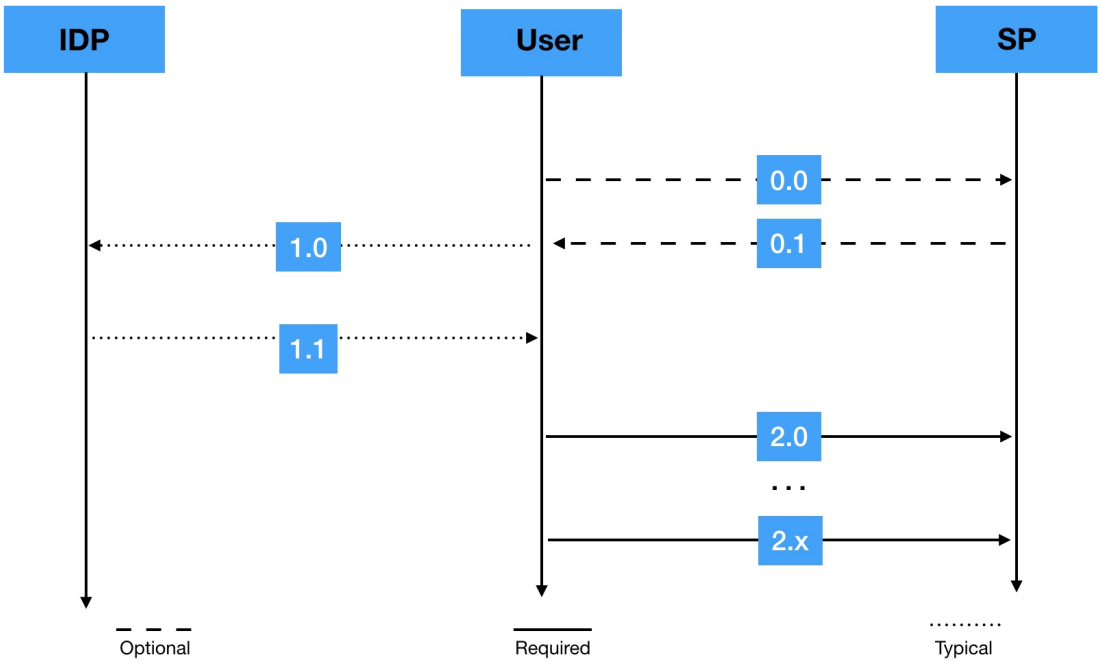


FIGURE 4.2 – Principaux acteurs et interactions du protocole

— User ⟷ SP :
  - 0.0 : l'utilisateur demande la liste des IdPs auxquels le SP fait confiance.
  - 0.1 : le SP renvoie la liste correspondante.

— User ⟷ IDP :
  - 1.0 : l'utilisateur choisit l'IdP, crée une demande d'authentification et l'envoie à l'IdP correspondant.
  - 1.1 : l'IdP vérifie la validité de la demande, authentifie l'utilisateur et répond ensuite avec le jeton d'authentification.

— User ⟷ SP :

- 2.0 : l'utilisateur crée un jeton éphémère, l'ajoute à celui qu'il a reçu et s'authentifie auprès du SP.
- 2.x : à chaque connexion, l'utilisateur reprend à l'étape 2.0 avec une légère modification.

Une fois la phase d'inscription et d'établissement de confiance terminée, nous pouvons authentifier les utilisateurs auprès du SP :

— sans qu'il soit nécessaire de communiquer directement entre l'IdP et le SP et avec le moins d'interaction possible,

— en utilisant n'importe quel service qui fait confiance à cet IdP une fois authentifié,

— sans qu'aucune information sur le client ne soit stockée du côté du SP.

Pour ce faire, nous allons utiliser le PRE comme technologie de base.

Dans la littérature, il est difficile de trouver une solution efficace qui combine ces trois avantages. Par exemple, OpenId nécessite une interaction directe entre l'IdP et le SP, il implique au moins cinq interactions et utilise JWT qui sont signées en utilisant un secret avec l'algorithme HMAC ou une paire de clés publique / privée utilisant RSA ou ECDSA. Le modèle de sécurité repose en tout cas sur le modèle de l'oracle aléatoire. Une signature est, de par sa conception, basée sur le protocole NIZK (Non-Interactive Zero Knowledge). Ces protocoles sont soit sécurisés dans le modèle standard mais inefficaces, soit efficaces mais généralement construits à partir de l'heuristique Fiat-Shamir FIAT et SHAMIR [1986] et donc sécurisés dans le modèle de l'oracle aléatoire. Nous avons décidé d'exclure l'utilisation de signatures et de créer ainsi un protocole qui pourrait s'appuyer sur des schémas cryptographiques qui se sont avérés sûrs dans le modèle standard tout en conservant un bon niveau d'efficacité. Bien sûr, il y a toujours un compromis entre les performances en termes de temps, de stockage et de niveau de sécurité.

Il existe plusieurs solutions qui réduisent le stockage et la complexité de la gestion des informations liées à l'identification, comme les accumulateurs. Ce schéma cryptographique a été proposé comme une alternative décentralisée aux signatures numériques dans la conception de protocoles distribués sécurisés BENALOH et DE MARE [1993]. Pour la vérification de l'appartenance à la méthode conventionnelle, un tiers de confiance signe numériquement les identifiants de chaque membre et distribue sa clé publique. Pour vérifier l'adhésion d'un utilisateur, il suffit de fournir son identité signée alors que la clé publique est accessible à tous (même les non-membres peuvent la vérifier). Lors de l'utilisation d'un accumulateur, celui-ci peut être initialisé et diffusé par un membre d'un groupe et pas nécessairement par un tiers de confiance. Les membres échangent alors leurs informations d'identification puis chacun calcule son témoin et l'accumulateur. Ce dernier peut ne pas être conservé puisqu'il peut être calculé. Pour les accumulateurs statiques, les membres du groupe ne peuvent pas changer, ce qui est restrictif, alors qu'avec les accumulateurs dynamiques, nous pouvons ajouter et retirer des membres du groupe. Le développement actuel des accumulateurs TREMEL [2013] ne permet pas encore une utilisation en condition réelle alors que la maturité des PRE est aujourd'hui une réalité puisque des implémentations très efficaces existent.

4.3.2 Contexte PRE

Nous avons opté pour l'utilisation des proxy de re-chiffrement pour utiliser la délégation des droits de déchiffrement d'une entité à une autre à des fins d'authentification. En effet, il existe déjà des proxy de re-signature (PRS) qui permettent de déléguer les droits de signature qui sont conçus spécialement pour la problématique d'authentification et de délégation. Néanmoins, ils présentent plusieurs inconvénients, principalement celui de donner le pouvoir de signature au nom d'une entité par une autre à laquelle la confiance est donnée, ce qui reste très sensible et

peut avoir des conséquences néfastes. Aussi, la majorité des PRS se basent sur l'utilisation d'applications bilinéaires coûteuses pour certaines solutions et/ou sur le modèle de preuve de sécurité qui repose sur les oracles aléatoires pour d'autres solutions. C'est ainsi que nous avons étudié la possibilité d'utiliser les PRE à des fins d'authentification.

Nous estimons qu'une clé de re-chiffrement peut faire l'objet d'un contrat de confiance entre l'IdP et le SP. Ces derniers doivent détenir des paires de clés PRE, l'IdP garde alors une paire de clés secrètes tandis que la clé de chiffrement du SP peut être publiée. Dans ce cas d'utilisation, l'IdP peut créer une clé de re-chiffrement pour chaque SP qui lui demande un contrat de confiance, ce qui correspond à la phase d'établissement de confiance. Maintenant, un SP peut re-chiffrer via cette clé tout message chiffré avec la clé de chiffrement secrète détenue par l'IdP. Une fois le message re-chiffré et déchiffré, il ne sera lisible que s'il a été généré correctement par l'IdP.

Cependant le PRE utilisé dans notre protocole doit répondre à certaines propriétés. Nous définissons ci-dessous les propriétés que nous considérons comme essentielles pour la réalisation de notre solution :

— *Unidirectionnel* : la délégation des droits de déchiffrement de l'IdP au SP ne permet pas à l'IdP de déchiffrer les messages chiffrés des SP.

— *Non-interactive* : la clé de re-chiffrement peut être générée par l'IdP sans besoin de connaitre la clé secrète des SP.

— *Collusion-safe* : si le proxy et les SP sont corrompus et complotent , ils ne pourront pas retrouver la clé privée de l'IdP.

— *Non-transitive* : le proxy de re-chiffrement ne peut pas re-déléguer les droits de déchiffrement. (avec $Rk_{a\to b}$ et $Rk_{b\to c}$ le proxy ne peut pas $Rk_{a\to c}$)

— *Non-transferable* : le proxy et le délégant ne peuvent pas redéfinir les droits de déchiffrement. (avec $Rk_{a\to b}$ et Pk_c et Sk_b on ne peut pas calculer $Rk_{a\to c}$)

— *Key-private* : en ayant la clé de re-chiffrement, il est impossible de retrouver les clé publiques du délégant et délégataire.

4.3.3 Le protocole proposé

Phase d'initialisation

Comme mentionné précédemment, une phase d'inscription est nécessaire, au cours de laquelle l'IdP stocke l'identifiant du client (id) et le secret pour l'authentifier (v). Nous avons pris comme exemple identifiant/secret d'identification pour illustrer notre protocole (voir tableau 4.1) mais d'autres méthodes d'identification peuvent être utilisées comme la biométrie. Pour les services, au moment de l'établissement de confiance, l'IdP doit avoir généré une paire de clés asymétriques secrètes (pks/sks) en plus de sa propre paire de clés asymétriques publique/privée (pki/ski). Cette paire de clés asymétriques secrètes sera utilisée pour créer un jeton d'authentification et des clés de re-chiffrement pour le service concerné ; l'autre paire de clés sera utilisée pour vérifier la validité de la demande d'authentification de l'utilisateur. Au lieu d'utiliser une paire de clés symétriques secrètes pré-partagées entre l'IDP et le SP afin de vérifier l'origine des jetons, nous utilisons une paire de clés secrètes asymétriques connues uniquement de l'IDP, grâce à laquelle nous générons la clé de re-chiffrement de l'IDP vers le SP correspondant. Comme le montre le tableau 4.2, l'IdP aura alors stocké pour chaque service sa clé publique ainsi que la clé de re-chiffrement générée pour celui-ci. L'utilisation d'un PRE à clé privée est alors indispensable. Cela permet d'éviter que la clé publique ne soit reconstruite par la clé de re-chiffrement et donc de garder secrète la paire de clés asymétriques. Côté service, il doit stocker cette clé de re-chiffrement ainsi que la clé publique de cet IdP (voir tableau 4.3). Le service peut, à son tour, avoir établi une relation de confiance avec d'autres IdP.

C1	C2	C3	...	Cn
id1	id2	id3	...	idn
v1	v2	v3	...	vn

TABLEAU 4.1 – Informations d'identification des utilisateurs stockées par l'IDP .

S1	S2	...	Sn	Services
pkp_1 $rk_{s\rightarrow p1}$	pkp_2 $rk_{s\rightarrow p2}$	... ...	pkp_n $rk_{s\rightarrow pn}$	pks/sks

TABLEAU 4.2 – Clés publiques, clés de re-chiffrement et paires de clés secrètes asymétriques stockées par l'IDP.

I1	I2	I3	...	In
pki_1	pki_2	pki_3	...	pki_n
$rk_{s1\rightarrow p}$	$rk_{s2\rightarrow p}$	$rk_{s3\rightarrow p}$	...	$rk_{sn\rightarrow p}$

TABLEAU 4.3 – Clés publiques et clés de re-chiffrements stockées par le SP.

Phase d'authentification

La figure 4.3 illustre les différentes communications requises pour authentifier un utilisateur ainsi que les informations à stocker au sein de chaque entité que nous allons détailler ci-dessous.

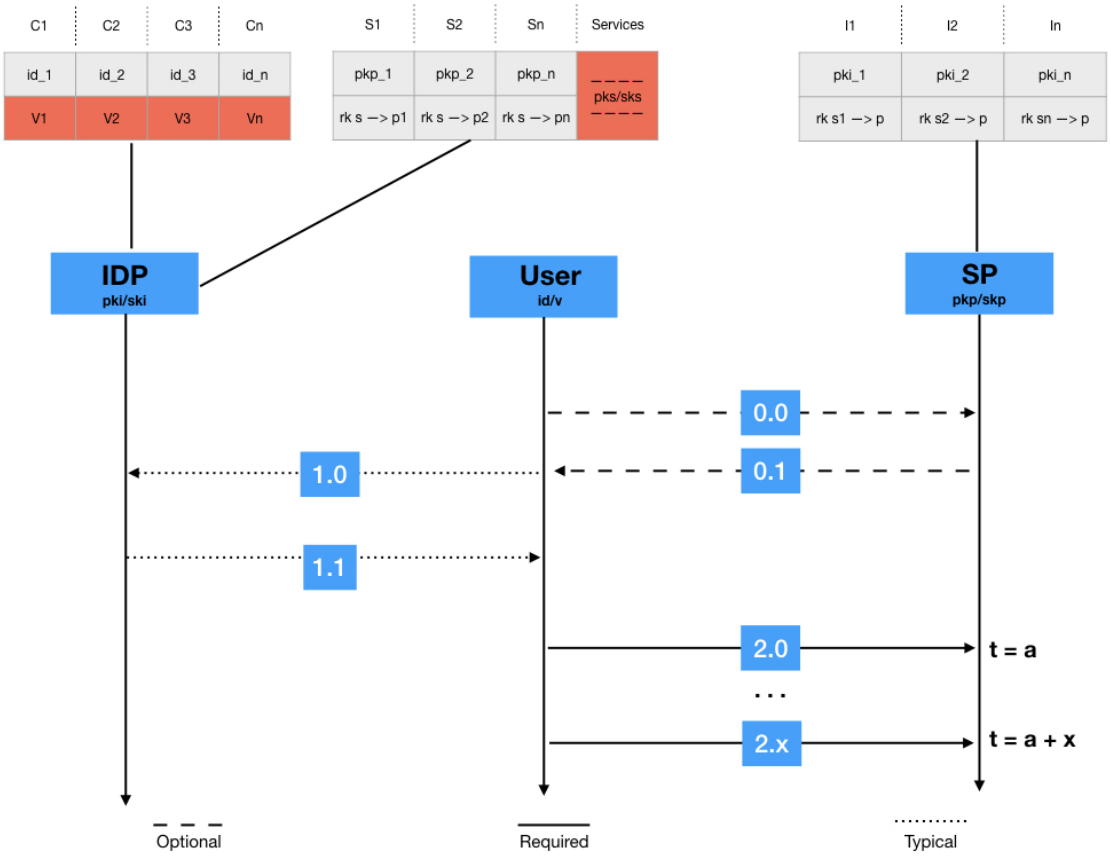


FIGURE 4.3 – Principaux acteurs et interactions du protocole

Tout d'abord, l'utilisateur demande au SP la liste des IdPs de confiance (0.0). Le SP renvoie ensuite sa liste d'IdP connus (0.1). L'utilisateur choisit l'IdP correspondant et envoie une demande d'authentification (Auth_req) qui contient des informations relatives à son identification et un jeton aléatoire, tous chiffrés avec la clé publique de l'IdP(1.0) afin de recueillir le jeton d'authentification (Auth_token). L'IdP, à la réception du Auth_req, le déchiffre et vérifie la validité des informations. S'il correspond à un utilisateur, ce dernier crée un Auth_token en chiffrant un "scope" contenant principalement un "timestamp", une date d'expiration et le jeton aléatoire créé par l'utilisateur(1.1). D'autres informations relatives à l'autorisation pourraient être ajoutées, toutefois, les questions liées au contrôle d'accès ne sont pas couvertes par le présent travail.

Pour chiffrer ce "scope", l'IdP utilise sa propre clé de chiffrement asymétrique secrète destinée aux services. L'utilisateur, en recevant son Auth_token, chiffre cette fois son jeton aléatoire avec la clé publique du SP qui correspond au jeton éphémère (Eph_token), le concatène avec le Auth_token reçu et l'envoie au SP(2.0). Le SP vérifie la validité de la réponse d'authentification Auth_resp. Il commence par rechiffrer la première partie de la réponse, ce qui pourrait également être fait par une partie indépendante semi-fiable. Ensuite, le SP déchiffre le message chiffré obtenu et la deuxième partie de la Auth_resp. Si les deux jetons aléatoires calculés correspondent à la même valeur, l'authentification est alors réussie. Le SP stocke le jeton aléatoire jusqu'à la prochaine connexion, le Auth_token pendant sa durée de validité et le nombre de connexions utilisées avec ce jeton. Le SP peut à ce moment communiquer les ressources demandées par C en utilisant la clé de session imbriquée dans Eph_token. À chaque connexion, l'utilisateur crée un nouveau Eph_token en chiffrant l'incrémentation du jeton aléatoire (par exemple, au moment de la connexion x (2.x), la valeur de ce jeton correspondra à $r+x$), la clé de session peut être modifiée mais ce n'est pas obligatoire et le Auth_token sera le même pendant sa durée de validité. Le SP n'aura plus qu'à vérifier l'existence de ce jeton, le nombre de connexions qui lui sont liées et la valeur du jeton aléatoire.

Nous décrivons ci-dessous les grandes lignes techniques du protocole à l'aide de la figure 4.4 :

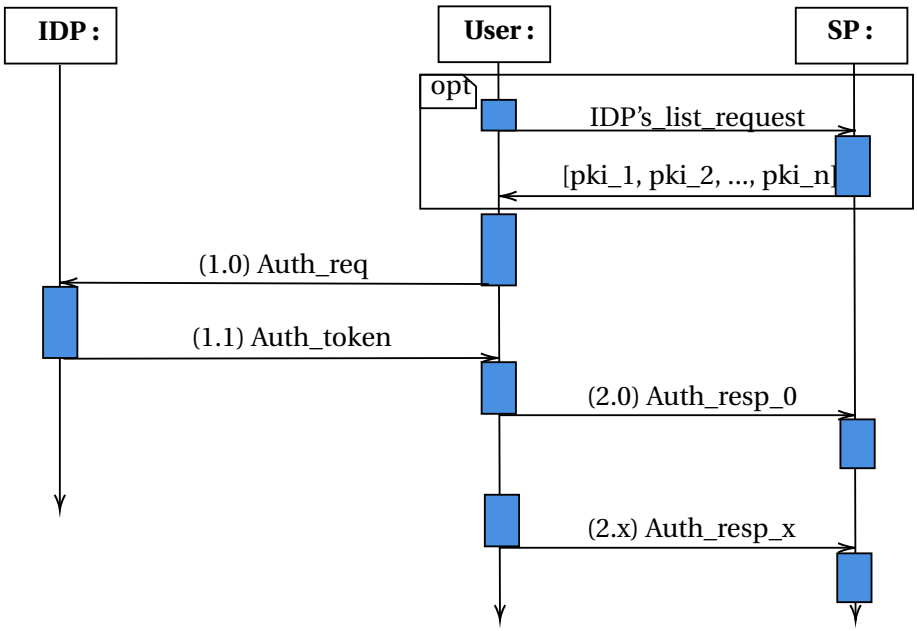


FIGURE 4.4 – Diagramme de séquence du diagramme proposé

Finalement, les différentes étapes présentées dans la figure 4.4 à partir de 1.0 correspondent à des processus cryptographiques bien précis que nous décrivons ci-dessous :

- 1.0 : Choisi $r \xleftarrow{\$} \{0,1\}^{l_0}$ puis calcule Auth_req = E$nc(pki, id||v||r)$.

— 1.0 → 1.1 : D$ec(ski, Auth_req)$, puis vérifie la validité de id & v.

- 1.1 : Auth_token = E$nc(pks, scope)$; $scope = (timestamp||expire\_date||r)$.

- 2.0 : Choisi $k \xleftarrow{\$} \{0,1\}^{l_1}$ puis calcule Eph_token = E$nc(pkp, pki||r||k)$; Auth_resp = Auth_token||Eph_token.

— 2.0 → 2.x : D$ec(skp,$ReE$ncrypt(rk_{s\to p},$A$uth_token))$; D$ec(skp,$E$ph_token)$ puis vérifie la validité de $scope$ & r.

- 2.x : Auth_token||E$nc(pkp, pki||r+x)$.

Les deux prochaines sections concernent la preuve de sécurité de notre protocole. Nous montrons en premier comment notre protocole se comportera face aux différents types d'attaques connues. Puis de manière plus formelle, nous allons présenter la preuve de sécurité en se basant sur la méthode dite BAN logique.

4.4 Preuve et complétude

4.4.1 Preuve

Un protocole de délégation d'authentification sécurisé devrait avoir les propriétés suivantes : la complétude, la sécurité contre les attaques par falsification, la sécurité contre les attaques par re-jeu et la sécurité contre l'attaque de l'homme du milieu. Nous allons ici présenter et définir ces propriétés et montrer que notre proposition les respecte.

- **Complétude** : le délégué (le SP) est toujours en mesure d'identifier la source de la réponse d'authentification créée par le délégataire (l'IdP) et le client (l'utilisateur), si les différents acteurs suivent le protocole.
  Preuve : lorsque le SP reçoit la réponse d'authentification Auth_resp = Auth_token||Eph_token, le re-chiffrement de Auth_token = E$nc(pks, scope)$ sera valide si et seulement si Auth_token a été créé par l'IdP. En effet, l'IdP est le seul à posséder pks. Le jeton Eph_token devrait contenir la même valeur aléatoire que celle trouvée dans le "scope". Ainsi, si l'utilisateur et l'IdP exécutent tous deux strictement le protocole, le SP identifiera toujours la source de la réponse d'authentification.

- **Sécurité contre les attaques par falsification** : lorsqu'un attaquant veut falsifier un jeton valide et l'envoie au SP concerné, le protocole est capable de résister à l'attaque de falsification.
  Preuve : pour lancer ce type d'attaque, l'attaquant doit créer un message chiffré valide correspondant à Auth_token = E$nc(pks, scope)$. Ensuite, il doit créer le jeton Eph_token qui est E$nc(pkp, pki||r)$. La deuxième partie pourrait être créée facilement par l'attaquant puisque pkp est connu. Cependant, la première partie qui correspond à Auth_token ne pourrait pas être créée à moins que l'attaquant connaisse la clé secrète pks.

- **Sécurité contre les attaques par re-jeu** : lorsqu'une transmission de jeton est malicieusement répétée par un attaquant, le SP concerné refuse la demande d'authentification.
  Preuve : un attaquant peut obtenir un jeton valide Auth_resp (c'est à dire E$nc(pks, scope)$ || E$nc(pkp, pki||r)$) en interceptant les communications. Lorsque l'attaquant veut se faire passer pour l'utilisateur, le jeton obtenu correspond au chiffrement d'un jeton r choisi aléatoirement par l'utilisateur, mais la valeur de ce jeton est incrémentée après chaque connexion. Par conséquent, la retransmission du même jeton Auth_resp n'aboutira pas à une authentification réussie. Le SP peut détecter l'attaque en stockant la dernière valeur de l'incrément du jeton aléatoire et la comparer avec celle reçue.

- **Sécurité contre l'attaque de l'homme du milieu** : un attaquant ne peut pas créer une demande/réponse d'authentification valable au nom des différents acteurs.

Preuve : selon notre protocole, si l'attaquant peut créer une demande d'authentification valide au nom de l'utilisateur (par exemple $Enc(pki, id||v||r) = Auth\_req$), il doit connaître les informations d'authentification relatives à l'utilisateur. Ceci est impossible car cela implique qu'il ait accès à la clé secrète permettant de déchiffrer le $Auth\_req$. Une deuxième façon de mener cette attaque consiste à créer une réponse d'authentification valide $Enc(pks, scope)||Enc(pkp, pki||r)$ au nom de l'utilisateur. Ceci est également impossible car cela implique que l'attaquant connaisse la valeur du jeton aléatoire choisi par l'utilisateur. La seule façon pour l'attaquant de connaître cette valeur est de casser le système de chiffrement, à savoir le PRE utilisé, ou de deviner la valeur du jeton aléatoire par une attaque de force brute. Cependant, il est impossible d'y parvenir pendant la durée de vie du jeton car la taille du jeton aléatoire est l'un des paramètres de sécurité de notre protocole.

4.4.2 Complétude

À première vue, nous pouvons constater que toute la communication est chiffrée. Un intercepteur n'est donc pas en mesure d'interpréter les paquets en une forme lisible par l'homme. Ceci est garanti par un chiffrement de bout en bout. De plus, les phases d'enregistrement et d'établissement de confiance se font en toute sécurité.

L'utilisation de jetons aléatoires joue un rôle majeur dans notre solution. Le jeton en lui-même ne peut pas authentifier le client mais seulement confirmer son origine ; il le fait en gardant secrètement la paire de clés asymétriques de l'IdP pks/sks destinée aux services. Par conséquent, si le re-chiffrement est effectué correctement, nous sommes sûrs que le jeton a été créé par l'IdP. Néanmoins, n'importe qui pourrait intercepter le jeton et l'envoyer à n'importe quel SP pour être authentifié. L'utilisation des jetons aléatoires nous permet de nous assurer que l'utilisateur concerné est bien l'entité qui a fait la demande d'authentification à l'IdP. Son incrémentation permet d'éviter les attaques de re-jeu car la valeur du jeton sera différente pour chaque connexion. La seule façon pour un attaquant de se faire passer pour l'utilisateur est de deviner la valeur du jeton aléatoire. En outre, il permet de s'authentifier dans les SP de manière anonyme. Il faut noter que le PRE doit avoir les propriétés expliquées dans la sous-section 3.3.2, qui sont : {Unidirectionnel, Non-interactive, Collusion-safe, Non-transitive, Non-transferable, Key-private}

4.5 Preuve formelle

En dehors de l'analyse que nous avons présentée dans la section 4.4, le protocole proposé semble résister intuitivement aux différentes attaques connues. Néanmoins une analyse de sécurité plus formelle doit être menée pour la validation du système. Afin d'appuyer notre preuve de sécurité, nous nous sommes basés sur la méthode dite BAN logique BURROWS et collab. [1989].

4.5.1 Définition du BAN logique

Proposée en 1989 par Burrows, Abadi et Needham, c'est l'une des premières méthodes formelles d'analyse de protocole d'authentification.Elle permet de répondre de manière formelle aux différentes questions qui se posent à chaque proposition d'un protocole d'authentification :

— Ce protocole fonctionne-t-il?

— Qu'est-ce que ce protocole permet de réaliser exactement?

— Quelle sont les hypothèses nécessaires au fonctionnement du protocole?

L'analyse BAN repose sur les croyances des parties de confiance impliquées dans les protocoles. Elle permet d'étudier l'évolution de ces croyances en fonctions des échanges entre les différentes entités. Nous constatons ici l'une des premières limites de la méthode puisqu'il n'est pas

possible d'introduire des entités ou des clés corrompues. Une vérification avec la logique BAN n'implique pas nécessairement qu'aucune attaque sur le protocole n'est possible. Une preuve avec la logique BAN est une bonne preuve d'exactitude pour un protocole d'authentification basé sur des hypothèses bien définies. Elle permet de répondre à de nombreuses questions de manière formelle et d'exclure certaines attaques et erreurs possibles.

Le protocole est formalisé en utilisant les termes suivants :

— Participants : toute entité ou acteur impliqué dans le déroulement du protocole est considéré comme un participant et noté par une lettre majuscule.

— Clés : toute clé symétrique ou asymétrique est prise en compte lors de la preuve de sécurité. Une paire de clés asymétriques est souvent notée : (K, K^{-1}) où K^{-1} est la clé privée correspondante à K.

— Messages : tout message en clair ou chiffré utilisé ou transmis est pris en compte. Lorsqu'il s'agit d'un message chiffré, nous le notons $\{X\}_K$, correspondant au chiffrement du message X utilisant la clé K. Une combinaison entre un message X et Y, où Y est un secret prouvant l'identité de l'émetteur de X est noté $\langle X \rangle_Y$.

4.5.2 Notations BAN :

A partir des termes définis précédemment, le protocole est traduit en se basant sur les notations et formules suivantes :

— P croit X : ce qui veut dire que le participant P, à ce moment précis du protocole croit que X est vrai. La notation correspondante est :

$$P \mid\equiv X$$

— P voit ou reçoit X : cette notation permet de formuler la réception d'un message par un participant, nous pouvons aussi dire que P détient le message X et noté comme suit :

$$P \lhd X$$

— P a envoyé ou a dit X : ce qui veut dire qu'à un certain moment lors de l'exécution ou en dehors de l'exécution du protocole, le participant P a envoyé un message X.

$$P \mid\sim X$$

— P contrôle X : le participant P a le contrôle total sur la déclaration X. Souvent utilisé pour décrire une autorité de certification ou un centre de distribution de clé et est noté :

$$P \Rightarrow X$$

— P a généré un aléa X : ici X est un message, une valeur ou une formule qui n'a jamais été envoyé auparavant lors de l'exécution du protocole. Nous le notons comme suit :

$$\#(X)$$

— P et Q partagent une clé K : ce qui se traduit par l'existence d'une clé secrète K utilisable seulement pour la communication entre P et Q. Elle est notée comme suit :

$$P \overset{K}{\leftrightarrow} Q$$

— P détient la clé publique K : P est le seul participant à voir la clé privée correspondante K^{-1}. La détention d'une clé publique K par un participant P est notée :

$$\overset{K}{\mapsto} P$$

— P et Q connaissent un secret X : X peut être alors utilisé pour prouver l'identité de P ou Q l'un à l'autre. Elle peut aussi être connue par d'autres participant auxquels P et Q font confiance. Nous le notons :

$$P \overset{X}{=\!=} Q$$

4.5.3 Règles BAN :

Maintenant que nous avons défini les termes et les différentes notations existantes, nous pouvons suivre les lois appelées aussi postulats logiques afin de générer de nouvelles déclarations.

— Vérification de l'origine du message : cette loi permet de dériver des croyances sur l'origine du message à travers l'interprétation des informations échangées lors de l'exécution du protocole. Trois règles sont définies :

- Pour les clés partagées : lorsque P voit un message X chiffré avec une clé partagée entre P et Q, alors P croit que Q a envoyé le message. Comme la clé secrète n'est connue que de P et Q, seuls P ou Q sont capables de produire le message et P sait ce qu'il a dit.

$$\frac{P \mid\equiv Q \overset{K}{\leftrightarrow} P, P \triangleleft \{X\}_K}{P \mid\equiv Q \mid\sim X}$$

- Pour les clés publiques : similaire à la règle précédente, elle postule le fait que lorsque P voit un message X qui est chiffré avec la clé privée K^{-1} de Q, il ne peut être envoyé que par Q.

$$\frac{P \mid\equiv \overset{K}{\mapsto} Q, P \triangleleft \{X\}_{K^{-1}}}{P \mid\equiv Q \mid\sim X}$$

- Pour les secrets partagés : lorsque P croit que le secret Y est partagé avec Q et voit $\langle X \rangle_Y$, alors P croit que Q a déjà dit X. Puisque Y garantit l'identité de P auprès de Q.

$$\frac{P \mid\equiv Q \overset{Y}{\rightleftharpoons} P, P \triangleleft \langle X \rangle_Y}{P \mid\equiv Q \mid\sim X}$$

— Vérification de la nouveauté des messages "nonce" ou "aléa" : lorsque P croit que X est un message récent (nonce) et que P croit qu'il a été dit par Q, alors P croit que Q croit toujours le message X.

$$\frac{P \mid\equiv \#(X), P \mid\equiv Q \mid\sim X}{P \mid\equiv Q \mid\equiv X}$$

— Vérification de la fiabilité de l'origine : cela signifie que nous faisons confiance à un participant Q pour faire des déclarations sur X. Lorsque P croit que Q contrôle X et que Q croit X alors nous déduisons que P croit X.

$$\frac{P \mid\equiv Q \Rightarrow X, P \mid\equiv Q \mid\equiv X}{P \mid\equiv X}$$

— Autres :

- Si P voit un message composé alors il peut voir une partie du message :

$$\frac{P \triangleleft (X, Y)}{P \triangleleft X}$$

- Si P voit la combinaison d'un message avec un secret partagé, il peut voir le message :

$$\frac{P \triangleleft \langle X \rangle_Y}{P \triangleleft X}$$

- Si P croit qu'il a une clé secrète partagée avec Q et qu'il voit un message chiffré avec cette clé secrète alors il croit que le message est vrai :

$$\frac{P \mid\equiv Q \overset{K}{\leftrightarrow} P, P \triangleleft \{X\}_K}{P \mid\equiv X}$$

- Si P croit qu'il détient sa vraie clé publique et qu'il voit un message chiffré avec cette clé alors il voit ce message mais ne croit pas forcément qu'il est vrai :

$$\frac{P \mid\equiv \overset{K}{\longmapsto} P, P \lhd \{X\}_K}{P \lhd X}$$

- Si P croit que Q détient sa clé publique et que P voit un message signé par Q alors P voit le message X. Le même postulat déduit que P croit que Q à dit X comme il l'a été défini plus haut mais pas forcément que X est vrai :

$$\frac{P \mid\equiv \overset{K}{\longmapsto} Q, P \lhd \{X\}_{K^{-1}}}{P \lhd X}$$

Nous pouvons remarquer que cette liste n'est pas exhaustive. Bien qu'il manque une base sémantique au formalisme, la logique BAN fournit une base intuitive et une technique puissante pour l'analyse de la sécurité Boyd et Mao [1993]. La flexibilité de la logique BAN permet d'ajouter des constructions et des règles appropriées. Nous trouvons plusieurs extensions de la méthode BAN comme GNY Gong et collab. [1990] qui ajoute la notion de possession et de non provenance. Cette méthode construit de nouveaux postulats logiques différents du BAN et peut être appliquée à d'autres protocoles cryptographiques en dehors des systèmes d'authentification. Dans le cadre de notre preuve, il s'agit d'analyser un système de délégation d'authentification. Aussi, les notions de possessions et de non provenance n'ont pas d'effet sur notre protocole, d'où l'utilisation du BAN logique avec une légère modification. Certaines notions qui n'existent pas dans les deux logiques BAN et GNY ont été définies en intégrant ainsi des éléments permettant d'interpréter les fonctions des PRE dans le BAN et implicitement pour la méthode GNY aussi.

4.5.4 Etapes de la méthode BAN

Avant de commencer l'analyse et suivre l'évolution du protocole, différentes étapes doivent être menées au préalable :

— L'idéalisation du protocole : la première étape consiste à idéaliser le protocole. Nous définissons les différents participants, les clés et les messages impliqués dans le système, puis nous traduisons chaque étape.

— La définition des objectifs et sous-objectifs du protocole : les objectifs et sous-objectifs dépendent du protocole étudié et de ce que nous voulons prouver. Les sous-objectifs sont dérivés directement des objectifs principaux et permettent de faciliter la preuve et d'avoir plus de visibilité. Pour prouver que les objectifs sont atteints, il suffit de prouver que tous les sous-objectifs sont atteints sans exception.

— La définition des hypothèses : en général, les hypothèses indiquent quelles clés sont initialement partagées entre les participants, quels participants ont généré de nouveaux nonces et quels participants sont fiables d'une certaine manière. Dans la plupart des cas, les hypothèses sont standards et évidentes pour le type de protocole considéré.

Maintenant que la méthode BAN est définie, nous allons présenter deux exemples. Le premier est un contre-exemple qui consiste à prouver que l'échange de Diffie-Hellman anonyme est vulnérable. Le deuxième présente une preuve formelle pour l'échange semi-statique de clés de Diffie-Hellman. Par la suite, nous allons prouver la sécurité de notre protocole en suivant la logique BAN. Néanmoins, certaines notions ont été introduites afin de pouvoir adapter cette méthode d'analyse aux technologies utilisées par notre protocole à savoir les PRE. Seuls les éléments nécessaires à l'interprétation de notre protocole ont été définis, cette extension peut être améliorée par la suite.

4.6 Exemple d'analyse sur l'échange de clés de Diffie-Hellman

4.6.1 Anonyme Diffie-Hellman

Le protocole Diffie-Hellman anonyme établit un secret partagé entre deux parties sans authentification. Il permet de garantir une sécurité contre les attaquants passifs où le secret partagé ne peut pas être découvert. Par contre, un attaquant actif pourrait prétendre être l'une des parties légitimes et monter une attaque d'homme du milieu. Nous allons prouver en utilisant le BAN logique, que cette méthode ne permet pas de garantir l'échange de clés entre deux entités.

Idéalisation du protocole

Soit q un nombre premier et g un générateur de $\mathbb{Z}_q$. Les valeurs de q et g sont connues. Le protocole d'échange de clés de Diffie-Hellman entre deux entités A et B procède comme suit :

1. A choisi un élément aléatoire N_a de $\mathbb{Z}_q$.
2. A calcule $\alpha = g^{N_a}$
3. A envoie α à B.
4. B choisi un élément aléatoire N_b de $\mathbb{Z}_q$.
5. B calcule $\beta = g^{N_b}$.
6. B envoie β à A.
7. A calcule $K = \beta^{N_a}$.
8. B calcule $K = \alpha^{N_b}$.

— Les participants : A et B sont les entités impliquées dans le protocole

— Les messages : α et β sont les messages échangés.

— Les clés : K est la seule clé présente qui correspond à la clé échangée à la fin du protocole.

Définition des objectifs et sous-objectifs du protocole

L'objectif que nous allons définir pour ce protocole est l'échange d'une clé secrète entre A et B, ce qui se traduit en :

- Objectifs :

  1 . $A \mid\equiv A \overset{K}{\leftrightarrow} B$

  2 . $B \mid\equiv B \overset{K}{\leftrightarrow} A$

Ces objectifs permettent de vérifier si les deux parties estiment elles-mêmes que la clé K est vraiment échangée entre A et B.

- Sous-objectifs :

  1.1. $A \mid\equiv N_a$

  1.2. $A \mid\equiv \beta$

  1.1. $B \mid\equiv N_b$

  1.2. $B \mid\equiv \alpha$

Les sous-objectifs 1.1 et 1.2 ainsi que 2.1 et 2.2 permettent de déduire respectivement les objectifs 1 et 2, dans le sens où chaque entité doit croire à la validité des deux parties de la clé pour croire qu'elles ont vraiment échangé la clé secrète. Ils sont alors dérivés directement des objectifs définis et peuvent reposer sur des hypothèses.

Définition des hypothèses et preuve :

Nous ne pouvons pas définir d'hypothèses pour ce protocole puisqu'aucune clé n'est connue ou partagée auparavant.

— A choisi $N_a \xleftarrow{\$} \mathbb{Z}_q$:
- $A \mid\equiv \#(N_a)$

— Implicitement on déduit :
- $A \mid\equiv N_a$ **(sous-objectif 1.1.)**

— A calcule $\alpha = g^{N_a}$:
- $A \mid\equiv \alpha$

— A envoie α à B :
- $B \triangleleft \alpha$

— B choisi $N_b \xleftarrow{\$} \mathbb{Z}_q$:
- $B \mid\equiv \#(N_b)$
- $B \mid\equiv N_b$ **(sous-objectif 1.2.)**

— B calcule $\beta = g^{N_b}$:
- $B \mid\equiv \beta$

— B envoie β à A :
- $A \triangleleft \beta$

Nous constatons que seuls deux sous-objectifs ont été atteints par le protocole et qu'aucun objectif ne peut être déduit de ces derniers, ce qui implique qu'aucun objectif défini pour ce protocole ne peut être atteint. Les sous-objectifs retrouvés correspondent au fait que les deux entités font confiance aux valeurs qu'ils ont créées eux même mais pas l'inverse. Ainsi, A par exemple ne peut pas croire que β est vrai, ceci étant dû au manque de signature, de secret ou de clé pré-partagée entre les deux entités. Ainsi, il n'est pas possible de déduire qu'une entité fait confiance à une valeur envoyée par l'autre entité. Nous verrons dans l'exemple suivant, l'impact d'une signature sur le protocole.

4.6.2 Semi-statique Diffie-Hellman

Il s'agit d'une variante très importante et utile du protocole Diffie-Hellman initial évoqué plus haut. Le protocole Diffie-Hellman semi-statique fixe une clé publique d'un serveur. Cette clé reste constante pendant de longues périodes et l'authenticité de cette dernière peut être vérifiée d'une manière ou d'une autre (par exemple en utilisant des certificats signés). Elle est ainsi utilisée par tous ceux qui souhaitent communiquer en toute sécurité avec le serveur en échangeant une clé secrète. Ce mécanisme est particulièrement utile pour sécuriser les connexions anonymes des clients. Il est actuellement utilisé par TLS 1.3 McKay et Cooper [2018].

Idéalisation du protocole

Soit q un nombre premier et g un générateur de $\mathbb{Z}_q$. Les valeurs de q et g sont connues. Le protocole d'échange de clés de Diffie-Hellman semi-statique entre deux entités A et B où B est le serveur procède comme suit :

1. A choisi un élément aléatoire N_a de $\mathbb{Z}_q$.
2. A calcule $\alpha = g^{N_a}$.
3. A envoie α à B
4. B signe α et β .
5. B envoie sa clé publique $\beta = g^{N_b}$ et $Sign(\beta||\alpha)$.
4. A calcule $K = \beta^{N_a}$.
5. B calcule $K = \alpha^{N_b}$.

— Les participants : A et B sont les entités impliquées dans le protocole.

— Les messages : α, β et $Sign(\beta||\alpha)$ sont les messages échangés.

— Les clés : K_B, K_B^{-1} est la paire de clés asymétriques de B et K la clé secrète échangée.

Définition des objectifs et sous-objectifs du protocole

L'objectif que nous allons définir pour ce protocole est l'échange d'une clé secrète entre A et B où seule A doit avoir la certitude que la clé est vraiment échangée avec B, ce qui se traduit en :

- Objectifs :

  1 . $A \mid\equiv A \overset{K}{\leftrightarrow} B$

  2 . $B \mid\equiv A \mid\equiv A \overset{K}{\leftrightarrow} B$

Ces objectifs permettent de vérifier si A estime lui-même que la clé K est vraiment échangée entre A et B et que B estime également que A croit en la clé.

- Sous-objectifs :

  1.1. $A \mid\equiv N_a$

  1.2. $A \mid\equiv \beta$

  2.1. $A \mid\equiv B \mid\equiv \alpha$

  2.2. $A \mid\equiv B \mid\equiv \beta$

Définition des hypothèses et preuve :

Dans ce protocole, une partie du message échangé est signé avec la clé privée du serveur. Afin de lire le message, il est nécessaire de le vérifier avec la clé publique. Il est supposé que toutes les entités détiennent la clé publique du serveur B, mais aussi que B croit en sa clé publique et également que A croit que B contrôle sa clé publique.

Hypothèse :

$A \mid\equiv \overset{K_B}{\longmapsto} B$

$B \mid\equiv \beta$

$A \mid\equiv B \Rightarrow \beta$

Les trois hypothèses présentées ci-dessus sont implicites, du fait que B détient un certificat permettant de vérifier la validité de sa clé publique. Il est supposé que toute entité peut avoir accès aux clés publiques soit sur un annuaire, soit directement auprès d'une autorité de confiance. Ainsi, A estime que β, c'est-à-dire K_B, est vraiment la clé publique de B et que ce dernier a le contrôle de sa clé.

Preuve :

- A choisi $N_a \xleftarrow{\$} \mathbb{Z}_q$:

1. $A \mid\equiv \#(N_a)$
2. $A \mid\equiv N_a$ **(sous-objectif 1.1.)**

- A calcule $\alpha = g^{N_a}$:

3. $A \mid\equiv \#(\alpha)$
4. $A \mid\equiv \alpha$

- A envoie α à B :

5. $B \triangleleft \alpha$
6. $B \mid\equiv \beta$

- B envoie sa clé publique $\beta = g^{N_b}$ et $Sign(\beta||\alpha)$:

7. $A \triangleleft \beta, \{\beta,\alpha\}_{K_B^{-1}}$

   L'hypothèse 1 et la formule 7 permet de déduire le postulat suivant : $\left(\frac{A\mid\equiv \xmapsto{K_B} B, A\triangleleft\{\beta,\alpha\}_{K_B^{-1}}}{A\mid\equiv B\mid\sim(\beta,\alpha)}\right)$
8. $A \mid\equiv B \mid\sim (\beta,\alpha)$

   La formule 3. permet de déduire le postulat suivant : $\frac{A\mid\equiv\#(\alpha)}{A\mid\equiv\#(\alpha,\beta)}$
9. $A \mid\equiv \#(\beta)$

   La formule 9 et 8 permet de déduire le postulat suivant : $\left(\frac{A\mid\equiv\#(\beta), A\mid\equiv B\mid\sim\beta}{A\mid\equiv B\mid\equiv\beta}\right)$
10. $A \mid\equiv B \mid\equiv \beta$ **(sous-objectif 2.1.)**

    L'hypothèse 2 et la formule 10 permet de déduire le postulat suivant : $\left(\frac{A\mid\equiv B\Rightarrow\beta, A\mid\equiv B\mid\equiv\beta}{A\mid\equiv\beta}\right)$
11. $A \mid\equiv \beta$ **(sous-objectif 1.2.)**

    La formule 3 et 8 permet de déduire le postulat suivant : $\left(\frac{A\mid\equiv\#(\alpha), A\mid\equiv B\mid\sim\alpha}{A\mid\equiv B\mid\equiv\alpha}\right)$
12. $A \mid\equiv B \mid\equiv \alpha$ **(sous-objectif 2.2.)**

Le premier sous-objectif est atteint implicitement, vu que la valeur de N_a est générée directement par A. Le deuxième sous-objectif est atteint cette fois puisque B détient un certificat de sa clé publique. Nous estimons ainsi que B a le contrôle sur sa clé publique ce qui nous permet d'arriver à la conclusion que A croit en β. Les deux autres sous-objectifs sont atteints grâce à la signature émise par B. Nous retrouvons alors que tous les sous-objectifs que nous avons définis ont été atteints, ce qui implique que le protocole répond aux objectifs définis.

4.7 Analyse de sécurité de notre protocole

4.7.1 Extension du BAN logique

Afin de pouvoir utiliser le BAN logique pour la preuve de notre protocole, nous proposons une extension de ce dernier. Certaines notions ne sont pas formalisées et doivent être incluses dans la preuve comme les clés de re-chiffrement ainsi que le re-chiffrement. Les nouveaux termes que nous allons utiliser pour formaliser ces notions seront :

- Clé de re-chiffrement : $K_{A\rightarrow B}$
- Re-chiffrement : $\{\{X\}_{K_A}\}_{K_{A\rightarrow B}}$

Nous définissons aussi des nouvelles règles permettant de faire des déductions à partir de nouveaux postulats utilisant les termes que nous venons de définir.

Le re-chiffrement d'un message X chiffré par la clé publique de A via une clé de re-chiffrement de A vers B correspond au chiffrement du message X avec la clé publique de B à condition que

l'entité qui exécute le re-chiffrement connaisse la clé de re-chiffrement :

$$\frac{P \mid\equiv \xrightarrow{K_{Q\to P}} Q, P \lhd \{\{X\}_{K_Q}\}_{K_{Q\to P}}}{P \lhd \{X\}_{K_P}}$$

La deuxième règle concerne un cas atypique que nous avons utilisé dans notre protocole. Il s'agit de l'utilisation d'une paire de clés asymétriques secrètes. Dans ce cas, la clé de chiffrement sera secrète ce qui n'est pas le cas d'utilisation habituelle d'un système de chiffrement asymétrique. Ceci nous permet de vérifier l'authenticité des messages chiffrés pour des entités bien définies. Ces entités doivent avoir reçu une clé de re-chiffrement avec laquelle ils pourront re-chiffrer puis déchiffrer le message. Si ces deux processus sont valides alors le message chiffré a bien été créé par le détenteur de la clé secrète. Nous intégrons ce principe selon la règle suivante :

$$\frac{P \mid\equiv Q \Rightarrow +K, P \lhd \{X\}_K, P \lhd X}{P \mid\equiv Q \mid\sim X}$$

Ainsi, nous supposons que si P croit que Q a le contrôle de sa paire de clés asymétriques secrètes $+(K, K^{-1})$ et que P arrive à voire le message chiffré avec la clé K et son correspondant en clair, alors nous déduisons que P croit que Q a dit ce message.

4.7.2 Idéalisation du protocole

Rappel sur le protocole :

- C choisi $r \xleftarrow{\$} \{0,1\}^{l_0}$.
- C calcule $Auth\_req = Enc(pki, id||v||r)$.
- C envoie $Auth\_req$ à l'IdP
- IdP calcule $Dec(ski, Auth\_req)$.
- IdP vérifie la validité de id & v.
- IdP calcule $Auth\_token = Enc(pks, scope)$; $scope = (timestamp||expire\_date||r)$.
- IdP envoie $Auth\_token$ à C.
- C choisi $k \xleftarrow{\$} \{0,1\}^{l_1}$.
- C calcule $Eph\_token = Enc(pkp, pki||r||k)$.
- C envoie $Auth\_resp = Auth\_token||Eph\_token$ au SP.
- Le SP calcule $Dec(skp, ReEncrypt(rk_{s\to p}, Auth\_token))$;
- Le SP calcule $Dec(skp, Eph\_token)$.
- Le SP vérifie la validité de $scope$ & r.
- C calcule $Auth\_token||Enc(pkp, pki||r + x||k)$.

— Les participants : les entités impliquées dans le protocole sont le fournisseur d'identités (IdP), l'utilisateur (C) et le fournisseur de services SP.

— Les messages : Eph_token, $Auth\_token$ et $Auth\_resp$ sont les messages échangés.

— Les clés : K_P correspond à la clé publique du fournisseur de services Pkp , K_S la clé Pks , K_I la clé publique Pki, $K_{S\to P}$ à la clé de re-chiffrement et K la clé de session échangée en fin de protocole entre l'utilisateur et le fournisseur de services.

4.7.3 Définition des objectifs et sous-objectifs du protocole

Pour notre protocole, ce qui permet d'authentifier l'utilisateur est en première partie la validation de l'identité de ce dernier par son IdP. Ceci se traduit par le chiffrement de la valeur aléa-

toire générée par l'utilisateur C avec la clé secrète des SP détenue par l'IdP. Puisque les informations d'identités ne sont pas divulguées aux SP et que la valeur aléatoire peut être interceptée, une deuxième partie est nécessaire pour la validation de l'authentification auprès du SP. Elle consiste à chiffrer la même valeur aléatoire mais cette fois par l'utilisateur à destination du SP. Ceci peut se traduire par un partage de message secret r. La première validation consiste dans le fait que l'IdP croit que C a partagé un secret avec lui. La deuxième validation permet de faire croire au SP que le même secret a été partagé avec lui par l'IdP.

- Objectifs :

  1 . SP $|\equiv$ IdP $\overset{r}{\leftrightharpoons}$ SP

  2 . IdP $|\equiv$ C $\overset{r}{\leftrightharpoons}$ IdP

Ces objectifs permettent de vérifier si l'IdP estime avoir échangé un secret avec l'utilisateur C (Objectif 2.). De la même manière, ils permettent de vérifier si le SP a vraiment échangé le même secret avec l'IdP (Objectif 1.).

- Sous-objectifs :

  1.1. IdP $|\equiv r$

  1.2. SP $|\equiv$ IdP $|\equiv r$

  2.1. C $|\equiv r$

  2.2. IdP $|\equiv$ C $|\equiv r$

Pour que SP croit qu'il vient d'échanger un secret avec l'IdP, il faut que l'IdP croit en la validité du secret partagé et que SP estime que l'IdP croit en la validité de ce dernier. Le même raisonnement s'applique pour l'échange de secret entre l'IdP et C.

4.7.4 Définition des hypothèses et preuve :

Comme expliqué dans la section 3.1, notre protocole suppose qu'une phase d'enregistrement et d'établissement d'une relation de confiance soit pré-établie. Elle se traduit par l'échange de messages secrets entre le fournisseur d'identités et l'utilisateur, ce qui correspond à (id, v). Au niveau de la confiance entre le fournisseur d'identités et le fournisseur de services elle se traduit par l'échange de la clé de re-chiffrement. Ainsi, nous supposons que le SP détient la clé de re-chiffrement et qu'il estime que l'IdP contrôle la clé asymétrique secrète à travers laquelle la clé de re-chiffrement a été générée. Nous supposons aussi que les "timestamps" sont considérés comme valides par les différentes entités. La dernière hypothèse suppose que l'IdP estime que C a le contrôle de l'aléa qu'il génère. Cette hypothèse ne peut pas s'appliquer sur le SP puisqu'il n'a aucune relation pré-établie avec l'utilisateur. Par contre, au niveau de l'IdP, les aléas reçus sont toujours énoncés avec l'identifiant de l'utilisateur en question, nous pouvons ainsi faire cette supposition.

Hypothèse :

1. IdP $|\equiv \overset{\mathrm{K_I}}{\longmapsto}$ IdP
2. SP $|\equiv \overset{\mathrm{K_P}}{\longmapsto}$ SP
3. SP $|\equiv \overset{\mathrm{K_{S\to P}}}{\longmapsto}$ SP
4. SP $|\equiv$ IdP $\Rightarrow +\mathrm{K_S}$
5. IdP $|\equiv$ C $\overset{(id,v)}{\Longleftrightarrow}$ IdP
6. IdP $|\equiv \#(\mathrm{T}s)$; SP $|\equiv \#(\mathrm{T}s)$
7. IdP $|\equiv$ C $\Rightarrow r$

Preuve :

- C choisi $r \xleftarrow{\$} \{0,1\}^{l_0}$:

1. C $|\equiv \#(r)$
2. C $|\equiv r$ **(sous objectif 2.1.)**

- C envoie $\mathrm{A}uth\_req = \mathrm{E}nc(pki, id||v||r)$ à l'IdP :

3. $\mathrm{I}d\mathrm{P} \lhd \{(id, v, r)\}_{\mathrm{K_I}}$

   L'hypothèse 1 et la formule 3 permettent de déduire le postulat : $\frac{\mathrm{I}d\mathrm{P}|\equiv \overset{\mathrm{K_I}}{\longmapsto} \mathrm{I}d\mathrm{P}, \mathrm{I}d\mathrm{P} \lhd \{\mathrm{X}\}_{\mathrm{K_I}}}{\mathrm{I}d\mathrm{P} \lhd \mathrm{X}}$
4. $\mathrm{I}d\mathrm{P} \lhd (id, v, r)$
5. $\mathrm{I}d\mathrm{P} \lhd \langle r \rangle_{(id,v)}$

   L'hypothèse 5 et la formule 5 permettent de déduire le postulat : $\frac{\mathrm{I}d\mathrm{P}|\equiv \mathrm{C} \overset{\mathrm{Y}}{\rightleftharpoons} \mathrm{I}d\mathrm{P}, \mathrm{I}d\mathrm{P} \lhd \langle \mathrm{X} \rangle_{\mathrm{Y}}}{\mathrm{I}d\mathrm{P}|\equiv \mathrm{C}|\sim \mathrm{X}}$
6. $\mathrm{I}d\mathrm{P} |\equiv \mathrm{C} |\sim r$
7. $\mathrm{I}d\mathrm{P} |\equiv \#(r)$ *

   La formule 6 et 7 permettent de déduire le postulat : $\frac{\mathrm{I}d\mathrm{P}\#(r), \mathrm{I}d\mathrm{P}|\equiv \mathrm{C}|\equiv r}{\mathrm{I}d\mathrm{P}|\equiv r}$
8. $\mathrm{I}d\mathrm{P} |\equiv \mathrm{C} |\equiv r$ **(sous objectif 2.2.)**

   L'hypothèse 7 et la formule 8 permettent de déduire le postulat : $\frac{\mathrm{I}d\mathrm{P}|\equiv \mathrm{C} \Rightarrow r, \mathrm{I}d\mathrm{P}|\equiv \mathrm{C}|\sim r}{\mathrm{I}d\mathrm{P}|\equiv \mathrm{C}|\equiv r}$
9. $\mathrm{I}d\mathrm{P} |\equiv r$ **(sous objectif 1.1.)**

- IdP envoie $\mathrm{A}uth\_token$ à C :

10. $\mathrm{C} \lhd \{r, \mathrm{T}_s\}_{\mathrm{K_S}}$

- C choisi $k \xleftarrow{\$} \{0,1\}^{l_1}$.

11. C $|\equiv \#(k)$

- C envoie $\mathrm{A}uth\_resp = \mathrm{A}uth\_token||\mathrm{E}ph\_token$ au SP :

12. $\mathrm{SP} \lhd (\{r, \mathrm{T}_s\}_{\mathrm{K_S}}, \{r, k\}_{\mathrm{K_P}})$
13. $\mathrm{SP} \lhd \{r, k\}_{\mathrm{K_P}}$

    L'hypothèse 2 et la formule 13 permettent de déduire le postulat : $\frac{\mathrm{SP}|\equiv \overset{\mathrm{K_P}}{\longmapsto} \mathrm{SP}, \mathrm{SP} \lhd \{\mathrm{X}\}_{\mathrm{K_P}}}{\mathrm{SP} \lhd \mathrm{X}}$
14. $\mathrm{SP} \lhd (r, k)$
15. $\mathrm{SP} \lhd \{r, \mathrm{T}_s\}_{\mathrm{K_S}}$

- Le SP calcule $\mathrm{D}ec(\mathrm{K^{-1}_P}, \mathrm{R}e\mathrm{E}ncrypt(\mathrm{K_{S \to P}}, \mathrm{A}uth\_token))$;

  L'hypothèse 3 et la formule 15 permettent de déduire le postulat : $\frac{\mathrm{SP}|\equiv \overset{\mathrm{K_{S \to P}}}{\longmapsto} \mathrm{SP}, \mathrm{SP} \lhd \{\{\mathrm{X}\}_{\mathrm{K_S}}\}_{\mathrm{K_{S \to P}}}}{\mathrm{SP} \lhd \{\mathrm{X}\}_{\mathrm{K_P}}}$
16. $\mathrm{SP} \lhd \{r, \mathrm{T}_s\}_{\mathrm{K_P}}$

    L'hypothèse 2 et la formule 16 permettent de déduire le postulat : $\frac{\mathrm{SP}|\equiv \overset{\mathrm{K_P}}{\longmapsto} \mathrm{SP}, \mathrm{SP} \lhd \{\mathrm{X}\}_{\mathrm{K_P}}}{\mathrm{P} \lhd \mathrm{X}}$
17. $\mathrm{SP} \lhd (r, \mathrm{T}_s)$

    L'hypothèse 6 permet de déduire le postulat : $\frac{\mathrm{SP}|\equiv \#(\mathrm{T}_s)}{\mathrm{SP}|\equiv \#(r, \mathrm{T}_s)}$
18. $\mathrm{SP} |\equiv \#(r, \mathrm{T}_s)$

    L'hypothèse 4 et les formules 15 et 17 permettent de déduire le postulat : $\frac{\mathrm{SP}|\equiv \mathrm{I}d\mathrm{P} \Rightarrow +\mathrm{K_S}, \mathrm{SP} \lhd \{(r, \mathrm{T}_s)\}_{\mathrm{K_S}}, \mathrm{SP} \lhd (r, \mathrm{T}_s)}{\mathrm{SP}|\equiv \mathrm{I}d\mathrm{P}|\sim (r, \mathrm{T}_s)}$
19. $\mathrm{SP} |\equiv \mathrm{I}d\mathrm{P} |\sim (r, \mathrm{T}_s)$
20. $\mathrm{SP} |\equiv \mathrm{I}d\mathrm{P} |\equiv (r, \mathrm{T}_s)$
21. $\mathrm{SP} |\equiv \mathrm{I}d\mathrm{P} |\equiv r$ **(sous objectif 1.2.)**

Nous retrouvons ainsi que les objectifs que nous avons définis ont été atteints grâce à notre protocole puisque les sous-objectifs qui en dérivent ont tous été atteints.

4.7.5 Discussion

L'objectif principal de notre protocole est d'authentifier l'utilisateur en question auprès du fournisseur de services. Néanmoins, le modèle BAN ne nous permet pas de présenter de manière directe cet objectif. Afin de l'interpréter par la logique BAN, nous avons repris les éléments qui permettent à l'utilisateur d'être authentifié. L'élément central est la valeur aléatoire générée par l'utilisateur. Si la valeur reçue depuis l'IdP par le SP via l'utilisateur est la même que celle qui a été échangée entre l'utilisateur et l'IdP, nous considérons que ce dernier a été identifié par l'IdP et que sa demande d'authentification a été validée par l'IdP. Elle est ainsi valide pour le SP. Cette valeur aléatoire n'est jamais révélée en clair mais envoyée à l'IdP et au SP chiffré. Cette valeur peut être vue alors comme étant un secret partagé entre C et l'IdP et entre l'IdP et le SP. Le même secret n'est pas considéré comme étant partagé directement entre C et le SP puisque seule l'IdP peut vérifier l'identité de l'utilisateur. Aussi, seul l'IdP peut faire valider la demande d'authentification de C auprès du SP, d'où la définition de ces deux objectifs : $\mathrm{SP} \mid\equiv \mathrm{I}d\mathrm{P} \overset{r}{\leftrightharpoons} \mathrm{SP}$ & $\mathrm{I}d\mathrm{P} \mid\equiv \mathrm{C} \overset{r}{\leftrightharpoons} \mathrm{I}d\mathrm{P}$.

Afin de faciliter l'enchaînement de la preuve, nous définissons d'abord des sous-objectifs dérivés des objectifs définis auparavant ce qui nous permet de diminuer le nombre de déductions à faire. Pour le premier objectif, afin que le SP croit que l'échange du secret est vrai entre lui et l'IdP, il faut que l'IdP lui-même croit en la validité du secret et que le SP, à son tour, estime que l'IdP croit en ce secret. De la même façon, nous avons dérivé les sous-objectifs du deuxième objectif.

Une fois les objectifs et sous-objectifs définis, nous avons présenté nos hypothèses. Les deux premières hypothèses concernent la détention des clés publiques par les différentes entités à savoir l'IdP et le SP. Nous présentons aussi la détention de la clé de re-chiffrement par le SP ainsi que la clé de chiffrement de l'IdP pour les messages d'authentification à destination des SP. Nous supposons aussi l'existence d'un secret partagé entre l'utilisateur et l'IdP. Toutes nos hypothèses jusqu'à maintenant sont directement liées à la phase d'enregistrement et d'établissement d'une relation de confiance. Néanmoins, d'autres hypothèses devaient être présentées comme la confiance des différentes entités pour les "timestamps" spécifiquement pour les IdP et les SP. Quant à la dernière hypothèse, nous supposons que la valeur générée par l'utilisateur est contrôlée par ce dernier. Puisque cette valeur est toujours associée à des identifiants vérifiables par l'IdP alors il croit que cette valeur est vraiment contrôlée par l'utilisateur.

Maintenant que les phases préliminaires du BAN logique ont été établies alors, il ne reste plus qu'à suivre l'évolution du protocole. En utilisant les différentes règles que nous avons présentées, nous arrivons à déduire les différents sous-objectif et ainsi vérifier que le protocole atteint les objectifs que nous lui avons définis.

4.8 Conclusion

Dans ce chapitre, nous avons d'abord présenté les différentes solutions existantes concernant la délégation d'authentification, en présentant leurs avantages et leurs inconvénients. Tous ces systèmes utilisent des signatures ou des chiffrements symétriques avec des clés pré-partagées et, par conséquent, aucune de ces solutions ne peut être instanciée efficacement en utilisant un modèle de sécurité basé sur le modèle standard. De plus, la communication directe entre les IdP et les SP est presque inévitable. Nous proposons alors une solution de délégation d'authentification originale basée sur l'utilisation des PRE SBAI et collab. [2021]. Notre solution ne nécessite aucune interaction entre les deux entités IdP et SP pendant la phase d'authentification et peut donc se faire de manière asynchrone. En termes de complexité, l'ensemble de la procédure peut être réalisée en cinq transitions dans le pire des cas, y compris la phase de découverte de l'IdP, et dans le

cas idéal, en trois transitions (1.0, 1.1, 2.0), ce qui reste inférieur aux autres solutions existantes. En effet, la solution la moins coûteuse en termes de nombre de communications nécessite au moins quatre interactions, par exemple avec OpenID en mode de flux implicite. Dans les autres cas, il faut au moins cinq transitions. Avec l'aide du PRE, nous pouvons utiliser le même jeton pendant toute sa durée de vie pour nous authentifier auprès de plusieurs SP, sans avoir à revenir aux IdP pour chaque connexion. Pour la répartition des tâches, une première authentification nécessitera deux chiffrements du côté utilisateur, un déchiffrement et un chiffrement pour l'IdP, et enfin un re-chiffrement et deux déchiffrements du côté du SP. Il est également possible que le processus de re-chiffrement soit effectué par un service indépendant pour alléger les SP.

Concernant la preuve de sécurité, nous nous sommes basés sur la méthode du BAN logique. Elle nous a permis de modéliser le protocole sous forme de postulats logiques et de prouver que notre protocole atteint l'objectif que nous lui avons défini. Ainsi, à l'aide des hypothèses, nous avons utilisé certaines règles pré-définies de la logique BAN mais nous avons aussi intégré d'autre règles pour l'adapter à l'utilisation des proxy de re-chiffrement. Ceci nous a amenés à déduire de nouveaux postulats qui nous ramènent aux objectifs définis.

Dans ce chapitre, nous passons brièvement en revue certaines solutions SSO et de délégation d'authentification. Nous avons ensuite présenté un nouveau protocole de délégation d'authentification basé sur l'utilisation du PRE. Grâce à ce protocole, un utilisateur peut s'authentifier une fois, recevoir l' Auth_token de l'IdP et utiliser tout service qui fait confiance à l'IdP de l'utilisateur pendant la durée de vie du jeton. Les questions d'autorisations n'ont pas été prises en compte dans ce travail mais ont été laissées pour une étude future. Les travaux futurs impliqueront l'examen de la solution NUNEZ et collab. [2012] et incluront une couche d'autorisations pour proposer un cadre complet d'authentification et de contrôle d'accès.

Conclusion générale

Le volume et la valeur des données augmentent de façon exponentielle. Les innovations technologiques ont introduit de nouvelles préoccupations en matière de confidentialité et de sécurité de l'information. Afin d'éviter toutes utilisations malveillantes des données, les données sensibles doivent alors être chiffrées. Ainsi seules les personnes possédant une clé de déchiffrement peuvent accéder à ces données. Cependant, plusieurs scénarios nécessitent le partage de ces données entre différentes entités.

Le proxy de re-chiffrement (PRE) est l'outil le mieux adapté à cette problématique puisqu'il permet de déléguer les droits de déchiffrement aux entités auxquelles le droit d'accès est accordé. Dans la première partie, nous commençons par définir certaines notions de base utiles pour la compréhension des constructions des différents crypto-systèmes étudiés. Par la suite, nous nous sommes concentrés sur la définition de sécurité des systèmes de chiffrement. Des exemples ont été présentés afin d'illustrer ces notions à travers des preuves permettant de vérifier la robustesse et/ou la vulnérabilité de l'algorithme en question. Nous avons conclu cette partie avec la présentation du contexte et la problématique du projet, à savoir la protection des données au sein du projet VERTPOM.

Dans la deuxième partie, nous avons présenté la définition du PRE ainsi que les différentes propriétés qu'il peut avoir. Certaines propriétés sont primordiales comme la résistance aux collusions tandis que d'autres sont liées au contexte et scénario d'utilisation. Nous avons par la suite cité les différents PRE proposés dans la littérature ainsi que les propriétés et le niveaux de sécurité atteints par ces derniers. Pour résoudre notre problématique, nous avons choisi d'utiliser le PRE en tant que service et nous avons créé la première librairie en JavaScript pour leur utilisation. Nous avons implémenté dans un premier temps l'algorithme de Chow et collab. [2010]. La première version étant vulnérable, nous avons implémenté sa variante proposée par Selvi et collab. [2017] tout en présentant la différence entre les deux variantes. Notre implémentation, permet d'optimiser les ressources en termes de temps et de stockage grâce à notre instanciation à base de courbes elliptiques. Le travail présenté dans cette partie a fait l'objet d'une publication dans la conférence internationale SECRYPT 2019 Sbai et collab. [2019].

En troisième partie, nous avons repris la définition de sécurité des PRE. Une fois cette notion de sécurité des PRE est formalisée, la vérifiabilité publique s'est avérée également très importante pour la construction d'un PRE sécurisé. Nous avons alors présenté les différentes méthodes permettant de concevoir un système de chiffrement publiquement vérifiable à partir d'une primitive de chiffrement asymétrique. Par la suite, nous avons étudié les différentes étapes de construction de l'algorithme de Chow et collab. [2010]. Cet algorithme répond à l'une des deux problématiques posé par Canetti et Hohenberger [2007], à savoir, concevoir un PRE unidirectionnel CCA sans utiliser le couplage. Nous nous sommes concentrés sur la deuxième problématique resté en suspens qui concerne la conception d'un PRE CCA dans le modèle standard sans utiliser le couplage. Nous montrons que le seul algorithme prétendant avoir résolu ce problème est vulnérable. Nous avons alors conçu notre propre PRE à base de l'algorithme de Cramer et Shoup [1998].Étant donné que la vérifiabilité publique ne peut pas être atteinte efficacement dans le modèle standard. Nous nous sommes basé sur la propriété de vérifiabilité désigné qui nous a obligé à utiliser une paire de clé au niveau des proxy. À la fin de ce chapitre, nous avons présenté la preuve de sécurité

de notre PRE. Ce travail a fait l'objet d'une publication dans la conférence internationale ICISSP 2020 SBAI et collab. [2020a].

La quatrième partie est consacrée à la problématique de délégation d'authentification. Nous avons commencé par présenter les méthodes existantes, puis l'idée générale de notre proposition. En utilisant la même technologie étudiée dans les chapitres précédents, nous montrons qu'il est possible de concevoir un protocole de délégation d'authentification en utilisant les PRE. Notre solution permet d'authentifier les utilisateurs de manière asynchrone, sans communication directe entre les fournisseurs d'identité et les fournisseurs de services. Afin de prouver la sécurité du protocole, nous nous sommes basés sur la logique BAN. Cette méthode ne pouvait pas être utilisé directement pour notre protocole puisqu'elle ne permettait pas de modéliser les différentes fonctionnalités d'un PRE. Une adaptation a été alors proposée, permettant d'intégrer de nouvelles règles prenant en compte l'utilisation des PRE. Le travail présenté dans cette partie a fait l'objet d'une publication dans la conférence internationale SECRYPT 2021 SBAI et collab. [2021].

Lors de la conception des différentes solutions proposées, certaines n'ont pas été testées à grande échelle. En perspectives à ce travail, il conviendra alors de mener des tests de performance et d'intégration du PREaaS ainsi que notre protocole de délégation d'authentification, afin de calculer le temps de réponse, les ressources et la capacité du PREaaS en conditions réelles.

Outre la nécessité de l'expérimentation de ces méthodes en conditions réelles, on peut discerner dès à présent quelques directions de recherche intéressantes :

- Étudier les méthodes de mapping des messages vers des points de courbes elliptiques, que ce soit des fonctions à sens unique utilisées par exemple pour l'implémentation de Hashed ElGamal ou des fonctions inversibles pouvant être utilisées pour l'implémentation classique d'ElGamal sur courbe elliptique.

- Continuer l'analyse et la conception des PRE avec paire de clés. En effet, ce type de PRE est rarement conçu vu la contrainte de transparence, mais grâce à ce dernier nous avons pu concevoir un PRE sécurisé contre les attaques CCA dans le modèle standard sans utiliser le couplage.

- Enfin, l'approche que nous avons utilisée au dernier chapitre pour prouver la sécurité du protocole de délégation d'authentification devrait faire l'objet d'une étude plus approfondie : créer une nouvelle variante du BAN logique incluant les nouveaux types de primitives cryptographiques existants et pourquoi pas créer un outils de modélisation et de vérification de protocole à base de cette variante.

Références

AONO, Y., X. BOYEN, L. WANG et collab.. 2013, «Key-private proxy re-encryption under lwe», dans *International Conference on Cryptology in India*, Springer, p. 1–18. 39

ATENIESE, G., K. BENSON et S. HOHENBERGER. 2009, «Key-private proxy re-encryption», dans *Cryptographers' Track at the RSA Conference*, Springer, p. 279–294. 38

ATENIESE, G., K. FU, M. GREEN et S. HOHENBERGER. 2006, «Improved proxy re-encryption schemes with applications to secure distributed storage», *ACM Transactions on Information and System Security (TISSEC)*, vol. 9, nº 1, p. 1–30. 35, 36, 38, 66

BARKER, E., L. CHEN et D. MOODY. 2014, *Recommendation for pair-wise key-establishment schemes using integer factorization cryptography*, US Department of Commerce, National Institute of Standards and Technology. 17

BELLARE, M. et P. ROGAWAY. 1993, «Random oracles are practical : A paradigm for designing efficient protocols», dans *Proceedings of the 1st ACM conference on Computer and communications security*, p. 62–73. 26

BEN-AROYA, I. et E. BIHAM. 1996, «Differential cryptanalysis of lucifer», *Journal of Cryptology*, vol. 9, nº 1, p. 21–34. 14

BENALOH, J. et M. DE MARE. 1993, «One-way accumulators : A decentralized alternative to digital signatures», dans *Workshop on the Theory and Application of of Cryptographic Techniques*, Springer, p. 274–285. 78

BLAZE, M., G. BLEUMER et M. STRAUSS. 1998, «Divertible protocols and atomic proxy cryptography», dans *International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, p. 127–144. 34, 38

BLEICHENBACHER, D. 1998, «Chosen ciphertext attacks against protocols based on the rsa encryption standard pkcs# 1», dans *Annual International Cryptology Conference*, Springer, p. 1–12. 22, 25

BONEH, D. et M. FRANKLIN. 2001, «Identity-based encryption from the weil pairing», dans *Annual international cryptology conference*, Springer, p. 213–229. 20, 32, 37

BONEH, D., A. SAHAI et B. WATERS. 2011, «Functional encryption : Definitions and challenges», dans *Theory of Cryptography Conference*, Springer, p. 253–273. 19

BORONAT, J.-P. 2017, «Véritable énergie du territoire positif et modulaire», URL http://www.ciac-it.com/images/Presse_Vertpom_09-10-2017.pdf. 5, 7

BOYD, C. et W. MAO. 1993, «On a limitation of ban logic», dans *Workshop on the Theory and Application of of Cryptographic Techniques*, Springer, p. 240–247. 86

BOYLE, E., S. GOLDWASSER et I. IVAN. 2014, «Functional signatures and pseudorandom functions», dans *International workshop on public key cryptography*, Springer, p. 501–519. 19

BRINKHAUS, S., D. CARLUCCIO, U. GREVELER, B. JUSTUS, D. LÖHR et C. WEGENER. 2011, «Smart hacking for privacy», dans *Proceeding of the 28th Chaos Communication Congress (28C3)*. 7

BURROWS, M., M. ABADI et R. M. NEEDHAM. 1989, «A logic of authentication», *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, vol. 426, nº 1871, p. 233–271. 83

CANETTI, R. 2001, «Universally composable security : A new paradigm for cryptographic protocols», dans *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, IEEE, p. 136–145. 38

CANETTI, R., O. GOLDREICH et S. HALEVI. 2004a, «The random oracle methodology, revisited», *J. ACM*, vol. 51, nº 4, doi :10.1145/1008731.1008734, p. 557–594, ISSN 0004-5411. URL https://doi.org/10.1145/1008731.1008734. 54

CANETTI, R., S. HALEVI et J. KATZ. 2004b, «Chosen-ciphertext security from identity-based encryption», dans *International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, p. 207–222. 57

CANETTI, R. et S. HOHENBERGER. 2007, «Chosen-ciphertext secure proxy re-encryption», dans *Proceedings of the 14th ACM conference on Computer and communications security*, ACM, p. 185–194. 38, 54, 63, 65, 66, 69, 72, 97

CANTOR, S., J. MOREH, R. PHILPOTT et E. MALER. 2004, «Metadata for the oasis security assertion markup language (saml) v2. 0», *OASIS Standard (March 2005)*. 75

CHAIDOS, P. et G. COUTEAU. 2018, «Efficient designated-verifier non-interactive zero-knowledge proofs of knowledge», dans *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, p. 193–221. 75

CHOW, S. S., J. WENG, Y. YANG et R. H. DENG. 2010, «Efficient unidirectional proxy re-encryption», dans *International Conference on Cryptology in Africa*, Springer, p. 316–332. 38, 39, 45, 47, 53, 54, 57, 58, 61, 62, 63, 65, 67, 69, 70, 71, 97

COOL, D. et A. D. KEROMYTIS. 2005, «Conversion and proxy functions for symmetric key ciphers», dans *International Conference on Information Technology : Coding and Computing (ITCC'05)-Volume II*, vol. 1, IEEE, p. 662–667. 37

CRAMER, R. et V. SHOUP. 1998, «A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack», dans *Annual International Cryptology Conference*, Springer, p. 13–25. 26, 27, 54, 57, 63, 67, 69, 70, 71, 97

DDN, D. D., C. DWORK et M. NAOR. 1991, «Non-malleable cryptography», dans *Proceedings of the 23rd Annual Symposium on the Theory of Computing, ACM*. 57

DE SANTIS, A., Y. DESMEDT, Y. FRANKEL et M. YUNG. 1994, «How to share a function securely», dans *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, p. 522–533. 56

DENG, R. H., J. WENG, S. LIU et K. CHEN. 2008a, «Chosen-ciphertext secure proxy re-encryption without pairings», dans *International Conference on Cryptology and Network Security*, Springer, p. 1–17. 38, 39

DENG, R. H., J. WENG, S. LIU et K. CHEN. 2008b, «Chosen-ciphertext secure proxy re-encryption without pairings», dans *International Conference on Cryptology and Network Security*, Springer, p. 1–17. 53, 57, 58, 59, 60, 61, 62, 69

DIFFIE, W. et M. HELLMAN. 1976, «New directions in cryptography», *IEEE transactions on Information Theory*, vol. 22, nº 6, p. 644–654. 16

EGOROV, M., D. NUÑEZ et M. WILKISON. 2018, «Nucypher : A proxy re-encryption network to empower privacy in decentralized systems», . 40

ELGAMAL, T. 1985, «A public key cryptosystem and a signature scheme based on discrete logarithms», *IEEE transactions on information theory*, vol. 31, nº 4, p. 469–472. 16, 18

FEISTEL, H. 1974, «Block cipher cryptographic system», US Patent 3,798,359. vii, 14

FIAT, A. et M. NAOR. 1993, «Broadcast encryption», dans *Annual International Cryptology Conference*, Springer, p. 480–491. 32

FIAT, A. et A. SHAMIR. 1986, «How to prove yourself : Practical solutions to identification and signature problems», dans *Conference on the theory and application of cryptographic techniques*, Springer, p. 186–194. 21, 78

FUJISAKI, E. et T. OKAMOTO. 1999, «Secure integration of asymmetric and symmetric encryption schemes», dans *Annual International Cryptology Conference*, Springer, p. 537–554. 19

GOLDREICH, O., S. MICALI et A. WIGDERSON. 1986, «How to prove all np statements in zero-knowledge and a methodology of cryptographic protocol design», dans *Conference on the Theory and Application of Cryptographic Techniques*, Springer, p. 171–185. 21, 27

GOLDWASSER, S. et S. MICALI. 1984, «Probabilistic encryption», *Journal of computer and system sciences*, vol. 28, nº 2, p. 270–299. 22, 28

GOLDWASSER, S., S. MICALI et C. RACKOFF. 1989, «The knowledge complexity of interactive proof systems», *SIAM Journal on computing*, vol. 18, nº 1, p. 186–208. 57

GONG, L., R. M. NEEDHAM et R. YAHALOM. 1990, «Reasoning about belief in cryptographic protocols.», dans *IEEE Symposium on Security and Privacy*, Citeseer, p. 234–248. 86

GREEN, M. et G. ATENIESE. 2007, «Identity-based proxy re-encryption», dans *Applied Cryptography and Network Security*, Springer, p. 288–306. 37, 38

GUERON, S. et V. KRASNOV. 2015, «Fast prime field elliptic-curve cryptography with 256-bit primes», *Journal of Cryptographic Engineering*, vol. 5, nº 2, p. 141–151. 49

HARDT, D., J. BUFU et J. HOYT. 2007, «Openid attribute exchange 1.0-final», *at, Dec*, vol. 5, p. 11. 76

HASAN, M. M. et H. T. MOUFTAH. 2015, «Encryption as a service for smart grid advanced metering infrastructure», dans *Computers and Communication (ISCC), 2015 IEEE Symposium on*, IEEE, p. 216–221. 39

HIARY, G. 2016, «A deterministic algorithm for integer factorization», *Mathematics of Computation*, vol. 85, nº 300, p. 2065–2069. 12

HIROSE, S. 2010, «On re-encryption for symmetric authenticated encryption», dans *Computer Security Symposium (CSS)*, vol. 2010. 37

HOFFSTEIN, J., J. PIPHER et J. H. SILVERMAN. 1998, «Ntru : A ring-based public key cryptosystem», dans *International Algorithmic Number Theory Symposium*, Springer, p. 267–288. 39

ICART, T. 2009, «How to hash into elliptic curves», dans *Annual International Cryptology Conference*, Springer, p. 303–316. 49

IVAN, A.-A. et Y. DODIS. 2003, «Proxy cryptography revisited.», dans *NDSS*. 35, 38

JIVANYAN, A., R. YEGHIAZARYAN, A. DARBINYAN et A. MANUKYAN. 2015, «Secure collaboration in public cloud storages», dans *CYTED-RITOS International Workshop on Groupware*, Springer, p. 190–197. 40

KANG, S., B. VEERAVALLI et K. M. M. AUNG. 2014, «Espresso : An encryption as a service for cloud storage systems», dans *IFIP International Conference on Autonomous Infrastructure, Management and Security*, Springer, p. 15–28. 39

KIRSHANOVA, E. 2014, «Proxy re-encryption from lattices», dans *International Workshop on Public Key Cryptography*, Springer, p. 77–94. 38, 39

LIBERT, B. et D. VERGNAUD. 2008, «Unidirectional chosen-ciphertext secure proxy re-encryption», dans *International Workshop on Public Key Cryptography*, Springer, p. 360–379. 38, 63, 69

MATROSOV, A., E. RODIONOV, D. HARLEY et J. MALCHO. 2010, «Stuxnet under the microscope», *ESET LLC (September 2010)*. 6

MCKAY, K. et D. COOPER. 2018, «Guidelines for the selection, configuration, and use of transport layer security (tls) implementations (2nd draft)», cahier de recherche, National Institute of Standards and Technology. 88

MERKLE, R. et M. HELLMAN. 1978, «Hiding information and signatures in trapdoor knapsacks», *IEEE transactions on Information Theory*, vol. 24, nº 5, p. 525–530. 17

NAOR, M. et M. YUNG. 1990, «Public-key cryptosystems provably secure against chosen ciphertext attacks», dans *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, p. 427–437. 57

NEEDHAM, R. M. et M. D. SCHROEDER. 1978, «Using encryption for authentication in large networks of computers», *Communications of the ACM*, vol. 21, nº 12, p. 993–999. 75

NIETO, J. M. G., M. MANULIS, B. POETTERING, J. RANGASAMY et D. STEBILA. 2012, «Publicly verifiable ciphertexts», dans *International Conference on Security and Cryptography for Networks*, Springer, p. 393–410. 57

NUNEZ, D., I. AGUDO et J. LOPEZ. 2012, «Integrating openid with proxy re-encryption to enhance privacy in cloud-based identity services», dans *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings*, IEEE, p. 241–248. 76, 95

NUÑEZ, D., I. AGUDO et J. LOPEZ. 2015, «Ntrureencrypt : An efficient proxy re-encryption scheme based on ntru», dans *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, p. 179–189. 38, 39

NUÑEZ, D., I. AGUDO et J. LOPEZ. 2017, «Proxy re-encryption : Analysis of constructions and its application to secure access delegation», *Journal of Network and Computer Applications*, vol. 87, p. 193–209. 39, 40

NUÑEZ, D., I. AGUDO et J. LOPEZ. 2015, «A parametric family of attack models for proxy re-encryption», dans *2015 IEEE 28th Computer Security Foundations Symposium*, p. 290–301, doi : 10.1109/CSF.2015.27. 55, 56

PERERA, C., C. H. LIU et S. JAYAWARDENA. 2015, «The emerging internet of things marketplace from an industrial perspective : A survey», *IEEE Transactions on Emerging Topics in Computing*, vol. 3, nº 4, p. 585–598. 1

POPA, R. A., N. ZELDOVICH et H. BALAKRISHNAN. 2011, «Cryptdb : A practical encrypted relational dbms», . 39

PURUSHOTHAMA, B., B. SHRINATH et B. AMBERKER. 2013, «Secure cloud storage service and limited proxy re-encryption for enforcing access control in public cloud», *International Journal of Information and Communication Technology*, vol. 5, nº 2, p. 167–186. 65, 66

QIN, Z., H. XIONG, S. WU et J. BATAMULIZA. 2016, «A survey of proxy re-encryption for secure data sharing in cloud computing», *IEEE Transactions on Services Computing*. 39, 48

RIVEST, R. L. 1997, «All-or-nothing encryption and the package transform», dans *International Workshop on Fast Software Encryption*, Springer, p. 210–218. 37

RIVEST, R. L., A. SHAMIR et L. ADLEMAN. 1978, «A method for obtaining digital signatures and public-key cryptosystems», *Communications of the ACM*, vol. 21, nº 2, p. 120–126. 16, 17

RONEN, E., A. SHAMIR, A.-O. WEINGARTEN et C. O'FLYNN. 2017, «Iot goes nuclear : Creating a zigbee chain reaction», dans *Security and Privacy (SP), 2017 IEEE Symposium on*, IEEE, p. 195–212. 6

SAHAI, A. et B. WATERS. 2005, «Fuzzy identity-based encryption», dans *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, p. 457–473. 19, 32

SAKURAI, K., T. NISHIDE et A. SYALIM. 2017, «Improved proxy re-encryption scheme for symmetric key cryptography», dans *2017 International Workshop on Big Data and Information Security (IWBIS)*, IEEE, p. 105–111. 37

SBAI, A., C. DROCOURT et G. DEQUEN. 2019, «Pre as a service within smart grid cities», dans *16th International Conference on Security and Cryptography*. 51, 97

SBAI, A., C. DROCOURT et G. DEQUEN. 2020a, «Cca secure unidirectional pre with key pair in the standard model without pairings», dans *6th International Conference on Information Systems Security and Privacy*. 71, 98

SBAI, A., C. DROCOURT et G. DEQUEN. 2020b, «Encrypted data sharing using proxy reencryption in smart grid», dans *International Conference on Electronic Engineering and Renewable Energy*, Springer, p. 161–167. 72

SBAI, A., C. DROCOURT et G. DEQUEN. 2021, «A new delegated authentication protocol based on pre», dans *18th International Conference on Security and Cryptography*. 94, 98

SCHNORR, C.-P. 1989, «Efficient identification and signatures for smart cards», dans *Conference on the Theory and Application of Cryptology*, Springer, p. 239–252. 21, 26

SCHNORR, C. P. et M. JAKOBSSON. 2000, «Security of signed elgamal encryption», dans *International Conference on the Theory and Application of Cryptology and Information Security*, Springer, p. 73–89. 26

SELVI, S. S. D., A. PAUL et C. PANDURANGAN. 2017, «A provably-secure unidirectional proxy re-encryption scheme without pairing in the random oracle model», dans *International Conference on Cryptology and Network Security*, Springer, p. 459–469. 38, 39, 45, 47, 48, 49, 50, 54, 58, 61, 62, 71, 97

SHAMIR, A. 1979, «How to share a secret», *Commun. ACM*, vol. 22, nº 11, doi :10.1145/359168.359176, p. 612–613, ISSN 0001-0782. URL https://doi.org/10.1145/359168.359176. 56

SHAMIR, A. 1982, «A polynomial time algorithm for breaking the basic merkle-hellman cryptosystem», dans *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, IEEE, p. 145–152. 17

SHAMIR, A. 1984, «Identity-based cryptosystems and signature schemes», dans *Workshop on the theory and application of cryptographic techniques*, Springer, p. 47–53. 19

SHANNON, C. E. 1948, «A mathematical theory of communication», *The Bell system technical journal*, vol. 27, n° 3, p. 379–423. 10

SHANNON, C. E. 1949, «Communication theory of secrecy systems», *The Bell system technical journal*, vol. 28, n° 4, p. 656–715. 10, 22, 26

SHAO, J. et Z. CAO. 2009, «Cca-secure proxy re-encryption without pairings», dans *International Workshop on Public Key Cryptography*, Springer, p. 357–376. 39, 63

SHAO, J., Z. CAO, X. LIANG et H. LIN. 2010, «Proxy re-encryption with keyword search», *Information Sciences*, vol. 180, n° 13, p. 2576–2587. 37

SORKIN, A. 1984, «Lucifer, a cryptographic algorithm», *Cryptologia*, vol. 8, n° 1, p. 22–42. 14

STARK, E., M. HAMBURG et D. BONEH. 2013, «Stanford javascript crypto library», . 49

SYALIM, A., T. NISHIDE et K. SAKURAI. 2011, «Realizing proxy re-encryption in the symmetric world», dans *International conference on informatics engineering and information science*, Springer, p. 259–274. 37

TANG, L., L. OUYANG et W.-T. TSAI. 2015, «Multi-factor web api security for securing mobile cloud», dans *Fuzzy Systems and Knowledge Discovery (FSKD), 2015 12th International Conference on*, IEEE, p. 2163–2168. 51

TIBOUCHI, M. et T. KIM. 2017, «Improved elliptic curve hashing and point representation», *Designs, Codes and Cryptography*, vol. 82, n° 1-2, p. 161–177. 49

TREMEL, E. 2013, «Real-world performance of cryptographic accumulators», *Undergraduate Honors Thesis, Brown University*. 78

WANG, A. X., W. WU et X. YANG. 2009, «On ddos attack against proxy in re-encryption and re-signature», *Engineering College of Chinese, PR China*. 54, 67, 70

WATANABE, Y., J. SHIKATA et H. IMAI. 2003, «Equivalence between semantic security and indistinguishability against chosen ciphertext attacks», dans *International Workshop on Public Key Cryptography*, Springer, p. 71–84. 25

WEI, P., X. A. WANG et X. YANG. 2010, «Proxy re-encryption schemes with proxy having its own public/private keys», dans *2010 2nd International Workshop on Database Technology and Applications*, IEEE, p. 1–4. 65, 66

XAGAWA, K. et K. TANAKA. 2010, «Proxy re-encryption based on learning with errors (mathematical foundation of algorithms and computer science)», . 38, 39

ZHANG, M., X. A. WANG, W. LI et X. YANG. 2013, «Cca secure publicly verifiable public key encryption without pairings nor random oracle and its applications.», *JCP*, vol. 8, n° 8, p. 1987–1994. vii, 53, 54, 56, 63, 64, 65, 66, 71

Liste des publications

PRE as a Service within Smart Grid City

In the context of Smart Grid Cities, legal obligations require that certain personal data must be stored in the long term and protected. To deal with confidentiality issues, we use the concept of Proxy Re-Encryption (PRE) which allows sharing encrypted data. We present the first implementation of the Chow's algorithm, and propose an optimized instantiation thanks to elliptic curves. This is the first unidirectional algorithm with CCA security that does not rely on pairing, which guarantees its high performance. This allows its use in real conditions. We have implemented it in JavaScript for direct use in a web browser by the user. In order to be able to process the data asynchronously, we then define the notion of PREaaS (Proxy Re-Encryption as a Service) that also allows use in a service-oriented context.
The 16th International Conference on Security and Cryptography (SECRYPT 2019)

CCA secure unidirectional PRE with key pair in the standard model without pairings

Secure Data sharing has become an ubiquitous need. One way of pursuing it is to use Proxy Re-Encryption (PRE), which allows delegation of decryption rights selectively. This work tackles the problem of designing a Proxy Re-Encryption that is unidirectional and CCA-secure in the standard model without pairings. In (Zhang et al., 2013) they propose a solution that makes the Cramer-Shoup encryption scheme publicly verifiable and use their result to construct a CCA secure PRE in the standard model. However, we show that their scheme is vulnerable against adaptive chosen ciphertexts attacks. Then we propose a new construction based on CramerShoup crypto-system (Cramer and Shoup, 1998), that is CCA secure without pairings nor random oracle.
The 6th International Conference on Information Systems Security and Privacy (ICISSP 2020)

Encrypted data sharing using Proxy ReEncryption in smart grid

In a rapidly changing territory, energy networks must be increasingly responsive and flexible. New models of multi-fluid management and energy production are being created and developed on national and international level. This involves the use, monitoring and supervision of many sensors that reports lot of data. This paper deals with the secure management of large amounts of data within the context of smart grid. We propose a solution based on proxy re-encryption designed primarily to allow decryption delegation, which allow a neat management of large amount of data while respecting the GDPR (General Data Protection Regulation) and security standards
2nd International Conference on Electronic Engineering and Renewable Energy 2020 (ICEERE 2020)

A New Delegated Authentication Protocol based on PRE

New trends highlight the use of delegated authentication solutions where identity providers do not need to synchronize user credentials with services. It is a facility for service providers and also for users who do not have to create multiple accounts. Different solutions for single sign-on and delegated authentication exist. Most of these solutions require many exchanges between

the different actors involved in the protocol, an additional TLS layer and/or the use of signature schemes which, in terms of security, rely on random oracles for reasons of efficiency. In this article, we recall the concept of the best known solutions (e.g. Kerberos, OpenID, ...), briefly discuss the possibility of using oneway accumulators and define the Proxy Re-Encryption (PRE). Next, we propose a new delegated authentication protocol that allows users to authenticate anonymously on insecure networks and therefore asynchronously without direct communication between identity providers and service providers while minimizing the number of interactions. We based our solution on the use of PRE which could be instantiated by schemes based on standard assumptions.
The 18th International Conference on Security and Cryptography (SECRYPT 2021)

Résumé

La cyber sécurité est un enjeu majeur pour le SmartGrid et les industries énergétiques. La manipulation des données issues des compteurs intelligents peut avoir des conséquences néfastes, particulièrement lorsque les systèmes de comptage sont connectés directement aux sources de production.

Dans le cadre du projet VertPom, nous nous sommes intéressés à deux problématiques majeures : la confidentialité des données de consommation et les systèmes d'authentification. Pour répondre aux problématiques de confidentialité, nous avons utilisé le concept des proxy de rechiffrement (PRE) qui permet le partage de données chiffrées. Nous avons étudié les systèmes existants et nous nous sommes intéressés aux constructions bénéficiant d'une sécurité CCA dans le modèle standard qui n'utilisent pas le couplage. Nous montrons l'existence d'une vulnérabilité dans un PRE existant puis nous proposons une nouvelle construction basée sur le système de chiffrement de Cramer-Shoup. Nous définissons aussi la notion de PREaaS (Proxy Re-Encryption as a Service) qui permet une utilisation dans un contexte orienté services.

S'agissant des problématiques d'authentification, nous présentons un nouveau protocole de délégation d'authentification. Notre solution permet aux utilisateurs de, s'authentifier anonymement sur des réseaux non sécurisés, de manière asynchrone sans communication directe entre les différents acteurs, tout en minimisant le nombre d'interactions.

Abstract

Cyber security is a major issue for the SmartGrid and energy industries. Manipulating data from smart meters can have harmful consequences, especially when metering systems are connected directly to production sources.

Within the VertPom project, we have focused on two major issues : the confidentiality of consumption data and authentication systems. To address the confidentiality issues, we used the concept of proxy re-encryption proxies (PRE) which allows the sharing of encrypted data. We have studied existing schemes and we are interested in constructions that achieve CCA security in the standard model without pairing. We point out a vulnerability in an existing PRE and then we propose a new construction based on the Cramer-Shoup encryption scheme. We also define the notion of PREaaS (Proxy Re-Encryption as a Service) which allows a use in a service-oriented context.

Regarding authentication issues, we present a new authentication delegation protocol. Our solution allows users to anonymously authenticate themselves on unsecured networks, asynchronously without direct communication between the different actors, while minimizing the number of interactions.