Sur quelques questions de cryptographie : Anonymat révocable et Une généralisation du chiffrement de Goldwasser-Micali

Davide Alessio

N° d'ordre : 4476 ANNÉE 2011

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : mathématiques et applications

Ecole doctorale Matisse

présentée par

Davide Alessio

préparée à l'unité de recherche 6625 du CNRS : IRMAR
Institut de Recherche Mathématique de Rennes
UFR de mathématiques

Sur quelques questions de cryptographie :

***Anonymat révocable* et**

Une généralisation du chiffrement de Goldwasser-Micali

Thèse soutenue à Rennes
le 13 décembre 2011

devant le jury composé de :

Roland GILLARD
Professeur émérite, Université Joseph-Fourier (Grenoble) / rapporteur

Jean-François MESTRE
Professeur, Université Denis-Diderot (Paris 7) / examinateur

Sylvain DUQUESNE
Professeur, Université de Rennes 1 / examinateur

Alain DURAND
Ingénieur, Technicolor / examinateur

Marc JOYE
Ingénieur, Technicolor / directeur de thèse

Antoine CHAMBERT-LOIR
Professeur, Université de Rennes 1 / directeur de thèse

REMERCIEMENTS

Je voudrais adresser mes premiers remerciements à mes directeurs de thèse, Marc Joye et Antoine Chambert-Loir, pour tout ce qu'ils m'ont apporté pendant ces années. Je les remercie pour leur patience, le temps qu'ils m'ont dédié pour discuter, répondre à mes questions et partager leur savoir. Marc a su m'initier à la sécurité prouvée, un sujet fondamental dans la recherche dans le domaine de la cryptographie. Antoine a su se rendre très disponible, m'aider dans un sujet qui ne pas le sien et donner un support mathématique à ma recherche.

Je tiens à remercier aussi Roland Guillard et Guillaume Hanrot pour avoir accepté d'être rapporteurs pour cette thèse. Une tâche sûrement alourdie par les nombreuse coquilles dont je l'avais farcie. Je les remercie pour avoir eu la patience et pris le temps de me les signaler au fin d'aboutir à une version, j'espère, plus agréable à lire.

Je remercie également Jean-François Mestre et Sylvain Duquesne pour leur disponibilité à compléter mon jury.

Je remercie chaleureusement Alain Durand pour faire partie du jury mais surtout pour son soutien pendant toute la durée de la thèse. Il a su y trouver de l'intérêt et il a été toujours disponible lorsque j'avais de questions. Il est à présent un des mes repères dans la vie professionnelle.

Je voudrais remercier tout le Security Lab de Technicolor où j'ai passé une très grand partie de temps de recherche. Je remercie en particulier Eric Diehl pour m'avoir accueillit au sein du laboratoire qu'il dirige et avoir rendu possible cette thèse CIFRE. Merci aussi aux (*ex*-)collègues pour avoir rendu les journées de travail très agréables, à partir des pauses café et déjeuner, sans oublier les pots. En discutant avec eux j'ai appris beaucoup dans le domaine de la sécurité informatique mais aussi sur la vie d'entreprise. Le deux chapitres en anglais, lors de leur soumission aux conférences, ont étés lus, relus et corrigés par de nombreux yeux, merci. Je remercie mon co-bureau Mohammed, qui avec Marc et Alain, compose l'équipe crypto, pour toutes les discussions qu'on a eu à propos de la cryptographie, pour m'avoir appris beaucoup sur le domaine du broadcast encryption et pour avoir éclairci mes nombreux doutes.

Je remercie Tancrède et Laila pour les nombreuses lectures et relectures du dernier chapitre et pour avoir signalé, entre autre, une erreur grossière qui avait échappée.

Merci enfin aux doctorants et le personnel administratif de l'IRMAR. Merci aux doctorants qui ont partagé leur bureau avec moi, Jérémy pour avoir rafraichi quelques détails de maths trop vieux dans ma mémoire, Mathilde et Clément, ... La présence d'Anna et Michele a sûrement aidé à satisfaire accompagné mes besoins de caféine.

Les remerciements ne seraient pas complets sans une pensée à tous les amis qui m'ont entouré et ont contribué à rendre ma vie à Rennes bien plus que agréable. Tous les nommer serait trop long et je risquerai d'en oublier, vous savez déjà qui vous êtes.

Un abbraccio particulier pour le p'tit groupe d'expats *ritals* qui me donne l'impression de n'avoir jamais quitté l'Italie. Je rappelle seulement les deux colonnes de ma *jeunesse* rennaise : Giulio et Andrea avec qui, en compagnie de quelques cocktails, j'ai refait le monde trop de fois. Ils sont à présent partis mais des très bons éléments ont pris le relais. Merci au Duc pour les kilomètres parcourus ensemble.

Merci à Isabelle et Patrick pour l'immense travail sur la première rédaction, pour lui avoir donnée une apparence française.

Ringrazio di tutto cuore i miei genitori Maurizio e Valentina, e mia sorella Sara per avermi sempre incoraggiato e sostenuto anche con questa lunga distanza che ci separa, per non avermi mai fatto pesare il fatto di esser partito da questo lato delle Alpi, tornare solo alle feste comandate e al contrario per aver sempre assecondato e sostenuto le mie scelte professionali. *Dai, Sara, adesso tocca a te !*

Merci Marie, avec toi tout est meilleur.

Molto bene ! Allons-y !

TABLE DES MATIÈRES

INTRODUCTION

Introduction et motivations.

Le chiffrement est sûrement une primitive fondamentale parmi les fonctions cryptographiques. Cela rend possible à deux parties, d'ordinaire appelées Alice et Bob, de communiquer au travers d'un canal non sécurisé en permettant qu'un adversaire, Eve, ne puisse pas comprendre leur communication.

Il pourrait sembler assez facile d'identifier les contraintes nécessaires pour le design d'un « bon » schéma de chiffrement. Il s'avère en réalité que la définition d'un modèle de sécurité rigoureux n'est pas trivial. Ceci dépend fortement du contexte. La sécurité même dépend du contexte.

Par exemple, la contrainte qu'il ne soit pas possible de récupérer le message en clair à partir du chiffré peut ne pas être suffisante dans certaines applications.

Organisation du document.

Ce document est le résultat d'un travail de thèse avec un contrat CIFRE (Conventions Industrielles de Formation par la REcherche [1]) financé par le ministère de l'Enseignement supérieur et de la Recherche, qui en confie la mise en œuvre à l'ANRT (Association Nationale de la Recherche et de la Technologie [2]). Pour mes travaux de thèse j'ai été intégré à l'équipe Security and Content Protection Labs [3] à Technicolor, Rennes. L'équipe, dirigée par Eric Diehl, compte une trentaine de personnes dans les deux laboratoires de Rennes et Hannover en Allemagne. L'équipe s'intéresse à la recherche dans plusieurs domaines autour de la protection des contenus pour les nouveaux media et la sécurité des données. Les domaines principaux de recherche et expertise sont : la cryptographie, le *watermarking* pour l'audio et le video, le *fingerprinting* video, les techniques de DRM (*Digital Rights Management*) pour applications professionnelles (adressées principalement à des grands producteurs et distributeurs de contenus), la protection de la copie pour les nouveaux media comme les disques Blu-Ray et la sécurité logicielle et des réseaux. Pendant mes années de thèse j'ai intégré le groupe Crypto du Security and Content Protection Labs, j'ai été encadré par Alain Durand et j'ai travaillé principalement avec Marc Joye.

Ce document s'ouvre avec un chapitre d'introduction à la cryptographie à clé publique. Nous donnerons les définitions et fixerons la notation pour la suite. En particulier nous allons illustrer formellement comment un algorithme de chiffrement est composé et nous

1. `http://www.anrt.asso.fr/fr/espace_cifre/accueil.jsp`
2. `http://www.anrt.asso.fr/`
3. `http://www.technicolor.com/en/hi/technology/advanced-development-center/security-content-protection-labs`

donnerons aussi le détail des sous algorithmes. Nous décrirons ensuite comment la sécurité d'une telle primitive est évaluée en définissant de façon rigoureuse un attaquant, en particulier cela signifie fixer son but et les moyens dont il dispose. Ce chapitre touchera aussi le chiffrement basé sur l'identité, une variante du chiffrement à clé publique introduit par Shamir pour résoudre des problèmes d'architecture. Finalement nous rappellerons la définition du chiffrement *broadcast* (ou le chiffrement pour diffusion) qui vise à trouver des solutions cryptographiques dans les cas où le destinataire du chiffré est un groupe de nombreux utilisateurs.

Le reste de ce document s'organise autour de deux parties. La première partie concerne l'*anonymat révocable*. Les résultats présentés font l'objet de deux papiers publiés à deux colloques internationaux sur les DRM. Le premier, résultat d'un travail fait en collaboration avec Marc Joye, (chapitre 2) a été présenté et publié au *Digital Rights Management Workshop 2009* à Chicago, Illinois, USA et illustre une méthode simple pour avoir un schéma à clé publique qui garantit l'anonymat du destinataire avec en plus la propriété que l'anonymat peut être révoqué sous certaines conditions. Des exemples d'utilisation sont donnés dans le chapitre. Nous proposons aussi une variante qui applique les principes du chiffrement basé sur l'identité. Dans le chapitre suivant (chapitre 3) nous nous intéressons au problème analogue dans un contexte du chiffrement pour la diffusion. Ce chapitre a été présenté et publié sous forme d'article au *7th IEEE International Workshop on Digital Rights Management Impact on Consumer Communications (DRM 2011)* à Las Vegas, Nevada.

La deuxième partie est consacrée à la généralisation d'un *schéma de chiffrement à clé publique*. Le schéma original est dû à Goldwasser et Micali. Dans leur travail les auteurs ont eu l'intuition, en premier, que, comme nous l'avons dit dans l'introduction, la sécurité dépend du contexte et ils ont donné, pour la première fois, la définition de sécurité sémantique. Notre travail généralise leur schéma en fournissant une famille de schémas de chiffrement où l'on peut retrouver l'original pour un choix particulier d'un paramètre. Notre travail est motivé par la recherche de l'amélioration de l'efficacité (en termes de bande passante) du schéma original, afin de pouvoir chiffrer des messages plus longs dans un chiffré de la même taille. Ce travail a généré un brevet qui est actuellement en cours de dépôt (**PCT/EP11/066883**).

Dans l'Annexe A nous pouvons retrouver un article publié avec Marc Éluard et Yves Maetz dans les *Proceedings of the 29th International Conference on Consumer Electronics (2011)* et présenté à la conférence en janvier 2011. Il s'agit d'un travail qui a généré deux brevets (**PCT/EP11/057345** et **EP 11160566.3**) et qui traite de mot de passe graphique. Il s'agit d'une méthode qui permet de remplacer, pendant l'authentification à un système, la donnée d'un code alphanumérique par une image. Les avantages sont à rechercher dans la facilité qu'a l'être humain à se rappeler de quelque chose si celle ci est associée à une image. Il existe des systèmes de mot de passe graphique, comme celui présenté dans ce document, qui demandent à l'utilisateur de créer une image en plaçant des objets sur un fond de son choix afin de créer une image plus complexe et « unique ». Un tel mot de passe présente aussi l'avantage d'être bien distribué et de rendre une attaque par dictionnaire plus difficile, parce qu'il n'existe pas d'ensemble d'images « ordinaire » comme l'on peut trouver pour les mots d'une langue.

CHAPITRE 1

CHIFFREMENT À CLÉ PUBLIQUE

1.1. Chiffrement à clé publique

Le chiffrement est un outil apte à empêcher toute entité à l'exception du (ou des) destinataire légitime d'accéder au contenu d'un message échangé. De plus, cet outil protège la confidentialité du message en empêchant que même une partie de l'information (autre que la taille) soit découverte. Historiquement les premiers schémas étaient appelés *symétriques* (ou à clé secrète), c'est à dire qu'une même clé peut être utilisée pour le chiffrement et pour le déchiffrement. C'est à partir du 1976 avec le travail de Diffie-Hellman [**DH76**] que la cryptographie moderne, dite *asymétrique* (ou à clé publique), fait sa première apparition dans la littérature. Contrairement à la cryptographie à clé secrète, deux clés différentes sont utilisées dans les deux phases de chiffrement et déchiffrement : l'émetteur chiffre des messages pour le destinataire en utilisant la clé publique de ce dernier, après l'avoir récupérée d'un serveur public. Le destinataire, de son côté, pour accéder au contenu du chiffré, utilise sa clé privée.

Nous allons donner maintenant la définition formelle d'un schéma de chiffrement à clé publique comme une suite d'algorithmes probabilistes.

Définition 1.1.1 **(Schéma de chiffrement à clé publique)**
Un schéma de chiffrement à clé publique est formé de quatre algorithmes :

- $\mathsf{Setup}(\lambda)$ **:** Cet algorithme est exécuté au tout début, il prend en entrée un paramètre de sécurité λ et donne en sortie les paramètres communs publics pour le schéma de chiffrement.
- $\mathsf{KeyGen}(\lambda)$ **:** *Algorithme de génération des clés.* Cet algorithme prend en entrée le paramètre de sécurité λ et les éventuels paramètres communs publics et donne en sortie un couple de clés : $(\mathsf{pk}, \mathsf{sk})$; où pk est la clé publique de chiffrement et sk est la clé secrète de déchiffrement.
- $\mathsf{Encrypt}(m, \mathsf{pk})$ **:** *Algorithme de chiffrement.* Cet algorithme prend en entrée le message à chiffrer m et la clé publique du destinataire pk et donne en sortie le message chiffré $c = \mathsf{Encrypt}(m, \mathsf{pk})$.
- $\mathsf{Decrypt}(c, \mathsf{sk})$ **:** *Algorithme de déchiffrement.* Cet algorithme prend en entrée un message chiffré c et une clé secrète sk et donne en sortie le message en clair m si le chiffré est valide et la clé secrète correspond à celle utilisée pour le chiffrement ou un symbole qui indique la non réussite (normalement indiqué par $\perp$) sinon.

Le premier algorithme, $\mathsf{Setup}(\cdot)$, est optionnel. Il existe de nombreux schémas de chiffrement qui ne requièrent pas une génération des paramètres communs. Nous allons en présenter quelques uns dans la suite.

Un tel schéma de chiffrement doit répondre à deux exigences de base :

Complétude : Un chiffré correctement engendré sera correctement déchiffré ;

Sécurité : La seule connaissance du chiffré et des paramètres publics ne doit pas être suffisante pour retrouver *facilement* le message original (ceci signifiant qu'inverser la fonction de chiffrement doit être difficile sans la connaissance de la clé secrète).

1.1.1. Notions de Sécurité. — L'intérêt d'un schéma de chiffrement est de protéger la communication entre les deux parties face à de nombreuses et différentes attaques. Dans le cours de l'histoire de la cryptologie différentes notions de sécurité ont été définies, à partir du travail de Goldwasser et Micali [**GM84**].

Afin de pouvoir décrire et définir la sécurité d'un schéma de chiffrement à clé publique nous devons d'abord définir le *scénario d'attaque*. Pour cela il faut donner le *but* de l'adversaire, à savoir la propriété du schéma qu'il cherche à casser et les *moyens* pour mener l'attaque, c'est à dire les ressources dont il dispose pour atteindre son but.

Nous commençons d'abord en classant les types d'attaques en fonction de la difficulté (décroissante) du but ; ensuite, nous nous attacherons à définir les adversaires en fonction des leurs moyens.

1.1.1.1. Le but de l'adversaire. — Un adversaire peut se mettre contre un schéma de chiffrement en essayant de récupérer :

- *la clé secrète*. L'adversaire récupère la clé secrète, utilisée pour le déchiffrement, de l'utilisateur attaqué, ceci permettant de déchiffrer tous messages générés avec la clé publique correspondante. Ce but d'attaque est le plus important et, en cas de réussite, on parle de schéma *totalement cassé*. Il est d'ordinaire noté UBK, de l'anglais *unbreakability*.
- *le message en clair*. L'adversaire vise à inverser la fonction de déchiffrement pour pouvoir accéder à tout message en clair sans, pour autant, connaître la clé secrète de l'utilisateur attaqué. Le but que cet attaque vise est de casser le sens unique de la fonction (à trappe) de chiffrement et ceci est normalement noté OW, de l'anglais *one-wayness*. En fait, qu'il trouve la clé ou non, la méthode par laquelle l'adversaire arrive à obtenir les messages en clair n'est pas important en pratique, le résultat est le même, le schéma est cassé.

Les deux notions que nous venons de décrire sont les buts de sécurité de base et sont les exigences minimales pour avoir un schéma sûr. Plus récemment deux nouvelles notions plus faibles ont été introduites et donc les schémas qui résistent à ces attaques sont plus sûrs :

- *la sécurité sémantique* : le but de cette attaque est d'obtenir (au moins) un bit d'information. Introduite par Goldwasser et Micali en 1984 [**GM84**] la sécurité sémantique est définie au travers d'un jeu entre l'adversaire et un challengeur. L'adversaire fournit deux messages en clair de même taille au challengeur qui en chiffre un des deux choisi au hasard. L'adversaire doit *distinguer* quel message a été choisi au hasard et a été chiffré. L'adversaire réussit s'il devine avec une probabilité meilleure qu'une réponse aléatoire. Cette notion est appelée aussi indistinguabilité (*indistinguishability* en anglais) et est notée IND dans la littérature. Elle signifie qu'un adversaire ne

peut déterminer entre les messages en clair qu'il a choisi lequel a été chiffré. Dans la pratique la sécurité sémantique contre les attaques dans le cas où les messages en clair possibles sont dans un ensemble « restreint » (exemple typique : une réponse affirmative ou négative).

– *la non-malléabilité* : cette notion a été introduite par Dolev, Dwork et Naor ([**DDN91, DDN00**]) puis reprise dans [**BDPR98, BS99**] et est un affaiblissement de la notion sur la sécurité sémantique. L'adversaire est capable de manipuler un chiffré c, sans connaître le clair correspondant, afin de produire un nouveau chiffré c' de telle sorte que les deux déchiffrements vérifient entre eux une relation non triviale. En simplifiant c'est à dire qu'il est capable de manipuler l'information contenue dans un chiffré sans la connaître directement. Cette notion est normalement notée NM de l'anglais *non-malleability*.

  Il a été démontré dans [**BDPR98**] que, dans le cas d'attaques à chiffré choisi, les deux notions sont équivalentes.

1.1.1.2. Les moyens de l'adversaire. — Nous passons maintenant à classer l'adversaire en fonction des moyens dont il dispose pour mener son attaque au schéma de chiffrement. Il peut donc monter une attaque :

– à *clair choisi* : l'adversaire possède seulement la clé publique de l'utilisateur attaqué et peut générer tout chiffré de son choix en chiffrant les messages selon sa volonté. Dans la littérature ceci est noté CPA, de l'anglais *chosen-plaintext attack*. Il est à noter que ce niveau est toujours accordé à l'adversaire, dans le contexte de la clé publique, en fait, l'adversaire est toujours capable de chiffrer tout message de son choix, en connaissant la clé (publique) attaquée.

– à *chiffré choisi* : l'adversaire peut accéder à un oracle pour déchiffrer tout chiffré de son choix (avec certaines contraintes : comme par exemple le nombre maximum de requêtes de déchiffrement, le chiffré challenge ne peut être pas déchiffré par l'oracle, ...). L'accès à cet oracle peut être du type adaptatif [**RS91**] ou non-adaptatif [**NY90**], suivant que l'adversaire peut modifier ses requêtes en fonction des réponses reçues précédemment ou non. Cet type d'adversaire est noté CCA-1 (resp. CCA-2) de l'anglais *non-adaptative chosen-ciphertext attack* (resp. *adaptative chosen-ciphertext attack*).

1.1.2. Niveau de sécurité d'un schéma de chiffrement à clé publique. — Quand on veut définir le niveau de sécurité d'un schéma de chiffrement à clé publique on le considère par rapport à sa résistance à un adversaire générique dont le but d'attaque et les moyens dont il dispose sont connus. La sécurité est définie par un couple $(\mathsf{but}, \mathsf{moyen}) \in \{\mathsf{UBK}, \mathsf{OW}, \mathsf{IND}, \mathsf{NM}\} \times \{\mathsf{CPA}, \mathsf{CCA\text{-}\{1,2\}}\}$.

Il est coutume de prouver la sécurité dans le *scénario* le plus défavorable possible pour l'utilisateur et de conséquence le plus favorable possible à l'adversaire, c'est à dire quand son *but* est le moins ambitieux et les *moyens* à sa disposition sont les plus puissants, à savoir le scénario NM-CCA-2. D'après le fait que sous une attaque à chiffré choisi la *non-malleabilité* est équivalente à la *sécurité sémantique*, il est normalement plus simple de démontrer la sécurité d'un schéma dans le scénario IND-CCA. De plus, comme nous le verrons par la suite, il est souvent suffisant de démontrer qu'un schéma atteint la sécurité face à des attaques à clair choisi (CPA).

La résistance à une attaque générique est décrite par le jeu suivant. Nous donnons le détail des étapes pour le jeu dans le cas IND-CCA, c'est à dire l'indistinguabilité contre une

attaque à chiffré choisi. Les acteurs d'un tel jeu sont au nombre de deux : le challengeur et l'adversaire. Le challengeur est le maître du jeu et peut générer une instance du schéma et donc la contrôler ; l'adversaire est mis en face d'une instance et il lui est demandé de casser le schéma.

Setup : Le challengeur exécute $\mathsf{Setup}(\lambda)$ et $\mathsf{KeyGen}(\lambda)$ afin d'obtenir une clé publique pk. Cette clé est donnée à l'adversaire et la clé secrète correspondante sk est gardée par le challengeur.

(Première) phase de requêtes : L'adversaire effectue de manière adaptative des requêtes de déchiffrement. Pour tout chiffré c de son choix soumis, le challengeur répond avec le clair $m = \mathsf{Decrypt}(c, \mathsf{sk})$ correspondant.

Challenge : Quand l'adversaire considère terminée la première phase de requête, il choisit deux messages m_0, m_1 de même taille et il les renvoie au challengeur. Le challengeur tire aléatoirement $b \leftarrow_R \{0,1\}$ et chiffre le message m_b pour obtenir le chiffré challenge : $c_* = \mathsf{Encrypt}(m_b, \mathsf{pk})$. Le challenge c_* est finalement renvoyé à l'adversaire.

(Seconde) phase de requêtes : L'adversaire peut recommencer une deuxième phase de requêtes de déchiffrement au challengeur, avec la contrainte de ne pas soumettre le chiffré challenge c_*.

Réponse : Terminée la seconde phase de requêtes, l'adversaire donne sa réponse $b' \in \{0,1\}$. L'adversaire gagne le jeu si $b' = b$ et autrement il perd.

1.1.3. Calcul de l'avantage. — L'adversaire $\mathcal{A}$ est vu comme un couple d'algorithmes probabilistes polynomiaux $(\mathcal{A}_1, \mathcal{A}_2)$ qui sont exécutés pendant les deux phases des requêtes. Généralement, il est noté avec s, *state* en anglais, une quantité calculée par $\mathcal{A}_1$, puis prise comme entrée par $\mathcal{A}_2$. Cette quantité sert à « faire communiquer » les deux composantes de $\mathcal{A}$ entre elles. Le nombre de requêtes de déchiffrement effectuées par l'adversaire $\mathcal{A}$ pendant les deux phases est noté, habituellement, avec q_D. Avec ce paramètre il est possible de calculer l'avantage $\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E}}(q_D, \mathcal{A})$ de l'adversaire $\mathcal{A}$ à gagner ce jeu et donc casser (au moins partialement) la sécurité du schéma de chiffrement :

$$\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E}}(q_D, \mathcal{A}) = \left| 2 \times \Pr\left[b' = b \;\middle|\; \begin{array}{c} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\lambda), (m_0, m_1, s) \leftarrow \mathcal{A}_1(\mathsf{pk}), \\ c_* = \mathsf{Encrypt}(m_b, \mathsf{pk}), \mathcal{A}_2(m_0, m_1, s, c_*) = b' \end{array} \right] - 1 \right|,$$

où la probabilité est calculée sur les choix aléatoires de b, les jetons aléatoires choisis par l'adversaire $\mathcal{A}$, du challengeur et des algorithmes probabilistes exécutés par ce dernier.

Précisons que la sécurité d'un schéma de chiffrement est mesurée au travers d'un *avantage* comme nous l'avons défini ci-dessous. Cette notion capture le vrai gain d'un tel adversaire. En effet il est possible de construire un simple adversaire qui retourne une valeur aléatoire b'. Un tel adversaire, comme on le voit aisément, a une probabilité de victoire dans le jeu précèdent de $\frac{1}{2}$. La notion d'avantage, donc, vise à capturer la différence entre la probabilité d'une réponse aléatoire et la probabilité que la bonne valeur de b soit effectivement devinée. La valeur est puis normalisée pour avoir des valeurs comme dans le contexte de la probabilité entre 0 (aucun avantage, l'adversaire devine aléatoirement une fois sur deux) et 1 (avantage parfait, l'adversaire devine chaque fois).

***Définition 1.1.2* (Niveau de sécurité IND-CCA).** — Il est défini (avec un abus de notation) par

$$\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E}}(q_D) = \sup_{\mathcal{A}\ \text{p.p.t}} \{\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E}}(q_D, \mathcal{A})\}$$

comme la borne supérieure des avantages, lorsque $\mathcal{A}$ varie parmi les algorithmes probabilistes polynomiaux en le paramètre de sécurité λ. Le schéma de chiffrement $\mathcal{E}$ est dit être (q_D)-IND-CCA sûr si $\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E}}(q_D)$ est une fonction qui tend vers 0, plus rapidement que l'inverse de toute fonction polynomiale, lors que le paramètre de sécurité λ tend vers l'infini.

Par analogie avec la définition précédente nous donnons la définition de la sécurité sémantique face à des attaques à clairs choisis, en interdisant simplement les requêtes de déchiffrement à l'adversaire :

Définition 1.1.3 **(Niveau de sécurité IND-CPA)**. — La définition précédente est valable aussi dans les deux cas de sécurité sémantique, le schéma de chiffrement $\mathcal{E}$ est dit être IND-CPA sûr si il est (0)-IND-CCA sûr.

Remarque 1.1.4. — Ici nous faisons deux remarques utiles pour la suite et qui sont importantes en pratique.

1.1.4. Transformations génériques de CPA à CCA. — Comme nous l'avons dit précédemment le niveau de sécurité CCA est plus élevé et donc il est préférable d'utiliser un schéma qui atteigne un tel niveau de sécurité. Dans la littérature il existe de nombreux schémas qui sont prouvés sûrs et résistants « seulement » aux attaques à clairs choisis (niveau CPA). Néanmoins, pour élever le niveau de sécurité à CCA de ces derniers algorithmes, il existe des transformations génériques. Il s'agit d'algorithmes qui prennent en entrée un schéma de niveau de sécurité CPA et donnent en sortie le même schéma modifié qui atteint le niveau de sécurité CCA. Nous pouvons citer comme exemples les transformations données par Fujisaki-Okamoto [**FO99a, FO99b**] ou REACT [**OP01**]. Le seul inconvénient possible de cet approche est une possible perte en efficacité. Il est possible que les transformations exigent une complexité ou une bande passante majeures.

1.1.5. Paradigme KEM-DEM. — Le chiffrement à clé publique a un désavantage : l'espace des clés publiques et secrètes, l'espace des messages en clair et chiffrés sont des sous-ensembles particuliers de la structure algébrique sur laquelle le schéma est basé. Ceci est souvent la cause d'une efficacité plus faible par rapport aux schémas symétriques. Ce problème devient plus important dès que la taille du message à chiffrer devient importante. Pour résoudre cette question, le paradigme KEM-DEM a été introduit. Il s'agit d'une méthode de chiffrement hybride composée d'une première partie (KEM : *key encapsulation method*) asymétrique et une seconde (DEM : *data encapsulation method*) symétrique. Suivant cette méthode, un message m à chiffrer sera chiffré sous un schéma asymétrique avec une clé aléatoire et cette dernière sera chiffrée à l'aide de la clé publique du destinataire. L'avantage de cette technique est de pouvoir chiffrer une petite quantité de données sous l'algorithme lent (asymétrique), en revanche la grosse partie des données est chiffrée avec un algorithme très efficace.

1.1.6. Exemples de schémas de chiffrement classiques. — Dans cette section nous allons illustrer quelques schémas de chiffrement classiques qui font l'objet de nos travaux de thèse dans les prochains chapitres.

1.1.6.1. Schéma de chiffrement El Gamal.— Le schéma de chiffrement El Gamal a été présenté par El Gamal dans [**Gam84**]. Donnons d'abord sa description :

$\mathsf{Setup}(\lambda)$ **:** L'algorithme $\mathsf{Setup}(\lambda)$ est exécuté avec le paramètre de sécurité λ comme entrée. Cela donne en sortie un groupe cyclique $\mathbb{G}$, d'ordre premier p, où la taille du premier p dépend du paramètre de sécurité. L'algorithme choisit ensuite un générateur g.

Les paramètres publics sont : $(g, p, \mathbb{G})$.

$\mathsf{KeyGen}(\lambda)$ **:** Pour générer sa propre paire de clés, l'utilisateur tire aléatoirement un entier non nul $x \in \mathbb{Z}/p\mathbb{Z}$ et il calcule $h = g^x$ dans $\mathbb{G}$.

La clé privée de l'utilisateur est : $\mathsf{sk} = x$, et sa clé publique est : $\mathsf{pk} = (h, g, \mathbb{G})$.

$\mathsf{Encrypt}(m, \mathsf{pk})$ **:** Pour chiffrer un message $m \in \mathbb{G}$, l'envoyeur récupère la clé publique du destinataire $\mathsf{pk} = (h, g, \mathbb{G})$, il tire une valeur aléatoire non nul $r \in \mathbb{Z}/p\mathbb{Z}$ et il calcule les deux quantités :

$$c_1 = h^r \cdot m \text{ et } c_2 = g^r.$$

Le chiffré en sortie est $c = (c_1, c_2)$.

$\mathsf{Decrypt}(c, \mathsf{sk})$ **:** Au destinataire, pour déchiffrer c, suffit de calculer $m = c_2 \cdot c_1^{-x}$.

Si nous regardons la taille des messages chiffrés et des clairs nous notons que le message double sa taille. Le clair est en effet un élément du groupe $\mathbb{G}$, le chiffré correspondant est composé de deux éléments du même groupe. De norme, les groupes choisis sont de la forme $(\mathbb{Z}/p\mathbb{Z})^*$ avec p un grand entier premier ou des sous-groupes des points rationnels sur une courbe elliptique.

Ce schéma est prouvé avoir la propriété de la sécurité sémantique pour des attaques à clair choisi (IND-CPA) sous l'hypothèse que le problème *Diffie-Hellman Décisionnel* (noté avec DDH) est un problème difficile. En gros, cela dit qu'il est calculatoirement infaisable dire si la terne (g^a, g^b, Z) où g est un élément générateur d'un groupe $\mathbb{G}$ d'ordre premier, a et b deux entiers choisis aléatoirement et $Z \in \mathbb{G}^*$, est une terne dite « Diffie-Hellman » (ou terne DDH), c'est à dire pour laquelle $Z = g^{ab}$. Ce problème décisionnel est lié au problème classique du *logarithme discret* (noté d'habitude DLOG). En gardant la même notation, si un adversaire est capable de retrouver x connaissant g et $h = g^x$, il peut aisément vérifier si $Z = (g^a)^b$ après avoir retrouvé b. Pour cette raison l'hypothèse DDH est considérée *plus forte* de l'hypothèse DLOG. Autrement dit, cela signifie qu'il existe des groupes dans lesquels il est considéré facile de reconnaître des ternes DDH mais il est considéré difficile de calculer le logarithme discret.

Il existe aussi une version calculatoire de l'hypothèse DDH, notée avec CDH comme *Diffie-Hellman Calculatoire*, qui affirme qu'il n'est pas possible de calculer efficacement g^{ab} en connaissant seulement $h' = g^a$ et $h'' = g^b$.

L'hypothèse CDH a été introduite par Diffie-Hellman en 1976 dans [**DH76**] et elle est à la base du schéma d'échange de clés Diffie-Hellman-Merkle. Indépendamment, à partir de l'année 1986 Victor Miller [**Mil86**] et Neal Koblitz [**Kob87**] ont proposé l'utilisation des courbes elliptiques dans la cryptographie. La structure particulière de groupe des points rationnels sur ces courbes permet d'utiliser des paramètres de taille plus petite tout en gardant un bon niveau de sécurité. À Miller [**Mil86**] est dû un exemple d'implémentation du schéma d'échange de clés Diffie-Hellman-Merkle qui utilise les courbes elliptiques.

1.1.6.2. Schéma de chiffrement de Paillier.— Nous donnons ici la définition du schéma de chiffrement dû à Pascal Pailler [**Pai99**]. Nous nous sommes intéressés à cet algorithme pour nos travaux car il introduit une fonction qui permet de calculer, dans un contexte particulier, le logarithme discret. Nous avons utilisé cette fonction dans nos travaux présentés dans le chapitre 3.

$\mathsf{KeyGen}(\lambda)$ **:** Pour générer sa paire des clés, l'utilisateur exécute un algorithme probabiliste qui donne en sortie deux (grands) nombres premiers p, q. Soit $N = pq$. Ensuite, il choisit un entier $g \in (\mathbb{Z}/N^2\mathbb{Z})^*$ d'ordre un entier non nul multiple de N.

La clé publique est $\mathsf{pk} = (N, g)$ tandis que la clé privée est la fonction d'Euler de l'entier N : $\mathsf{sk} = \varphi(N) = (p-1)(q-1)$.

$\mathsf{Encrypt}(m, \mathsf{pk})$ **:** Pour chiffrer un message $m \in (\mathbb{Z}/N\mathbb{Z})^*$, l'envoyeur récupère la clé publique du destinataire $\mathsf{pk} = (N, g)$ et il calcule le chiffré

$$c = g^m \cdot r^N \pmod{N^2} \text{ avec } r \in_R (\mathbb{Z}/N\mathbb{Z})^* \text{ aléatoire.}$$

Finalement le chiffré c du message m est retourné.

$\mathsf{Decrypt}(c, \mathsf{pk})$ **:** Pour déchiffrer un message, le destinataire calcule

$$m = \frac{L_N(c^{\varphi(N)})}{L_N(g)} \mod N$$

où la fonction $L_N : \{u \in \mathbb{Z}/N^2\mathbb{Z} | u \equiv 1 \pmod{N}\} \to \mathbb{Z}/N\mathbb{Z}$ est définie comme suit :

$$u \mapsto L_N(u) = \frac{u-1}{N} \mod N^2.$$

La fonction L_N retourne, pour certains éléments du groupe $(\mathbb{Z}/N^2\mathbb{Z})^*$, le logarithme discret. Pour le déchiffrement il s'agit de calculer le logarithme discret de $c^{\varphi(N)}$ en base g. Plus des détails sur cette fonction peuvent être trouvés dans le chapitre 3.

Ce schéma de chiffrement est prouvé être IND-CPA (sécurité sémantique face à des attaques à clair choisi) sous l'hypothèse de la *résiduosité composite*.

Le message chiffré est un élément modulo N^2 et donc sa taille égal à deux fois la taille du module RSA utilisé, à savoir dans la pratique la taille du chiffré devient de 2048 bits (pour un message en clair jusqu'à 1024 bits).

1.2. Chiffrement basé sur l'identité

Un des points les plus sensibles dans l'utilisation d'un schéma à clé publique est de garantir que la clé publique de chiffrement que l'on utilise pour chiffrer un message appartienne bien au destinataire désigné. En effet, si ce lien n'est pas assuré, une attaque très simple, dite « par le milieu » ou *man-in-the-middle* en anglais, est possible ; il s'agit, pour l'adversaire, de publier une clé publique dont il connaît la clé secrète correspondante sous l'identité de l'utilisateur attaqué. Si l'émetteur ne vérifie pas la légitimité du propriétaire de la clé, le message est bien chiffré sous la clé de l'utilisateur « attaqué » et peut être déchiffré par l'adversaire en utilisant la clé privée. Pour contrer cette famille d'attaques il est utilisé une variété des méthodes : les utilisateurs de PGP [1] ou GPG [2], par exemple, certifient les clés dans le système au travers d'un « *web of trust* », un paradigme introduit

1. `http://www.pgpi.org/` et `http://en.wikipedia.org/wiki/Pretty_Good_Privacy`
2. `http://www.gnupg.org/`

par Phil Zimmerman, l'inventeur de PGP, en 1992, en gros ils signent une clé d'un utilisateur après avoir vérifié sa paternité. Puis la confiance dans une clé publique est évaluée en fonction des signatures qu'elle a reçu. Plus généralement il est utilisé une infrastructure de certification des clés publiques appelée *PKI* (de l'anglais *Public Key Infrastructure*). Grossièrement, il s'agit d'une *autorité (ou tiers) de confiance* (*third trusted authority (or party)* en anglais) qui « certifie » qu'une clé publique donnée appartient bien à un utilisateur donné. Cela signifie que l'autorité de confiance *confirme*, au travers d'une signature, que la clé publique utilisée appartient bien à un utilisateur donné. Le déploiement d'une telle infrastructure demande donc quelques passages de plus lors de l'utilisation d'un schéma de chiffrement. Lors de la génération des clés, un utilisateur doit prouver de façon sûre son identité à l'autorité de confiance qui certifie avec sa signature le lien entre utilisateur et clé publique. Lors du chiffrement l'émetteur doit récupérer la clé publique et vérifier le certificat lié à la clé du destinataire.

C'est pour faire face différemment à ce problème que Shamir a introduit en 1984 [**Sha85**] le concept de *cryptographie basée sur l'identité*. Dans ce type de chiffrement la clé publique est remplacée par une chaîne quelconque de caractères comme : un identifiant alphanumérique, une adresse mail ou un numéro de téléphone portable. Dans ce contexte, l'autorité de confiance génère et conserve les paramètres publics ainsi qu'une clé secrète maître. Pour communiquer de façon sécurisée avec le destinataire, l'émetteur chiffre le message à l'aide de l'identifiant public. À son tour, afin de déchiffrer le message, le destinataire demande à l'autorité la clé secrète correspondant à son identité (ou une parmi ses identités). Comme la clé secrète est générée par le tiers de confiance avec l'identifiant public comme entrée, cette dernière peut être requise même après le chiffrement du message.

Nous donnons maintenant la définition formelle d'un schéma de chiffrement basé sur l'identité comme suite d'algorithmes probabilistes. La définition du schéma requiert la présence d'une autorité tierce de confiance $\mathcal{TTP}$ qui est chargée aussi de la génération des clés des utilisateurs. Cette autorité possède un jeu de deux clés : msk, la clé secrète maître, pour générer les clés de déchiffrement des utilisateurs et mpk, la clé maître publique, qui fait partie des paramètres publics du schéma et elle est donc connue par tout émetteur du message chiffré.

Définition 1.2.1 **(Schéma de chiffrement basé sur l'identité)**

Un schéma de chiffrement basé sur l'identité est la donnée de quatre algorithmes probabilistes :

- $\mathsf{Setup}(\lambda)$ **:** Cet algorithme est exécuté pendant l'initialisation du schéma. En prenant en entrée un paramètre de sécurité λ, ceci donne en sortie la clé publique maître mpk du système et la clé secrète maître correspondante msk gardée par l'autorité $\mathcal{TTP}$.
- $\mathsf{Extract}(\mathsf{msk}, \mathsf{ID})$ **:** Cet algorithme est exécuté par l'autorité de confiance $\mathcal{TTP}$, il prend en entrée la clé maître secrète msk et un identifiant d'utilisateur ID et il donne en sortie la clé secrète de déchiffrement $\mathsf{sk}_{\mathsf{ID}}$ pour l'utilisateur ID.
- $\mathsf{Encrypt}(\mathsf{mpk}, \mathsf{ID}, m)$ **:** Cet algorithme prend comme entrée la clé globale du schéma mpk, l'identifiant du destinataire ID et le message à chiffrer m. Il donne comme sortie le message chiffré $c = \mathsf{Encrypt}(\mathsf{mpk}, \mathsf{ID}, m)$.
- $\mathsf{Decrypt}(\mathsf{mpk}, \mathsf{ID}, \mathsf{sk}_{\mathsf{ID}}, c)$ **:** Cet algorithme, déterministe, prend comme entrée le chiffré c, la clé secrète de déchiffrement du destinataire $\mathsf{sk}_{\mathsf{ID}}$, l'identifiant public correspondant ID et les paramètres publics mpk et donne en sortie le message en clair m si le chiffré est valide ou un symbole de non réussite ($\perp$) sinon.

1.2.1. Notion de sécurité pour les schémas basées sur l'identité. — Comme nous avons vu précédemment pour le chiffrement à clé publique (1.1.1) pour définir les différents niveaux de sécurité d'un schéma il est nécessaire de donner le contexte de l'attaque en indiquant le but et les moyens de l'adversaire. En revanche, concernant la cryptographie basée sur l'identité, c'est l'adversaire qui décide quelles identités attaquer. La sécurité de tels schémas est définie selon deux modèles :

1. la *sécurité forte* (*full security* dans la littérature en langue anglaise) implique que l'adversaire choisit les identités attaquées adaptativement, éventuellement après avoir vu les paramètres du système ;
2. la sécurité dite *selective-ID*, qui au contraire impose à l'adversaire de déclarer les identités attaquées avant que le système soit déployé, et donc avant que les paramètres publics soient effectivement connus ou même générés.

Il est clair que la *full security*, donnant le choix de l'identité à attaquer après avoir eu la connaissance des paramètres publics, est une notion de sécurité plus forte et donc préférable.

Le niveau de sécurité est défini, comme dans le cas du chiffrement à clé publique, par un jeu. Nous donnons dans la suite le détail du niveau IND-ID-CCA, l'indistinguabilité contre une attaque à chiffré choisi, dans le modèle de la sécurité forte. Ceci représente le niveau de sécurité le plus haut qu'un schéma de chiffrement basé sur l'identité puisse atteindre.

Setup : Le challengeur exécute $\mathsf{Setup}(\lambda)$ pour obtenir le jeu de clés maître $(\mathsf{msk}, \mathsf{mpk})$. La clé maître publique, mpk, est donnée à l'adversaire $\mathcal{A}$.

(Première) phase de requêtes : L'adversaire effectue de manière adaptative deux types des requêtes :
- requêtes d'*extraction* : l'adversaire demande l'extraction de la clé privée correspondante à un identifiant de son choix ID_i, le challengeur exécute $\mathsf{Extract}(\mathsf{ID}_i)$ et transmet la clé secrète obtenue $\mathsf{sk}_{\mathsf{ID}_i}$ à l'adversaire $\mathcal{A}$;
- requêtes de *déchiffrement* : l'adversaire demande le déchiffrement du chiffré c_i pour l'utilisateur ID_i ; le challengeur répond avec la sortie de $\mathsf{Decrypt}(\mathsf{mpk}, \mathsf{ID}, \mathsf{sk}_{\mathsf{ID}}, c_i)$ après avoir obtenu la clé $\mathsf{sk}_{\mathsf{ID}_i} = \mathsf{Extract}(\mathsf{ID}_i)$.

Challenge : Quand l'adversaire considère terminée la première phase de requête, l'adversaire choisit une identité ID_*, dont il n'a pas demandé l'extraction de la clé secrète et deux messages m_0, m_1 de même taille et il les renvoie au challengeur. Le challengeur tire aléatoirement $b \leftarrow_R \{0,1\}$ et chiffre le message m_b pour obtenir le chiffré challenge : $c_* = \mathsf{Encrypt}(\mathsf{mpk}, \mathsf{ID}_*, m_b)$. Le challenge c_* est finalement renvoyé à l'adversaire.

(Seconde) phase de requêtes : L'adversaire peut recommencer une deuxième phase de requêtes au challengeur comme dans la première phase, avec la contrainte de ne pas soumettre le chiffré challenge c_* ou l'extraction de la clé secrète pour l'identifiant ID_*.

Réponse : Terminée la seconde phase de requêtes, l'adversaire donne sa réponse $b' \in \{0,1\}$. L'adversaire gagne le jeu si $b' = b$ et autrement il perd.

Nous remarquons que l'adversaire a à sa disposition un oracle $\mathsf{Extract}(\cdot)$ qui retourne les clés privées des utilisateurs sur lesquels il est interpelé. Cela veut représenter la possibilité que l'adversaire ait corrompu un sous ensemble d'utilisateurs ou alternativement que l'adversaire soit un groupe d'utilisateurs malicieux envers un tiers.

1.2.1.1. Calcul de l'avantage. — Comme dans le cas du chiffrement à clé publique nous considérons l'adversaire $\mathcal{A}$ comme composé de deux algorithmes probabilistes polynomiaux $(\mathcal{A}_1, \mathcal{A}_2)$ qui sont exécutés respectivement pendant la première et la deuxième phase de requêtes comme indiqué dans la description du jeu ci-dessous, nous notons avec q_D le nombre de requêtes de déchiffrement effectuées par l'adversaire (indistinctement par $\mathcal{A}_1$ ou $\mathcal{A}_2$) pendant les deux phases. Nous notons aussi avec q_E le nombre de requêtes d'extraction de clés de déchiffrement pendant tout le jeu, à nouveau faites indistinctement par $\mathcal{A}_1$ ou $\mathcal{A}_2$. Avec ces paramètres il nous est possible de calculer l'avantage $\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{IBE}}(q_E, q_D, \mathcal{A})$ de l'adversaire $\mathcal{A}$ pour gagner ce jeu et donc casser (au moins partiellement) la sécurité du schéma de chiffrement basé sur l'identité $\mathcal{IBE}$.

$$\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{IBE}}(q_E, q_D, \mathcal{A}) = \left| 2 \times \Pr\left[b' = b \;\middle|\; \begin{array}{c} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\lambda), \\ (\mathsf{ID}, m_0, m_1, s) \leftarrow \mathcal{A}_1(\mathsf{mpk}), \\ c_* = \mathsf{Encrypt}(\mathsf{mpk}, \mathsf{ID}, m_b), \\ \mathcal{A}_2(\mathsf{ID}, m_0, m_1, s, c_*) = b' \end{array} \right] - 1 \right|,$$

où la probabilité est calculée sur le choix aléatoire de b dans $\{0, 1\}$, les jetons aléatoires choisis par l'adversaire $\mathcal{A}$, le challengeur et les algorithmes probabilistes exécutés par ce dernier.

***Définition 1.2.2* (Niveau de sécurité IND-ID-CCA).** — On définit $\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{IBE}}(q_E, q_D) = \sup_{\mathcal{A}\ \text{p.p.t}}\{\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{IBE}}(q_E, q_D, \mathcal{A})\}$ comme la borne supérieure des avantages, lorsque $\mathcal{A}$ varie parmi les algorithmes probabilistes polynomiaux en le paramètre de sécurité λ. Le schéma de chiffrement basé sur l'identité $\mathcal{IBE}$ est dit être (q_E, q_D)-IND-ID-CCA sûr si $\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{IBE}}(q_E, q_D)$ est une fonction qui tend vers 0 plus rapidement que l'inverse de toute fonction polynomiale, dès lors que le paramètre de sécurité λ tend vers l'infini.

Par analogie avec la définition précédente nous donnons la définition de la sécurité sémantique face à des attaques à clairs choisis, simplement en interdisant les requêtes de déchiffrement à l'adversaire :

***Définition 1.2.3* (Niveau de sécurité IND-ID-CPA).** — La définition précédente est valable aussi dans les deux cas de sécurité sémantique. Le schéma de chiffrement $\mathcal{IBE}$ est dit être IND-ID-CPA sûr si il est $(0, 0)$-IND-ID-CCA sûr.

1.2.1.2. Sécurité dans le modèle Selective-ID. — Du jeu donné précédemment pour la sécurité forte nous pouvons aisément déduire celui correspondant pour le modèle *selective-ID* (indiqué avec sID) de moindre sécurité :

- L'indistinguabilité contre une attaque à chiffré choisi (IND-sID-CCA) se définit comme celui pour IND-ID-CCA en imposant à l'adversaire de déclarer l'identité attaquée avant la phase d'initialisation (Setup).
- L'indistinguabilité contre une attaque à clair choisi est définie de manière similaire en empêchant, de plus, à l'adversaire toutes requêtes de déchiffrement.

1.2.2. Un schéma d'exemple basé sur l'identité : *Boneh-Franklin.* — Le schéma dont nous donnons la description dans cette section est la première solution pratique au problème soulevé par Shamir dans [**Sha85**]. Ce schéma utilise une fonction bilinéaire, c'est à dire une fonction qui est bilinéaire sur les deux entrées. À savoir les fonctions utilisées sont les couplages de Weil ou Tate sur les groupes des points rationnels des courbes elliptiques.

$\mathsf{Setup}(\lambda)$ **:** L'algorithme $\mathsf{Setup}(\cdot)$ est exécuté par l'autorité de confiance pour générer les paramètres publics. L'algorithme donne en sortie deux groupes $\mathbb{G}_1$ e $\mathbb{G}_2$ d'ordre p et une application bilinéaire $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ et un générateur g pour le groupe $\mathbb{G}_1$. L'algorithme tire aussi une valeur aléatoire $x \in (\mathbb{Z}/p\mathbb{Z})^*$ et pose $y = g^x$. Soit l la taille maximale des messages, l'algorithme choisit deux fonctions d'hachage : $\mathcal{H} : \{0,1\}^* \to \mathbb{G}_1$ (fonction pour codifier les identités) et $\mathcal{G} : \mathbb{G}_2 \to \{0,1\}^l$.

Donc les paramètres publics sont : $\mathsf{mpk} = (\mathbb{G}_1, \mathbb{G}_2, p, e(\cdot,\cdot), g, y, \mathcal{H}, \mathcal{G}, l)$ et la clé secrète correspondante : $\mathsf{msk} = x$.

$\mathsf{Extract}(\mathsf{msk}, \mathsf{ID})$ **:** L'autorité de confiance pour générer la clé secrète de l'utilisateur ID calcule d'abord sa clé publique $\mathsf{pk}_{\mathsf{ID}} = h_{\mathsf{ID}} = \mathcal{H}(\mathsf{ID})$ et ensuite pose la clé secrète comme $\mathsf{sk}_{\mathsf{ID}} = h_{\mathsf{ID}}^x$.

$\mathsf{Encrypt}(\mathsf{mpk}, \mathsf{ID}, m)$ **:** Pour chiffrer un message m (de taille maximale l) pour le destinataire ID, l'envoyeur tire un élément $r \in_R (\mathbb{Z}/p\mathbb{Z})^*$, ensuite il calcule la clé publique $\mathsf{pk}_{\mathsf{ID}}$ au travers de la fonction d'hachage $\mathcal{H}$. Il pose $w = e(y, h_{\mathsf{ID}})^r$. Finalement il calcule $c_1 = g^r$ et $c_2 = m \oplus \mathcal{G}(w)$.

Le chiffré en sortie est $c = (c_1, c_2)$.

$\mathsf{Decrypt}(\mathsf{mpk}, \mathsf{ID}, \mathsf{sk}, c)$ **:** Pour retrouver le message m à partir du chiffré c au destinataire il suffit de calculer $w' = e(c_1, \mathsf{sk}_{\mathsf{ID}})$ et ensuite démasquer le message en clair en calculant : $m = c_2 \oplus \mathcal{G}(w')$.

Il est prouvé que le schéma décrit ci-dessous a la sécurité sémantique sous l'hypothèse que le Problème Diffie-Hellmann Bilinéaire (BDH) soit difficile. La version décrite est résistante *seulement* aux attaques à clair choisi, mais, comme nous avons déjà remarqué, il existe des transformations génériques, notamment celle du à Fujisaki et Okamoto [**FO99a, FO99b**] qui permet d'élever le niveau de IND-CPA à IND-CCA.

À noter, la taille maximale pour les messages en clair est due à la taille de la sortie de fonction $\mathcal{G}(\cdot)$. Sans cette limitation, si un message « long » est chiffré, une partie ne sera pas masquée par l'opération de chiffrement, en rendant les attaques à la sécurité sémantique triviales.

1.3. Anonymat ou *Key-privacy*

Normalement quand nous parlons des buts de sécurité pour la cryptographie nous sommes focalisés sur la « protection » du contenu du message chiffré. Les différentes définitions données précédemment ont en effet pour but de capturer différents niveaux de protection de données chiffrées : la sécurité sémantique, la non-malléabilité, le sens unique de la fonction de chiffrement ou la récupération de la clé secrète. Une telle « protection » interdit à l'adversaire d'apprendre n'importe quelle information contenue dans le chiffré mais est donc limitée au contenu des messages. Un autre aspect est de protéger l'identité du destinataire du message chiffré. Cet aspect est normalement défini comme *anonymat*. Cette notion de sécurité vise donc à protéger, en plus des informations contenues dans le chiffré, la clé publique utilisée pour le générer. Nous voulons souligner que la *key-privacy* n'a rien à voir avec la sécurité des données : dans le contexte de la clé publique la connaissance de la clé qui a généré un chiffré n'aide pas à son déchiffrement (sans la possession de la clé secrète correspondante). Cette notion de sécurité vise à protéger le destinataire dans le contexte de la création du chiffré même. En d'autres termes nous voulons que le

chiffré ne donne aucune information sur la clé utilisée pour sa création, pour cette raison nous parlons de receveur anonyme.

Remarque 1.3.1. — Nous parlons plus correctement de *Key-privacy* dans le contexte de la clé publique au vu du fait que normalement les clés sont des chaînes aléatoires qui n'ont pas un lien réel avec le propriétaire. Un tel lien clé-personne physique peut être reconstruit avec une recherche dans les répertoires publics qui contiennent les clés avec leurs certificats.

En revanche, dans le contexte du chiffrement basé sur l'identité, les clés peuvent être une chaîne de caractères au choix de l'émetteur et normalement contiennent une référence claire et immédiate à l'utilisateur propriétaire.

Avant de donner la définition formelle pour la notion d'anonymat il est utile d'observer que le concept même de *privacy* du receveur implique l'existence d'un environnement homogène. Il est en effet impossible de parler d'anonymat si le format du chiffré laisse fuir des informations sur le destinataire. Dans un contexte non homogène l'anonymat faillirait pour des raisons simples : différents utilisateurs peuvent utiliser différents schémas de chiffrement pour protéger leur données ou il se peut que même en utilisant le même cryptosystème ils utilisent des tailles des clés différentes. Idéalement pour aspirer à des schémas qui protègent l'identité du receveur les utilisateurs doivent utiliser le même paramètre de sécurité (λ) et partager les mêmes paramètres publics.

1.3.1. Key-privacy dans le chiffrement à clé publique. — Nous donnons dans cette section la définition formelle de la key-privacy dans le chiffrement à clé publique à travers un jeu. La notion est similaire à celle de la sécurité sémantique, mais dans cette formalisation l'adversaire doit distinguer entre deux clés publiques possibles.

Setup : Après avoir exécuté l'algorithme d'initialisation $\mathsf{Setup}(\lambda)$, le challengeur exécute $\mathsf{KeyGen}(\lambda)$ deux fois afin de générer deux clés publiques : pk_0 et pk_1. Les deux clés sont données à l'adversaire et les clés secrètes correspondantes sk_0 et sk_1 sont gardées par le challengeur.

(Première) phase de requêtes : L'adversaire effectue de manière adaptative des requêtes de déchiffrement pour la clé de son choix. Pour tout chiffré c, de son choix, soumis, le challengeur répond avec le clair $m = \mathsf{Decrypt}(c, \mathsf{sk}_i)$ où i est choisi par l'adversaire.

Challenge : Lorsque l'adversaire considère terminée la première phase de requête, il choisit un message m et il le renvoie au challengeur. Le challengeur tire aléatoirement $b \leftarrow \{0, 1\}$ et chiffre le message m sous la clé pk_b pour obtenir le chiffré challenge : $c_* = \mathsf{Encrypt}(m, \mathsf{pk}_b)$. Le challenge c_* est finalement renvoyé à l'adversaire.

(Seconde) phase de requêtes : L'adversaire peut recommencer une deuxième phase de requêtes de déchiffrement au challengeur, avec la contrainte de ne pas soumettre le chiffré challenge c_*.

Réponse : Terminée la seconde phase de requêtes, l'adversaire donne sa réponse $b' \in \{0, 1\}$. L'adversaire gagne le jeu si $b' = b$ et autrement il perd.

1.3.1.1. Calcul de l'avantage. — Nous calculons ici l'avantage d'un adversaire dans le jeu illustré précédemment. Afin de plus aisément représenter l'adversaire $\mathcal{A}$ dans les deux phases, ce dernier est représenté par un couple d'algorithmes probabilistes polynomiaux $(\mathcal{A}_1, \mathcal{A}_2)$ qui sont exécutés dans les deux phases respectives. Nous notons avec q_D le nombre

de requêtes de déchiffrement faites par l'adversaire au challengeur lors des deux phases. Ce paramètre nous permet de calculer l'avantage $\mathsf{Adv}_{\mathcal{E}}^{\mathsf{key\text{-}ind}}(q_D, \mathcal{A})$ de l'adversaire $\mathcal{A}$ pour gagner le jeu. L'avantage est donné par la formule suivante :

$$\mathsf{Adv}_{\mathcal{E}}^{\mathsf{key\text{-}ind}}(q_D, \mathcal{A}) = \left| 2 \times \Pr\left[b' = b \;\middle|\; \begin{array}{c} (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{KeyGen}(\lambda), (\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow \mathsf{KeyGen}(\lambda), \\ (m, s) \leftarrow \mathcal{A}_1(\mathsf{pk}_1, \mathsf{pk}_2), c_* = \mathsf{Encrypt}(\mathsf{pk}_b, m), \\ \mathcal{A}_2(\mathsf{pk}_1, \mathsf{pk}_2, m, s, c_*) = b' \end{array} \right] - 1 \right|,$$

où la probabilité est calculée sur le choix aléatoire de b, les jetons aléatoires choisis par l'adversaire $\mathcal{A}$, du challengeur et des algorithmes probabilistes exécutés par ce dernier.

Nous pouvons donc maintenant définir l'indistinguabilité des clés (ou *key-privacy*) pour une attaque à chiffré choisi.

Définition 1.3.2 **(Key-privacy pour une attaque à chiffré choisi. IK-CCA)**
Soit $\mathsf{Adv}_{\mathcal{E}}^{\mathsf{key\text{-}ind}}(q_D) = \sup_{\mathcal{A}}(\mathsf{Adv}_{\mathcal{E}}^{\mathsf{key\text{-}ind}}(q_D, \mathcal{A}))$ la borne supérieure des avantages, lorsque $\mathcal{A}$ varie parmi les algorithmes probabilistes polynomiaux en le paramètre de sécurité λ. Nous disons que le schéma de chiffrement $\mathcal{E}$ a la propriété de *key-privacy* si $\mathsf{Adv}_{\mathcal{E}}^{\mathsf{key\text{-}ind}}(q_D)$ est une fonction négligeable du paramètre de sécurité λ et il est noté $(q_D) - \mathsf{IK\text{-}CCA}$.

De façon analogue, comme dans le cas de la sécurité sémantique, nous pouvons définir la *key-privacy* pour des attaques à clair choisi en imposant que l'adversaire n'ait pas accès à un oracle de déchiffrement.

Définition 1.3.3 **(Key-privacy pour une attaque à clair choisi. IK-CPA)**
Un schéma de chiffrement est dit avoir la propriété de *key-privacy* contre une attaque à clair choisi s'il a la propriété $(0) - \mathsf{IK\text{-}CCA}$.

1.3.2. Anonymat dans le chiffrement basé sur l'identité. — Dans cette section nous présentons la définition formelle d'anonymat pour un schéma de chiffrement basé sur l'identité. Nous traitons les détails pour le cas de la *sécurité forte*. La définition est donnée, comme dans le cas précédent par le biais d'un jeu contre un adversaire.

Setup : Le challengeur exécute l'algorithme d'initialisation $\mathsf{Setup}(\lambda)$ afin de générer la clé maître publique mpk. La clé est donnée à l'adversaire et la clé maître secrète correspondante msk est gardée par le challengeur.

(Première) phase de requêtes : L'adversaire effectue de manière adaptative deux types des requêtes :
- requêtes d'*extraction* : l'adversaire demande l'extraction de la clé privée correspondant à un identifiant de son choix ID_i, le challengeur exécute $\mathsf{Extract}(\mathsf{ID}_i)$ et transmet la clé secrète obtenue $\mathsf{sk}_{\mathsf{ID}_i}$ à l'adversaire $\mathcal{A}$;
- requêtes de *déchiffrement* : l'adversaire demande le déchiffrement du chiffré c pour l'utilisateur ID_i ; le challengeur répond avec la sortie de $\mathsf{Decrypt}(\mathsf{mpk}, \mathsf{ID}_i, \mathsf{sk}_{\mathsf{ID}_i}, c)$.

Challenge : Lorsque l'adversaire considère que la première phase de requêtes est terminée, il choisit deux identités ID_1 et ID_2, dont il n'a pas demandé l'extraction de la clé secrète et un message m et il les renvoie au challengeur. Le challengeur tire aléatoirement $b \leftarrow \{0, 1\}$ et chiffre le message m sous l'identité ID_b pour obtenir le chiffré challenge : $c_* = \mathsf{Encrypt}(\mathsf{mpk}, \mathsf{ID}_b, m)$. Le challenge c_* est finalement renvoyé à l'adversaire.

(Seconde) phase de requête : L'adversaire peut recommencer une deuxième phase de requêtes au challengeur comme dans la première phase, avec la contrainte de ne pas soumettre le chiffré challenge c_* ou l'extraction de la clé secrète pour les identifiants ID_1 et ID_2.

Réponse : Terminée la seconde phase de requêtes, l'adversaire donne sa réponse $b' \in \{0,1\}$. L'adversaire gagne le jeu si $b' = b$ et autrement il perd.

1.3.2.1. Calcul de l'avantage. — Comme pour les cas vus précédemment nous allons considérer l'adversaire $\mathcal{A}$ comme un couple d'algorithmes probabilistes polynomiaux $(\mathcal{A}_1, \mathcal{A}_2)$ qui sont exécutés respectivement dans les deux phases du jeu ci-dessous. Nous notons avec q_D la somme des requêtes de déchiffrement faites pendant les deux sessions du jeu et, de façon analogue, q_E désigne la totalité des requêtes d'extraction de la clé secrète faites par $\mathcal{A}_1$ et $\mathcal{A}_2$ dans les deux phases. Ces deux quantités nous permettent de calculer l'avantage $\mathsf{Adv}^{\mathsf{anon}}_{\mathcal{IBE}}(q_D, q_E, \mathcal{A})$ selon la formule suivante :

$$\mathsf{Adv}^{\mathsf{anon}}_{\mathcal{IBE}}(q_D, q_E, \mathcal{A}) = \left| 2 \times \Pr\left[b' = b \;\middle|\; \begin{array}{c} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\lambda), \\ (\mathsf{ID}_1, \mathsf{ID}_2, m, s) \leftarrow \mathcal{A}_1(\mathsf{mpk}), \\ c_* = \mathsf{Encrypt}(\mathsf{ID}_b, m), \\ \mathcal{A}_2(\mathsf{mpk}, \mathsf{ID}_1, \mathsf{ID}_2, m, s, c_*) = b' \end{array} \right] - 1 \right|,$$

où la probabilité est calculée sur le choix aléatoire du bit $b \in \{0,1\}$, sur les jetons aléatoires choisis par l'adversaire $\mathcal{A}$, le challengeur et les algorithmes probabilistes exécutés par ce dernier.

Avec la définition d'avantage ci-dessous nous pouvons finalement donner la définition d'un schéma de chiffrement basé sur l'identité *anonyme* face à une attaque à chiffré choisi.

Définition 1.3.4 **(Anonymat pour une attaque à chiffré choisi. ANON-CCA)**
Soit $\mathsf{Adv}^{\mathsf{anon}}_{\mathcal{IBE}}(q_D, q_E) = \max_{\mathcal{A}}\{\mathsf{Adv}^{\mathsf{anon}}_{\mathcal{IBE}}(q_D, q_E, \mathcal{A})\}$ où le maximum est calculé sur la variation de l'algorithme $\mathcal{A}$ parmi les possibles algorithmes probabilistes polynomiaux. Le schéma de chiffrement basé sur l'identité $\mathcal{IBE}$ est dit être *anonyme* si $\mathsf{Adv}^{\mathsf{anon}}_{\mathcal{IBE}}(q_D, q_E)$ est une fonction négligeable en le paramètre de sécurité λ et il est noté (q_D, q_E)-ANON-CCA.

De façon analogue à celle vue dans le cas de *key-privacy*, il est possible définir l'*anonymat* dans le cas où un adversaire peut accéder seulement à un oracle d'extraction de clés secrètes. Cela donne la définition suivante, correspondante à l'attaque à clair choisi.

Définition 1.3.5 **(Anonymat pour une attaque à clair choisi. ANON-CPA)**
Un schéma de chiffrement base sur l'identité $\mathcal{IBE}$ est dit être *anonyme* face aux attaques à clair choisi s'il est $(0, q_E)$-ANON-CCA. Dans ce cas il est noté (q_E)-ANON-CPA.

1.3.3. Exemples de schémas classiques de chiffrement avec la propriété de l'anonymat. — Le schéma de chiffrement dû à El Gamal [**Gam84**] (décrit dans la Section 1.1.6.1) a la sécurité sémantique et si un groupe d'utilisateurs partage les mêmes paramètres publics $(g, p, \mathbb{G})$, ce schéma de chiffrement rencontre aussi la propriété d'anonymat.

Nous donnons ici la description d'un schéma de chiffrement dû à Cramer et Shoup [**CS98**] qui présente aussi cette propriété l'indistinguabilité des clés.

$\mathsf{Setup}(\lambda)$ **:** L'algorithme $\mathsf{Setup}(\lambda)$ est exécuté avec le paramètre de sécurité λ comme entrée. Cela donne en sortie un groupe cyclique $\mathbb{G}$, d'ordre premier p, où la taille

du premier p dépend du paramètre de sécurité. L'algorithme choisit ensuite deux générateurs g_1 et g_2.

Les paramètres publics sont : $(g_1, g_2, p, \mathbb{G})$.

$\mathsf{KeyGen}(\lambda)$ **:** Pour générer sa paire de clés un utilisateur tire au hasard des éléments dans $(\mathbb{Z}/p\mathbb{Z})^*$:

$$x_1, x_2, y_1, y_2, z \in (\mathbb{Z}/p\mathbb{Z})^*.$$

Puis il calcule :

$$c = g_1^{x_1} g_2^{x_2}, d = g_1^{y_1} g_2^{y_2}, h = g_1^z.$$

Il choisit une fonction d'hachage $\mathcal{H}$ et il publie la clé publique : $\mathsf{pk} : (g_1, g_2, c, d, h, \mathcal{H})$. La clé de déchiffrement correspondante est : $\mathsf{sk} : (x_1, x_2, y_1, y_2, h)$.

$\mathsf{Encrypt}(m, \mathsf{pk})$ **:** Pour chiffrer un message $m \in \mathbb{G}$, l'envoyeur récupère la clé publique du destinataire pk, il tire un élément aléatoire $r \in_R (\mathbb{Z}/p\mathbb{Z})^*$ et calcule :

$$u_1 = g_1^r, u_2 = g_2^r, e = h^r \cdot m, \alpha = \mathcal{H}(u_1, u_2, e), v = c^r d^{r\alpha}.$$

Puis il envoie le chiffré $\tilde{c} = (u_1, u_2, e, v)$ au destinataire.

$\mathsf{Decrypt}(\tilde{c}, \mathsf{sk})$ **:** Pour déchiffrer un message $\tilde{c}$, l'utilisateur destinataire exécute l'algorithme $\mathsf{Decrypt}(\cdot, \cdot)$ comme il suit :

1. calcule $\alpha = \mathcal{H}(u_1, u_2, e)$,
2. vérifie si $u_1^{x_1 + y_1\alpha} u_2^{x_2 + y_2\alpha} = v$,
3. l'algorithme donne en sortie :

$$m = \begin{cases} e \cdot u_1^{-z} & \text{si la condition est vérifiée} \\ \perp & \text{sinon (déchiffrement non valide)} \end{cases}.$$

Il a été démontré que ce schéma atteint la sécurité des données (sécurité sémantique) pour des attaques a chiffré choisi, sous l'hypothèse que le problème DDH soit « difficile » dans le groupe $\mathbb{G}$. Dans l'hypothèse où les utilisateurs partagent les paramètres publics ce schéma est montré avoir la propriété de la *key-privacy* pour une attaque à chiffré choisi ($\mathsf{IK\text{-}CCA}$). Plus de détails peuvent être trouvés dans [**BBDP01**].

CHAPITRE 2

UNE CONSTRUCTION SIMPLE POUR UN SCHÉMA DE CHIFFREMENT À CLÉ PUBLIQUE AVEC LA PROPRIÉTÉ DE L'ANONYMAT RÉVOCABLE : LE CAS DE L'ÉMETTEUR HONNÊTE.

Ce chapitre est constitué de l'article « A Simple Construction for Public-Key Encryption with Revocable Anonymity: The Honest-Sender Case », publié avec Marc Joye dans les *Proceedings of the nineth ACM workshop on Digital Rights Management* et présenté par moi au même workshop en novembre 2009 à Chicago, USA. La référence complète est disponible : [**AJ09**].

2.1. Introduction

In numerous scenarios, the recipient's identity in a transmission needs to be kept private. This allows users to maintain some privacy. Protecting communication content may be not enough, as already observed in a couple of papers (e.g., [**BBW06, BBDP01, KTY07**]). For example, by analyzing the traffic between an antenna and a mobile device, one can recover some information about [at least] user's position and some details about the use of her mobile device. This information leaks easily during all day: it is a common habit, indeed, to use a mobile phone every day and to keep it (almost) always switched on. A similar problem exists in the context of electronic commerce. If no anonymity is provided, users' preferences can be known. The knowledge of this information enables profiling users and sending them targeted advertisements or selling profiles to other commercial entities. It should be conceivable that buyers can make their choices and shop on Internet without risking to be profiled. On the other hand, it is understandable that, in some situations, there may be a need to be able to revoke the anonymity; for example in the case of a legal dispute — while keeping the content private. This paper provides such a solution, offering a trade-off between the needs of the different involved parties.

We propose constructions for encryption schemes with revocable anonymity. As a result, we get schemes keeping private the recipient's identity in a transmission, but offering to some trusted party the possibility to discover it.

Back to the wireless communication case, we can imagine antennas broadcasting messages without leaking receivers' identifiers to mobile devices and in case of network misuse, find out the user identifier and revoke her. The primitive can be deployed on a shared network (or storage) system where all data are encrypted. If someone overuses the resources quota, the system administrator can easily individuate that user and take appropriate actions (as a warning, reduce/revoke rights, ...).

2.1.0.1. Related work. — Formal notions of anonymity in the public-key setting appear in [**BBDP01**]. There should be no way for an adversary to distinguish between a message

sent to a given recipient and one addressed to a random one. In [**IP08**], Pointcheval and Izabachène analyze different anonymity levels for identity-based encryption schemes ([**Sha85**]).

Concurrent to our work, Kiayias *et al.* introduce and model in [**KTY07**] the concept of *group encryption*. This is the analogue for encryption of group signatures [**CvH91**]. Group encryption allows one to conceal the identity of the recipient of a given ciphertext among a set of legitimate receivers. However, in case of misuse, some authority (the group manager) is capable of recovering the recipient's identity. Furthermore, in addition to security and privacy properties, group encryption offers *verifiability*: a sender can convince a verifier that the formed ciphertext can be decrypted by a group member.

2.1.0.2. Our contribution. — Key privacy in public-key encryption assumes a "homogeneous" environment. Indeed, if users make use of different cryptosystems or of the same cryptosystem but with keys of different lengths, anonymity is likely to be lost. The notion of anonymity is therefore is restricted to users sharing the same cryptosystem and common parameters. This implicitly defines a group.

In this paper, we relax the requirements for group encryption. In the particular context of media broadcasting or wireless communications, we face a different situation where the sender (the broadcaster or the wireless emitter) can be trusted. This relaxation is justified by the fact that, in practical uses of the infrastructure, the sender has no interest in cheating because of business and reputation aspects. Moreover, it is very unlikely that an attacker can impersonate the sender, due to the particular material infrastructure needed (expensive, powerful, ...). Such an attacker should, indeed, mute the licit signals and substitute them with illicit ones, keeping all existing communications alive and faking the attacked ones. For similar reasons, we can also suppose that there are no collusions between senders and recipients. We refer to this setting as the *honest-sender case*.

The rest of this paper is organized as follows. In the next section, we introduce some background on public-key encryption. In Section 2.3, we define the notion of revocable anonymity. Section 2.4 is the core of our paper. We describe a simple and generic transformation to get a public-key encryption scheme with revocable anonymity. Finally, we present applications of our transformation in Section 2.5.

2.1.0.3. Acknowledgments. — We are grateful to the anonymous referees and to Alain Durand, Mohamed Karroumi and Nicolas Prigent for useful comments.

2.2. Preliminaries

In this section, we review classical notions for public-key encryption, dealing both with data-privacy and key-privacy. We also introduce some useful notation.

2.2.0.4. Public-key encryption scheme. — In order to better capture the property that users may share some common parameters in a homogeneous environment, the key generation algorithm is divided in two sub-algorithms: the *common-key generation* algorithm and the *key generation* algorithm.

Following the syntax of [**BBDP01**], we define a *public-key encryption scheme* is a tuple of four algorithms $(\mathsf{CKeyGen}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$:

Common-key generation : The common-key generation algorithm $\mathsf{CKeyGen}$ takes on input some security parameter λ and outputs some common key $I \stackrel{R}{\leftarrow} \mathsf{CKeyGen}(\lambda)$.

Key generation : The key generation algorithm KeyGen is a randomized algorithm that takes on input I and returns a matching pair of public key and secret key for some user: $(\mathsf{upk}, \mathsf{usk}) \xleftarrow{R} \mathsf{KeyGen}(I)$.

Encryption : Let $\mathcal{M}$ denote the message space. The encryption algorithm Enc is a randomized algorithm that takes on input a public key upk and a plaintext $m \in \mathcal{M}$, and returns a ciphertext c. We write $c \leftarrow \mathsf{Enc}_{\mathsf{upk}}(m)$.

Decryption : The decryption algorithm Dec takes on input secret key usk (matching upk) and ciphertext c and returns the corresponding plaintext m or a special symbol $\perp$ indicating that the ciphertext is invalid. We write $m \leftarrow \mathsf{Dec}_{\mathsf{usk}}(c)$ if c is a valid ciphertext and $\perp \leftarrow \mathsf{Dec}_{\mathsf{usk}}(c)$ if it is not.

We require that $\mathsf{Dec}_{\mathsf{usk}}(\mathsf{Enc}_{\mathsf{upk}}(m)) = m$ for any message $m \in \mathcal{M}$.

2.2.0.5. Indistinguishability of encryptions. — The notion of *indistinguishability of encryptions* [**GM84**] captures a strong notion of [data]-privacy: The adversary should not learn any information whatsoever about a plaintext given its encryption beyond the length of the plaintext.

We view an adversary $\mathcal{A}$ as a pair $(\mathcal{A}_1, \mathcal{A}_2)$ of probabilistic algorithms. This corresponds to adversary $\mathcal{A}$ running in two stages. In the "find" stage, algorithm $\mathcal{A}_1$ takes on input a public key upk and outputs two equal-size messages m_0 and $m_1 \in \mathcal{M}$ and some state information s. In the "guess" stage, algorithm $\mathcal{A}_2$ receives a challenge ciphertext c which is the encryption of m_b under upk and where b is chosen at random in $\{0, 1\}$. The goal of $\mathcal{A}_2$ is to recover the value of b from s and c.

A public-key encryption scheme is said *semantically secure* (or *indistinguishable*) if

$$\Pr\left[\begin{array}{l} I \xleftarrow{R} \mathsf{CKeyGen}(\lambda), (\mathsf{upk}, \mathsf{usk}) \xleftarrow{R} \mathsf{KeyGen}(I), \\ (m_0, m_1, s) \leftarrow \mathcal{A}_1(\mathsf{upk}), b \xleftarrow{R} \{0,1\}, c \leftarrow \mathsf{Enc}_{\mathsf{upk}}(m_b) \end{array} : \mathcal{A}_2(s, c) = b \right] - \frac{1}{2}$$

is negligible in the security parameter for any polynomial-time adversary $\mathcal{A}$; the probability is taken over the random coins of the experiment according to the distribution induced by $\mathsf{CKeyGen}$ and KeyGen and over the random coins of the adversary.

As we are in the public-key setting, it is worth noting that adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is given the public key upk and so can encrypt any message of its choice. In other words, the adversary can mount chosen-plaintext attacks (CPA). Hence, we write *IE-CPA* the security notion achieved by a semantically secure encryption scheme. (1)

A stronger scenario is to give the adversary an adaptive access to a decryption oracle. The previous definition readily extends to this model. Adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is allowed to submit any ciphertext of its choice and receives the corresponding plaintext (or $\perp$); the sole exception is that $\mathcal{A}_2$ may not query the decryption oracle on challenge ciphertext c. Likewise, we write *IE-CCA* the corresponding security notion.

1. We deviate from the usual notation of IND-CPA to emphasize the fact that indistinguishability is about encryptions.

2.2.0.6. Indistinguishability of keys. — Analogously, the notion of *indistinguishability of keys* captures a strong requirement about key privacy: The adversary should not be able to link whatsoever a ciphertext with its underlying encryption key.

As before, we view an adversary $\mathcal{A}$ as a pair $(\mathcal{A}_1, \mathcal{A}_2)$ of probabilistic algorithms. In the "find" stage, algorithm $\mathcal{A}_1$ takes on input two public keys upk_0 and upk_1 and outputs a message m and some state information s. Then in the "guess" stage, algorithm $\mathcal{A}_2$ receives a challenge ciphertext c which is the encryption of m under upk_b where b is chosen at random in $\{0,1\}$. The goal of $\mathcal{A}_2$ is to recover the value of b from s and c.

More formally, a public-key encryption scheme is said *anonymous* (or *key-private*) if

$$\Pr\left[\begin{array}{l} I \xleftarrow{R} \mathsf{CKeyGen}(\lambda), (\mathsf{upk}_0, \mathsf{usk}_0) \xleftarrow{R} \mathsf{KeyGen}(I), \\ (\mathsf{upk}_1, \mathsf{usk}_1) \xleftarrow{R} \mathsf{KeyGen}(I), (m, s) \leftarrow \mathcal{A}_1(\mathsf{upk}_0, \mathsf{upk}_1), : \\ b \xleftarrow{R} \{0,1\}, c \leftarrow \mathsf{Enc}_{\mathsf{upk}_b}(m) \end{array} \mathcal{A}_2(s,c) = b \right] - \frac{1}{2}$$

is negligible in the security parameter for any polynomial-time adversary $\mathcal{A}$; the probability is taken over the random coins of the experiment according to the distribution induced by $\mathsf{CKeyGen}$ and KeyGen and over the random coins of the adversary.

This definition of anonymity gives rise to the security notion of *IK-CPA* or *indistinguishability of keys under chosen-plaintext attacks*. If the adversary is given adaptive access to a decryption oracle, the corresponding security notion is *IK-CCA* or *indistinguishability of keys under chosen-ciphertext attacks*.

Of course, the goals of data-privacy and key-privacy can be combined to define extended security notions. A public-key encryption scheme achieves *IND-CPA security* if it is both IE-CPA and IK-CPA. Likewise, a public-key encryption scheme achieves *IND-CCA security* if it is both IE-CCA and IK-CCA.

2.3. Revocable Anonymity

As exemplified in the introduction, there are several use cases where the sender is trustful and has no incentive to cheat. However, while it may be useful to keep the identity of the receiver private, it may also be useful to have some means to discover the identity of a receiver in case of misuse or dispute. This section formally defines the notion of revocable anonymity.

2.3.1. Formal definition. — We augment the definition of public-key encryption scheme so that anonymity can be revoked. The formalization shares many parts with that of a (regular) public-key encryption scheme. The main differences are *(i)* the generation of a pair of keys for tracing purposes in the common-key generation and *(ii)* a tracing algorithm for recovering the intended recipient of a ciphertext.

More formally, a *public-key encryption scheme with revocable anonymity* is a tuple of five algorithms $(\mathsf{CKeyGen}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Trace})$:

Common-key generation : The key generation algorithm $\mathsf{CKeyGen}$ takes on input some security parameter λ and

1. generates the opener's secret and public keys osk and opk;
2. outputs some common key I: $I \xleftarrow{R} \mathsf{CKeyGen}(\lambda)$.

Here, element I may include a copy of public key opk.

Key generation : The key generation algorithm KeyGen is a randomized algorithm that takes on input I and returns a matching pair of public key and secret key for some user: $(\mathsf{upk}, \mathsf{usk}) \xleftarrow{R} \mathsf{KeyGen}(I)$.

Encryption : Let $\mathcal{M}$ denote the message space. The encryption algorithm Enc is a randomized algorithm that takes on input a public key upk and a plaintext $m \in \mathcal{M}$ and returns a ciphertext c. We write $c \leftarrow \mathsf{Enc}_{\mathsf{upk}}(m)$.

Decryption : The decryption algorithm Dec takes on input secret key usk (matching upk) and ciphertext c and returns the corresponding plaintext m or a special symbol $\perp$ indicating that the ciphertext is invalid. We write $m \leftarrow \mathsf{Dec}_{\mathsf{usk}}(c)$ if c is a valid ciphertext and $\perp \leftarrow \mathsf{Dec}_{\mathsf{usk}}(c)$ if it is not.

Tracing : The tracing algorithm Trace takes on input secret opening key osk and ciphertext c and returns the corresponding user's public key upk or a special symbol $\perp$ indicating that the ciphertext is invalid. We write $\mathsf{upk} \leftarrow \mathsf{Trace}_{\mathsf{osk}}(c)$ if c is a valid ciphertext and $\perp \leftarrow \mathsf{Trace}_{\mathsf{osk}}(c)$ if is not.

We require for a public-key encryption scheme with revocable anonymity the two following properties:

***Correctness* :** For any message $m \in \mathcal{M}$ and for any pair of matching public key/secret key returned by the key generation algorithm, $(\mathsf{upk}, \mathsf{usk}) \xleftarrow{R} \mathsf{KeyGen}(I)$, one has $\mathsf{Dec}_{\mathsf{usk}}(\mathsf{Enc}_{\mathsf{upk}}(m)) = m$; and

***Traceability* :** For any message $m \in \mathcal{M}$ and for any pair of matching public key/secret key returned by the key generation algorithm, $(\mathsf{upk}, \mathsf{usk}) \xleftarrow{R} \mathsf{KeyGen}(I)$, one has $\mathsf{Trace}_{\mathsf{osk}}(\mathsf{Enc}_{\mathsf{upk}}(m)) = \mathsf{upk}$.

Given a [well formed] ciphertext, the first property (*correctness*) ensures that the intended recipient will always recover the corresponding plaintext while the second property (*traceability*) ensures that the tracing authority will always discover the recipient, if needed.

2.3.2. Non-malleability. — In this section we add a further requirement to enforce the *traceability* property and discuss the notion of *non-malleability*.

2.3.2.1. A limited solution. — A simple way to get revocable anonymity is to *(i)* encrypt the message as usual with a key-private public-key encryption scheme, *(ii)* encrypt the recipient's identity under a trusted public encryption-key, and *(iii)* define the resulting ciphertext as the concatenation of the two encryptions and make it available to the recipient.

It is easily verified that the above scheme meets the *correctness* and *traceability* properties. But it also suffers from limitations, even in the honest-sender-case. The *traceability* property resides only in the presence of the encryption of the recipient's identity. If, for various reasons, this encryption is modified or suppressed, the trusted authority would no longer be able to discover the recipient's identity. On the other hand, the recipient might still be able to decrypt the ciphertext and recover the corresponding plaintext message.

Remark 2.3.1. — We note that signing the ciphertext (or similar techniques) does not solve the problem — or does, in addition, require compliant decrypting hardware to check

the ciphertext validity and return the corresponding plaintext message only if the test is successful.

2.3.2.2. Non-malleability. — It is useful to introduce some notation. We extend the decryption algorithm to vectors of ciphertexts (denoted in boldface). If $\boldsymbol{c} = (c_1, \ldots, c_n)$, $\mathsf{Dec}_{\mathsf{sk}}(\boldsymbol{c})$ stands for $(\mathsf{Dec}_{\mathsf{sk}}(c_1), \ldots, \mathsf{Dec}_{\mathsf{sk}}(c_n))$.

Basically, with non-malleability [**DDN00, BDPR98**], the goal of the adversary is, given a ciphertext $\hat{c}$, to output a vector $\boldsymbol{c}$ of ciphertexts whose decryption, $\mathsf{Dec}_{\mathsf{sk}}(\boldsymbol{c})$, is "meaningfully" related to $\mathsf{Dec}_{\mathsf{sk}}(\hat{c})$. We write $\mathcal{R}(\mathsf{Dec}_{\mathsf{sk}}(\boldsymbol{c}), \mathsf{Dec}_{\mathsf{sk}}(\hat{c})) = 1$ for some relation $\mathcal{R}$. Suppose now that the ciphertext output by the above scheme (limited solution) is non-malleable. As a result, if the tracing authority is not able to trace the ciphertext then its intended recipient would not be able to decrypt it. This prevents ciphertext modifications.

The requirement of non-malleability can be captured as the advantage of an adversary $\mathcal{A}$ running some experiment [**BDPR98**] (see also [**BS99**]).

Define experiment Exp_b by

$$\begin{cases} I \xleftarrow{R} \mathsf{CKeyGen}(\lambda), (\mathsf{upk}, \mathsf{usk}) \xleftarrow{R} \mathsf{KeyGen}(I), \\ (\mathcal{M}, s) \leftarrow \mathcal{A}_1(\mathsf{upk}, \mathsf{opk}), \\ m_0, m_1 \xleftarrow{R} \mathcal{M}, \hat{c} \leftarrow \mathsf{Enc}_{\mathsf{upk}}(m_1), \\ (R, \boldsymbol{c}) \leftarrow \mathcal{A}_2(\mathcal{M}, s, \hat{c}) \end{cases} : \quad R(\mathsf{Dec}_{\mathsf{usk}}(\boldsymbol{c}), m_b) = 1 \ .$$

Ciphertext vector $\boldsymbol{c}$ returned by the adversary is supposed to be valid and not containing $\hat{c}$ (i.e., $\perp \notin \mathsf{Dec}_{\mathsf{usk}}(\boldsymbol{c})$ and $\hat{c} \notin \boldsymbol{c}$).

This leads to the property of non-malleability, adapted to our purposes:

Non-malleability **:** For any probabilistic polynomial-time adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the advantage

$$\Pr[\mathsf{Exp}_1(\lambda) = 1] - \Pr[\mathsf{Exp}_0(\lambda) = 1]$$

has to be negligible in security parameter λ whenever $\mathsf{Trace}_{\mathsf{osk}}(\boldsymbol{c}) \neq \mathsf{upk}$.

In the previous definition, the inequality $\mathsf{Trace}_{\mathsf{osk}}(\boldsymbol{c}) \neq \mathsf{upk}$ should not be understood in the strict sense as a mere application of the trace algorithm to the components of $\boldsymbol{c}$. It should be interpreted as "the tracing manager (owning secret opening key osk) cannot recover upk from $\boldsymbol{c}$."

2.4. Transformation

We are now ready to present our transformation. It takes on input a key-private public-key encryption scheme, say $\mathcal{S}_{\mathrm{Msg}}$, and outputs a key-private public encryption scheme with revocable anonymity. Input scheme $\mathcal{S}_{\mathrm{Msg}} = (\mathsf{CKeyGen}, \mathsf{KeyGen}, \mathsf{MsgEnc}, \mathsf{MsgDec})$ is supposed to meet [at least] the IK-CPA security notion.

The transformation requires the presence of a trusted party, called *tracing authority*. The tracing authority owns a pair of keys $(\mathsf{opk}, \mathsf{osk})$ registered for public-key encryption scheme $\mathcal{S}_{\mathrm{ID}} = (\mathsf{IDCKeyGen}, \mathsf{IDKeyGen}, \mathsf{IDEnc}, \mathsf{IDDec})$, meeting [at least] the IE-CPA security notion.

The output scheme $\mathcal{S}'$, after the transformation, is detailed into the following five algorithms: $\mathcal{S}' = (\mathsf{RevCKeyGen}, \mathsf{RevKeyGen}, \mathsf{RevEnc}, \mathsf{RevDec}, \mathsf{Trace})$. Specifically, we have:

Common-key generation : This algorithm takes on input a security parameter λ and outputs common parameter: $I \xleftarrow{R} \mathsf{RevCKeyGen}(\lambda) = \mathsf{CKeyGen}(\lambda)$ —so for the first algorithm the new scheme borrows the first algorithm of the original $\mathcal{S}_{\mathrm{Msg}}$. As mentioned before, I may include a copy of public key opk.

This algorithm also selects a non-malleable symmetric encryption scheme $\mathcal{E}$ and a second-preimage resistant key-derivation function KDF, whose descriptions are made public.

Key generation : This randomized algorithm is run by some user. Given as an input common parameter I, it outputs a pair of matching public encryption key and secret decryption key, $(\mathsf{upk}, \mathsf{usk}) \xleftarrow{R} \mathsf{RevKeyGen}(I) = \mathsf{KeyGen}(I)$.

Public-key usk is registered and certified by some authority (typically, the tracing authority).

Encryption : This randomized algorithm is used to encrypt messages. It takes on input public keys opk and upk (i.e., tracing authority's public key and recipient's public-key), message m to be encrypted, and outputs:

$$\begin{aligned} c &\leftarrow \mathsf{RevEnc}_{\mathsf{opk}}(\mathsf{upk}, m) \\ &= (\mathsf{IDEnc}_{\mathsf{opk}}(\mathsf{upk}), \mathsf{MsgEnc}_{\mathsf{upk}}(\mathcal{E}_k(m)) \end{aligned}$$

with $k = \mathrm{KDF}(\mathsf{IDEnc}_{\mathsf{opk}}(\mathsf{upk}))$ is a key derived from the first part with key derivation function KDF.

Decryption : This algorithm is run by the intended recipient, owning secret key usk. Taking on input ciphertext (c_1, c_2) and usk, this algorithm computes $k' = \mathrm{KDF}(c_1)$ and next $\mathcal{E}_{k'}^{-1}(\mathsf{MsgDec}_{\mathsf{usk}}(c_2))$, which yields corresponding plaintext message m (or $\perp$).

Tracing : This algorithm is run by the tracing authority. On input the first part c_1 of ciphertext $c = (c_1, c_2)$ and secret opening key osk, it returns the public key corresponding to the intended recipient (or $\perp$), $\mathsf{upk} \leftarrow \mathsf{Trace}_{\mathsf{osk}}(c_1) = \mathsf{IDDec}_{\mathsf{osk}}(c_1)$.

2.4.0.3. Discussion. — We remind that we are in the honest-sender setting. In particular, the sender does not encapsulate fake or wrong identities.

The building blocks needed in the transformation require to satisfy some properties. The transformation splits the RevEnc algorithm into two sub-algorithms; i.e., IDEnc and MsgEnc. This relaxes security requirements about RevEnc components. We explain below the choices that were made.

– Encryption schemes $\mathcal{S}_{\mathrm{ID}}$ and $\mathcal{S}_{\mathrm{Msg}}$ are required to be respectively IE and IK so as to prevent that the recipient's identifier (namely, upk) leaks from ciphertext $(\mathsf{IDEnc}_{\mathsf{opk}}(\mathsf{upk}), \mathsf{MsgEnc}_{\mathsf{upk}}(\mathcal{E}_k(m))$.

Remark that IDEnc algorithm is used to output ciphertexts intended only to be decrypted by the tracing authority. This algorithm is therefore not necessarily required to be anonymous. Remark also that key $k = \mathrm{KDF}(c_1)$ is randomized because $c_1 = \mathsf{IDEnc}_{\mathsf{opk}}(\mathsf{upk})$ is randomized.

– Key derivation function KDF is required to be second-preimage resistant. If this requirement is not met, it would be possible to substitute c_1 with a different c_1' having the same image through the key derivation function but disallowing the tracing of upk.

– Symmetric encryption scheme $\mathcal{E}$ is required to be non-malleable.

Remember that in the *non-malleability* definition, the relation "$\mathsf{Trace}_{\mathsf{osk}}(\boldsymbol{c}) \neq \mathsf{upk}$" means that the tracing authority is not able to recover upk from $\boldsymbol{c}$.

Suppose now that a valid ciphertext $c = (c_1, c_2)$ is "perturbed" into $\bar{c} = (\bar{c}_1, c_2)$ such that $\bar{c}_1$ and c_1 only differ in a few bits. In that case, the recipient can by exhaustive search correct $\bar{c}_1$ into c_1 and recover the corresponding plaintext. But, the tracing authority can in the same way recover the recipient's identifier (upk) from corrupted $\bar{c}_1$. We can therefore assume that c_1 is corrupted into $\bar{c}_1$ such that its correct value cannot be recovered by the intended recipient. The goal of the non-malleability requirement on $\mathcal{E}$ is to prevent the recipient to infer information related to m from $\mathcal{E}_k(m) = \mathsf{MsgDec}_{\mathsf{usk}}(c_2)$ because $k = \mathrm{KDF}(c_1)$ is unknown to her (and cannot be guessed).

2.5. Applications

We give two applications of our generic transformation. This first one is based on a traditional public-key infrastructure (PKI) while the second one relies on the identity-based paradigm.

To simplify the presentation, we assume that the group manager (i.e., the authority in charge of setting up and maintaining the system) and the tracing authority are a single entity, called hereafter *system authority*. It is however easy to adapt the schemes to deal with two separate entities.

2.5.1. PKI-based solution. — In [**TY98**], Tsiounis and Yung demonstrated that El Gamal encryption scheme [**Gam84**] in a prime-order group $\mathbb{G}$ is IE-CPA under the decisional Diffie-Hellman assumption. Later, Bellare *et al.* [**BBDP01**] showed that it also achieves IK-CPA security under the same assumption. Hence, applying our generic transformation, we get a public-key encryption scheme with revocable anonymity. The description of the resulting scheme is detailed below.

Common-key generation : Taking as an input some security parameter λ, a group $\mathbb{G}$ of prime order p and a generator $g \in \mathbb{G}$ are selected. The common key is $I = \{p, g\}$.

The message space is denoted by $\mathcal{M}$. Let also a non-malleable symmetric encryption scheme $\mathcal{E} : \mathcal{K} \times \mathcal{M} \to \mathcal{M}$, a second-preimage resistant key-derivation function $\mathrm{KDF} : \mathbb{G} \to \mathcal{K}$, and a cryptographic hash function $H : \mathbb{G} \to \mathcal{M}$.

The secret opening key is some random element $s \in \mathbb{Z}_p$ and the corresponding public opening key is $h = g^s$.

(The public system parameters are $\{\mathcal{M}, p, g, h, \mathcal{E}, \mathrm{KDF}, H\}$.)

Key generation : In order to join the system, user $\mathcal{U}_i$ randomly chooses $x_i \in \mathbb{Z}_p$ and computes $y_i = g^{x_i}$. $\mathcal{U}_i$'s public key is $\mathsf{upk}_i = \{y_i\}$ while $\mathcal{U}_i$'s secret key is $\mathsf{usk}_i = \{x_i\}$.

Encryption : To send a message $m \in \mathcal{M}$ to user $\mathcal{U}_i$, one proceeds as follows:

- pick at random $r_1 \in \mathbb{Z}_p$ and compute $c_1 = (g^{r_1}, y_i \cdot h^{r_1})$;
- compute $k = \mathrm{KDF}(c_1)$, pick at random $r_2 \in \mathbb{Z}_p$, and compute $c_2 = (g^{r_2}, H({y_i}^{r_2}) \oplus \mathcal{E}_k(m))$.

The ciphertext is $c = (c_1, c_2)$.

Decryption : Upon receiving ciphertext $c = (c_1, c_2)$, user $\mathcal{U}_i$ recovers plaintext m as:

- letting $c_2 = (\varphi_1, \varphi_2)$, compute $t = H({\varphi_1}^{x_i})$ using her secret key x_i;
- compute $k' = \mathrm{KDF}(c_1)$ and obtain m as $\mathcal{E}_{k'}^{-1}(t \oplus \varphi_2)$.

Tracing : The intended recipient of ciphertext $c = (c_1, c_2)$ is recovered by the system authority using secret opening key s as:
- letting $c_1 = (\vartheta_1, \vartheta_2)$, compute $y' = \vartheta_2 \cdot {\vartheta_1}^{-s}$;
- check in the list of user's public keys whether there is some $y_j = y'$;
- if so, intended recipient is $\mathcal{U}_j$.

The efficiency of this scheme can be improved by choosing a single random element in $\mathbb{Z}_p$ (i.e., $r_1 = r_2$). A ciphertext is then given by $(g^{r_1}, y_i \cdot h^{r_1}, H({y_i}^{r_1}) \oplus \mathcal{E}_k(m))$.

El Gamal scheme provides data privacy and anonymity against chosen-plaintext attacks. A scheme with revocable anonymity can be similarly obtained by considering the IE-CCA Cramer-Shoup scheme [**CS98**], later proven secure in the IK-CCA sense in [**BBDP01**].

2.5.2. Identity-based solution. — The previous scheme can be adapted to fit the identity-based setting. We replace El Gamal scheme with Boneh-Franklin's BasicIdent scheme [**BF01**]. This latter scheme further requires a bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ such that the bilinear Diffie-Hellman assumption holds [**BF03**].

Applying our generic transformation, we get:

Common-key generatiion : Taking as an input some security parameter λ, groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p, a generator $g \in \mathbb{G}$, and a non-degenerate bilinear pairing $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ are selected.

The message space is $\mathcal{M}$. Let also an encoding function $\mu : \{0,1\}^* \to \mathbb{G}$, a non-malleable symmetric encryption scheme $\mathcal{E} : \mathcal{K} \times \mathcal{M} \to \mathcal{M}$, a second-preimage resistant key-derivation function $\mathrm{KDF} : \mathbb{G} \to \mathcal{K}$, and a cryptographic hash function $H : \mathbb{G}_T \to \mathcal{M}$.

The secret opening keys is some random element $s \in \mathbb{Z}_p$ and the corresponding public opening key is $h = g^s$.

(The public system parameters are $\{\mathcal{M}, p, g, \hat{e}, h, \mu, \mathcal{E}, \mathrm{KDF}, H\}$.)

Key generation : In order to join the system, user $\mathcal{U}_i$ obtains from the system authority her corresponding secret key $\mathsf{usk}_i = \{g_i\}$, where $g_i = \mu(\mathcal{U}_i)^s$.

Encryption : To send a message $m \in \mathcal{M}$ to user $\mathcal{U}_i$, one proceeds as follows:
- pick at random $r_1 \in \mathbb{Z}_p$ and compute $c_1 = (g^{r_1}, \mu(\mathcal{U}_i) \cdot h^{r_1})$;
- compute $k = \mathrm{KDF}(c_1)$, pick at random $r_2 \in \mathbb{Z}_p$, and compute $c_2 = (g^{r_2}, H({z_i}^{r_2}) \oplus \mathcal{E}_k(m))$ where $z_i = \hat{e}(\mu(\mathcal{U}_i), h)$.

The ciphertext is $c = (c_1, c_2)$.

Decryption : Upon receiving ciphertext $c = (c_1, c_2)$, user $\mathcal{U}_i$ recovers plaintext m as:
- letting $c_2 = (\varphi_1, \varphi_2)$, compute $t = H(\hat{e}(g_i, \varphi_1))$ using her secret key g_i;
- compute $k' = \mathrm{KDF}(c_1)$ and obtain m as $\mathcal{E}_{k'}^{-1}(t \oplus \varphi_2)$.

Tracing : The intended recipient of ciphertext $c = (c_1, c_2)$ is recovered by the system authority using secret opening key s as:
- letting $c_1 = (\vartheta_1, \vartheta_2)$, compute $g' = \vartheta_2 \cdot {\vartheta_1}^{-s}$;
- check in the list of users whether there is some $\mathcal{U}_j$ such that $\mu(\mathcal{U}_j) = g'$;
- if so, intended recipient is $\mathcal{U}_j$.

Here too, the efficiency of the scheme can be improved by choosing a single random element in $\mathbb{Z}_p$.

CHAPITRE 3

UN SCHÉMA DE CHIFFREMENT BROADCAST QUI PRÉSERVE L'ANONYMAT AVEC RÉVOCABILITÉ.

Ce chapitre est constitué de l'article « A Privacy-Preserving Broadcast Encryption Scheme with Revocability » publié dans les *Proceedings of the seventh IEEE International Workshop on Digital Rights Management Impact on Consumer Communications (DRM 2011)* et présenté par moi au même workshop en janvier 2011. L'article original est dans [**Ale11**].

3.1. Introduction

In many scenarios, it is crucial that the distributed content to a (large) set of users is kept private. In a commercial context, a broadcaster is interested in distributing digital content only to the paying customers. Similarly, an online store selling electronic books or digital music and movies wishes to keep the distributed content protected from any user but the proper customer. Companies are interested in distributing confidential information inside the company itself to a particular subset of employees (e.g.: accounting members, top management, ...) avoiding people outside the group (or competitors) to learn about the information. To solve this problem, Fiat and Naor formulated the "Broadcast Encryption" problem and provided a first solution in [**FN94**]. After their seminal paper, many different solutions in both public-key and secret-key cryptography have been developed solving different aspects of the main problem.

As already noticed by A. Barth *et al.* in [**BBW06**] sometimes protecting only the broadcast content could be not enough. In a commercial context an online seller would prefer to keep its list of customers secret and protect it from competing companies, e.g. to avoid competitors advertising. Digital media providers could be interested in protecting their customers' privacy avoiding them to be profiled by the analysis of their purchases. A company, giving a call for tenders would like to keep recipients unrevealed to maintain the competition fair.

As already seen in Chapter 2 ([**AJ09**]), complete anonymity of customers is not conceivable. Broadcaster would like to keep the ability to know with whom he is trading, at least in case of legal disputes. Authors in Chapter 2 ([**AJ09**]) provide a trade off between the complete recipient's anonymity and the possibility for a third party, a Trusted Authority, to trace a given transmission.

In this paper, we propose a construction that achieves an open privacy-preserving broadcast encryption allowing every user in the system to act as a broadcaster. With this broadcast encryption scheme, the recipients' privacy is guaranteed. On the other hand, the Tracing Authority (a trusted entity in the system) is able to revoke the anonymity.

Thus revealing the set of recipients for a given broadcast ciphertext. Furthermore, with the same key, Tracing Authority is able to reverse the encryption and recover the plaintext.

Such a scheme is useful in a Digital Rights Management (DRM) system allowing a balanced trade off between the needs of traceability of the broadcaster and the customers' wishes of complete anonymity and privacy. Back to the Digital Media Provider example, it is possible to deploy a system where users keep their privacy protected but, in case of non respect of granted rights by someone, the Tracing Authority is able to easily identify a user and revoke his rights.

3.1.0.1. Related Work. — Formal notions of anonymity in the public-key setting appear in [**BBDP01**]. There should be no way for an adversary to distinguish between a message sent to a given recipient and one addressed to a random one. In [**IP08**], Pointcheval and Izabachène analyze different anonymity levels for identity-based encryption schemes ([**Sha85**]).

3.1.0.2. Our Contribution. — Key privacy in public-key encryption assumes an "homogeneous" environment. Indeed, if users make use of different cryptosystems or of the same cryptosystem but with keys of different lengths, anonymity is likely to be lost. The notion of anonymity is therefore restricted to users sharing the same cryptosystem and common parameters. This implicitly defines a group. In the context of broadcast encryption, the group is explicit. All users in the system share the public parameters generated by the Group Manager during the setup stage. So, all public-keys share all "macro-characteristics" making the notion of privacy meaningful.

To achieve our goal, in this paper, we use a version of Paillier's cryptosystem ([**Pai99**]) revisited by Bresson *et al.* ([**BCP03**]) that provides a double trapdoor. The double trapdoor allows a double decryption process and then two different entities can decrypt the same ciphertext in two different ways, using two different *independent* secrets. Moreover, one of those secrets works as a "global" trapdoor that can be used as universal secret key.

The rest of the paper is organized as follows. In the next section we give some background about public-key broadcast encryption scheme and anonymity. In Section 3.3 we define the Revocable Anonymity. Section 3.4 fully describes our contribution. Finally, we conclude in Section 3.5.

3.2. Preliminaries

In this section, we review classical notions for broadcast encryption, dealing both with data-privacy and key-privacy. We also introduce some useful notation.

3.2.0.3. Broadcast encryption scheme. — In order to better capture the property that users share some common parameters in a homogeneous environment, the key generation algorithm is divided in two sub-algorithms: the *setup* (or common parameters generation) algorithm and the *key generation* algorithm.

Following the syntax of [**BBDP01**], we define a *broadcast encryption scheme* is a tuple of four algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$:

- **Setup** The common parameters generation algorithm Setup takes on input some security parameter κ and outputs some common parameters $I \xleftarrow{R} \mathsf{Setup}(\kappa)$.
- **Key generation** The key generation algorithm KeyGen is a randomized algorithm that takes on input I and returns a matching pair of public key and secret key for

some user: $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{R} \mathsf{KeyGen}(I)$. This algorithm is run once when a new user joins the system.
- **Encryption** Let $\mathcal{M}$ denote the message space. The encryption algorithm Enc is a randomized algorithm that takes as input a set of public keys $\mathcal{S} = \{\mathsf{pk}_i\}_i$ and a plaintext $m \in \mathcal{M}$, and returns a ciphertext c. We write $c \leftarrow \mathsf{Enc}_{\mathcal{S}}(m)$.
- **Decryption** The decryption algorithm Dec takes as input a secret key sk_i and a ciphertext c and returns the corresponding plaintext m if the corresponding $\mathsf{pk}_i \in \mathcal{S}$ or a special symbol $\perp$ indicating that $\mathsf{pk}_i \notin \mathcal{S}$ or that the ciphertext is invalid. We write $m \leftarrow \mathsf{Dec}_{\mathsf{sk}_i}(c)$ if c is a valid ciphertext and $\mathsf{pk} \in \mathcal{S}$ and $\perp \leftarrow \mathsf{Dec}_{\mathsf{sk}_i}(c)$ otherwise.

We require that $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathcal{S}}(m)) = m$ for any message $m \in \mathcal{M}$ and for any $(\mathsf{sk}, \mathsf{pk})$ issued by the KeyGen algorithm with $\mathsf{pk} \in \mathcal{S}$.

3.2.0.4. Indistinguishability of encryptions. — The notion of *indistinguishability of encryptions* [**GM84**] captures a strong notion of [data]-privacy: The adversary should not learn any information whatsoever about a plaintext given its encryption beyond the length of the plaintext.

We view an adversary $\mathcal{A}$ as a pair $(\mathcal{A}_1, \mathcal{A}_2)$ of probabilistic algorithms. This corresponds to adversary $\mathcal{A}$ running in two stages. In the "find" stage, algorithm $\mathcal{A}_1$ takes as input a set $\mathcal{S}$ of public keys pk_i and outputs two equal-size messages m_0 and $m_1 \in \mathcal{M}$ and some state information s. In the "guess" stage, algorithm $\mathcal{A}_2$ receives a challenge ciphertext c which is the encryption of m_b under $\mathcal{S}$ and where b is chosen at random in $\{0, 1\}$. The goal of $\mathcal{A}_2$ is to recover the value of b from s and c.

A (broadcast) encryption scheme is said *semantically secure* (or *indistinguishable*) if the adversary's advantage is a negligible function in the security parameter κ.

As we are in the public-key setting, it is worth noting that adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is given all public keys pk in the system and so can encrypt any message of its choice. In other words, the adversary can mount chosen-plaintext attacks (CPA). Hence, we write *IE-CPA* the security notion achieved by a semantically secure encryption scheme.

A stronger scenario is to give the adversary an adaptive access to a decryption oracle. The previous definition readily extends to this model: adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is allowed to submit any ciphertext of its choice and receives the corresponding plaintext (or $\perp$); the sole exception is that $\mathcal{A}_2$ may not query the decryption oracle on challenge ciphertext c. Likewise, we write *IE-CCA* the corresponding security notion.

3.2.0.5. Indistinguishability of keys. — Analogously, the notion of *indistinguishability of keys* captures a strong requirement about key privacy: The adversary should not be able to link whatsoever a ciphertext with its underlying encryption keys.

As before, we view an adversary $\mathcal{A}$ as a pair $(\mathcal{A}_1, \mathcal{A}_2)$ of probabilistic algorithms. In the "find" stage, algorithm $\mathcal{A}_1$ takes on input two different sets of public keys $\mathcal{S}_0$ and $\mathcal{S}_1$ with the same cardinality and outputs a message m and some state information s. Then in the "guess" stage, algorithm $\mathcal{A}_2$ receives a challenge ciphertext c which is the encryption of m for recipients' set $\mathcal{S}_b$ where b is chosen at random in $\{0, 1\}$. The goal of $\mathcal{A}_2$ is to recover the value of b from s and c.

A broadcast encryption scheme is said *anonymous* (or *key-private*) if the adversary's advantage in winning the above game is a negligible function in the security parameter κ.

This definition of anonymity gives rise to the security notion of *IK-CPA* or *indistinguishability of keys under chosen-plaintext attacks*. If the adversary is given adaptive

access to a decryption oracle, the corresponding security notion is *IK-CCA* or *indistinguishability of keys under chosen-ciphertext attacks*.

Of course, the goals of data-privacy and key-privacy can be combined to define extended security notions. A broadcast encryption scheme achieves *IND-CPA security* if it is both IE-CPA and IK-CPA. Likewise, a broadcast encryption scheme achieves *IND-CCA security* if it is both IE-CCA and IK-CCA.

3.3. Revocable Anonymity

As said in the introduction there are several cases, especially in business related situations, where a complete anonymity of one party is not desirable. However, while it is useful to keep the identity of the receiver private, it may also be useful to have some means to discover the identity of a receiver in case of misuse or dispute. This section formally defines the notion of revocable anonymity in the broadcast encryption context.

3.3.1. Formal definition. — We augment the definition of broadcast encryption scheme so that anonymity can be revoked. The formalization shares many parts with that of a (regular) broadcast encryption scheme. The main differences are *(i)* the generation of a key for tracing purposes in the setup and *(ii)* a tracing algorithm for recovering the intended recipient of a ciphertext.

More formally, a *privacy-preserving broadcast encryption scheme with revocability* is a tuple of five algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Trace})$. Algorithms $\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}$ are as in general description of broadcast encryption scheme (see Section 3.2).

- **Setup** The setup algorithm Setup takes on input some security parameter κ and generates the tracer's secret key tsk and some common parameters I: $(I, \mathsf{tsk}) \xleftarrow{R} \mathsf{Setup}(\kappa)$.
- **Key generation, Encryption, Decryption** See Section 3.2.
- **Tracing** The tracing algorithm Trace takes on input tracing secret key tsk and ciphertext c and returns the corresponding set $\mathcal{S}$ of users' public keys pk_i or a special symbol $\perp$ indicating that the ciphertext is invalid. We write $\mathcal{S} \leftarrow \mathsf{Trace}_{\mathsf{tsk}}(c)$ if c is a valid ciphertext and $\perp \leftarrow \mathsf{Trace}_{\mathsf{tsk}}(c)$ otherwise.

We require for a broadcast encryption scheme with revocable anonymity the following two properties:

- ***Correctness*** For any message $m \in \mathcal{M}$, for any pair of matching public key/secret key returned by the key generation algorithm, $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{R} \mathsf{KeyGen}(I)$ and for any set $\mathcal{S}$ containing pk one has $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathcal{S}}(m)) = m$; and
- ***Traceability*** For any message $m \in \mathcal{M}$, for any pair of matching public key/secret key returned by the key generation algorithm, $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{R} \mathsf{KeyGen}(I)$ and for any set $\mathcal{S}$ containing pk one has $\mathsf{pk} \in \mathsf{Trace}_{\mathsf{tsk}}(\mathsf{Enc}_{\mathcal{S}}(m))$.

Given a [well formed] ciphertext, the first property (*correctness*) ensures that the intended recipient will always recover the corresponding plaintext while the second property (*traceability*) ensures that the tracing authority will always discover the (set of) recipients, if needed.

3.4. Our Construction

In this section, we present a scheme providing key-private broadcast encryption with revocable anonymity.

Before describing our construction, it is useful to introduce some mathematical notions.

Following the notation in [**BCP03**], let N be a safe-prime modulus, i.e. $N = pq$ is a product of safe primes, meaning that $p = 2p' + 1$ and $q = 2q' + 1$ with p' and q' primes. Let then consider the group $\mathbb{QR}_{N^2} = \{x \in \mathbb{Z}^*_{N^2} | x \equiv y^2 \mod N^2, \exists y \in \mathbb{Z}_{N^2}\}$, the group of quadratic residues modulo N^2, its order is $\frac{\lambda(N^2)}{2} = pqp'q' = \frac{N\lambda(N)}{2}$ where $\lambda(N) = 2p'q'$ is the Carmichael function mod N.

One can define the partial discrete logarithm function modulo N, a function that, for a certain subset of $\mathbb{Z}_{N^2}$, using the factorisation of the modulus N, outputs the discrete logarithm computed modulo N. More specifically: let $\mathcal{U} = \{u \in \mathbb{Z}_{N^2} | u \equiv 1 \pmod N\}$. We define the partial discrete logarithm modulo N the function $L_N : \mathcal{U} \to \mathbb{Z}_N$ as:

$$u \mapsto L_N(u) = \frac{u-1}{N} \mod N^2.$$

In our construction we need a *strongly correct* encryption scheme: where decrypting a ciphertext c using a secret key not corresponding to the one used to produce it, outputs the $\perp$ symbol (a symbol to indicate that decryption failed). Usually this property is not required by typical public-key encryption schemes. One can easily add the strong correctness property to a generic public-key encryption scheme by adding a padding function to be run on the plaintext *before* to encrypt it and a verification function during the decryption process to test if the decrypted plaintext is well padded. As example, we can use a function $\mu_{k_0} : \mathcal{M} \to \mathbb{Z}_N$; $x \mapsto \mu_{k_0}(x) = 10 \ldots 0 || x$ where the string $10 \ldots 0$ counts k_0 0s. Another simple example is $\mu_{\mathcal{H}} : \mathcal{M} \to \mathbb{Z}_N$; $x \mapsto \mu_{\mathcal{H}}(x) = \mathcal{H}(x) || x$ where $\mathcal{H}(\cdot)$ is an cryptographic hash function and $||$ is the concatenation operator. Or, at cost of increasing the size of the ciphertext, a strongly correct encryption scheme like Cramer-Shoup can be used [**CS98**].

In the rest of the paper, we assume this pair of encoding and decoding functions and we consider then that the encryption scheme achieves the strong correctness.

In our construction, each private key is a random element a_{ID} (for an user with identifier ID) and the corresponding public key is $h_{\mathsf{ID}} \equiv g^{a_{\mathsf{ID}}} \pmod{N^2}$ where g is a square in $\mathbb{Z}^*_{N^2}$. The encryptor draws a random exponent r and encrypts a ciphertext per each recipient using his public key pk_i. He finally sends the concatenation of all encryptions as a whole ciphertext. In our proposal, the ciphertext counts $n + 1$ group elements, this improves the trivial solution where the ciphertext has length $2n$ group elements. To decrypt the ciphertext the recipient ID has to decrypt every single ciphertext until the decryption is successful, if the decryption process fails, user is not in the set of targeted users.

Tracing Authority is then able to decrypt the ciphertext issued by the encryptor and potentially, in case of need, reveal the set of recipients.

Here follows the detailed description:

– **Setup** During the setuip a probabilistic algorithm is run with a security parameter κ as input and generates two safe primes p, q. Let $N = pq$ as in a standard RSA construction. This algorithm picks also a random square $g \in \mathbb{Z}^*_{N^2}$ of order $\lambda(N)/2$; for example: draw a random $\alpha \in \mathbb{Z}^*_{N^2}$ and set $g \equiv \alpha^{2N} \mod N^2$.

  The common system parameters are $\mathsf{pp} : \{(N, g)\}$.

  The Tracing key is $\mathsf{tsk} = \{\lambda(N)\}$ and is given to the Tracing Authority.

– **Key Generation** This algorithm is run to generate user's public and private key pair: $(\mathsf{pk}, \mathsf{sk})$ and it is run once when a new user joins the system. The KeyGen algorithm is a probabilistic algorithm that takes as input the pp and outputs a random exponent $a \in \mathbb{Z}_N^*$ as secret key sk and the corresponding public key $\mathsf{pk} : h \equiv g^a \pmod{N^2}$.
The key pair is: $(\mathsf{pk}, \mathsf{sk}) : (h, a)$.
The Group Manager certifies the public key pk of the user.
– **Encryption** Given a properly padded message $\dot{m} \in \mathbb{Z}_N$ to be encrypted for a set $\mathcal{S} = \{\mathsf{pk}_i\}$ of n privileged recipients, Enc algorithm proceeds as follows:
1. draw a random $r \in \mathbb{Z}_N$;
2. for each $\mathsf{pk}_i \in \mathcal{S}$ compute: $c_i \equiv h_i^r(1 + \dot{m}N) \pmod{N^2}$;
3. concatenate $\{c_i\}_{\mathsf{pk}_i \in \mathcal{S}}$ in random order;
4. output the ciphertext $C = (C_1, C_2) = (g^r, c_1||c_2|| \dots ||c_n)$.
– **Decryption** Dec algorithm, given a ciphertext C and a secret key $\mathsf{sk} = \{a_{\mathsf{ID}}\}$, proceeds as follows to decrypt the ciphertext and recover the plaintext:
1. parse the ciphertext C as the pair (C_1, C_2), then C_2 is parsed as concatenation of c_i;
2. decrypt every c_i until the correct one is found: compute
$$\tilde{m} \equiv \frac{c_i}{C_1^{a_{\mathsf{ID}}}} - 1 \equiv \frac{h_i^r(1 + \dot{m}N)}{(g^r)^{a_{\mathsf{ID}}}} - 1 \mod N^2$$
and
$$\dot{m} = \frac{\tilde{m}}{N} \text{ over the integers.}$$
It is easily verifiable that the decryption is correct if and only if index i corresponds to secret key a_{ID}. If, indeed, i does not match with the secret key a_{ID} the probability that the decryption process outputs a well padded message is negligible.
– **Tracing** Using Trace algorithm the Tracing Authority is able to decrypt the ciphertext and to recover the identity of the recipients.
To decrypt the ciphertext, Tracing Authority picks c_i at random and computes:
$$\dot{m} \equiv \frac{L_N(c_i^{\lambda(N)})}{\lambda(N)} \mod N$$
where the function $L_N(\cdot)$ is the partial discrete logarithm as defined above.
To reveal a recipient the Tracing Authority:
1. computes $\tilde{c}_i \equiv c_i \mod N$ (notice that $c_i \equiv h_i^r \mod N$)
2. computes then $\frac{L_N(\tilde{c}_i)}{L_N(g^r)} = \frac{ar}{r} = a \mod N$

Checking then for a correspondence in a database containing the t-uples ($a \mod N$, $a_{\mathsf{ID}}, \mathsf{ID}$), the Tracing Authority is able to link the partial secret key $a \mod N$ with the secret key a_{ID} and finally identify the recipient with identifier ID.

Remark 3.4.1. — We state that the database contains a plain copy of users' secret key to keep the scheme simple. It is preferable and raccomended to use a techique to store all the keys in a secure way avoiding any leakage, e.g.: the database stores a hashed version of the public key; A malicious user could only test a key against the database. This technique allows also a faster search during the tracing operation.

The scheme presented here achieves a key-private broadcast encryption allowing the Tracing Authority to revoke the anonymity of (some) recipients if needed. We stress the fact the Group Manager and the Tracing Authority do not share any common secret. So, one is able to deploy a system where roles of Group Manager and Tracing Authority are separated. The role of Group Manager is to certify the link between the user's identifier and his public key and he can delegate the Tracing role to a third trusted party, as a judge or an independent entity.

The previous description leads to a decryption time that is linear in the number of recipients (actually $\frac{n}{2}$ where n represents the size of recipients' set). This can be improved with the solution presented in [**BBW06**]. Authors add a label l_i to each ciphertext component c_i getting $c'_i = l_i||c_i$ and then sort the component in the whole concatenation by this latter l_i. This label is randomly generated via an operation similar to a Diffie-Hellman key-exchange; this can be applied to our scheme setting: $l_i = \mathcal{H}(h_i^r)$. User is able to compute the same label on his side as: $l'_i = \mathcal{H}(C_1^{a_{\mathsf{ID}}})$.

3.5. Conclusion

In this paper, we presented a practical solution for a privacy-preserving broadcast encryption. Advantageously, the presented solution offers a balanced trade-off between privacy and traceability.

CHAPITRE 4

SCHÉMA DE CHIFFREMENT BASÉ SUR LA RÉSIDUOSITÉ QUADRATIQUE

4.1. Définitions et prérequis

Dans cette section nous allons rappeler quelques définitions et fixer la notation que nous utiliserons dans la suite. Commençons d'abord avec la définition classique de la résiduosité n-ième, ensuite nous étudierons en particulier la résiduosité quadratique et plus tard nous généraliserons ce dernier concept et nous présenterons quelques outils, comme le symbole de Legendre, définis initialement pour la résiduosité quadratique et ensuite également adaptés au contexte des résidus de puissance n-ième.

Définition 4.1.1 **(Résidus de puissance n-ième)**. — Soit N un entier positif. Pour tout entier $n \geq 2$ nous pouvons définir :

$$((\mathbb{Z}/N\mathbb{Z})^*)^n = \{x^n \,|x \in (\mathbb{Z}/N\mathbb{Z})^*\}$$

l'ensemble des *résidus de puissance n-ième* modulo N. Si pour un $a \in (\mathbb{Z}/N\mathbb{Z})^*$, il n'existe aucun élément $x \in \mathbb{Z}/N\mathbb{Z}$ tel que $a \equiv x^n \mod N$ alors a est appelé *non-résidu de puissance n-ième* modulo N.

Il existe une caractérisation des éléments résidus de puissance n-ième *modulo* un entier N :

Théorème 4.1.2. — *Soient a, n des entiers positifs avec $n \geq 2$. Soit N un nombre premier et supposons que $\gcd(a, N) = 1$ alors a est un résidu de puissance n-ième si et seulement si*

$$a^{\frac{\varphi(N)}{\gcd(n,\varphi(N))}} \equiv 1 \mod N.$$

Donnons la preuve de ce Théorème :

Démonstration. — Soit a un résidu n-ième *modulo* N. Soit $x \in (\mathbb{Z}/N\mathbb{Z})^*$ une solution de $x^n \equiv a \pmod N$. Puis que $\gcd(a, N) = 1$ alors $\gcd(x, N) = 1$. Alors :

$$\begin{aligned} a^{\frac{\varphi(N)}{\gcd(n,\varphi(N))}} &\equiv (x^n)^{\frac{\varphi(N)}{\gcd(n,\varphi(N))}} \\ &\equiv (x^{\varphi(N)})^{\frac{n}{\gcd(n,\varphi(N))}} \\ &\equiv 1^{\frac{n}{\gcd(n,\varphi(N))}} \\ &\equiv 1 \pmod N. \end{aligned}$$

Inversement, soit $a^{\frac{\varphi(N)}{\gcd(n,\varphi(N))}} \equiv 1 \pmod N$, alors $\alpha^{i\frac{\varphi(N)}{\gcd(n,\varphi(N))}} \equiv 1 \pmod N$ pour un certain $i \in \mathbb{Z}$ et α une racine primitive de l'unité. Par le Petit Théorème de Fermat

nous savons que $\varphi(N) \mid i\frac{\varphi(N)}{\gcd(n,\varphi(N))}$ et donc que $\gcd(n,\varphi(N)) \mid i$ puisque $\frac{i}{\gcd(n,\varphi(N))}$ est un entier. Il existe donc un entier k tel que $\gcd(n,\varphi(N)) \cdot k = i$. Pour la définition de $\gcd(\cdot,\cdot)$ il existe deux entiers u, v tels que : $\gcd(n,\varphi(N)) = un + v\varphi(N)$. Ceci nous donne : $unk + v\varphi(N)k = i$.

Donc après avoir substitué la valeur de a on trouve :

$$a \equiv \alpha^i \equiv \alpha^{unk+v\varphi(N)k} \equiv \alpha^{unk} \cdot \alpha^{v\varphi(N)k} \equiv (\alpha^{uk})^n \pmod{N}$$

qui montre que a est un résidu n-ième *modulo* N. □

Dans notre cas, nous travaillons modulo un entier premier, ce Théorème offre un critère rapide pour discriminer les résidus et les non-résidus de puissance n-ième. Soit p un entier premier. Pour tout entier a tel que $\gcd(a,p) = 1$ nous avons a est résidu de puissance n-ième si et seulement si :

$$a^{\frac{p-1}{\gcd(n,p-1)}} \equiv 1 \mod p.$$

De ce critère, il est possible de définir le *Symbole de Legendre* : une fonction définie dans $(\mathbb{Z}/p\mathbb{Z})^*$ à valeurs dans $\{-1,1\}$ qui permet de discriminer les éléments qui sont des résidus quadratiques et ceux qui n'en sont pas. Considérons donc le cas $n = 2$, soit p un entier premier. Nous définissons le *Symbole de Legendre* comme suit :

***Définition 4.1.3* (Symbole de Legendre).** — Soit p un entier premier, soit $a \in (\mathbb{Z}/p\mathbb{Z})^*$. Le *Symbole de Legendre de* $a \mod p$, noté $\left(\frac{a}{p}\right)$, vaut :

$$\left(\frac{a}{p}\right) = \begin{cases} +1 & \text{si } a \text{ est un résidu quadratique } \textit{modulo } p \\ -1 & \text{si } a \text{ n'en est pas un résidu quadratique } \textit{modulo } p \end{cases}.$$

Remarque 4.1.4. — Nous observons d'abord que $\left(\frac{a}{p}\right)^2 = 1$ pour tout $a \in (\mathbb{Z}/p\mathbb{Z})^*$ et donc, que d'après le critère précèdent, nous pouvons alors écrire :

$$\left(\frac{a}{p}\right) = a^{\frac{p-1}{2}} \pmod{p}$$

pour tout élément $a \in (\mathbb{Z}/p\mathbb{Z})^*$.

De plus, nous pouvons prendre cette dernière égalité comme définition du Symbole de Jacobi.

En suivant la définition alternative donnée dans la Remarque précédente nous pouvons généraliser celle-ci pour un entier n diffèrent de 2.

***Définition 4.1.5* (Résidu de puissance n-ième modulo un entier premier p.)**
Soit p un entier premier ($p > 2$), soit $n \geq 2$ un entier tel que n soit un diviseur de $p-1$. Alors nous définissons le symbole :

$$\left(\frac{a}{p}\right)_n = a^{\frac{p-1}{n}} \pmod{p}$$

comme le *Symbole de Résiduosité n-ième* modulo p, et avec $a^{\frac{p-1}{n}} \pmod{p}$ on considère l'élément le plus petit, pris en valeur absolue, qui le représente ; c'est à dire compris dans l'intervalle $[-\frac{p-1}{2}, \cdots, 0, \cdots, +\frac{p-1}{2}]$. L'élément $\left(\frac{a}{p}\right)_n$ est, d'après sa définition, une *racine n-ième de l'unité dans l'ensemble* $(\mathbb{Z}/p\mathbb{Z})^*$.

Le Symbole que nous avons défini ci-dessus a les propriétés suivantes : supposons que a et b soient deux entiers premiers avec p.

1. si $a \equiv b \pmod{p}$ alors $\left(\frac{a}{p}\right)_n \equiv \left(\frac{b}{p}\right)_n$;

2. $\left(\frac{ab}{p}\right)_n = \left(\frac{a}{p}\right)_n \cdot \left(\frac{b}{p}\right)_n \pmod p$
3. $\forall a \in (\mathbb{Z}/p\mathbb{Z})^*$ il vaut $\left(\frac{a^n}{p}\right)_n = \left(\frac{a}{p}\right)_n^n = 1$;
4. $\left(\frac{1}{p}\right)_n = 1$ et $\left(\frac{-1}{p}\right)_n = (-1)^{\frac{p-1}{n}}$.

Les propriétés sont immédiatement vérifiées par la définition de $\left(\frac{\cdot}{p}\right)_n$ et les propriétés de l'algèbre modulaire.

Nous pouvons aussi étendre cette fonction aux entiers non premiers. Les notions que nous avons présentées sont étendues par multiplicativité ; nous donnons ci-dessous explicitement la définition de Résiduosité Quadratique :

***Définition 4.1.6* (Résiduosité Quadratique modulo un entier composé)**
Soit $N = \prod_{i=0}^{n} p_i^{e_i}$, où les p_i sont des entiers premiers impairs et les e_i des entiers positifs. Donné un élément $a \in (\mathbb{Z}/N\mathbb{Z})^*$, c'est à dire tel que $\gcd(a, N) = 1$, nous pouvons définir le *Symbole de Jacobi* modulo N, comme le produit de symboles de Legendre correspondants aux facteurs de N et il sera noté comme pour le Symbole de Legendre :

$$\left(\frac{a}{N}\right) = \prod_{i=0}^{n} \left(\frac{a}{p_i}\right)^{e_i}.$$

De la définition du Symbole de Jacobi, il est naturel de définir l'ensemble suivant :

$$\mathbb{J}(N) = \left\{ a \in (\mathbb{Z}/N\mathbb{Z})^* \middle| \left(\frac{a}{N}\right) = 1 \right\},$$

appelé aussi sous ensemble des éléments *pseudo-quadratiques* modulo N.

Un sous ensemble de importance particulière dans la suite est celui des résidus quadratiques *modulo* N :

$$\mathbb{QR}(N) = \left\{ a \in (\mathbb{Z}/N\mathbb{Z})^* \middle| \left(\frac{a}{p_i}\right) = 1 \, \forall i \right\}.$$

Dans la suite nous sommes amenés à calculer des Symboles 4-ièmes *modulo* un entier composite. Donnons ici la définitions en généralisant celle ci-dessus.

Soit i une racine primitive 4-ième de l'unité, c'est à dire que $i^l \neq 1$, $l = 1, 2, 3$ et $i^4 = 1$. Soit $N = pq$ où p, q sont deux nombres premiers dans $\mathbb{Z}$. Considérons p, q comme éléments de $\mathbb{Z}[i]$. Soit $\alpha \in \mathbb{Z}[i]$ un élément non nul. α est dit être un élément *premier* si pour tout $a, b \in \mathbb{Z}[i]$ tels que $\alpha \mid ab$ alors $\alpha \mid a$ ou $\alpha \mid b$.

Soit $\alpha \in \mathbb{Z}[i]$ un élément non nul, soit $\pi \in \mathbb{Z}[i]$ un élément premier qui ne divise pas α. Nous pouvons alors définir le *résidu de puissance* 4-*ième* modulo π comme :

$$\left(\frac{\alpha}{\pi}\right)_4 = i^l$$

où l est l'entier in $\{0, \ldots, 3\}$ tel que $\alpha^{\frac{N(\pi)-1}{4}} \equiv i^l \pmod \pi$.

Nous notons que dans cette définition est équivalente à celle donnée dans 4.1.5 puisque $N(\pi) = p$.

Passons maintenant à définir l'équivalent du Symbole de Jacobi :

***Définition 4.1.7* (Résiduosité Quartique modulo un entier composé)**
Soit $N = \prod_{i=0}^{n} p_i^{e_i}$, où les p_i sont des entiers premiers impairs et les e_i des entiers positifs. Soit $p_i = \pi_i \bar{\pi}_i$ où les π_i sont des éléments premiers dans $\mathbb{Z}[i]$. Donné un élément $a \in (\mathbb{Z}/N\mathbb{Z})^*$ nous pouvons définir le *Symbole de Résiduosité Quartique* modulo N, comme

le produit de Symboles de Legendre correspondants aux facteurs de N et il sera noté comme pour le Symbole de Legendre :

$$\left(\frac{a}{N}\right)_4 = \prod_{i=0}^{n} \left(\frac{a}{\pi_i}\right)_4^{e_i}.$$

Le Symbole de Résiduosité Quartique satisfait aussi les propriétés suivantes : supposons que α et β soient deux entiers premiers avec π.

1. si $\alpha \equiv \beta \pmod{\pi}$ alors $\left(\frac{\alpha}{\pi}\right)_4 \equiv \left(\frac{\beta}{\pi}\right)_4$;
2. $\left(\frac{\alpha\beta}{\pi}\right)_4 = \left(\frac{\alpha}{\pi}\right)_4 \cdot \left(\frac{\beta}{\pi}\right)_4 \pmod{\pi}$
3. $\forall \alpha \in (\mathbb{Z}/\pi\mathbb{Z})^*$ il vaut $\left(\frac{\alpha^4}{\pi}\right)_4 = \left(\frac{\alpha}{\pi}\right)_4^4 = 1$.

Les propriétés sont immédiatement vérifiées par la définition de $\left(\frac{\cdot}{\pi}\right)_4$ et les propriétés de l'algèbre modulaire.

Un algorithme rapide pour le calcul de ce Symbole est décrit dans la Section 5 de [**DF03**].

Une définition analogue est donnée pour le Symbole de Résiduosité 2^k-ième *modulo* un premier p. Soit ζ une racine 2^k-ième primitive de l'unité. L'entier premier p peut être factorisé en éléments premiers dans $\mathbb{Z}[\zeta]$ comme $p = \pi\sigma_1(\pi)\cdots\sigma_{2^k-1}(\pi)$, où $\sigma_i(\cdot)$ est le morphisme de conjugaison.

Soit α un élément non nul et soit π un élément premier qui ne divise pas α. Nous définissons le *Symbole de Résiduosité 2^k-ième* modulo p la valeur :

$$\left(\frac{\alpha}{\pi}\right)_{2^k} = \zeta^l$$

où l est l'entier tel que $\alpha^{\frac{N(\pi)-1}{2^k}} \equiv \zeta^l \pmod{\pi}$.

Le Symbole d'ordre 2^k-ième satisfait les propriétés suivantes : comme dans le cas précèdent supposons que α et β soient deux entiers premiers avec π.

1. si $\alpha \equiv \beta \pmod{\pi}$ alors $\left(\frac{\alpha}{\pi}\right)_{2^4} \equiv \left(\frac{\beta}{\pi}\right)_{2^k}$;
2. $\left(\frac{\alpha\beta}{\pi}\right)_{2^k} = \left(\frac{\alpha}{\pi}\right)_{2^k} \cdot \left(\frac{\beta}{\pi}\right)_{2^k} \pmod{\pi}$
3. $\forall \alpha \in (\mathbb{Z}/\pi\mathbb{Z})^*$ il vaut $\left(\frac{\alpha^{2^k}}{\pi}\right)_{2^k} = \left(\frac{\alpha}{\pi}\right)_{2^k}^{2^k} = 1$.

Encore une fois la définition du *Symbole de Jacobi d'ordre* 2^k est donnée par multiplicativité. Soit $N = pq$ et soit $p = \pi\sigma_1(\pi)\cdots\sigma_{2^k-1}(\pi)$ et $q = \rho\sigma_1(\rho)\cdots\sigma_{2^k-1}(\rho)$ leur décomposition en éléments premiers dans $\mathbb{Z}[\zeta]$. Alors le Symbole *modulo* N est défini comme :

$$\left(\frac{\alpha}{N}\right)_{2^k} = \left(\frac{\alpha}{\pi}\right)_{2^k}\left(\frac{\alpha}{\rho}\right)_{2^k}.$$

Étudions à présent les notions vues précédemment appliquées au contexte de la cryptographie. Dorénavant l'entier composite N sera de la forme dite « RSA-*secure* », c'est à dire le produit de deux nombres premiers de taille « assez grande » pour éviter que N soit factorisé facilement.

Dans le cas où la factorisation de N n'est pas connue, il n'est pas possible en effet d'appliquer le critère vu précédemment et, au contraire, distinguer les éléments résidus et ceux non-résidus quadratiques modulo un composite, est un problème difficile.

Dire que déterminer la résiduosité quadratique d'un élément choisi aléatoirement *modulo* un composite N est « difficile » si la factorisation de N n'est pas connue (au sens juste expliqué) peut être réécrit plus formellement :

Définition 4.1.8 **(Hypothèse de la Résiduosité Quadratique (ou *Quadratic Residuosity Assumption*))**

Soit $\mathsf{RSAGen}(\cdot)$ un algorithme probabiliste qui, recevant en entrée un paramètre de sécurité κ, retourne deux nombres premiers p et q et leur produit $N = pq$. Alors, pour tout algorithme probabiliste en temps polynomial $\mathcal{B}$, prenant en entrée un entier composé et un élément *pseudo-quadratique* z, qui donne en sortie une réponse du type « $z \in \mathbb{QR}(N)$ » ou « $z \notin \mathbb{QR}(N)$ » nous avons :

$$\left|\Pr\left[\mathcal{B}\left(\begin{array}{l} N \leftarrow \mathsf{RSAGen}(\kappa), \\ z \in_R \mathbb{J}(N) \end{array}\right) = \text{« } z \in \mathbb{QR}(N) \text{ »} \,\middle|\, z \in \mathbb{QR}(N)\right] - \right.$$
$$\left.\Pr\left[\mathcal{B}\left(\begin{array}{l} N \leftarrow \mathsf{RSAGen}(\kappa), \\ z \in_R \mathbb{J}(N) \end{array}\right) = \text{« } z \in \mathbb{QR}(N) \text{ »} \,\middle|\, z \notin \mathbb{QR}(N)\right]\right|$$

est une fonction de κ qui tend vers 0, plus rapidement que l'inverse de toute fonction polynomiale, lors que le paramètre de sécurité κ tend vers l'infini, pour tout choix d'algorithme probabiliste en temps polynomial $\mathcal{B}$ qui donne en sortie une des deux réponses possibles, prenant en entrée un challenge pour le problème de la Résiduosité Quadratique, où la probabilité est calculée sur l'expérience obtenue avec $(N, p, q) \leftarrow \mathsf{RSAGen}(1^\kappa)$ et sur les choix aléatoires de $z \in \mathbb{QR}(N)$ et $z \in \mathbb{J}(N) \setminus \mathbb{QR}(N)$.

Pour résumer, cette définition signifie que pour tout algorithme *efficace*, c'est à dire probabiliste en temps polynomial, les probabilités d'obtenir une même réponse en sortie (z est un résidu quadratique dans la définition ci-dessus) avec en entrée les deux possibilités différentes est environ la même. En simplifiant, cette expression décrit la différence de probabilité qu'un algorithme a de répondre correctement ou de donner une réponse fausse. Il faut remarquer qu'aucune autre hypothèse n'est faite sur l'algorithme $\mathcal{B}$; la définition de $\mathcal{B}$ est volontairement la plus générique possible. Ceci afin de ne pas restreindre le comportement de l'adversaire. Qu'importe les procédés qu'il met en œuvre pour réussir, il suffit juste qu'il soit « crédible ». Il faut noter que l'algorithme peut aussi donner une réponse fausse en sortie.

4.2. Schéma de chiffrement de Goldwasser-Micali

Dans cette section nous allons présenter la description du schéma de chiffrement à clé publique donnée par Goldwasser et Micali en 1984 [**GM84**].

Dans l'histoire de la recherche dans le domaine de la cryptologie ce schéma a un une importance remarquable. Grâce a ce travail une nouvelle notion de sécurité a été définie. En effet si d'un côté il est « facile » d'énumérer les propriétés qu'un « bon » schéma de chiffrement doit avoir de l'autre il est compliqué de donner une rigoureuse définition de *sécurité*. Demander seulement que la fonction de chiffrement soit à sens unique n'est souvent pas assez. Il peut être requis que la connaissance d'une information partielle sur le message en clair ne permet pas de retrouver d'autre nouvelle information. Cette notion est désormais la référence pour tout schéma générique. Ceci a été formalisée dans la définition de *sécurité sémantique* par Goldwasser et Micali dans leur papier [**GM84**]. Ils ont aussi

introduit la définition de *indistinguabilité des chiffrés*, une notion de sécurité équivalente mais préférable d'un point de vue pratique car il est plus simple de la traiter. En simplifiant il est demandé à un adversaire de reconnaître lequel, parmi deux messages en clair de son choix, a été chiffré sous une clé publique qu'il connaît. Les détails de cette notion sont donnés dans le Chapitre 1, dans la section 1.1.2. Habituellement reconnaître quel message a été chiffré est aussi indiqué comme « récupérer un *bit* d'information ».

Du concept de sécurité sémantique il est évident de s'attendre à ce que le chiffrement doit être probabiliste. Un parmi les premier schémas de ce type est celui donné par Goldwasser-Micali dans [**GM84**]. Le schéma est basé sur le problème Décisionnel de la Résiduosité Quadratique (*Decisional Quadratic Residuosity* en langue anglaise et il est noté, d'ordinaire, *QR*), défini dans la Section 4.1.8.

Comme l'on verra dans la suite l'intérêt reste surtout dans le domaine théorique car un point de faiblesse de ce schéma est l'expansion du chiffré. En effet à tout *bit* du message en clair il correspond un entier *modulo* N. Une telle expansion est un vrai problème dans une utilisation pratique qui en rendre impraticable le déploiement. Vues son élégance et simplicité, plus son importance historique il est intéressant de rechercher comment l'améliorer. Après la description du schéma dans sa forme originale nous présentons deux améliorations et successivement la notre.

Donnons à présent la description détaillée du schéma de Goldwasser-Micali. Nous notons que l'algorithme d'initialisation Setup n'est pas présent, cela est dû à l'absence dans ce schéma des paramètres publics communs.

$\mathsf{KeyGen}(1^\kappa)$ **:** Donné comme entrée le paramètre de sécurité κ, l'algorithme KeyGen génère deux nombres premiers p, q et il calcule leur produit $N = pq$. L'algorithme définit aussi un entier $y \in (\mathbb{Z}/N\mathbb{Z})^*$ tel que $y \in \mathbb{J}(N) \setminus \mathbb{QR}(N)$.

La clé publique est $\mathsf{pk} = (N, y)$ et la clé privée correspondante est $\mathsf{sk} = p$.

$\mathsf{Encrypt}(\mathsf{pk}, m)$ **:** Soit $\mathcal{M} = \{0, 1\}$ l'espace des messages en clair. Pour générer le chiffré c du message $m \in \mathcal{M}$ l'algorithme de chiffrement $\mathsf{Encrypt}$:

- prend un $x \in_R (\mathbb{Z}/N\mathbb{Z})^*$ aléatoirement et
- calcule $c \equiv y^m x^2 \mod N$.

Le chiffré en sortie est c.

$\mathsf{Decrypt}(\mathsf{sk}, c)$ **:** L'algorithme de déchiffrement $\mathsf{Decrypt}$ pour récupérer le message en clair m à partir du chiffré c procède en calculant d'abord $z = \left(\frac{c}{p}\right)$, puis en donnant en sortie le message m tel que $(-1)^m = z$.

Avant de donner la preuve de sécurité, montrons la complétude du schéma, à savoir que les fonctions de chiffrement et de déchiffrement sont bien l'une l'inverse de l'autre. Si le message chiffré c a été correctement généré nous obtenons que :

$$z = \left(\frac{c}{p}\right) = \left(\frac{y^m x^2}{p}\right) = \left(\frac{y^m}{p}\right) \cdot \left(\frac{x^2}{p}\right) = \left(\frac{y}{p}\right)^m \cdot 1 = (-1)^m,$$

car y n'est pas non plus un carré *modulo* p. Si cela avait été le cas, comme $\left(\frac{y}{N}\right) = +1$ on aurait eu aussi $\left(\frac{y}{q}\right) = +1$ et y serait un carré *modulo* N, contrairement à l'hypothèse.

Cette équation montre la complétude du schéma de Goldwasser-Micali. Nous notons pareillement que l'on ne sait pas calculer le symbole $\left(\frac{c}{p}\right)$ sans connaître N et que le symbole de Jacobi ne donne aucune information : $\left(\frac{c}{N}\right) = \left(\frac{y^m x^2}{N}\right) = \left(\frac{y}{N}\right)^m \cdot \left(\frac{x^2}{N}\right) = 1$.

4.2.1. Sécurité du schéma de Goldwasser-Micali. — Démontrons maintenant la sécurité IND-CPA : l'indistinguabilité face à des attaques par clair choisi. Dorénavant dans cette section nous nous référons simplement à la sécurité en sous-entendant IND-CPA. Le niveau de sécurité sera explicité, si différent, par la suite. Comme nous l'avons annoncé avant la description du schéma, la sécurité de celui-ci est basée sur le problème dit de la Résiduosité Quadratique. Nous prouvons la sécurité via une réduction du schéma au problème « difficile », c'est à dire pour lequel la communauté mathématique aujourd'hui ne connaît pas d'algorithme efficace pour le résoudre, et ainsi estime qu'il n'y en a pas.

Nous allons montrer maintenant la sécurité du schéma par contradiction. Supposons que le schéma ne soit pas sûr et donc qu'il existe un adversaire $\mathcal{A}$ capable de distinguer entre les chiffrements de deux messages et montrons comment nous pouvons utiliser sa capacité pour résoudre une instance quelconque du problème de la Résiduosité Quadratique. Soit

$$\Pr\left[b = b' \;\middle|\; \begin{array}{c}(N,y) \leftarrow \mathsf{KeyGen}(1^\kappa), s \leftarrow \mathcal{A}_1((N,y)) \\ b \in_R \{0,1\}, c_* = \mathsf{Enc}((N,y),b), b' \leftarrow \mathcal{A}_2(s,(N,y),c_*)\end{array}\right] = \delta(\kappa)$$

la probabilité que l'adversaire $\mathcal{A}$, considéré comme un couple d'algorithmes $(\mathcal{A}_1, \mathcal{A}_2)$, répondant correctement au jeu décrit dans la définition de la sécurité sémantique. L'objectif dans la preuve sera de montrer que cette fonction $\delta(\kappa)$ doit être très proche de la fonction constante $\frac{1}{2}$. Cela signifiant que la différence entre les deux fonctions est une fonction négligeable en le paramètre de sécurité κ pour des valeurs assez grandes du paramètre de sécurité.

Remarquons aussi que dans ce jeu, le choix pour les deux messages en clair est imposé. En effet l'adversaire doit choisir deux messages différents et il n'y a que deux messages possibles : 0 et 1.

Soit (N_*, V_*) un challenge bien construit pour le problème de la Résiduosité Quadratique, soit donc $N_* = pq$ un entier donné en sortie par l'algorithme $\mathsf{RSAGen}(\kappa)$ dont la factorisation est inconnue et $V_* \in \mathbb{J}(N_*)$ un entier dont le symbole de Jacobi est égal à $+1$. Il nous est demandé de déterminer si V_* est un résidu quadratique *modulo* N_* ou non. Nous générons une instance du schéma de chiffrement de Goldwasser-Micali en simulant l'exécution de l'algorithme $\mathsf{KeyGen}(\kappa)$. Nous introduisons la notation $\mathsf{KG}(\kappa)$ à représenter l'algorithme $\mathsf{KeyGen}(\kappa)$ simulé, c'est à dire où la condition sur y est relaxée en $y \in \mathbb{J}(N)$. Nous posons la clé publique $(N, y) = (N_*, V_*) \leftarrow \mathsf{KG}(\kappa)$ et nous envoyons cette clé publique à l'adversaire $\mathcal{A}$. Nous remarquons que, sous l'hypothèse de la Résiduosité Quadratique, du point de vue de l'adversaire cette clé publique est bien générée et indistinguable d'une clé publique sortie d'une vraie instance du schéma. Du point de vue de l'adversaire les deux algorithmes $\mathsf{KeyGen}(\cdot)$ et $\mathsf{KG}(\cdot)$ sont indistinguables. Effectivement, si l'adversaire $\mathcal{A}$ peut distinguer une clé d'une instance « simulée » d'une instance correcte cela signifie qu'il peut détecter que V_* n'est pas un carré *modulo* N. Ceci représente donc une instance du schéma de Goldwasser-Micali du point de vue de l'adversaire $\mathcal{A}$.

Nous chiffrons le message (choisi aléatoirement) $b \in_R \{0,1\}$ pour obtenir le chiffré challenge, nous tirons aléatoirement un élément $x \in (\mathbb{Z}/N\mathbb{Z})^*$ et nous calculons la quantité suivante :

$$c_* = y^b x^2 \pmod{N}.$$

Comme nous pouvons aisément le voir, le chiffré est bien construit et sous l'hypothèse de la Résiduosité Quadratique est indistinguable d'un chiffré issu d'une vraie instance du schéma.

Le message chiffré c_* est donné à l'adversaire $\mathcal{A}$ qui retourne son bit de réponse b'. Si l'adversaire a deviné donc $b = b'$ alors la réponse au challenge initial est « $V_* \notin \mathbb{QR}(N_*)$ » c'est à dire que V_* est un non-résidu quadratique. Dans le cas contraire si l'adversaire ne répond pas correctement donc $b \neq b'$ alors réponse au challenge initial est « $V_* \in \mathbb{QR}(N_*)$ », à savoir V_* est un résidu quadratique. Ou pour résumer en symboles, la réponse de l'algorithme $\mathcal{B}$ au challenge sur la Résiduosité Quadratique est :

$$\mathsf{guess}_{\mathcal{QA}} = \mathcal{B}\left(\begin{array}{c} N \leftarrow \mathsf{RSAGen}(\kappa), \\ V_* \in_R \mathbb{J}(N) \end{array}\right) = \left\{\begin{array}{l} \text{« } V_* \notin \mathbb{QR}(N) \text{ » : si } b = b' \\ \text{« } V_* \in \mathbb{QR}(N) \text{ » : si } b \neq b' \end{array}\right. .$$

Nous avons réduit le Problème Décisionnel de la Résiduosité Quadratique à rompre la distinguabilité du schéma de chiffrement de Goldwasser-Micali. En effet nous avons décrit comment utiliser l'adversaire $\mathcal{A}$ comme part d'un algorithme probabiliste en temps polynomial $\mathcal{B}$ pour résoudre le problème de la Résiduosité Quadratique. L'algorithme $\mathcal{B}$, de fait, utilisant $\mathcal{A}$ comme une fonction interne, pris en entrée un couple challenge pour le Problème Décisionnel de la Résiduosité Quadratique, donne en sortie une réponse.

Plus exactement, dans ce cas, nous parlons de réduction polynomiale. Pour résumer cela signifie qu'une solution algorithmique pour la distinguabilité des chiffrés peut être efficacement transformée en une solution pour le problème de la Résiduosité Quadratique sans perte d'efficacité. Appelons $\mathcal{R}$ cette réduction.

Par ailleurs l'hypothèse sur la Résiduosité Quadratique 4.1.8 nous dit que un tel algorithme qui tourne en temps polynomial n'a pas une meilleure réussite de celui qui répond au hasard et donc la relation donnée dans la définition 4.1.8 qui est valable pour *tout algorithme probabiliste en temps polynomiale* doit être également vérifiée, comme un cas particulier, par l'algorithme $\mathcal{B}$ que nous avons décrit ci-dessus.

Nous calculons l'avantage de la réduction :

$$\epsilon_{\mathsf{QRA}}(\kappa) = \left|\Pr\left[\mathcal{B}\left(\begin{array}{l} N \leftarrow \mathsf{RSAGen}(\kappa), \\ z \in_R \mathbb{J}(N) \end{array}\right) = \text{« } z \in \mathbb{QR}(N) \text{ »} \middle| z \in \mathbb{QR}(N)\right] - \right.$$
$$\left.\Pr\left[\mathcal{B}\left(\begin{array}{l} N \leftarrow \mathsf{RSAGen}(\kappa), \\ z \in_R \mathbb{J}(N) \end{array}\right) = \text{« } z \in \mathbb{QR}(N) \text{ »} \middle| z \notin \mathbb{QR}(N)\right]\right| .$$

Et au travers de la réduction $\mathcal{R}$ nous pouvons réécrire cette relation en fonction de l'algorithme $\mathcal{A}$: nous observons que la réduction $\mathcal{R}$ a pour seul rôle de transférer les données entre le challengeur et l'algorithme $\mathcal{A}$ et elle n'introduit aucun biais, ceci nous donne directement :

$$\epsilon_{\mathsf{QRA}}(\kappa) = \left|\Pr\left[b = b' \middle| \begin{array}{c} (N,y) \leftarrow \mathsf{KG}(1^\kappa), s \leftarrow \mathcal{A}_1((N,y)) \\ b \in_R \{0,1\}, c_* = \mathsf{Enc}((N,y),b), b' \leftarrow \mathcal{A}_2(s,(N,y),c_*) \\ y \in \mathbb{QR}(N) \end{array}\right] - \right.$$
$$\left.\Pr\left[b = b' \middle| \begin{array}{c} (N,y) \leftarrow \mathsf{KG}(1^\kappa), s \leftarrow \mathcal{A}_1((N,y)) \\ b \in_R \{0,1\}, c_* = \mathsf{Enc}((N,y),b), b' \leftarrow \mathcal{A}_2(s,(N,y),c_*) \\ y \notin \mathbb{QR}(N) \end{array}\right]\right| .$$

Nous observons que dans la simulation, si $V_* = y \notin \mathbb{QR}(N)$ alors tout fonctionne comme dans une vraie instance du schéma et l'adversaire est en face d'un vrai challenge. Le deuxième terme de la soustraction représente la probabilité qu'un adversaire a de réussir le jeu sur la distinguabilité face à une vraie instance et d'après l'hypothèse sur l'adversaire,

la probabilité de réussite est :

$$\Pr\left[b = b' \;\middle|\; \begin{array}{c}(N,y) \leftarrow \mathsf{KeyGen}(1^\kappa), s \leftarrow \mathcal{A}_1((N,y)) \\ b \in_R \{0,1\}, c_* = \mathsf{Enc}((N,y),b), b' \leftarrow \mathcal{A}_2(s,(N,y),c_*)\end{array}\right] = \delta(\kappa).$$

Dans le cas contraire, quand $V_* = y \in \mathbb{QR}(N)$, le chiffré de b est un carré choisi aléatoirement dans $(\mathbb{Z}/N\mathbb{Z})^*$ indépendamment du choix aléatoire du bit b. Effectivement, quand y est un résidu quadratique *modulo* N nous avons $c_* = y^b x^2 \in \mathbb{QR}(N)$, avec $x \in_R (\mathbb{Z}/N\mathbb{Z})^*$ choisi aléatoirement, le chiffré c_* est un élément bien distribué dans $((\mathbb{Z}/N\mathbb{Z})^*)^2$, l'ensemble de résidus quadratiques *modulo* N. Cela nous indique que, sans devoir faire aucune hypothèse calculatoire, l'adversaire $\mathcal{A}$ n'obtient aucune information à partir du chiffré c_* donc la probabilité qu'il devine le bit b dans le jeu est

$$\Pr\left[b = b' \;\middle|\; \begin{array}{c}(N,y) \leftarrow \mathsf{KeyGen}(1^\kappa), s \leftarrow \mathcal{A}_1((N,y)) \\ b \in_R \{0,1\}, c_* = \mathsf{Enc}((N,y),b), b' \leftarrow \mathcal{A}_2(s,(N,y),c_*)\end{array}\right] = \frac{1}{2}.$$

Ceci nous donne :

$$\begin{aligned}\epsilon_{\mathsf{QRA}}(\kappa) = \Bigg|\Pr\left[\mathcal{A}\left(\begin{array}{c}N_* \leftarrow \mathsf{RSAGen}(\kappa), \\ V_* \in_R \mathbb{J}(N)\end{array}\right) = \text{« } V_* \in \mathbb{QR}(N) \text{ »} \;\middle|\; V_* \in \mathbb{QR}(N)\right] - \\ \Pr\left[\mathcal{A}\left(\begin{array}{c}N_* \leftarrow \mathsf{RSAGen}(\kappa), \\ V_* \in_R \mathbb{J}(N)\end{array}\right) = \text{« } V_* \in \mathbb{QR}(N) \text{ »} \;\middle|\; V_* \notin \mathbb{QR}(N)\right]\Bigg| = \\ = \left|\frac{1}{2} - \delta(\kappa)\right|.\end{aligned}$$

L'algorithme $\mathcal{B}$ a donc un avantage $\epsilon_{\mathsf{QRA}}(\kappa)$ de résoudre correctement le challenge du problème de la Résoduosité Quadratique ; de l'hypothèse QRA (4.1.8) nous savons que tout algorithme a un avantage négligeable de réussite, ceci implique que pour $\mathcal{B}$ doit valoir le même. Et de ceci $\delta(\kappa)$ doit être négligeable aussi. En effet, de la dernière égalité nous pouvons en déduire que l'avantage de l'adversaire $\mathcal{A}$ pour casser la sécurité sémantique du schéma de Goldwasser-Micali est :

$$\mathsf{Adv}_{\mathcal{A}} = |2 \cdot \delta(\kappa) - 1| \leq 2\varepsilon(\kappa),$$

qui reste une fonction négligeable dans le paramètre de sécurité κ pour des valeurs « suffisamment grandes » du paramètre.

Pour résumer, il n'existe pas d'adversaire capable de casser la sécurité sémantique du schéma de chiffrement de Goldwasser-Micali avec une probabilité différente de $\frac{1}{2}$, c'est à dire qu'il n'existe pas d'attaque au schéma meilleure de deviner au hasard la valeur du bit b.

4.3. Généralisation du schéma de chiffrement de Goldwasser-Micali

Dans cette section nous allons décrire une généralisation du schéma de Goldwasser-Micali.

Nous observons que le schéma de chiffrement de Goldwasser-Micali a une grande expansion du chiffré : tout bit du clair est chiffré singulièrement avec un entier *modulo* N. L'expansion du chiffré pour le schéma original de Goldwasser-Micali est linéaire : un message de taille l bits est chiffré dans un message de taille $l \cdot \log N$ bits. Pour cette raison il peut être difficilement utilisé, même couplé avec le paradigme du KEM-DEM.

Un premier essai de généralisation et amélioration de ce schéma est dû à Benaloh et Fischer [**Ben87, CF85**]. Ils ont proposé d'utiliser un entier r de taille k bits pour pouvoir chiffrer des messages (vus comme des entiers) en clair $m < r$ dans un seul entier composite N. En revanche, pour déchiffrer, l'utilisateur doit procéder par recherche exhaustive dans l'intervalle $[0, r[$ qui est plus efficace qu'une recherche dans l'espace complet $\{0,1\}^k$.

En détails :

KeyGen(1^κ) **:** Donné comme entrée un paramètre de sécurité κ l'algorithme KeyGen génère deux nombres premiers p et q et calcule leur produit $N = pq$. Il définit aussi un entier impair premier r de taille k bits et tel que $r \mid p-1$, $r^2 \nmid p-1$ et $r \nmid q-1$. Il définit aussi un entier $y \in (\mathbb{Z}/N\mathbb{Z})^*$ tel que $y^{\varphi(N)/r} \not\equiv 1 \pmod N$.

Encrypt **:** Pour chiffrer un message de k bits $m < r$, l'envoyeur calcule $c \equiv y^m x^r \pmod N$ où x est un entier choisi aléatoirement.

Decrypt **:** L'utilisateur, destinataire d'un chiffré, pour récupérer le clair recherche le $m \in [0, r[$ tel que :

$$(y^{\varphi(N)/r})^m \equiv c^{\varphi(N)/r} \pmod N.$$

Le message, déchiffré de c, m est donné en sortie.

Le schéma est démontré étant sûr sous l'hypothèse de la Résiduosité Première, une généralisation de l'hypothèse de la Résiduosité Quadratique. L'hypothèse affirme qu'il est *difficile* (au sens expliqué précédemment) de distinguer les éléments qui sont des résidus premiers et ceux qui ne le sont pas *modulo* un entier N dont la factorisation est inconnue.

Pour rester praticable cette solution requiert que l'espace des messages ne soit pas *trop* étendu, disons $k < 20$.

Ce schéma a été successivement amélioré par Naccache et Stern dans [**NS98**]. Les auteurs ont montré comment le déchiffrement peut être accéléré en utilisant R produit de « petits » premiers plutôt qu'un seul premier. Le message est ensuite relevé *modulo* N via le Théorème des Restes Chinois. Ceci permet de partager la recherche exhaustive dans des espaces plus petits.

En détails le schéma de Naccache-Stern :

KeyGen **:** Donné comme entrée un paramètre de sécurité κ, l'algorithme donne en sortie deux entiers premiers p, q et leur produit $N = pq$. Il donne aussi un entier $R = \prod_i^k r_i$, produit des entiers premiers r_i avec $r_i \neq r_j$ pour $i \neq j$ et tel que $r \mid \varphi(N)$ mais $r_i^2 \nmid \varphi(N)$, $\forall r_i$. Il choisit un élément $g \in (\mathbb{Z}/N\mathbb{Z})^*$ tel que $\mathrm{ord}(g) \mid R$.

La clé publique est $\mathsf{pk} = (N, g)$. La clé secrète correspondante est $\mathsf{sk} = (p, q)$.

Encrypt **:** Soit $\mathcal{M} = \{m | m < R\}$. Pour chiffrer un message $m \in \mathcal{M}$, l'envoyeur calcule le chiffré $c = g^m \mod N$.

Puis, le chiffré c est envoyé au destinataire.

Decrypt **:** Le déchiffrement est basé sur le Théorème des Restes Chinois.

L'algorithme calcule $c_i = c^{\varphi(N)/r_i} \pmod N$, $\forall r_i$. Puis il trouve, par recherche exhaustive le message $m_i \in \{0, \ldots, r_i - 1\}$ tel que l'égalité suivante soit vérifiée :

$$g^{m_i \varphi(N)/r_i} \equiv c_i \pmod N.$$

Après avoir récupéré tous les messages m_i, l'algorithme calcule la solution m à :

$$\begin{cases} m & \equiv & m_1 \pmod{r_1} \\ & \cdots & \\ m & \equiv & m_k \pmod{r_k} \end{cases}.$$

L'algorithme donne enfin le message déchiffré m en sortie.

Ce schéma est prouvé indistinguable sous l'hypothèse de la Résiduosité Première.

L'avantage de ce dernier schéma par rapport à celui donné pas Benaloh et Fischer est que la recherche exhaustive est décomposée en k recherches plus rapides. L'expansion des chiffrés est assez similaire et comparable pour des messages *courts*, c'est à dire tant que $k \leq \log r$. Le schéma Naccache-Stern devient plus intéressant et efficace pour des grands messages.

Une comparaison entre ces schémas et notre proposition ci-dessous se trouve après la preuve de sécurité dans la Section 4.4.4.

4.3.1. Notre schéma. — Nous donnons ici la description de notre schéma : il s'agit d'une amélioration du schéma de Goldwasser-Micali qui permet d'obtenir une expansion mineur du chiffré. La description est similaire à celle de l'algorithme de Goldwasser-Micali et nous utiliserons la même notation. Comme pour Goldwasser-Micali l'algorithme Setup est absent. Ce travail a été à l'origine d'un brevet qui est en cours de dépôt (**PCT/EP11/066883**). Nous donnons ici la description générale d'un schéma de la famille indexé par le paramètre k entier positif :

$\mathsf{KeyGen}(1^\kappa)$ **:** Donné comme entrée le paramètre de sécurité κ, l'algorithme KeyGen génère deux nombres premiers p et q tel que $p, q \equiv 1 \pmod{2^k}$ et il calcule leur produit $N = pq$. Les premiers p et q sont tels que nous avons :

$$\left(\frac{-1}{p}\right)_{2^k} = -\left(\frac{-1}{q}\right)_{2^k} = -1.$$

L'algorithme définit aussi un entier $y \in (\mathbb{Z}/N\mathbb{Z})^*$ tel que $y \in \mathbb{J}(N) \setminus \mathbb{QR}(N)$. Il est requis que

$$\left(\frac{y}{N}\right)_{2^i} = 1 \; \forall i \leq k.$$

La clé publique est $\mathsf{pk} = (N, y, k)$ et la clé privée correspondante est $\mathsf{sk} = p$.

$\mathsf{Encrypt}(\mathsf{pk}, m)$ **:** Soit $\mathcal{M} = \{0,1\}^k$ l'espace des messages en clair. Le chiffré du message $m \in \mathcal{M}$, vu comme le développement d'un entier m dans l'ensemble $\{0, \cdots, 2^k - 1\}$, est le suivant : l'algorithme de chiffrement $\mathsf{Encrypt}$:

- prend un $x \in_R (\mathbb{Z}/N\mathbb{Z})^*$ aléatoirement et
- calcule $c \equiv y^m x^{2^k} \mod N$.

Le chiffré en sortie est c.

$\mathsf{Decrypt}(\mathsf{sk}, c)$ **:** L'algorithme de déchiffrement $\mathsf{Decrypt}$ pour récupérer le message en clair m à partir du chiffré c procède comme il suit :

- il calcule $z = \left(\frac{c}{p}\right)_{2^k}$, comme défini dans la Définition 4.1.5 ;
- il trouve $m \in \{0, \cdots, 2^k - 1\}$ tel que la relation

$$\left[\left(\frac{y}{p}\right)_{2^k}\right]^m = z \pmod{p}$$

soit vérifiée.

Nous notons que dans le détail de l'algorithme $\mathsf{Decrypt}(\mathsf{sk}, c)$ il y a une recherche par exhaustion du message en clair correspondant. Cette recherche est évidement non efficace. Cette description est donnée ici pour sa similarité avec le cas particulier du schéma de

Goldwasser-Micali ($k = 1$). Nous donnerons une description d'une méthode plus efficace pour récupérer le message bit à bit après la preuve de sécurité.

Avant de montrer que le schéma ci-dessus détaillé possède la sécurité sémantique, nous donnons la preuve que le schéma est correct ; un message bien chiffré est correctement déchiffré par un utilisateur qui possède la clé secrète correspondante à celle publique utilisée pour le chiffrement, autrement dit : les fonctions de chiffrement et déchiffrement sont l'une l'inverse de l'autre quand elles sont calculées avec les clés publiques et secrètes correspondantes.

Soit $a = \left(\frac{y}{p}\right)_{2^k}$. L'élément $a \in (\mathbb{Z}/p\mathbb{Z})^*$ et d'ordre 2^k, en fait soit $n = \text{ord}(a)$ l'ordre de a comme élément de $(\mathbb{Z}/p\mathbb{Z})^*$, alors, par la définition du Symbole de Jacobi nous savons que $a = \left(\frac{y}{p}\right)_{2^k} = y^{\frac{p-1}{2^k}} \pmod p$. Alors $n \mid 2^k$. Ceci implique que $n = 2^{k'}$ pour un $k' \leq k$; mais si $k \neq k'$ alors $a^{2^{k'}} \equiv 1 \pmod p$ est en contradiction avec l'hypothèse que $y \notin \mathbb{QR}(N)$. Si l'ordre de a est maximale, alors l'algorithme de déchiffrement donne en sortie le seul et unique $m \in \{0, \ldots, 2^k - 1\}$ tel que $\left[\left(\frac{y}{p}\right)_{2^k}\right]^m = z \pmod p$.

4.3.2. Sécurité du Schéma de chiffrement de Goldwasser-Micali généralisé. — Dans cette section nous donnons la preuve de sécurité pour le schéma, au sens IND-CPA, de manière explicite pour la valeur $k = 2$. Dans ce cas le schéma résultant obtient en sortie un chiffré de taille réduite à la moitié par rapport au schéma de Goldwasser-Micali classique. Nous nous restreindrons au cas $k = 2$ aussi dans les preuves des Lemmes. Le même raisonnement et la même technique peuvent être utilisés pour étendre le résultat au cas général d'un schéma de chiffrement indexé par un entier $k \geq 2$.

Théorème 4.3.1. — *Sous l'*Hypothèse de la Résiduosité Quadratique*, le schéma de Goldwasser-Micali généralisé a la propriété IND-CPA, c'est à dire qu'il est* sémantiquement sûr *face aux attaques à clair choisi.*

La preuve est donnée via une réduction au problème difficile de la Résiduosité Quadratique. Nous allons montrer par l'absurde qu'un adversaire réel pour la sécurité sémantique du schéma, s'il existe, peut être utilisé pour résoudre efficacement une instance du problème de la Résiduosité Quadratique. Nous construisons un algorithme $\mathcal{B}$ qui utilise l'adversaire $\mathcal{A}$ comme une fonction interne pour donner la réponse à un challenge sur le problème de la Résiduosité Quadratique.

Avant de montrer ce théorème, nous montrons d'abord un *Lemme* qui sera utilisé dans la preuve du Théorème 4.3.1.

Nous allons montrer que sous l'hypothèse de la Résiduosité Quadratique le chiffrement du message $m = 0$ est indistinguable du chiffrement du message $m_i = 2^i \, \forall i \leq k$.

Remarque 4.3.2. — Avant de donner la preuve du Théorème et des Lemmes nous faisons une observation qui nous sera utile dans la suite.

Soit $z \in \mathbb{J}(N)$ un élément non nul. Nous savons que $p \equiv q \equiv 1 \pmod 4$ donc -1 est un carré *modulo* p et *modulo* q. Il n'est pas une restriction de supposer que $\left(\frac{z}{N}\right)_4 = 1$. Si ceci n'était pas le cas nous pouvons substituer z avec son opposé $-z \mod N$. Puisque -1 est un carré *modulo* N nous savons que $z \in \mathbb{QR}(N)$ si et seulement si $-z \in \mathbb{QR}(N)$ et nous calculons son Symbole $\left(\frac{-z}{N}\right)_4 = \left(\frac{-1}{N}\right)_4 \left(\frac{z}{N}\right)_4 = 1$ car par hypothèse $\left(\frac{-1}{N}\right)_4 = -1$.

Lemme 4.3.3. — *Sous l'Hypothèse de la Résiduosité Quadratique les distributions des chiffrés de la forme $m = 2^i \; \forall i < k$ sont indistinguables du chiffré du message $m = 0$ pour le schéma de Goldwasser-Micali généralisé.*

Nous donnons la preuve explicite de ce Lemme dans le cas $k = 2$.

Démonstration. — Nous allons démontrer que sous l'hypothèse de la Résiduosité Quadratique le chiffrement du message $m = 0$ est indistinguable du chiffrement du message respectivement $m = 2^0 = 1$ et $m = 2^1 = 2$.

Soit $(N, y, 2)$ une clé publique comme dans la définition du schéma. Il s'agit de démontrer que il n'existe pas d'adversaire efficace capable de faire la distinction entre le chiffre $c_0 \equiv y^0 x^4 \pmod N$ et $c_1 \equiv y^{2^i} x^4 \pmod N$ pour $i = 0, 1$.

Nous le montrons par l'absurde. Soit $\mathcal{A}$ un distingueur capable de discerner entre les deux chiffrements. Définissons sa probabilité de réussite dans le jeu sur l'indistinguabilité comme :

$$\Pr\left[b = b' \;\middle|\; \begin{array}{c}(N, y, 2) \leftarrow \mathsf{KeyGen}(1^\kappa), s \leftarrow \mathcal{A}_1((N, y, 2)),\\ b \in_R \{0, 1\}, c_* = y^{2^i b} x^4 \pmod N\\ b' \leftarrow \mathcal{A}_2(c_*, s), y \notin \mathbb{QR}(N)\end{array}\right] = \delta(\kappa).$$

Réduisons cette adversaire $\mathcal{A}$, vu comme un couple d'algorithmes probabilistes en temps polynomial, à un algorithme capable de résoudre la Résiduosité Quadratique. Soit, à cette fin, (N, z) un challenge pour la Résiduosité Quadratique : $(N, p, q) \leftarrow \mathsf{RSAGen}(\kappa)$ et $p \equiv 1 \equiv q \pmod 4$, $z \in \mathbb{J}(N)$. Détaillons un algorithme $\mathcal{B}$ qui, pris comme entrée le challenge, donne en sortie la réponse à la question : « $z \in \mathbb{QR}(N)$? » et calculons son avantage.

$\mathcal{B}$ transforme le challenge donné en entrée dans une clé publique en posant : $\left(N, y = \left(\frac{z}{N}\right)_4 z, 2\right)$.

Observons que $\left(\frac{z}{N}\right)_4$ vaut ± 1 et que $\left(\frac{\pm 1}{N}\right) = 1$. De ceci dérive que $\left(\frac{y}{N}\right) = 1$. La clé est donc indistinguable d'une vraie clé à moins de pouvoir discerner entre résidus et non-résidus quadratiques *modulo* N.

L'algorithme $\mathcal{A}_1$ prend en entrée la clé publique et donne en sortie un message $m_1 = 2^i (i \in \{0, 1\})$ sur lequel être challengé. $\mathcal{B}$ calcule le chiffré challenge :

$$c \equiv y^{2^i b} x^4 \pmod N$$

avec le bit $b \in_R \{0, 1\}$ et $x \in_R (\mathbb{Z}/N\mathbb{Z})^*$.

$\mathcal{A}_2$ reçoit en entrée le challenge c et donne en sortie sa réponse b'.

$\mathcal{B}$ transforme ce bit b en une réponse pour le challenge initial :

$$\mathsf{guess}_{\mathcal{QA}} = \mathcal{B}\left(\begin{array}{c}N \leftarrow \mathsf{RSAGen}(\kappa),\\ z \in_R \mathbb{J}(N)\end{array}\right) = \left\{\begin{array}{l}\text{« } z \notin \mathbb{QR}(N) \text{ » : si } b = b'\\ \text{« } z \in \mathbb{QR}(N) \text{ » : si } b \neq b'\end{array}\right.$$

Calculons l'avantage de l'algorithme $\mathcal{B}$:

$$\left|\Pr\left[b = b' \,|\, z \in \mathbb{QR}(N)\right] - \Pr\left[b = b' \,|\, z \notin \mathbb{QR}(N)\right]\right|.$$

Réécrivons le en fonction de l'algorithme $\mathcal{A}$:

$$\left|\Pr\left[b = b' \;\middle|\; \begin{array}{c}(N, y, 2) \leftarrow \mathsf{KG}(1^\kappa), s \leftarrow \mathcal{A}_1((N, y, 2))\\ b \in_R \{0, 1\}, c_* = y^{2^i b} x^4 \pmod N,\\ b' \leftarrow \mathcal{A}_2(c_*, s), z \in \mathbb{QR}(N)\end{array}\right] - \Pr\left[b = b' \;\middle|\; \begin{array}{c}(N, y, 2) \leftarrow \mathsf{KG}(1^\kappa), s \leftarrow \mathcal{A}_1((N, y, 2)),\\ b \in_R \{0, 1\}, c_* = y^{2^i b} x^4 \pmod N\\ b' \leftarrow \mathcal{A}_2(c_*, s), z \notin \mathbb{QR}(N)\end{array}\right]\right|.$$

Le second terme, quand $z \notin \mathbb{QR}(N)$, représente la probabilité que l'adversaire a face à une vraie instance du schéma de deviner la valeur du bit b, cette probabilité vaut, pour l'hypothèse faite sur $\mathcal{A}$, $\delta(\kappa)$.

Le premier terme, d'un autre côté, quand $z \in \mathbb{QR}(N)$, représente la probabilité que l'adversaire donne la bonne réponse mis en face avec une fausse instance. Calculons cette probabilité.

Si $z \in \mathbb{QR}(N)$, alors il existe $t \notin \mathbb{QR}(N)$, $j \geq 1$ tel que $z \equiv t^{2^j} \pmod N$. Si $z \equiv t^2 \pmod N$ et $t \in \mathbb{QR}(N)$ je peux réitérer à nouveau en raisonnant sur t à la place que sur z. Si cette itération termine pour un $j \leq k$ j'arrête et j'obtiens $z \equiv t^{2^j} \pmod N$. Sinon si $z \equiv t^{2^k} \pmod N$ avec $t \in \mathbb{QR}(N)$ alors $z \equiv (-t)^{2^k} \pmod N$ et $-t \notin \mathbb{QR}(N)$.

Traitons les deux cas : $j \geq 2$ et $j = 1$ séparément.

Cas $j \geq 2$: Si $j \geq 2$ (c'est à dire $2^j \geq 4$) alors le challenge est réécrit de la forme suivante :

$$c \equiv y^{2^i b} x^4 \equiv \left(\left(\frac{t^{2^j}}{N} \right)_4 t^{2^j} \right)^{2^i b} x^4 \equiv t^{2^{i+j} b} x^4 \pmod N$$

et donc c est un résidu quartique bien distribué dans $((\mathbb{Z}/N\mathbb{Z})*)^4$ indépendamment de la valeur du bit b. L'adversaire donc ne peut rien apprendre du chiffré. La probabilité qu'il donne la bonne réponse est donc égale à $\frac{1}{2}$.

Cas $j = 1$: Supposons maintenant que z est juste un carré et non une puissance quatrième. Soit alors $z \equiv t^2 \pmod N$.

Nous avons à nouveau deux cas :

1. $\left(\frac{z}{N}\right)_4 = \left(\frac{t}{N}\right) = +1$, ou
2. $\left(\frac{z}{N}\right)_4 = \left(\frac{t}{N}\right) = -1$.

Dans le premier cas ($\left(\frac{t}{N}\right) = +1$) le chiffré challenge c devient :

$$c \equiv y^{2^i b} x^4 \equiv \left(t^2\right)^{2^i b} x^4 \equiv t^{2^{1+i} b} x^4 \pmod N$$

qui, comme dans le cas précèdent est un résidu quartique indépendamment de la valeur du bit b si $i \geq 1$. Dans le cas opposé ($i = 0$) nous pouvons parcourir un raisonnement similaire en changeant la fonction de chiffrement avec la fonction $m \mapsto y^{2m} x^4 \pmod N$.

Dans le second cas ($\left(\frac{t}{N}\right) = -1$) nous pouvons faire le même raisonnement avec $-z \pmod N$ à la place de z. En utilisant $-z$, comme nous l'avons observé auparavant dans la Remarque 4.3.2, la clé reste bien formée et nous avons $\left(\frac{t}{N}\right) = +1$.

Nous avons montré que, si l'adversaire est face à une fausse instance, où la clé est formée à partir d'un élément quadratique, il ne peut donner la bonne réponse qu'une fois sur deux. En symboles :

$$\Pr\left[b = b' \;\middle|\; \begin{array}{c} (N, y, 2) \leftarrow \mathsf{KG}(1^\kappa), s \leftarrow \mathcal{A}_1((N, y, 2)), \\ b \in_R \{0, 1\}, c_* = y^{2^i b} x^4 \pmod N \\ b' \leftarrow \mathcal{A}_2(c_*, s), y \in \mathbb{QR}(N) \end{array} \right] = \frac{1}{2}.$$

Substituant les valeurs dans la formule de l'avantage pour l'algorithme $\mathcal{B}$ nous obtenons :

$$\left| \delta(\kappa) - \frac{1}{2} \right| = \varepsilon_{\mathsf{QRA}}(\kappa)$$

d'où l'avantage de $\mathcal{B}$ dans la résolution d'un challenge est :

$$\varepsilon_{\mathsf{QRA}}(\kappa) = \frac{1}{2} + \delta(\kappa)$$

qui n'est pas négligeable si $\delta(\kappa)$ n'est pas une fonction négligeable.

De l'hypothèse QRA nous savons que $\varepsilon_{\mathsf{QRA}}(\kappa)$ est négligeable, ceci nous dit aussi qu'un adversaire générique ne peut avoir qu'un avantage négligeable. □

Corollaire 4.3.4. — *Les chiffrements des messages $m_0, m_1 \in \mathcal{M}$ de la forme $m = 2^i \pmod{2^k}$ sont indistinguables entre eux.*

Soit $m_0 = 2^i$ et $m_1 = 2^j$ (supposons que soit $i < j$), appelons leur différence $l = j - i$. Pour la preuve l'idée est de raisonner par induction (descendante) sur l'exposant j du message $m_1 = 2^j$ et analyser successivement les cas $l = 1, \ldots$. Il s'agit d'une situation analogue à celle vue dans la preuve du Lemme 4.3.3, qui représente le cas $j = 0$ coïncidant avec le cas $j = k$ et la démonstration procéde de la même manière.

Remarque 4.3.5. — Nous observons que de ce Théorème et de son Corollaire il vient que sous l'Hypothèse de la Résiduosité Quadratique il est *infaisable* de distinguer les éléments qui sont *puissances de la forme* 2^i.

Avant de donner la démonstration du Théorème, nous montrons un deuxième Lemme.

Lemme 4.3.6. — *Sous l'Hypothèse de la Résiduosité Quadratique si les distributions $y^m x^{2^k} \pmod{N}$, chiffrements des messages m, sont indistinguables de la distribution $x^{2^k} \pmod{N}$ pour $1 \leq m \leq j$, alors la distribution des chiffrés du message $j+1$ est aussi indistinguable de $x^{2^k} \pmod{N}$.*

Démonstration. — Donnons la preuve explicite du cas particulier $k = 2$. Supposons que les distributions générées par les chiffrements des messages $m = 1$ et $m = 2$ soient indiscernables de celle générée par le chiffrement du message $m = 0$.

Autrement dit soit $(N, y, 2)$ une clé publique pour le schéma de chiffrement de Goldwasser-Micali généralisé.

Soit par définition $\mathcal{D}_i = \{N, y, 2, c \equiv y^i x^4 \pmod{N} : x \in (\mathbb{Z}/N\mathbb{Z})^*\}$ la distribution des chiffrements du message $i \in \{0, 1, 2, 3\}$. Par hypothèse nous avons $\mathcal{D}_0$ et $\mathcal{D}_1$ indistinguables et c'est également le cas de $\mathcal{D}_0$ et $\mathcal{D}_2$.

Nous affirmons que de ceci vient que $\mathcal{D}_0$ et $\mathcal{D}_3$ sont indistinguables si l'Hypothèse de la Résiduosité Quadratique tient.

Procédons par l'absurde et supposons qu'il existe un distingueur $\mathcal{A}$ capable de deviner la valeur du bit $b \in_R \{0, 1\}$ en prenant comme entrée le challenge $c \equiv y^{3b} x^4 \pmod{N}$. Soit, par définition :

$$\delta_i(\kappa) = \Pr\left[b = b' \;\middle|\; \begin{array}{c} (N, y, 2) \leftarrow \mathsf{RSAGen}(\kappa), s \leftarrow \mathcal{A}_1((N, y, 2)) \\ b \in_R \{0, 1\}, c \equiv y^{ib} x^4 \pmod{N} \\ b' \leftarrow \mathcal{A}_2(s, c) \end{array} \right]$$

la probabilité que l'algorithme $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ discerne les deux distributions des chiffrés des messages i et 0. Définissons de la même manière son avantage comme :

$$\epsilon_i(\kappa) = \left| \delta_i(\kappa) - \frac{1}{2} \right|.$$

Par hypothèse nous savons que $\delta_1(\kappa)$, la probabilité de discerner entre $\mathcal{D}_0$ et $\mathcal{D}_1$, est une fonction très proche de $\frac{1}{2}$ puisque les deux distributions sont indistinguables. Notons la comme : $\delta_1(\kappa) = \frac{1}{2} + \epsilon_1(\kappa)$, où $\epsilon_1(\kappa)$ est une fonction négliegeable.

Montrons comment réduire cet adversaire à un algorithme capable de discriminer les deux distributions $\mathcal{D}_0$ et $\mathcal{D}_2$.

Soit $c_* \equiv y^{2b}x^4 \pmod N$ un challenge bien construit. La réduction modifie c_* en c'_* en calculant :

$$c'_* \equiv c_* \cdot y \equiv y^{2b+1}x^4 \pmod N.$$

Notons que le challenge est, sous l'hypothèse que $\mathcal{D}_0$ et $\mathcal{D}_1$ soient indistinguables, indistinguable d'un challenge pour $\mathcal{A}$ avec une probabilité de $1 - \epsilon_1(\kappa)$.

Détaillons maintenant la stratégie de réponse de la réduction au challenge initial :

$$\mathsf{guess} = \begin{cases} \text{« } b'' = 1 \text{ » : si } b' = 1 \\ \text{« } b'' = 0 \text{ » : si } b' = 0 \end{cases}.$$

Calculons l'avantage que la réduction $\mathcal{B}$ a dans la distinction d'un élément de la distribution $\mathcal{D}_0$ d'un de $\mathcal{D}_2$.

$$\begin{aligned} \mathsf{Adv}_{\mathcal{B}} &= \left| \Pr\left[b'' = b\right] - \frac{1}{2} \right| = \\ &= \left| \Pr\left[b'' = 1 \,|b = 1\right] \Pr\left[b = 1\right] + \Pr\left[b'' = 0 \,|b = 0\right] \Pr\left[b = 0\right] - \frac{1}{2} \right| = \\ &= \left| \Pr\left[b'' = 1 \,|b = 1\right] \cdot \frac{1}{2} + \Pr\left[b'' = 0 \,|b = 0\right] \cdot \frac{1}{2} - \frac{1}{2} \right|. \end{aligned}$$

Le choix du bit b est aléatoire dans $\{0, 1\}$ et donc $\Pr[b = 1] = \Pr[b = 0] = \frac{1}{2}$. Quand $b = 1$ le distingueur $\mathcal{A}$ se trouve devant un challenge du type $c'_* \equiv y^3x^4 \pmod N$ et il donne en sortie la bonne réponse $b' = 1$ avec une probabilité de $\delta_3(\kappa)$. La valeur en sortie b' est transformée dans la réponse $b'' = 1$ qui est correcte avec la même probabilité. Quand $b = 0$ le distingueur $\mathcal{A}$ est face à un élément $c'_* \equiv yx^4 \pmod N$ qui est traité par le distingueur comme un élément de la forme $x^4 \pmod N$ sauf s'il arrive à discerner la différence entre $\mathcal{D}_0$ et $\mathcal{D}_1$. Il répondra donc le bit correct b avec la probabilité $\delta_3(\kappa)$ dans le cas il ne distingue pas les distributions et ceci arrive avec une probabilité de $1 - \epsilon_1(\kappa)$. Quand $b = 0$ la probabilité que $\mathcal{A}$ donne la bonne réponse est (au moins, nous ne pouvons pas savoir comme il répond quand il fait la distinction entre $\mathcal{D}_0$ et $\mathcal{D}_1$) de $\delta_3(\kappa) \cdot (1 - \epsilon_1(\kappa))$.

$$\begin{aligned} \mathsf{Adv}_{\mathcal{B}} &= \left| \frac{1}{2}\delta_3(\kappa) + \frac{1}{2}\delta_3(\kappa)\left(1 - \epsilon_1(\kappa)\right) - \frac{1}{2} \right| = \\ &= \frac{1}{2}\left|2\delta_3(\kappa) - \epsilon_1(\kappa)\delta_3(\kappa) - 1\right|. \end{aligned}$$

Or $\mathsf{Adv}_{\mathcal{B}}$ peut être une fonction négligeable que seulement si $|2\delta_3(\kappa) - \epsilon_1(\kappa)\delta_3(\kappa) - 1|$ est une fonction négligeable. Ou en réécrivant en fonction de $\epsilon_3(\kappa)$:

$$\left| \left(\frac{1}{2} + \epsilon_3(\kappa)\right)\left(1 - \frac{\epsilon_1(\kappa)}{2}\right) - \frac{1}{2} \right|$$

est négligeable. D'où vient que $\epsilon_3(\kappa)$ est négligeable. □

Montrons maintenant le Théorème 4.3.1 sur la sécurité sémantique du schéma de Goldwasser-Micali généralisé.

Démonstration. — Procédons par l'absurde. Supposons donc qu'il existe un adversaire $\mathcal{A}$ capable de casser la sécurité sémantique pour $k = 2$. Nous définissons $\mathcal{A}$ comme un algorithme probabiliste en temps polynomial. Pour simplifier la notation nous considérons l'algorithme $\mathcal{A}$ comme un couple d'algorithmes $(\mathcal{A}_1, \mathcal{A}_2)$ qui tournent dans les deux phases

d'attaque. En symboles sa probabilité de succès est :

$$(1) \qquad \Pr\left[b = b' \;\middle|\; \begin{array}{c}(N,y,2) \leftarrow \mathsf{KeyGen}(1^\kappa), (m_0,m_1,s) \leftarrow \mathcal{A}_1((N,y,2)),\\ b \in_R \{0,1\}, c_* = \mathsf{Enc}((N,y,2),m_b), b' \leftarrow \mathcal{A}_2(c_*,s)\end{array}\right] = \delta(\kappa).$$

Montrons comment un tel adversaire peut être réduit à un algorithme pour la résolution d'un challenge QR. Soit donc (N,z) un challenge pour le problème QR. C'est à dire $N = pq$, avec $p \equiv q \equiv 1 \pmod 4$ et $\left(\frac{-1}{p}\right)_4 = -\left(\frac{-1}{q}\right)_4 = -1$, $z \in \mathbb{J}(N)$.

Pour répondre à ce défi il faut déterminer si $z \in \mathbb{QR}(N)$ ou non.

Utilisons ce challenge pour simuler une instance de notre schéma de chiffrement. Posons la clé publique $\mathsf{pk} : (N, y := \left(\frac{z}{N}\right)_4 \cdot z, 2)$. La clé publique simulée a la bonne forme : en fait N est bien construit et $\left(\frac{y}{N}\right) = 1$.

Dans les détails :

$$\left(\frac{y}{N}\right) = \left(\frac{\left(\frac{z}{N}\right)_4 \cdot z}{N}\right) \begin{cases} \left(\frac{z}{N}\right) = 1 & \text{si } \left(\frac{z}{N}\right)_4 = 1 \\ \left(\frac{-z}{N}\right) = \left(\frac{-1}{N}\right)\left(\frac{z}{N}\right) = 1 \cdot 1 = 1 & \text{si } \left(\frac{z}{N}\right)_4 = -1 \end{cases}$$

De ceci et de la Remarque 4.3.2 nous pouvons supposer sans perte de généralité que $\left(\frac{z}{N}\right)_4 = 1$ dans la suite de la preuve.

La clé publique est donc bien simulée et, sous l'hypothèse de la Résiduosité Quadratique, est indistinguable d'une créée par l'algorithme de génération des clés.

Ayant obtenu en entrée la clé publique, l'algorithme $\mathcal{A}_1$ choisit deux messages $m_0, m_1 \in \{0,\ldots,3\}$ avec la seule contrainte que $m_0 \neq m_1$.

Le message m_b, choisi aléatoirement entre les deux, est chiffré pour obtenir le chiffré challenge :

$$c_b \equiv y^{m_b} x^4 \pmod 4$$

où x est un élément aléatoire dans $(\mathbb{Z}/N\mathbb{Z})^*$.

L'algorithme $\mathcal{A}_2$ retourne son bit b' de réponse. Nous transformons cette réponse pour une réponse au challenge sur le problème de la Résiduosité Quadratique comme suit :

$$\mathsf{guess}_{\mathcal{QA}} = \mathcal{B}\left(\begin{array}{c} N \leftarrow \mathsf{RSAGen}(\kappa), \\ z \in_R \mathbb{J}(N)\end{array}\right) = \begin{cases} \text{« } z \notin \mathbb{QR}(N) \text{ » : si } b = b' \\ \text{« } z \in \mathbb{QR}(N) \text{ » : si } b \neq b' \end{cases}$$

Nous avons donc explicité un algorithme $\mathcal{B}$ qui, pris comme entrée (N,z), challenge pour le Problème de la Résiduosité Quadratique, donne en sortie une réponse ($\mathsf{guess}_{\mathcal{QA}}$) au même problème.

Calculons l'avantage de cet algorithme $\mathcal{B}$ à donner la bonne réponse au challenge :

$$\epsilon_{\mathsf{QRA}}(\kappa) := \left|\Pr\left[\mathcal{B}\left(\begin{array}{c} N \leftarrow \mathsf{RSAGen}(\kappa), \\ z \in_R \mathbb{J}(N)\end{array}\right) = \text{« } z \notin \mathbb{QR}(N) \text{ »} \;\middle|\; z \in \mathbb{QR}(N)\right] - \right.$$
$$\left. \Pr\left[\mathcal{B}\left(\begin{array}{c} N \leftarrow \mathsf{RSAGen}(\kappa), \\ z \in_R \mathbb{J}(N)\end{array}\right) = \text{« } z \notin \mathbb{QR}(N) \text{ »} \;\middle|\; z \notin \mathbb{QR}(N)\right]\right|.$$

Remarquons que notre simulation ne réduit pas l'avantage de l'algorithme $\mathcal{A}$ à donner la réponse correcte, il s'agit d'un transfert de réponse et il n'y a pas de dégradation ;

l'inégalité peut être réécrite directement pour $\mathcal{A}$ comme il suit :

$$\epsilon_{\mathsf{QRA}}(\kappa) = \left| \Pr\left[b = b' \;\middle|\; \begin{array}{c} (N, y, 2) \leftarrow \mathsf{KeyGen}(1^\kappa), (m_0, m_1, s) \leftarrow \mathcal{A}_1((N, y, 2)), \\ b \in_R \{0,1\}, c_* = \mathsf{Enc}((N, y, 2), m_b), \\ b' \leftarrow \mathcal{A}_2(c_*, s), z \in \mathbb{QR}(N) \end{array} \right] - \right.$$
$$\left. \Pr\left[b = b' \;\middle|\; \begin{array}{c} (N, y, 2) \leftarrow \mathsf{KeyGen}(1^\kappa), (m_0, m_1, s) \leftarrow \mathcal{A}_1((N, y, 2)), \\ b \in_R \{0,1\}, c_* = \mathsf{Enc}((N, y, 2), m_b), \\ b' \leftarrow \mathcal{A}_2(c_*, s), z \notin \mathbb{QR}(N) \end{array} \right] \right|.$$

Par définition, le deuxième terme de la soustraction est la probabilité de l'algorithme $\mathcal{A}$ dans le cassage de la sécurité sémantique. Nous savons que par hypothèse (voir 1) cet avantage vaut $\delta(\kappa)$.

Passons à présent à l'analyse du premier terme :

$$\Pr\left[b = b' \;\middle|\; \begin{array}{c} (N, y, 2) \leftarrow \mathsf{KeyGen}(1^\kappa), (m_0, m_1) \leftarrow \mathcal{A}_1((N, y, 2)), \\ b \in_R \{0,1\}, c_* = \mathsf{Enc}((N, y, 2), m_b), \\ b' \leftarrow \mathcal{A}_2(c_*), z \in \mathbb{QR}(N) \end{array} \right].$$

Ceci représente le cas où l'élément z du challenge (N, z) est un résidu quadratique, le fait implique que l'instance que nous avons simulée n'est pas une vraie instance.

Comme nous avons montré dans le Lemme 4.3.3 et 4.3.6 les distributions des chiffrés sont indistinguables sous l'hypothèse de la Résiduosité Quadratique. Ceci va montrer qu'un adversaire n'a qu'un avantage négligeable dans le jeu de la sécurité sémantique.

D'après le Lemme 4.3.6 nous savons que cette probabilité est majorée par $\frac{1}{2} + \epsilon_{QRA}(\kappa)$.

En substituant les valeurs calculées nous obtenons que :

$$\left| \frac{1}{2} + \epsilon_{QRA}(\kappa) - \delta(\kappa) \right| = \epsilon_{QRA}(\kappa)$$

équivalent à dire que :

$$\delta(\kappa) = \frac{1}{2} + 2\epsilon_{QRA}(\kappa).$$

De la valeur de la probabilité on peut rapidement calculer l'avantage d'un tel adversaire comme :

$$\begin{aligned} \mathsf{Adv} &= \left| 2 \times \Pr\left[b = b' \;\middle|\; \begin{array}{c} (N, y, 2) \leftarrow \mathsf{KeyGen}(1^\kappa), \\ (m_0, m_1) \leftarrow \mathcal{A}_1((N, y, 2)), \\ b \in_R \{0,1\}, c_* = \mathsf{Enc}((N, y, 2), m_b), \\ b' \leftarrow \mathcal{A}_2(c_*) \end{array} \right] - 1 \right| = \\ &= \left| 2 \times \left(\frac{1}{2} + 2\epsilon_{QRA}(\kappa) \right) - 1 \right| = \\ &= 4\epsilon_{QRA}(\kappa) \end{aligned}$$

qui est une fonction négligeable. □

Remarque 4.3.7. — Nous avons montré la sécurité du schéma de Goldwasser-Micali généralisé pour le choix du paramètre $k = 2$. La preuve peut être étendue à un cas plus général avec un $k > 2$ au choix. Le schéma de la preuve est le suivant : d'abord il faut montrer que, sous l'hypothèse de la Résiduosité Quadratique, les distributions des chiffrements des messages de la forme $m_i = 2^i \pmod{2^k}$ sont indistinguables de la distribution du chiffrement du message $m_0 = 0$. Pour faire cela il faut procéder de façon analogue au cas détaillé dans ce chapitre à partir du message $m = 2^{k-1}$ à descendre. Puis, montrer que si le chiffrement du message $m_0 = 0$ est indistinguable des chiffrements des messages

respectivement $m = 1, 2, \ldots, j$, alors le chiffrement du message $m = j + 1$ est indistinguable aussi du chiffré de $m_0 = 0$. Ceci donne que les distributions des chiffrements sont indistinguables entre elles.

Pour la preuve nous avons supposé que l'adversaire puisse calculer les symboles d'ordre 2^k-ième *modulo* N. Nous remarquons que la définition du Symbole 2^k-ième est dépendante du choix de ζ une racine *primitive* 2^k-ième de l'unité. En particulier si l'élément y de la clé publique n'est pas un carré, il existe un choix de racine primitive tel que son symbole 2^k-ième *modulo* N soit égal à $+1$. Pour le calcul explicite et efficace du Symbole 2^k-ième il est nécessaire d'expliciter les lois de réciprocité et les lois complémentaires. Pour des valeurs petites de k ($k = 1, 2$) il existe des formules explicites (Section 5 de [**DF03**]). Pour les autres valeurs le problème est ouvert. Une exposition générale et complète est disponible dans [**Lem00**]. Pour le calcul du Symbole 2^k-ième *modulo* N ($N = pq$) il est demandé aussi de connaître l'élément $\nu = \pi\rho$ où π, ρ sont des éléments premiers de $\mathbb{Z}[\zeta]$ de norme respectivement p et q.

4.4. Remarques sur l'implémentation

Dans cette section nous aborderons quelques détails d'implémentation. Nous donnerons d'abord une méthode de déchiffrement plus efficace de celle présentée dans la description du schéma même qui prévoyait une recherche exhaustive dans l'espace des messages. Par la suite, nous nous attarderons sur le choix du paramètre k qui donne la mesure du gain par rapport au schéma de Goldwasser-Micali original.

4.4.1. Déchiffrement rapide. — Comme nous l'avons défini dans la description du schéma de Goldwasser-Micali généralisé l'extension immédiate de la fonction $\mathsf{Decrypt}(\cdot)$ prévoit une recherche exhaustive dans l'espace entier des messages en clair $\mathcal{M} = \left\{0, \ldots, 2^k - 1\right\}$. Ceci est à l'origine d'un problème de performances. D'un coté nous voudrions avoir un k « le plus grand possible » pour pouvoir comprimer le plus possible un message dans le même élément du chiffré. De l'autre une recherche exhaustive qui requiert, en moyenne, 2^{k-1} essais pour déchiffrer les messages et devient rapidement impraticable à l'augmentation du paramètre k. Cette méthode de « devine et vérifie » (ou *guess and check* en langue anglaise) est aussi la limite de plusieurs schémas de chiffrement proposés sur des problèmes similaires. Pour notre schéma il existe une méthode efficace pour déchiffrer le message et retrouver le clair. Avec cet algorithme le coût calculatoire croît *linéairement* plutôt que exponentiellement comme la recherche exhaustive et reste donc utilisable pour des plus grandes valeurs de k. Cette solution est basée sur l'algorithme de Tonelli et Shanks pour calculer les racines carrées *modulo* un entier premier p. Plus de détails sur cet algorithme peuvent être trouvés dans [**MvOV96**] et [**Coh93**] En résumant il s'agit de calculer la racine carrée *modulo* p du chiffré c. Soit $p = 2^k s + 1$ un entier premier avec s impair. L'algorithme cité donne en sortie une racine de la forme $x = c^{(s+1)/2} z^{t/2}$ où z est un élément d'ordre 2^k. L'algorithme original est modifié pour donner en sortie la valeur de t qui correspond au message en clair que nous cherchons à récupérer.

Soit $m \in \mathcal{M}$ un message en clair. Considérons son expansion binaire : $m = \sum_{i=0}^{k-1} m_i 2^i$ où $m_i \in \{0, 1\}$. Soit maintenant $c = y^m x^{2^k} \mod N$ son chiffré, avec x un entier non nul,

alors

$$\begin{aligned}\left(\frac{c}{p}\right)_{2^i} &= \left(\frac{y^m x^{2^k}}{p}\right)_{2^i} = \left(\frac{y^{\sum_{j=0}^{k-1} m_j 2^j}}{p}\right)_{2^i} = \left(\frac{y^{\sum_{j=0}^{i-1} m_j 2^j}}{p}\right)_{2^i} \cdot \left(\frac{y^{\sum_{j=i}^{k-1} m_j 2^j}}{p}\right)_{2^i} = \\ &= \left(\frac{y^{\sum_{j=0}^{i-1} m_j 2^j}}{p}\right)_{2^i} \cdot \left(\frac{y^{\sum_{j=0}^{k-i-1} m_{j+i} 2^j}}{p}\right)_{2^i}^{2^i} = \left(\frac{y^{\sum_{j=0}^{i-1} m_j 2^j}}{p}\right)_{2^i} \cdot 1 = \\ &= \left(\frac{y^{\sum_{j=0}^{i-1} m_j 2^j}}{p}\right)_{2^i} = \\ &= \left(\frac{y}{p}\right)_{2^i}^{\sum_{j=0}^{i-1} m_j 2^j} \pmod p.\end{aligned}$$

Nous pouvons donc exploiter cette égalité pour récupérer les bits m_i du message un à la fois, à partir de celui de plus petit poids. Réécrivons l'algorithme $\mathsf{Decrypt}(\cdot,\cdot)$:

$\mathsf{Decrypt}(\mathsf{sk}, c)$ **:** L'algorithme de déchiffrement $\mathsf{Decrypt}$ pour récupérer le message en clair m à partir du chiffré c procède comme il suit :

- il pose $m = 0$,
- pour $i \in \{1, \ldots, k\}$
  1. il calcule $z = \left(\frac{c}{p}\right)_{2^i} \pmod p$ et $t = \left(\frac{c}{p}\right)_{2^i}^m \pmod p$
  2. si $z \neq t$ alors $m \leftarrow m + 2^{i-1}$
- il donne en sortie m.

Remarque 4.4.1. — L'algorithme qui a été présenté ci-dessus calcule le message en clair, interprété comme un entier, un bit à la fois en partant de la droite. Ceci requiert le calcul les symboles 2^i-ièmes dans l'ordre croissant du paramètre i ; et donc cela requiert une exponentiation pour chaque symbole à calculer, pour un total de k exponentiations modulaires *modulo* p.

Une méthode qui permettrait d'être plus rapide consisterait en le calcul du message en clair à partir de la gauche, du bit du plus grand poids, car, donné le symbole i-ième, le calcul du symbole $i-1$-ième requiert seulement le calcul d'un carré. Malheureusement nos recherches dans cette direction n'ont pas donné de résultats utiles.

4.4.2. Choix des paramètres. — Dans la description du schéma nous avons requis, pour la génération des clés, deux nombres premiers $p, q \equiv 1 \pmod{2^k}$ pour un entier k et d'un élément y non nul, qui ne soit pas un résidu quadratique *modulo* N, le produit de p et q. Cette dernière condition sur la non-résiduosité équivaut à requérir que $\left(\frac{y}{p}\right) = \left(\frac{y}{q}\right) = -1$. Il faut donc générer un élément $y \in (\mathbb{Z}/N\mathbb{Z})^*$ tel que *(i)* $y \mod p$ soit primitif dans $(\mathbb{Z}/p\mathbb{Z})^*$ et *(ii)* $y \mod q$ soit primitif dans $(\mathbb{Z}/q\mathbb{Z})^*$.

Si la factorisation de $\varphi(p) = p - 1$ est connue, trouver un tel élément n'est pas difficile, il suffira de prendre un y qui soit premier avec $\varphi(p)$ (ou resp. avec $\varphi(q)$).

Pour cette raison nous proposons, pour la création des paramètres de la clé publique de générer un entier p qui soit *k-quasi-premier*, c'est à dire de la forme $p = 2^k p' + 1$ pour un entier premier (impair) p'. Pareillement nous proposons de générer l'entier q comme un *k-quasi-premier* : $q = 2^k q' + 1$ pour un q' entier premier (impair). Pour faire face aux attaques classiques, propres aux schémas basés sur des problèmes inhérents à la factorisation, comme dans les schémas basés sur RSA, pour la génération des clés, il est

requis d'utiliser des *safe primes* ; ce contexte requiert de générer des *k-quasi-safe primes*. Un algorithme efficace pour un telle génération des entiers *k-quasi-safe primes* peut être trouvé dans [**JP06**].

Passons à présent à l'analyse de la génération de l'élément y de la clé publique. Comme nous avons vu dans la description du schéma, nous avons besoin d'un élément non nul y de $(\mathbb{Z}/N\mathbb{Z})^*$ tel que $\left(\frac{y}{p}\right) = -1 = \left(\frac{y}{q}\right)$.

Soit ζ une racine primitive 2^k-ième de l'unité. Notons d'abord que si $p \equiv 1 \equiv q \pmod{2^k}$ il vient que :

$$\left(\frac{-1}{p}\right) = (-1)^{(p-1)/2} = 1$$

car $\frac{p-1}{2}$ est pair. De manière analogue nous notons que $\left(\frac{-1}{q}\right) = (-1)^{(q-1)/2} = 1$. Nous obtenons que -1 est un résidu quadratique in $\mathbb{Z}/N\mathbb{Z}$ et en conséquence $i = \sqrt{-1} \in (\mathbb{Z}/N\mathbb{Z})$. Cela permet de ne travailler qu'avec des entiers *modulo* N pour la génération des clés et des chiffrés et en conséquence de maintenir la rapidité des algorithmes comme pour le schéma de Goldwasser-Micali *standard*.

Si p est un *k-quasi-safe prime* il vient que :

$$\left(\frac{\zeta y}{p}\right) = \left(\frac{\zeta}{p}\right)\left(\frac{y}{p}\right) = \zeta^{(p-1)/2}\left(\frac{y}{p}\right) = -1 \cdot \left(\frac{y}{p}\right) \pmod{p}$$

car le symbole de ζ est égal à -1. En effet :

$$\left(\frac{\zeta}{p}\right) = \zeta^{\frac{p-1}{2}} = (\zeta^{\frac{p-1}{2^k}})^{2^{k-1}} = -1.$$

Puisque étant $\frac{p-1}{2^k}$ impair nous avons que $\zeta^{\frac{p-1}{2^k}}$ est une racine primitive.

De manière analogue nous pouvons faire le même raisonnement pour q.

D'après cette observation il est facile de construire un élément $y \in \mathbb{J}(N) \setminus \mathbb{QR}(N)$ comme nous le souhaitons. Le détail des étapes nécessaires est le suivant :

1. choisir aléatoirement deux éléments : $y_p \in (\mathbb{Z}/p\mathbb{Z})^*$ et $y_q \in (\mathbb{Z}/q\mathbb{Z})^*$;
2. si $\left(\frac{y_p}{p}\right) = 1$ alors poser $y_p \leftarrow \zeta y_p \pmod{p}$;
3. si $\left(\frac{y_q}{q}\right) = 1$ alors poser $y_q \leftarrow \zeta y_q \pmod{q}$;
4. donner en sortie $y = y_p + p(p^{-1}(y_q - y_p) \mod q)$

En discutant sur les paramètres et la génération des clés nous devons aussi tenir compte des attaques connues pour la factorisation d'un entier composé du type RSA. En particulier, si d'un coté le fait que $p \equiv q \equiv 1 \pmod{2^k}$ nous permet de comprimer une partie de la clé publique car les derniers bits du *modulo* N sont connus, de l'autre cela implique qu'une partie de la clé secrète soit connue, aussi. Cette forme particulière des nombres premiers expose aussi le schéma aux attaques présentées par Coppersmith in [**Cop96, Cop97**] qui utilisant un procédé basé sur l'algorithme LLL peut donner la factorisation de N si au moins la moitié « droite » d'un des ses facteurs est connue. Pour l'éviter, il suffira d'imposer un paramètre k inférieur à $\min(\log p, \log q)/2$. Pour des paramètres de taille courante ceci signifie que N est de 1024 bits et pour simplifier disons p, q de taille autour de 512 bits (dans la réalité les deux entiers premiers ne peuvent pas être trop « proches » sinon d'autres attaques sont connues), pour éviter l'attaque publiée par Coppersmith il sera nécessaire de maintenir la taille de k plus petite que 256 bits environ.

4.4.3. Propriétés du schéma. — Une propriété intéressante du schéma original et qui est conservée dans notre généralisation est le *chiffrement homomorphique*, c'est à dire que les opérations arithmétiques faites sur les chiffrés correspondent à des opérations faites sur les messages en clair.

En particulier, soient c_1 et c_2 deux chiffrés respectivement des messages m_0 et m_1, alors $c_1 = y^{m_1} x_1^{2^k} \pmod N$ et $c_2 = y^{m_2} x_2^{2^k} \pmod N$.

Le chiffré $c = c_1 \cdot c_2$ correspond au chiffrement de la somme *modulo* N des messages en clair :

$$c = c_1 \cdot c_2 = y^{m_1} x_1^{2^k} \cdot y^{m_2} x_2^{2^k} \equiv y^{m_0+m_1} (x_1 x_2)^{2^k} \equiv y^{m'} x'^{2^k} \mod N.$$

Cette propriété est considérée, d'ordinaire, comme une faiblesse ; en effet elle expose le schéma aux attaques par chiffré choisi. Cependant, cette malléabilité des chiffrés peut être parfois souhaitée et exploitée pour des autres fins.

Par exemple, dans le contexte du vote électronique, cette propriété permet d'agréger toutes les voix chiffrées et de déchiffrer le chiffré qui représente la somme des voix, pour garantir un vote secret et anonyme.

4.4.4. Expansion du chiffré. — Comme nous l'avons ébauché avant de décrire le schéma, notre généralisation permet de pouvoir chiffrer des messages en clair plus longs sans une expansion pénalisante du chiffré. De la description il résulte que nous pouvons chiffrer un message en clair de k bits dans un entier N. Au vu des attaques proposées par Coppersmith dans un contexte similaire, nous voyons qu'une limite à ne pas dépasser reste $\min(\log p, \log q)/2$. Dans des cas réels ceci nous permet de chiffrer dans un seul entier de taille « standard » (pour les schémas du type RSA, 1024 bits) des messages assez grands. Cependant, dans une application « typique » le message à chiffrer reste *relativement* court, étant normalement une clé symétrique (comme une clé pour 128-AES) dans une construction KEM-DEM.

Le schéma que nous avons présenté est une généralisation naturelle du schéma de Goldwasser-Micali. On veut comparer les performances avec celles d'autres schémas qui se proposent pour améliorer le schéma d'origine. Nous considérons un cas réel de chiffrement hybride (le paradigme KEM-DEM) où le chiffrement asymétrique est utilisé pour chiffrer une clé aléatoire d'un schéma symétrique. Nous reportons dans un tableau les valeurs d'expansion du chiffré d'un message en clair de k *bits* pour les différentes généralisations du schéma de Goldwasser-Micali. Le « message » est considéré être *court* (128 ou 256 *bits*) car il s'agit d'une clé symétrique (à utiliser avec, par exemple, 128- ou 256-*bit* AES).

TABLE 1. Expansion du chiffré lors d'une utilisation tipyque

Schéma de chiffrement	**Hypothèse**	**Taille du chiffré**
Goldwasser-Micali [**GM84**]	Résiduosité Quadratique (QR)	$k \cdot \log_2 N$
Benaloh-Fisher [**CF85**]	Résiduosité Première (PR)	$\lceil \frac{k}{\log_2 r} \rceil \cdot \log_2 N$
Naccache-Stern [**NS98**]	Résiduosité Première (PR)	$\log_2 N$
Notre schéma	Résiduosité Quadratique (QR)	$\log_2 N$

L'expansion du schéma de Goldwasser-Micali est la majeure, mais le schéma est basé sur une hypothèse considérée comme standard. Les performances du schéma de Benaloh-Fisher sont comparables à celles du schéma dû à Naccache-Stern pour des messages courts ($k \leq \log_2 r$). D'ailleurs pour des autres valeurs plus grandes du paramètre k, le schéma

de Naccache et Stern est à préférer aussi pour l'opération de déchiffrement qui est plus rapide. Si on considère notre schéma et celui par Naccache et Stern on remarquera que l'expansion du chiffré est la même : un chiffré typique a la taille d'un entier composite, soit $\log N$ bits. Le déchiffrement de notre schéma est plus rapide et, comme pour le schéma de Goldwasser-Micali, l'hypothèse calculatoire sous-jacente est le problème *standard* de la Résiduosité Quadratique.

ANNEXE A

MOT-DE-PASSE GRAPHIQUE BASÉ SUR L'ACTION : « CLIQUE UN SECRET »

Ce chapitre est constitué de l'article « Action-Based Graphical Password: "Click-a-Secret" » publié avec Marc Éluard et Yves Maetz dans les *Proceedings of the 29th International Conference on Consumer Electronics (2011)* et présenté à la conférence en Janvier 2011. La référence complète est disponible : [**ÉMA11**].

Ce travail est à l'origine d'un brevet (**EP 11160566.3**) pour une méthode d'authentification basée sur des mots de passe graphiques. Pendant les discussions qui ont abouti à ce travail nous avons élaboré une autre idée pour donner à un système la capacité de s'apercevoir d'être sous une attaque à force brute et déclencher, si ceci est prévu, une réaction opportune. Ce dernier travail, aussi, a été breveté (**PCT/EP11/057345**).

A.1. Abstract

This paper relates to a novel graphical system named "Click-a-Secret" that allows entering a secret through interactions with an image. Seamless modifications of the image will create a new image and derive corresponding secret. In this article, we will focus on access control related applications. However, other usages are possible in various cryptographic systems (e.g.: secret key to encrypt data, etc.). It is well known that humans use long-term memory for storing pictures therefore leading to better recall of images compared to text. This leads to improved memorization of graphical passwords.

A.2. Introduction

Knowledge-based authentication is a method of authentication used to prove the identity of someone. It requires the knowledge of a secret. The most commonly used technique is textual password. Graphical passwords are an alternative in knowledge-based authentication and have been studied in the last decade as potential replacement for textual password. Images replace text and it is well known that images are easier to remember than text [**She67**].

Graphical passwords can be classified in three main categories [**BCO09**]:

- Recall-based systems (drawmetric systems) in which users must recall and reproduce a secret by drawing it,
- Recognition-based systems (cognometric systems) where users have to remember and retrieve a set of images,
- Cued-recall systems (locimetric systems) where users point and click on specific locations of the image.

"Click-a-Secret" is a combination of locimetric and cognometric systems where users have to remember and retrieve a set of images and recreate a personal image by point and click interaction. Entering a password corresponds to recreating this image.

A.3. Principle

A.3.1. Description. — The user creates a personal image by interacting with an image initially displayed as illustrated in Figure 1. This image contains some particular regions where interactions are possible, hereafter named Gecu (Graphical Element Chosen by User).

FIGURE 1. Initial and personal images

A Gecu corresponds to a specific graphical element present in the initial image (for example, people, animals, vehicles, signs, architectural elements and so on). Interactions with a Gecu will replace it by an alternate version. The set of Gecu and alternate versions is pre-defined. All alternate versions must be integrated seamlessly in the initial image. The goal is to obtain a personal image that is coherent with the initial image.

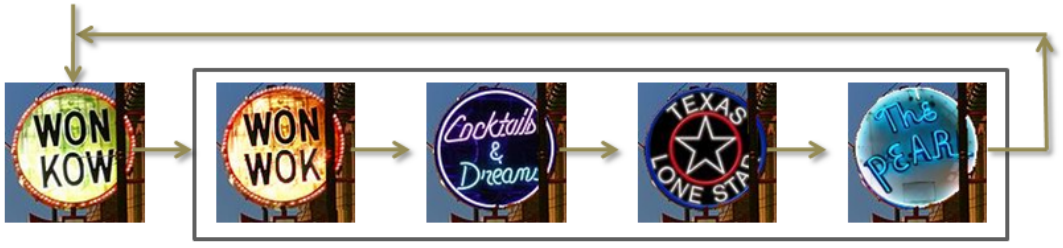

FIGURE 2. Gecu and its alternate versions

The interactions allow building the secret, either using the complete list of interactions or more simply by using only the final image. The construction and usage of the secret depends on the application. Section § A.4.1 presents different examples.

A.3.2. Life cycle of the secret. — Like traditional knowledge-based authentication, the life cycle is composed of three phases: enrolment, usage, and renewal.

A.3.2.1. Enrolemnt. — The enrolment starts after the identification of a new user (with no password) or password renewal requested by the user. Enrolment consists of the following steps:

– Choice: the user interacts to choose a new password,
– Reminder: the new password is displayed to the user,
– Practice: the user practices the password entry,
– Error: the user made an error during its practice.

Interactions between the phases are done according to the diagram of Figure 3.

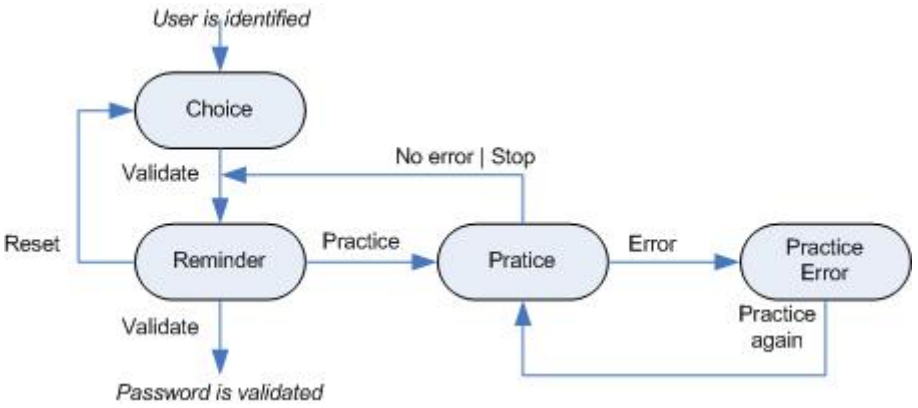

FIGURE 3. Enrollment screens

For the choice screen, an initial picture is displayed. Some visual cues allow the user to identify the Gecus. For example, each Gecu may be surrounded by a thin orange rectangle and the active Gecu is highlighted. If the user clicks on it, this Gecu will be replaced by an alternate version. When he is satisfied with the resulting picture, the user validates his personal image. This leads to the reminder screen. In the reminder screen, the personal image is displayed with the modified Gecus visually highlighted. This screen allows the user to memorize his personal image. If the picture does not suit him, he can reset it to choose another one. If he is confident to remember it later, then he validates directly. If he still has some doubts, he may enter the practice screen. The practice screen permits to the user to train the entry of the personal picture. The initial picture is shown to the user. He interacts with Gecus to obtain his personal image. When he believes the personal image is complete, he asks to validate. If the user retrieved perfectly his personal image or if the user asks to stop the practice, the reminder screen is displayed again. In the other case, the practice error screen highlights the mistakes. In the practice error screen, good alternate versions are then indicated by green rectangles and errors or missing alternate versions are shown in red rectangles. At this stage the user has to practice again and so returns to the practice screen with the initial image.

A.3.2.2. Usage. — The initial picture is shown to the user. He interacts with Gecus to obtain his personal image. The secret is generated after validation by the user.

A.3.2.3. Renewal. — Once the user is authenticated, he can renew his password through a new enrolment process.

A.3.3. User Interface considerations. —

A.3.3.1. Image choice. — The choice of the predefined initial picture is essential. In order to provide a system with a large number of different final images, the initial picture should be rich of characteristic elements for which we can easily define Gecus and coherent alternate Gecus. Furthermore, the personal image created by the user should be seamlessly different from initial image to minimize shoulder surfing.

A.3.3.2. Alternate version presentation. — The choice of alternate versions of Gecus by the user may depend on their presentation and the interaction process. In a simple implementation, the interaction can be resumed on a single click on a Gecu. Each click would present the next alternate version. In such implementation, the order of alternate versions must be random to prevent using too frequently the first elements. In another implementation, the different alternate versions of a Gecu are displayed through a pie or ribbon menu when clicking on it. The ribbon menu allows to access easily many elements while pie menu can present efficiently only a small number of elements.

A.4. Results

A.4.1. Use cases. — The access to personal content or resources is generally protected by a textual password. Entering alphanumeric characters is often not user friendly, particularly on devices without keyboard or with restricted keyboards like tablets, smartphones, connected TVs, etc. In the case of touchscreen devices, "Click-a-Secret" could be used to replace for example the 4-digit pin code generally used to unlock the access to personal content or resources. Graphical resources may be provided either by the device manufacturer or the service operator. "Click-a-Secret" could also be implemented in online services to allow personalization of access using graphical resources familiar to the service. For example, in the domain of online games, players could create its personal image using graphical elements from the game itself.

A.4.2. Secret Space. — To compute the theoretical size of secret space we schematize the system as follow: given an initial image, it is composed of g Gecus and each one has v alternate versions, excluding the initial picture. To simplify notations and formulas we consider that the same number of versions is available per Gecu. To enter a secret, user modifies a number i of Gecu of his choice. We take into account the order of modification of Gecus. The number of potential personal images (secrets) is given by the formula:

$$\#\{\text{Secrets}\} = \sum_{i=0}^{g} A_g^i \cdot \nu^i = \sum_{i=0}^{g} \frac{g!}{(g-i)!}\nu^i.$$

Table 1 presents a comparison of the secret space for the use cases introduced in Section § A.4.1 between the textual version and "Click-a-Secret". It shows that for pin-codes, we achieve 50 % improvement of secret space while reducing the length of the secret. For traditional user login, we obtain similar secret space while keeping reasonable settings and similar secret length.

	Pin Code		User Login	
	Textual	C-a-S	Textual	C-a-S
# Gecu	4	6	8	16
# modified Gecu	4	3	8	8
# variations per Gecu	10	5	62	6
space of secrets	10000	15000	2.18×10^{14}	8.72×10^{14}

TABLE 1. Comparison of secret space

A.4.3. Conclusion. — The advantages of "Click-a-Secret" are multiple. In addition to the use of long term memory for a better recall, it introduces a new practice phase that should improve the memorization by a training phase. It is well adapted to devices with no keyboard. By choosing the appropriate predefined images, "Click-a-Secret" can be easily integrated in applications that already use graphical environments. Finally, graphical environments are much more fun than textual environments. We will implement a proof of concept in order to perform user evaluations. The analysis of the results will give some hints to refine our approach particularly regarding the usability issues and obtain a practical graphical authentication system.

CHAPITRE B

CONTRIBUTIONS

Brevets d'inventions :

EP 11160566.3 : Ce brevet traite d'une méthode pour générer une valeur secrète (un mot-de-passe, par exemple) à partir d'une image. Demande de brevet européenne.

PCT/EP11/057345 : Ce brevet traite d'une méthode qui permet à un système d'authentification de s'apercevoir s'il une attaque à force brute a lieu et de déclencher les actions prévues. Demande de brevet internationale et européenne.

PCT/EP11/066883 : Ce brevet décrit la méthode de généralisation du chiffrement de Goldwasser-Micali donnée dans ce manuscrit. Demande de brevet internationale et européenne.

(sans numéro) : Un quatrième brevet qui est en cours de finalisation avant la phase de dépôt et qui traite la sécurité de *média* physiques.

BIBLIOGRAPHIE

[AJ09] Davide Alessio and Marc Joye, *A simple construction for public-key encryption with revocable anonymity : the honest-sender case*, Digital Rights Management Workshop (Ehab Al-Shaer, Hongxia Jin, and Gregory L. Heileman, eds.), ACM, 2009, pp. 11–16.

[Ale11] Davide Alessio, *A privacy-preserving broadcast encryption scheme with revocability*, pp. 318–322, IEEE Comput. Soc, 2011.

[BBDP01] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval, *Key-privacy in public-key encryption*, ASIACRYPT (Colin Boyd, ed.), Lecture Notes in Computer Science, vol. 2248, Springer, 2001, pp. 566–582.

[BBW06] Adam Barth, Dan Boneh, and Brent Waters, *Privacy in encrypted content distribution using private broadcast encryption*, Financial Cryptography (Giovanni Di Crescenzo and Aviel D. Rubin, eds.), Lecture Notes in Computer Science, vol. 4107, Springer, 2006, pp. 52–64.

[BCO09] Robert Biddle, Sonia Chiasson, and Paul C Van Oorschot, *Graphical passwords : Learning from the first generation*, Technical Report TR0909 School of Computer Science Carleton University (2009), 1–20.

[BCP03] Emmanuel Bresson, Dario Catalano, and David Pointcheval, *A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications*, ASIACRYPT (Chi-Sung Laih, ed.), Lecture Notes in Computer Science, vol. 2894, Springer, 2003, pp. 37–54.

[BDPR98] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway, *Relations among notions of security for public-key encryption schemes*, in Krawczyk [**Kra98**], pp. 26–45.

[Ben87] Josh Daniel Cohen Benaloh, *Verifiable secret-ballot elections*, Ph.D. thesis, Yale University, New Haven, CT, USA, 1987.

[BF01] Dan Boneh and Matthew K. Franklin, *Identity-based encryption from the weil pairing*, CRYPTO (Joe Kilian, ed.), Lecture Notes in Computer Science, vol. 2139, Springer, 2001, pp. 213–229.

[BF03] _____, *Identity-based encryption from the weil pairing*, SIAM J. Comput. **32** (2003), no. 3, 586–615.

[BS99] Mihir Bellare and Amit Sahai, *Non-malleable encryption : Equivalence between two notions, and an indistinguishability-based characterization*, in Wiener [**Wie99**], pp. 519–536.

[CF85] Josh D. Cohen and Michael J. Fischer, *A robust and verifiable cryptographically secure election scheme*, 26th Annual Symposium on Foundations of Computer Science (FOCS '85), IEEE Computer Society, 1985, pp. 372–382.

[Coh93] Henri Cohen, *A course in computational algebraic number theory*, Springer-Verlag New York, Inc., New York, NY, USA, 1993.

[Cop96] Don Coppersmith, *Finding a small root of a bivariate integer equation : Factoring with high bits known*, Advances in Cryptology — EUROCRYPT '96 (U. Maurer, ed.), Lecture Notes in Computer Science, vol. 1070, Springer-Verlag, 1996, pp. 179–189.

[Cop97] ———, *Small solutions to polynomial equations, and low exponent rsa vulnerabilities*, J. Cryptology **10** (1997), no. 4, 233–260.

[CS98] Ronald Cramer and Victor Shoup, *A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack*, in Krawczyk [**Kra98**], pp. 13–25.

[CvH91] David Chaum and Eugène van Heyst, *Group signatures*, EUROCRYPT, 1991, pp. 257–265.

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor, *Non-malleable cryptography (extended abstract)*, STOC, ACM, 1991, pp. 542–552.

[DDN00] ———, *Nonmalleable cryptography*, SIAM J. Comput. **30** (2000), no. 2, 391–437.

[DF03] Ivan Damgård and Gudmund Skovbjerg Frandsen, *Efficient algorithms for gcd and cubic residuosity in the ring of eisenstein integers*, FCT (Andrzej Lingas and Bengt J. Nilsson, eds.), Lecture Notes in Computer Science, vol. 2751, Springer, 2003, pp. 109–117.

[DH76] W. Diffie and M. E. Hellman, *New directions in cryptography*, IEEE Trans. Inform. Theory **IT-22** (1976), 644–654.

[ÉMA11] Marc Éluard, Yves Maetz, and Davide Alessio, *Action-based graphical password : "click-a-secret"*, pp. 265–266, IEEE Comput. Soc, 2011.

[FN94] Amos Fiat and Moni Naor, *Broadcast encryption*, Advances in Cryptology — CRYPTO '94 (Yvo G. Desmedt, ed.), Lecture Notes in Computer Science, vol. 839, Springer-Verlag, 1994, pp. 480–491.

[FO99a] Eiichiro Fujisaki and Tatsuaki Okamoto, *How to enhance the security of public-key encryption at minimum cost*, Public Key Cryptography (Hideki Imai and Yuliang Zheng, eds.), Lecture Notes in Computer Science, vol. 1560, Springer, 1999, pp. 53–68.

[FO99b] ———, *Secure integration of asymmetric and symmetric encryption schemes*, in Wiener [**Wie99**], pp. 537–554.

[Gam84] Taher El Gamal, *A public key cryptosystem and a signature scheme based on discrete logarithms*, CRYPTO, 1984, pp. 10–18.

[GM84] Shafi Goldwasser and Silvio Micali, *Probabilistic encryption*, J. Comput. Syst. Sci. **28** (1984), no. 2, 270–299.

[IP08] Malika Izabachène and David Pointcheval, *New anonymity notions for identity-based encryption*, SCN (Rafail Ostrovsky, Roberto De Prisco, and Ivan Visconti, eds.), Lecture Notes in Computer Science, vol. 5229, Springer, 2008, pp. 375–391.

[JP06] Marc Joye and Pascal Paillier, *Fast generation of prime numbers on portable devices : An update*, Cryptographic Hardware and Embedded Systems – CHES 2006 (L. Goubin and M. Matsui, eds.), Lecture Notes in Computer Science, vol. 4249, Springer-Verlag, 2006, pp. 160–173.

[Kob87] N. Koblitz, *Elliptic curve cryptosytems*, Mathematics of Computation **48** (1987), no. 177, 203–209.

[Kra98] Hugo Krawczyk (ed.), *Advances in cryptology - crypto '98, 18th annual international cryptology conference, santa barbara, california, usa, august 23-27, 1998, proceedings*, Lecture Notes in Computer Science, vol. 1462, Springer, 1998.

[KTY07] Aggelos Kiayias, Yiannis Tsiounis, and Moti Yung, *Group encryption*, ASIACRYPT (Kaoru Kurosawa, ed.), Lecture Notes in Computer Science, vol. 4833, Springer, 2007, pp. 181–199.

[Lem00] F. Lemmermeyer, *Reciprocity laws : from euler to eisenstein*, Springer monographs in mathematics, Springer, 2000.

[Mil86] Victor S. Miller, *Use of elliptic curves in cryptography*, CRYPTO (H. C. Williams, ed.), Lecture Notes in Computer Science, vol. 218, Springer, 1986, Lecture Notes in Computer Science No. 218, pp. 417–426.

[MvOV96] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone, *Handbook of applied cryptography*, CRC Press, Boca Raton, Florida, 1996, URL : `http://cacr.math.uwaterloo.ca/hac`.

[NS98] David Naccache and Jacques Stern, *A new public key cryptosystem based on higher residues*, ACM Conference on Computer and Communications Security 1998 (CCS '98), ACM Press, 1998, pp. 59–66.

[NY90] Moni Naor and Moti Yung, *Public-key cryptosystems provably secure against chosen ciphertext attacks*, STOC, ACM, 1990, pp. 427–437.

[OP01] Tatsuaki Okamoto and David Pointcheval, *React : Rapid enhanced-security asymmetric cryptosystem transform*, CT-RSA (David Naccache, ed.), Lecture Notes in Computer Science, vol. 2020, Springer, 2001, pp. 159–175.

[Pai99] Pascal Paillier, *Public-key cryptosystems based on composite degree residuosity classes*, Advances in Cryptology – EUROCRYPT '99 (J. Stern, ed.), Lecture Notes in Computer Science, vol. 1592, Springer-Verlag, 1999, pp. 223–238.

[RS91] Charles Rackoff and Daniel R. Simon, *Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack*, CRYPTO (Joan Feigenbaum, ed.), Lecture Notes in Computer Science, vol. 576, Springer, 1991, pp. 433–444.

[Sha85] Adi Shamir, *Identity-based cryptosystems and signature schemes*, Advances in Cryptology, Proceedings of CRYPTO '84 (G. R. Blakley and D. C. Chaum, eds.), Lecture Notes in Computer Science, vol. 196, Springer, 1985, pp. 47–53.

[She67] Roger N Shepard, *Recognition memory for words, sentences, and pictures*, Journal Of Verbal Learning And Verbal Behavior **6** (1967), no. 1, 156–163.

[TY98] Yiannis Tsiounis and Moti Yung, *On the security of El Gamal based encryption*, Public Key Cryptography (Hideki Imai and Yuliang Zheng, eds.), Lecture Notes in Computer Science, vol. 1431, Springer, 1998, pp. 117–134.

[Wie99] Michael J. Wiener (ed.), *Advances in cryptology - crypto '99, 19th annual international cryptology conference, santa barbara, california, usa, august 15-19, 1999, proceedings*, Lecture Notes in Computer Science, vol. 1666, Springer, 1999.

RÉSUMÉ

Le chiffrement est sûrement une primitive fondamentale parmi les fonctions cryptographiques. Cela rend possible à deux parties, d'ordinaire appelées Alice et Bob, de communiquer au travers d'un canal non sécurisé en permettant qu'un adversaire, Eve, ne puisse pas comprendre leur communication.

Il pourrait sembler assez facile d'identifier les contraintes nécessaires pour le design d'un « bon » schéma de chiffrement. Il s'avère en réalité que la définition d'un modèle de sécurité rigoureux n'est pas trivial. Ceci dépend fortement du contexte. La sécurité même dépend du contexte.

Ce document s'ouvre avec un chapitre d'introduction à la cryptographie à clé publique. Nous décrirons ensuite comment la sécurité d'une telle primitive est évaluée en définissant de façon rigoureuse un attaquant, en particulier cela signifie fixer son but et les moyens dont il dispose.

La suite de ce document s'organise autour de deux parties. La première partie concerne l'*anonymat révocable*. Nous obtenons et présentons deux schémas pour garantir l'anonymat pour l'envoyeur d'un message chiffré mais avec la possibilité pour un troisième acteur de confiance, si le cas le nécessite, de révéler l'identité de l'envoyeur. Cette primitive a été appliquée dans les contextes du chiffrement à clé publique et du chiffrement *broadcast*.

Dans la deuxième partie, nous nous dédions à l'étude et à l'amélioration d'un schéma de chiffrement à clé publique dont l'original est dû à Goldwasser et Micali. Notre travail généralise leur schéma en fournissant une famille de schémas de chiffrement. Notre travail est motivé par la recherche de l'amélioration de l'efficacité (en termes de bande passante) du schéma original, afin de pouvoir chiffrer des messages plus longs dans un chiffré de la même taille.

ABSTRACT

Encryption is one of the most important cryptographic primitives. It enables two parties, Alice and Bob, to communicate over an insecure channel in such a way that an eavesdropper, Eve, cannot understand what is being exchanged. Although it seems an easy task to identify properties that a "good" encryption scheme must fulfill, it turns out that rigorously defining the right security notion is not trivial at all. Security is context sensitive.

We open this manuscript introducing public-key cryptography. Then we survey the security of such a primitive, defining rigorously an attacker, i.e. stating his goal and means.

The manuscript has two main parts. The first one is about *revocable anonymity*. We present two original schemes preserving the sender's privacy but keeping the ability foe a third trusted part to reveal the sender identity under certain conditions, e.g.: under a court request. This primitive has been applied in the public-key encryption context and in broadcast encryption as well.

In the second part, we describe and improve a public-key encryption scheme due to Goldwasser and Micali. Our work generalises their scheme providing a family of encryption schemes. The original scheme can be recovered for a given choice of the k parameter. Our goal was to improve the bandwidth of the original scheme letting a larger message to be encrypted in a cipher of the same length.