École normale supérieure
Département d'informatique

Équipe CASCADE
INRIA

Université Paris 7
Denis Diderot

L'anonymat dans les protocoles cryptographiques

Thèse

présentée et soutenue publiquement le jeudi 1er octobre 2009

pour l'obtention du

Doctorat de l'université Paris VII - Denis Diderot
(spécialité informatique)

Malika Izabachène

Composition du jury:

Directeur de thèse:	David Pointcheval	(École Normale Supérieure)
Rapporteurs:	Xavier Boyen	(Université de Stanford)
	Jean-Sébastien Coron	(Université de Luxembourg)
Examinateurs:	Hervé Chabanne	(Sagem Sécurité, Télécom Paristech)
	Arnaud Durand	(Université Paris VII - Denis Diderot)
	Louis Goubin	(Université de Versailles)
	David Naccache	(Université Paris II - Panthéon-Assas)
	Damien Vergnaud	(École Normale Supérieure)

Travaux effectués au Laboratoire d'Informatique de l'École Normale Supérieure
45 rue d'Ulm, 75230 Paris Cedex 05

Table des matières

Remerciements

Introduction

L'anonymat est un concept universel à de nombreux égards : notre identité peut rester secrète dans des circonstances sociales, politiques exposant parfois des intérêts personnels ou économiques. Dans certains contexte, l'anonymat peut demander à être percée, par exemple lorsque nous ne connaissons pas l'identité d'un criminel ; parfois, on cherche au contraire à l'obtenir. Aujourd'hui, l'avancée massive des technologies de l'internet renforce énormément l'importance de cette notion : désormais, l'anonymat n'est plus seulement l'affaire des diplomates et des militaires mais celle de tous. La cryptographie offre un grand nombre d'outils renforçant sa démocratisation dans les protocoles de votes électroniques, de transactions en ligne, des correspondances *via* des logiciels, etc. . Dans cette thèse, nous apportons quelques éléments de réponses à ce désir de maintien d'anonymat, en nous concentrant sur les protocoles de chiffrement et d'authentification. En premier lieu, l'introduction de l'anonymat dans ces deux types de protocoles soulève énormément de questions : alors que l'authentification est proche de la notion d'identification, comment garantir l'authentification de la source tout en préservant son anonymat ? Comment envoyer un message chiffré avec une clé publique connue de tous sans révéler la clé publique utilisée pour chiffrer ? Le concept d'anonymat suscite de nombreux paradoxes qui amènent à manier cette notion avec une précaution et une attention toutes particulières : définir le contexte dans lequel elle s'applique et préciser le niveau de sécurité que l'on veut obtenir sont des procédés très classiques dans l'application du concept d'anonymat.

De manière à préciser ces nuances, nous définirons dans une première partie les outils et mécanismes autour desquelles cette notion d'anonymat s'est construite. Nous présenterons alors nos travaux en deux temps : nous rappellerons d'abord en quoi consiste l'anonymat pour le chiffrement, en mettant l'accent sur les notions de sécurité et les éléments constructifs pour les schémas de chiffrement anonymes, et en particulier pour le chiffrement à base d'identités. Ce dernier concept, très commode pour la gestion des certificats nécessite la contribution d'une autorité de confiance, capable de dériver les secrets de tous les utilisateurs. Dans l'article [85], en collaboration avec David Pointcheval, nous avons étendu la notion d'anonymat en considérant cette autorité comme attaquant potentiel. Cette première phase de présentation sera pour nous l'occasion d'introduire cette nouvelle notion de sécurité. Ensuite, nous étudierons le problème de l'anonymat révocable dans un contexte multi-acteurs : en collaboration avec David Pointcheval et Damien Vergnaud, nous avons conçu une nouvelle primitive intégrant de multiples fonctionnalités [86] ; nous décrirons également ce concept dans cette partie.

Dans un deuxième temps, nous nous intéresserons aux protocoles d'échange de clé authentifiés à deux et trois parties *via* un canal non sécurisé (ni confidentiel, ni authentifié), permettant l'authentification d'un membre. Nous montrerons comment garantir l'anonymat d'un participant dans chacun des deux scénarios. Avant de présenter nos résultats, nous introduirons le modèle de sécurité sur lequel nous nous sommes appuyés [17, 14] et

les extensions que nous avons pu y apporter. Avec Michel Abdalla et David Pointcheval, nous avons considéré l'anonymat du client dans le cas à trois parties, en adaptant le protocole de l'article [4]. Nous montrerons comment réaliser un protocole garantissant l'anonymat du client en utilisant un protocole d'interrogation confidentielle de bases de données ($\mathcal{PIR}$) (ou *Private Information Retrieval* en anglais) : il s'agit d'une primitive qui résout une problématique ancienne que nous présenterons dans la première partie. Nous prouverons la sécurité de ce protocole dans le modèle de l'oracle aléatoire. Un deuxième exemple viendra s'ajouter à cette phase d'introduction de l'anonymat dans les protocoles d'authentification : il s'agira d'un protocole d'échanges de clés à deux partie (authentifié) générique, IB-PAKE construit à partir d'une primitive de chiffrement anonyme : par là même, nous proposerons une application de la nouvelle notion d'anonymat introduite précédemment.

Enfin, nous achèverons notre présentation par une conclusion de nos travaux.

Première partie

Primitives cryptographiques pour l'anonymat

Chapitre 1

Primitives de chiffrement

Dans cette partie, nous présentons les primitives propres à la cryptographie à clé publique : les schémas de chiffrement et les schémas de signature. Nous définissons les notions de sécurité caractérisant chacun d'eux : la première classe de schémas assure la confidentialité des données et la seconde garantit l'authentification et l'intégrité des données. Nous insisterons davantage sur les définitions liées au chiffrement, nous ferons juste quelques rappels informels pour les schémas de signature.

Sommaire

1.1 Schémas de chiffrement

Un cryptosystème est défini par la donnée de quatre (ou trois selon le contexte) algorithmes : un premier algorithme génère les paramètres communs du système, un second renvoie des clés publiques et secrètes, un troisième, typiquement probabiliste permet de chiffrer un message (sans utiliser de secret particulier) et un dernier, quant à lui déterministe, retrouve le message à partir d'un chiffré en utilisant la clé secrète.

1.1.1 Définitions et notions de sécurité

Nous notons $\mathcal{M}, \mathcal{C}$ les espaces des messages clairs et des messages chiffrés respectivement et $\mathcal{R}$ un espace de probabilité fixé. On note λ le paramètre de sécurité. Formellement, les algorithmes impliqués dans un schéma de chiffrement sont définis de la manière suivante :

- **Algorithme de génération de paramètres** Setup **:** prend en entrée le paramètre de sécurité λ et renvoie les paramètres publics du système, params ;
- **Algorithme de génération des clés** KeyGen **:** prend en entrée les paramètres publics du système, params et renvoie une paire clé publique/clé privée, pk/sk ;
- **Algorithme de chiffrement** Encrypt **:** est un algorithme probabiliste qui prend en entrée la clé publique pk, un message m dans $\mathcal{M}$ et un aléa r et renvoie un chiffré c associé au message m. Afin d'alléger les notations, on notera le chiffré d'un message m par $c = \mathsf{Encrypt}(\mathsf{pk}, m)$ ou $c = \mathsf{Encrypt}(\mathsf{pk}, m; r)$; si la clé publique est clairement spécifiée par le contexte, on pourra écrire c sans la préciser.
- **Algorithme de déchiffrement** Decrypt **:** prend en entrée un chiffré c et la clé secrète sk de déchiffrement, il renvoie le message $m \in \mathcal{M}$ associé à $c \in \mathcal{C}$ ou un symbole d'erreur.

Remarque 1 *En règle générale, la fonction* Setup *est l'identité mais dans certains systèmes lorsque plusieurs utilisateurs doivent s'entendre sur des paramètres communs, il peut être commode de séparer les algorithmes* Setup *et* KeyGen*. On verra que pour les schémas de chiffrement anonyme, cette condition est nécessaire. Par exemple, pour le cryptosystème ElGamal, le paramètre publique peut être le nombre premier déterminant le groupe cyclique.*

Un schéma de chiffrement est dit consistant si pour tout paramètre public params, toute paire clé publique/clé secrète, pk/sk, tout message $m \in \mathcal{M}$, tout chiffré de m sous la clé pk se déchiffre sous la clé sk sur le message m.

Plus formellement, étant donné un paramètre de sécurité λ, on a :

$$\forall m \in \mathcal{M}, \quad \forall\ \mathsf{params} \leftarrow \mathsf{Setup}(\lambda) \quad \forall\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{params})$$

$$\forall\ c \in \mathcal{C} \quad \exists\ r \in \mathcal{R} \quad \mathsf{Encrypt}(\mathsf{pk}, m; r) = c \Longrightarrow\ \mathsf{Decrypt}(\mathsf{sk}, c) = m$$

Cette condition peut être assouplie de telle sorte à n'être vérifiée que pour une fraction écrasante de paires $(\mathsf{pk}, \mathsf{sk})$ et de chiffrés $c \in \mathcal{C}$; dans ce cas, on parle de *consistance calculatoire*.

Notions de Sécurité
la fonction de chiffrement doit garantir la confidentialité des messages, ce qui se formalise de différentes manières selon le niveau de sécurité que l'on veut garantir : on ne veut pas que l'adversaire puisse apprendre un bit du message. En 1982, Godwasser et Micali ont formalisé un critère de sécurité important pour le chiffrement asymétrique , la sécurité sémantique. L'idée est la suivante : considérons un attaquant $\mathcal{A}$ générant deux messages m_0 et m_1 de tailles égales. Supposons que nous lui renvoyons le chiffré c_b de l'un des deux messages m_0 et m_1, où b est un bit choisi aléatoirement. De plus, afin de renforcer le modèle, on peut très bien lui donner le pouvoir de déchiffrer certains chiffrés (excepté le chiffré challenge). On dit que l'algorithme de chiffrement est sémantiquement sûr résistant aux attaques à chiffrés choisis si l'attaquant ne parvient pas à deviner le bit b avec probabilité significativement plus grande que $\frac{1}{2}$. Plus précisément, nous formalisons la notion de sécurité sémantique par l'expérience suivante :

$\mathsf{Expérience}\ \mathsf{Exp}^{\mathsf{ind\text{-}cca}\text{-}b}_{\mathcal{E},\mathcal{A}}(\lambda)$
$\mathsf{CSet} \stackrel{\text{déf}}{=} \emptyset; \mathsf{params} \leftarrow \mathsf{Setup}(\lambda)\,;$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{params})\,;$
$(m_0, m_1, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathcal{O}\mathsf{Decrypt}()}(\mathtt{FIND}, \mathsf{params}, \mathsf{pk})\,;$
$c_b \stackrel{R}{\leftarrow} \mathsf{Encrypt}(\mathsf{params}, \mathsf{pk}, m_b)\,;$
$b' \leftarrow \mathcal{A}_2^{\mathcal{O}\mathsf{Decrypt}()}(\mathtt{GUESS}, \mathsf{params}, \mathsf{pk}, c_b, \mathsf{state})\,;$
$\mathtt{si}\ \ c_b \notin \mathsf{CSet}\,;$
$\mathtt{retourner}\ b'\ \mathtt{sinon}\ \mathtt{retourner}\ 0$

$\mathcal{O}\mathsf{Decrypt}(c)$
$\quad \mathsf{CSet} \leftarrow \mathsf{CSet} \cup \{c\}\,;$
$\quad m \leftarrow \mathsf{Decrypt}(\mathsf{sk}, c)\,;$
$\quad \mathtt{retourner}\ m$

On définit l'avantage de $\mathcal{A}$ dans l'expérience ci-dessus par :

$$\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathcal{E},\mathcal{A}} = \left|\Pr[\mathsf{Exp}^{\mathsf{ind\text{-}cca\text{-}1}}_{\mathcal{E},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{ind\text{-}cca\text{-}0}}_{\mathcal{E},\mathcal{A}}(\lambda) = 1]\right|$$

On dit qu'un schéma de chiffrement est IND-CCA-sûr résistant aux attaques adaptatives à chiffrés choisis (ou IND-CCA2-sûr), si pour tout attaquant polynomial $\mathcal{A}$, l'avantage de $\mathcal{A}$ est une fonction négligeable en le paramètre de sécurité λ. Lorsqu'on ne peut pas garantir la sécurité en donnant accès à l'oracle de déchiffrement, on parle de schéma de chiffrement IND-CPA-sûr : le schéma est dit sémantiquement sûr résistant aux attaques à clairs choisis.

1.1.2 Schéma de chiffrement hybride

Le chiffrement hybride est un concept assez intuitif qui consiste à décomposer les procédures impliquées dans un schéma de chiffrement en deux mécanismes : un premier mécanisme, appelé *encapsulation de clés ($\mathcal{KEM}$)*, asymétrique chiffre une clé symétrique et un second mécanisme, *data encapsulation mechanism, ($\mathcal{DEM}$)*, chiffre un message avec la clé secrète précédente en utilisant des techniques symétriques.

Définitions d'un schéma $\mathcal{KEM}$

Un mécanisme d'*encapsulation de clés* a la même structure qu'un *schéma de chiffrement asymétrique* excepté que l'algorithme de chiffrement ne prend que la clé publique du destinataire en entrée. Cet algorithme génère une paire constituée d'une clé K et de son encapsulation C. Plus précisément, un schéma $\mathcal{KEM}$ est défini par la donnée de quatre algorithmes polynomiaux $\mathcal{KEM} \stackrel{\text{déf}}{=} \langle \mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps} \rangle$, chacun défini de la manière suivante :

- **Algorithme de génération de paramètres** Setup **:** prend en entrée le paramètre de sécurité λ et renvoie les paramètres publics du schéma params.
- **Algorithme de génération des clés** KeyGen **:** prend en entrée les paramètres publics du schéma, params et renvoie une paire clé publique/clé privée, pk/sk.
- **Algorithme d'*encapsulation* de clés** Encaps **:** cet algorithme probabiliste prend en entrée la clé publique du destinataire pk et un aléa r ; il génère une clé aléatoire K dans S_K, l'ensemble des clés, et son *encapsulation* C.
- **Algorithme de *décapsulation* de clés** Decaps **:** cet algorithme déterministe prend en entrée la clé secrète du destinataire sk et l'encapsulation C d'une clé ; il retrouve la clé K telle qu'il existe un aléa $r \in \mathcal{R}$ tel que $(K, C) = \mathsf{Encaps}(\mathsf{pk}; r)$; si cette clé n'est pas définie ou si $K \notin S_K$, cet algorithme renvoie un message d'erreur.

Ce mécanisme doit également spécifier un entier positif, que l'on notera ℓ précisant la longueur de la clé renvoyée par l'algorithme Encaps.

La primitive $\mathcal{KEM}$ doit vérifier une propriété de *consistance* qui se déduit directement de celle énoncée précédemment pour le chiffrement au paragraphe 1.1.1.

Notions de sécurité pour les schémas $\mathcal{KEM}$

Nous définissons la sécurité pour le mécanisme d'*encapsulation de clés* sous les attaques à chiffrés choisis de la manière suivante : dans une première phase, le challenger génère des paramètres publics params et une paire de clé publique/clé secrète, pk/sk qu'il renvoie à l'attaquant $\mathcal{A}$. Ce dernier a accès à un oracle de décapsulation puis renvoie un bit b. Le challenger calcule un chiffré C_1^* et une clé K_1^*, générés par l'algorithme Encaps. Il choisit également une clé aléatoire K_0^* dans l'ensemble S_K, où S_K est l'ensemble des clés muni d'une distribution uniforme[1]. Ensuite, le challenger choisit un bit b et renvoie (K_b^*, C_1^*) à l'attaquant $\mathcal{A}$. Dans une seconde phase, l'attaquant a accès à un oracle de décapsulation. Enfin, il renvoie sa réponse pour le bit b.

Plus formellement, nous définissons l'expérience associée à un attaquant $\mathcal{A}$ et un schéma $\mathcal{KEM} \stackrel{\text{déf}}{=} \langle \mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps} \rangle$ de la manière suivante :

$\mathsf{Expérience}\ \mathsf{Exp}^{\mathsf{cca}-b}_{\mathcal{KEM},\mathcal{A}}(\lambda)$
$\mathsf{CSet} \stackrel{\text{déf}}{=} \{\emptyset\}$; $\mathsf{params} \leftarrow \mathsf{Setup}(\lambda)$;
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{params})$;
$\mathsf{state} \leftarrow \mathcal{A}_1^{\mathcal{O}\mathsf{Decaps}()}(\texttt{FIND}, \mathsf{params}, \mathsf{pk})$;
$(C_1^*, K_1^*) \stackrel{R}{\leftarrow} \mathsf{Encaps}(\mathsf{params}, \mathsf{pk})$;
$K_0^* \stackrel{R}{\leftarrow} S_K ; b \stackrel{R}{\leftarrow} \{0,1\}$;
$b' \leftarrow \mathcal{A}_2^{\mathcal{O}\mathsf{Decaps}()}(\texttt{GUESS}, \mathsf{params}, \mathsf{pk}, C_1^*, K_b^*, \mathsf{state})$;
$\texttt{si}\ C_1^* \notin \mathsf{CSet}$;
$\texttt{retourner}\ b'\ \texttt{sinon retourner}\ 0$

$\mathcal{O}\mathsf{Decaps}(c)$
$\mathsf{CSet} \leftarrow \mathsf{CSet} \cup \{c\}$;
$K \leftarrow \mathsf{Decaps}(\mathsf{sk}, c)$;
$\texttt{retourner}\ K$

On définit l'avantage de $\mathcal{A}$ dans l'expérience ci-dessus par :

$$\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathcal{KEM},\mathcal{A}} = \left|\Pr[\mathsf{Exp}^{\mathsf{cca\text{-}1}}_{\mathcal{KEM},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{cca\text{-}0}}_{\mathcal{KEM},\mathcal{A}}(\lambda) = 1]\right|$$

On dit qu'un schéma $\mathcal{KEM}$ est CCA-sûr, résistant aux attaques adaptatives à chiffrés choisis, si pour tout attaquant polynomial $\mathcal{A}$, l'avantage de $\mathcal{A}$ est une fonction négligeable en le paramètre de sécurité λ.

[1] D'autres distributions non triviales peuvent également être considérées.

Définitions d'un schéma $\mathcal{DEM}$

Un mécanisme d'*encapsulation de messages*, $\mathcal{DEM}$ est une sorte d' "enveloppe digital" qui permet de garantir le maintien de la confidentialité et de l'intégrité d'un message, en utilisant des méthodes de chiffrement symétrique. Il existe plusieurs implémentations possibles. Nous définissons juste l'interface abstraite et nous renvoyons au rapport de Shoup [117] pour une description détaillée des implémentations classiques de ces schémas.

Un schéma $\mathcal{DEM}$ est défini par la donnée d'un entier ℓ spécifiant la longueur de la clé symétrique utilisée et de deux algorithmes polynomiaux $\mathcal{DEM} \stackrel{\text{déf}}{=} \langle \mathsf{Encrypt}_{\mathsf{DEM}}, \mathsf{Decrypt}_{\mathsf{DEM}} \rangle$ chacun défini de la manière suivante :

- **Algorithme de chiffrement d'un message** $\mathsf{Encrypt}_{\mathsf{DEM}}$ **:** prend en entrée une clé symétrique de chiffrement K (de taille ℓ) et un message m ; cet algorithme renvoie un chiffré C du message m sous la clé K ;
- **Algorithme de génération de déchiffrement** $\mathsf{Decrypt}_{\mathsf{DEM}}$ **:** prend en entrée une clé symétrique K (de taille ℓ) et un chiffré C ; cet algorithme renvoie un message m associé au chiffré C sous la clé K ; si ce message n'est pas défini, cet algorithme renvoie un message d'erreur.

Notions de sécurité pour les schémas $\mathcal{DEM}$

Pour les schémas $\mathcal{DEM}$, nous considérons la notion de sécurité définie par l'expérience $\mathsf{Exp}^{\mathsf{cca}-b}_{\mathcal{DEM},\mathcal{A}}$ suivante : considérons un adversaire générant deux messages m_0, m_1. Le challenger choisit une clé aléatoire de taille ℓ et un bit aléatoire b. Il génère le chiffré du message m_b sous la clé K qu'il retourne à $\mathcal{A}$. Dans une seconde phase, ce dernier pose des requêtes de déchiffrement sauf pour le chiffré challenge. Enfin, il renvoie sa réponse b' pour le bit b.

On définit l'avantage de $\mathcal{A}$ dans l'expérience décrites par :

$$\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathcal{DEM},\mathcal{A}} = \left|\Pr[\mathsf{Exp}^{\mathsf{cca\text{-}1}}_{\mathcal{DEM},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{cca\text{-}0}}_{\mathcal{DEM},\mathcal{A}}(\lambda) = 1]\right|$$

Syntaxe du chiffrement hybride

À présent, nous pouvons décrire l'approche systématique de construction d'un schéma de chiffrement à clé publique en combinant les deux mécanismes $\mathcal{KEM}$ et $\mathcal{DEM}$. Tout d'abord, afin de rendre compatibles ces deux schémas, les clés renvoyées par l'algorithme Encaps doivent avoir la même taille que les clés de chiffrement du schéma $\mathcal{DEM}$. Nous supposons cette condition vérifiée.

Un schéma de chiffrement hybride $\mathcal{H}$-$\mathcal{PKE}$ est un schéma de chiffrement à clé publique, construit à partir d'un mécanisme d'*encapsulation de clés* $\mathcal{KEM}$ et d'*encapsulation de messages* $\mathcal{DEM}$. Plus précisément :

- **Algorithme** Setup **de génération de paramètres de** $\mathcal{H}$-$\mathcal{PKE}$ **:** est défini par l'algorithme Setup du schéma $\mathcal{KEM}$.
- **Algorithme** KeyGen **de génération de clés de** $\mathcal{H}$-$\mathcal{PKE}$ **:** est défini par l'algorithme KeyGen du schéma $\mathcal{KEM}$.
- **Algorithme de chiffrement** $\mathsf{Encrypt}$ **:** prend en entrée une clé publique pk, un message m. Dans un premier temps, cet algorithme fait appel à l'algorithme Encaps qui retourne (K, C). Dans un second temps, il fait appel à l'algorithme $\mathsf{Encrypt}_{\mathsf{DEM}}$ du schéma $\mathcal{DEM}$ qui chiffre le message m sous la clé K et produit un chiffré C'. Il retourne le chiffré $c \stackrel{\text{déf}}{=} C \| C'$.
- **Algorithme de déchiffrement** $\mathsf{Decrypt}$ **:** prend un entrée la clé secrète sk du destinataire et un chiffré c. Cet algorithme décompose le chiffré c en $C \| C'$, puis

"désencapsule" C sous la clé sk pour retrouver la clé K. Cette clé est alors utilisée pour déchiffrer C' avec l'algorithme $\mathsf{Decrypt}_{\mathsf{DEM}}$.

1.1.3 Schéma de signature

Un schéma de signature est défini par la donnée de quatre algorithmes ; mais à présent les rôles de l'émetteur et du destinataire sont duaux : le signataire a besoin d'un secret pour produire sa signature ; en revanche, tout le monde peut vérifier la validité d'une paire message/signature.

Un schéma de signature digitale est spécifié par la donnée de quatre algorithmes polynomiaux $\langle \mathsf{Setup}, \mathsf{KeyGen}, \mathcal{S}, \mathcal{V} \rangle$, chacun défini de la manière suivante :

- Setup : prend en entrée le paramètre de sécurité λ et renvoie les paramètres publics du système params ;
- KeyGen prend en entrée les paramètres publics params et renvoie une paire clés publique/clé privée, vpk/sk ;
- $\mathcal{S}$ prend en entrée la clé secrète du signataire sk, un message m à signer et renvoie la signature σ du message m ;
- $\mathcal{V}$ est un algorithme déterministe qui prend en entrée un message m, une signature σ, la clé publique du signataire vpk et renvoie 0 ou 1, suivant que σ est la signature associée au message m ou pas.

Nous définissons brièvement les propriétés requises pour un tel schéma :

Consistance

Pour toute signature valide, c.à.d. générée correctement par l'algorithme $\mathcal{S}$, l'algorithme de vérification renvoie toujours 1. Plus exactement, étant donné un paramètre de sécurité λ,

$$\forall m \in \mathcal{M} \quad \forall\ \mathsf{vpk}, \mathsf{sk} \quad (\mathsf{vpk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\lambda), \quad \mathcal{V}(\mathsf{vpk}, m, \mathcal{S}(\mathsf{sk}, m)) = 1$$

Types d'attaques

Nous pouvons distinguer différents types d'attaques déterminant les moyens de l'adversaire, nous décrivons ci-dessous les plus couramment utilisés :

Attaques sans messages : pour cette attaque, l'attaquant connaît juste la clé publique du signataire qu'il cherche à attaquer ; la seule possibilité pour lui est de vérifier la validité d'une paire message/signature. Il s'agit donc de l'attaque la plus faible

Attaques à messages connus : pour cette attaque, l'attaquant connaît la clé publique et une liste de paires messages/signatures de taille bornée[2], choisie et produite par le challenger/le signataire.

Attaques à messages choisis : pour cette attaque, l'attaquant a accès à un oracle de signature. Il peut obtenir une liste de messages/signatures de son choix et éventuellement indépendante de la clé publique, dans ce cas, on dit que l'attaquant est générique ; si l'attaque dépend de la clé publique, on parle d'attaques orientées. De plus, le choix des requêtes peut dépendre des signatures précédemment obtenues, on parle alors d'attaques adaptatives.

[2] polynomiale en le paramètre de sécurité.

Niveaux de sécurité

La définition de falsification de signature peut avoir plusieurs sens selon le niveau de sécurité auquel on fait référence. Il existe là encore une hiérarchie de niveaux permettant de caractériser le but de l'attaquant :

Le cassage total : fait référence à la capacité de l'attaquant à retrouver la clé secrète de signature. Tout schéma de signature doit se prémunir de ces attaques.

La falsification universelle : est la capacité de l'attaquant à produire une signature pour tout message n'ayant pas été l'objet d'une requête posée à l'oracle.

La falsification existentielle : est la capacité de l'attaquant à produire une signature pour un message de son choix.

Exemple : nous donnons un exemple simple de signature : la signature RSA :

- $\mathsf{KeyGen}(\lambda)$: cet algorithme choisit un entier e tel que $\mathsf{pgcd}(e, \phi(n)) = 1$, avec $n = pq$ et p et q deux nombres premiers de taille $\lfloor l/2 \rfloor$ bits. La clé publique est (n, e) et la clé secrète sk est l'entier d, tel que $d = e^{-1} \mod \phi(n)$.
- $\mathcal{S}(\mathsf{sk}, m)$ produit la signature $\mathcal{S}(\mathsf{sk}, m) = m^d \mod n$ d'un message $m \in \mathbb{Z}_n$,
- $\mathcal{V}(\mathsf{vpk}, m, \sigma)$ renvoie le résultat du test $m \stackrel{?}{=} \sigma^e \mod n$.

Théorème 1 *La signature RSA est universellement falsifiable sous des attaques à messages choisis et existentiellement falsifiable sous des attaques sans messages.*

Démonstration: Supposons qu'on demande à l'attaquant de produire une signature d'un message m. Ce dernier choisit $a \in \mathbb{Z}_n$ et il demande la signature de a et m/a. Le produit des deux signatures obtenues est la signature du message m. De plus, l'attaquant peut facilement produire un paire message/signature valide en renvoyant (σ, m), avec $m = \sigma^e \mod n$, pour $\sigma \in \mathbb{Z}_n$. □

Plusieurs schémas résistants aux falsifications ont été proposés. Nous ne citerons que quelques-uns d'entre eux. En 1988, en adoptant une approche probabiliste, Goldwasser, Micali et Yao ont proposé un schéma dont la sécurité repose sur la difficulté de factoriser un module RSA et d'inverser la fonction RSA (voir paragraphe 1.2.1 pour une description du problème RSA). Ce schéma est résistant aux falsifications existentielles sous les attaques à messages connus. Cette construction a été améliorée par Bellare, Goldwasser, Micali et Rivest dans [75]. Naor et Yung [102] ont proposé une transformation générique d'un schéma de signature résistante aux falsifications existentielles sous les attaques à messages choisis. Cette construction reste malgré tout inefficace ; elle repose sur l'existence de *permutations à sens unique*. Nous ne rentrerons pas en détail dans la description de ces constructions théoriques, car nous ne les utiliserons pas par la suite.

Schéma Full Domain Hash

Une autre approche, peut-être plus naturelle pour empêcher la falsification, même si elle n'est pas possible pour tous les schémas, est d'utiliser une fonction de hachage $\mathcal{H}$ à valeurs dans le domaine spécifié par le schéma de signature. Cette technique, aussi appelée "*hash then invert*" permet de détruire la structure algébrique du schéma de signature. Pendant longtemps, les propriétés de la fonction de hachage requises pour garantir la sécurité n'étaient pas toujours bien spécifiées, ce qui aboutissait à des protocoles vulnérables. Elles se sont précisées grâce aux attaques, qui ont permis de déterminer pas à pas les propriétés

requises par la fonction de hachage. Pour montrer la correction et la sécurité des schémas dans le contexte de ce paradigme, on modélise souvent la fonction de hachage par un oracle aléatoire, même si cette condition sur la fonction est très forte.

Nous décrivons ci-dessous le schéma Full Domain Hash dérivé de la signature RSA :

- $\mathcal{S}^{\mathcal{H}(*)}(\mathsf{sk}, m)$ produit la signature $\mathcal{S}(\mathsf{sk}, m) = y^d \mod n$ RSA du message m avec $y = \mathcal{H}(m)$.
- $\mathcal{V}(\mathsf{vpk}, m, \sigma)$ calcule d'abord $y = \mathcal{H}(m)$ et renvoie le résultat du test $y \stackrel{?}{=} \sigma^e \mod n$.

Dans l'article [18], Bellare et Rogaway ont montré la sécurité du schéma précédent en proposant une réduction au problème RSA. En 2000, Coron [60] a proposé une amélioration du facteur de réduction. D'autres approches apportent une réduction fine à ce même problème, le schéma probabiliste PSS (*Probabilistic Signature Scheme*) de Bellare et Rogaway [18]. Ce schéma peut être vu comme suite logique puisqu'il s'inscrit dans la classe des schémas "hash-then-invert", excepté que le haché est randomisé avec un aléa frais pour chaque nouvelle signature.

La cryptographie à base de couplages a contribué à l'élaboration de schémas avec des preuves de sécurité dans le modèle standard. Cet outil se révèle être très commode pour la construction de schémas de signatures : sa structure fournit un test de validité publique immédiat pour le problème Diffie-Hellman décisionnel, que nous définirons paragraphe 1.2.2. Par exemple Boneh, Lynn, et Shacham ont construit dans [33] un schéma de signature pour lequel la sécurité est équivalente au problème Diffie-Hellman calculatoire CDH (pour certaines courbes) (voir définition du CDH paragraphe 1.2.2) dans le modèle de l'oracle aléatoire. En 2004, Boneh et Boyen [27] ont proposé un schéma de signature efficace à base de couplages dans le modèle standard. Nous reviendrons sur les constructions basées sur les couplages dans le contexte du chiffrement à la partie II.

1.2 Hypothèses calculatoires

À présent, nous définissons les hypothèses calculatoires utilisées pour la construction de cryptosytèmes asymétriques. Pour certaines d'entre elles, nous ferons quelques rappels mathématiques qui permettront de sélectionner les instances pour lesquelles les problèmes introduits sont difficiles.

1.2.1 Problèmes liés à la factorisation

Problème RSA

Définition 1 *Nous définissons l'ensemble suivant :*

$$H_\lambda \stackrel{déf}{=} \{n \mid n = p \cdot q, \text{ où } p \text{ et } q \text{ premiers de taille } \frac{\lambda}{2} \text{ bits}\}.$$

Par la suite, nous parlerons de module RSA lorsque, étant donné un paramètre de sécurité λ, $n \in H_\lambda$.

Par la suite, nous supposerons la difficulté du problème de la factorisation d'un module $n \in H_\lambda$. Pour $n \in H_\lambda$, nous considérons les deux fonctions suivantes :

- la fonction d'Euler que l'on note $\phi(n)$, où $\phi(n) = (p-1)(q-1)$,
- et la fonction de Carmichaël que l'on note $\lambda(n)$ qui à tout module n associe le plus petit entier t tel que pour tout $w \in \mathbb{Z}_n^*, w^t = 1 \mod n$.

On appelle générateur RSA, un algorithme probabiliste qui prend en entrée un paramètre de sécurité λ puis renvoie un entier $n \in H_\lambda$ et deux entiers e, d tels que $ed = 1 \mod \phi(n)$.

Problème RSA : étant donnés un entier $n \in H_\lambda$, un entier e et un élément $y \in \mathbb{Z}_n$, déterminer l'entier $x \in \mathbb{Z}_n$ tel que $y = x^e \mod n$.

Hypothèse : étant données des paramètres n, e renvoyés par un générateur RSA et un entier $y \in \mathbb{Z}_n$, résoudre le problème RSA est difficile.

Juste un petit mot sur la relation entre le problème RSA et la factorisation : il est évident que si on sait résoudre la factorisation, on sait résoudre le problème RSA. En revanche, l'implication inverse fait aujourd'hui partie des controverses : même si en pratique on résout le problème RSA en factorisant le module, cette approche n'est *a priori* pas la seule possible.

Problème de la résiduosité quadratique, RQ

Le problème de la résiduosité quadratique a été introduit en 1984 par Goldwasser et Micali dans [73]. Avant de l'introduire, faisons quelques petits rappels :

Définition 2 *Pour tout $y \in \mathbb{Z}_n^*$, on note $\mathcal{Q}_n(y) = 0$ s'il existe $w \in \mathbb{Z}_n^*$ tel que $y = w^2 \mod n$. On dit alors que y est un résidu quadratique modulo n. Sinon, $\mathcal{Q}_n(y) = 1$, et dans ce cas, y n'est pas un résidu quadratique modulo n.*

On considère uniquement les entiers $n \in H_\lambda$ et $y \in \mathbb{Z}_n^*$ tels que $\left(\frac{y}{n}\right) = 1$, où $\left(\frac{y}{n}\right)$ est le symbole de Jacobi de y modulo n. On note alors $J_n \stackrel{\text{déf}}{=} \{y \in \mathbb{Z}_n^* \mid \left(\frac{y}{n}\right) = 1\}$.

Problème de la résiduosité quadratique : étant donnés un entier $n \in H_\lambda$ et un élément $y \in J_n$, décider si $\mathcal{Q}_n(y) = 0$.

Hypothèse : le problème de décider la résiduosité quadratique est un problème difficile. En revanche, si la factorisation de n est connue, on peut calculer $\mathcal{Q}_n(y)$. en temps $O(|n|^3)$.

Problème de la haute résiduosité, RH

Le problème de la haute résiduosité a été introduite en 1999 par Paillier [105] et a donné naissance à un schéma de chiffrement que nous décrirons juste après.

Définition 3 *Un élément z est un $n_{ième}$ résidu modulo n^2 s'il existe un élément $y \in \mathbb{Z}_{n^2}^*$ tel que $z = y^n \mod n^2$.*

Problème de la haute résiduosité : étant donnés un paramètre de sécurité λ, un entier $n \in H_\lambda$ et un élément $y \in \mathbb{Z}_{n^2}^*$, décider si y est un $n_{\text{ième}}$ résidu modulo n^2.

Hypothèse : si la factorisation de n n'est pas connue, décider la haute résiduosité d'un élément dans $\mathbb{Z}_{n^2}^*$ est un problème difficile.

1.2.2 Problèmes liés au logarithme discret

Soit $\mathcal{KG}_{\mathsf{LD}}$ un algorithme de génération qui renvoie la description d'un groupe $\mathbb{G}$ d'ordre un grand premier q et un générateur de ce groupe g ; on note cette description $\langle g, q, \mathbb{G}\rangle$.

Problème du logarithme discret, LD

Problème du logarithme discret, LD **:** étant donnés une description $\langle g, q, \mathbb{G}\rangle$ renvoyée par l'algorithme $\mathcal{KG}_{\mathsf{LD}}$ et un élément $y \in \mathbb{G}$, déterminer l'entier relatif $x < q$ tel que $y = g^x$; l'entier x est appelé le logarithme discret de y en base g.

Hypothèse LD **:** étant donnés une description $\langle g, q, \mathbb{G}\rangle$ et un élément y aléatoire dans $\mathbb{G}$, déterminer le logarithme discret de y en base g est un problème difficile.

Problème Diffie-Hellman calculatoire, CDH

Problème Diffie-Hellman calculatoire, CDH **:** étant donnés une description $\langle g, q, \mathbb{G}\rangle$ renvoyée par l'algorithme $\mathcal{KG}_{\mathsf{LD}}$, trois éléments $u, v = u^a, w = u^b$ aléatoires de $\mathbb{G}$, calculer l'élément $h \stackrel{\text{déf}}{=} u^{a\cdot b} \in \mathbb{G}$.

Hypothèse CDH **:** résoudre le problème Diffie-Hellman est un problème difficile.

On dit que h est le Diffie-Hellman de v et w en base u et on écrit $h = \mathsf{CDH}_{\mathbb{G},u}(v, w)$. Lorsque le groupe et la base sont précisés sans ambiguïté par le contexte, nous noterons simplement $h = \mathsf{CDH}(v, w)$.

Nous définissons également le langage des quadruplets[3] Diffie-Hellman de la manière suivante :

$$\mathcal{L}_{\mathsf{DH}_{\mathbb{G}}} \stackrel{\text{déf}}{=} \{(u, v, w, h) \mid h = \mathsf{CDH}_{\mathbb{G},u}(v, w)\}.$$

La version décisionnelle du problème précédent peut également être difficile, nous la définissons ci-dessous.

Problème Diffie-Hellman décisionnel, DDH

Problème Diffie-Hellman décisionnel, DDH **:** étant donnés une description $\langle g, q, \mathbb{G}\rangle$ renvoyée par l'algorithme $\mathcal{KG}_{\mathsf{LD}}$ et quatre éléments $u, v = u^a, w = u^b, h$ aléatoires de $\mathbb{G}$, décider si h est le Diffie-Hellman de v et w en base u, c.à.d. si $(u, v, w, h) \in \mathcal{L}_{\mathsf{DH}_{\mathbb{G}}}$.

Hypothèse DDH **:** étant donnés une description $\langle g, q, \mathbb{G}\rangle$ renvoyée par l'algorithme $\mathcal{KG}_{\mathsf{LD}}$ et quatre éléments $u, v = u^a, w = u^b, h$ aléatoires de $\mathbb{G}$, décider le problème Diffie-Hellman est un problème difficile.

1.3 Schémas de chiffrement homomorphes

Dans cette partie, nous nous intéressons aux cryptosystèmes homomorphes, qui sont très souvent utilisés lorsqu'il s'agit d'effectuer des opérations sur des données chiffrées tout en préservant leur confidentialité. Soit $\mathcal{E}\mathrm{nc} \stackrel{\text{déf}}{=} \langle \mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}\rangle$ un schéma de chiffrement. On suppose ici que la fonction Setup est la fonction identité. Soient $\otimes$ et $\oplus$ des lois de groupe dans $\mathcal{R}$ et $\mathcal{M}$ respectivement, où $\mathcal{R}$ est un espace de probabilité et $\mathcal{M}$ est l'ensemble des messages.

Définition 4 *On dit que* Encrypt *est une fonction de chiffrement homomorphe s'il existe un morphisme de groupes F_{pk} qui, étant donnés deux chiffrés des messages m_1 et m_2 dans $\mathcal{M}$, renvoie le chiffré de $m_1 \oplus m_2$. Plus formellement, la fonction F_{pk} est définie de la manière suivante : $F_{\mathsf{pk}} : \mathcal{C} \times \mathcal{C} \mapsto \mathcal{C}$ et définie par :*

$$F_{\mathsf{pk}}(\mathsf{Encrypt}(m_1; r_1), \mathsf{Encrypt}(m_2; r_2)) = \mathsf{Encrypt}(m_1 \oplus m_2; r_1 \otimes r_2).$$

[3] lorsque le base est sans ambiguïté, on parlera de triplets.

Cryptosystème de Goldwasser-Micali

L'algorithme KeyGen prend en entrée un paramètre de sécurité λ et renvoie $n \in H_\lambda$ et $g \in \mathbb{Z}_n^*$ tel que $\mathcal{Q}_n(g) = 1$ et $\left(\frac{g}{n}\right) = 1$. On définit $\mathsf{pk} = (n, g)$ et $\mathsf{sk} = (p, q)$.

Pour simplifier, on présente la fonction de chiffrement pour $\mathcal{M} = \mathbb{Z}_2$, définie de la manière suivante :

$$\mathsf{Encrypt} : \mathbb{Z}_2 \times \mathcal{R} \mapsto \mathbb{Z}_n^* \text{ avec } \mathsf{Encrypt}(m; r) = g^m \cdot r^2 \mod n \text{ et } \mathsf{pk} = (n, g).$$

La fonction $\mathsf{Encrypt}$ possède les propriétés suivantes :

– tout d'abord pour m et m' deux éléments de $\mathbb{Z}_2$ et $r, r' \overset{R}{\leftarrow} \mathbb{Z}_n^*$, on a :

$$\mathsf{Encrypt}(m; r) \cdot \mathsf{Encrypt}(m'; r') = \mathsf{Encrypt}(m \oplus m' \mod 2; r \cdot r' \mod n) \mod n$$

– de plus,

$$\mathsf{Encrypt}(m; r)^c = \mathsf{Encrypt}(m \cdot c \mod 2; r^c \mod n) \mod n.$$

L'algorithme $\mathsf{Decrypt}$ prend en entrée la clé secrète $\mathsf{sk} = (p, q)$ et calcule $\mathcal{Q}_n(c)$: si c est un résidu quadratique alors $m = 0$; sinon m vaut 1.

Cryptosystème de Paillier

Ce cryptosystème a été introduit par Pascal Paillier en 1999 dans [105]. Ce schéma de chiffrement est sémantiquement sûr contre les attaques à messages choisis sous l'hypothèse de la haute résiduosité.

Pour le déchiffrement, nous définissons l'ensemble S_n et la fonction L :

$$S_n \overset{\text{déf}}{=} \{u < n^2 |\ u = 1 \mod n\} \quad \text{et pour } u \in S_n, \quad L(u) \overset{\text{déf}}{=} \frac{u-1}{n} \mod n^2.$$

L'algorithme KeyGen prend en entrée un paramètre de sécurité λ et renvoie $n \in H_\lambda$ et g un élément non nul de $\mathbb{Z}_{n^2}^*$ d'ordre un multiple non nul de n ; on prend $g = 1 + n$. On définit $\mathsf{pk} = (n, g)$ et $\mathsf{sk} = (p, q)$.

Pour chiffrer un message $m \in \mathbb{Z}_n$, on choisit $r \overset{R}{\leftarrow} \mathbb{Z}_n^*$ et on calcule :

$$\mathsf{Encrypt}(m, r) = g^m \cdot r^n \mod n^2$$

La fonction $\mathsf{Encrypt}$ est homomorphe puisque :

– pour m et m' deux messages de $\mathbb{Z}_n$ et r, r' des éléments aléatoires de $\mathbb{Z}_n^*$, on a :

$$\mathsf{Encrypt}(m; r) \cdot \mathsf{Encrypt}(m'; r')) = \mathsf{Encrypt}(m + m' \mod n; r \cdot r' \mod n) \mod n^2$$

– de plus : $\mathsf{Encrypt}(m, r)^c = \mathsf{Encrypt}(c \cdot m \mod n; r^c \mod n) \mod n^2$

Définition 5 *Soit $n \in H_\lambda$ et $g = 1 + n$. Pour $w \in \mathbb{Z}_{n^2}^*$, on appelle la $n_{ième}$ classe de résiduosité de w en base g, l'unique $m \in \mathbb{Z}_n$ pour lequel il existe $r \in \mathbb{Z}_n^*$ tels que $\mathcal{E}_g(m, r) = w$. On note w_g cette classe.*

$\mathsf{KeyGen}(\lambda)$: renvoie un module RSA n,
$\texttt{retourner}\ \mathsf{pk} = (n, g \overset{\text{déf}}{=} (1+n))$ et $\mathsf{sk} = (p, q)$;

$\mathsf{Encrypt}(\mathsf{pk}, m)$: pour chiffrer $m \in \mathbb{Z}_n$:
choisir $r \overset{R}{\leftarrow} \mathbb{Z}_n^*$;
calculer $\mathsf{Encrypt}(m, r) = g^m \cdot r^n \mod n^2$;

$\mathsf{Decrypt}(\mathsf{sk}, c)$:
vérifier que $c < n^2$ sinon $\texttt{retourner}\ \perp$;
calculer $\lambda \overset{\text{déf}}{=} \lambda(n)$ et $C = L(c^\lambda \mod n^2)$;
$\texttt{retourner}\ m = \frac{C}{g^\lambda \mod n^2} \mod n$

FIG. 1.1 – Cryptosystème de Paillier

Pour déchiffrer c, on cherche la classe c_g de c en base $1+n$. Pour tout $c \in \mathbb{Z}_{n^2}^*$, on a $L(c^\lambda \mod n^2) = \lambda c_g \mod n$; La preuve de ce résultat est dans l'article de Paillier [105]. Pour déchiffrer un chiffré c, on calcule donc :

$$\begin{aligned} c^\lambda &= (1+n)^{m\lambda} r^{n\lambda} = (1+n)^{m\lambda} \\ &= 1 + m\lambda n \mod n^2. \end{aligned}$$

On applique la fonction L et on obtient :

$$L(c^\lambda \mod n^2) = L(1 + m\lambda n) = \lambda m, \text{ où } m = c_g.$$

On donne une description du cryptosystème figure 1.1.

Cryptosystème de Damgard-Jürik

La construction que nous allons présenter a été introduite par Damgard-Jurik [63] peu après l'introduction du cryptosystème précédent. Il s'agit d'une généralisation du schéma de Pailler par laquelle on peut chiffrer des messages de longueur quelconque, sans modifier les paramètres publics de départ, et avec la même clé privée de déchiffrement. De plus, la sécurité sémantique de ce cryptosystème repose sur la même hypothèse calculatoire que celle posée ci-dessus. On choisit toujours les mêmes notations et définitions pour le module n.

En généralisant ce qui précède, on a : $\mathbb{Z}_{n^s} \times \mathbb{Z}_n^* \simeq \mathbb{Z}_{n^{s+1}}^*$ (la preuve de ce résultat peut être trouvée dans l'article [63]). Le cas $s = 1$ correspond au cryptosystème de Paillier.

Soit s un entier fixé tel que $s < p, q$. On prend toujours $g = 1 + n$. On considère la fonction de chiffrement $\mathsf{Encrypt} : \mathbb{Z}_{n^s} \times \mathbb{Z}_n^* \mapsto \mathbb{Z}_{n^{s+1}}^*$ définie par :

$$\mathsf{Encrypt}(m, r) = g^m \cdot r^{n^s} \mod n^{s+1}, \text{ avec } m \in \mathbb{Z}_{n^s} \text{ et } r \overset{R}{\leftarrow} \mathbb{Z}_n^*$$

Cette fonction vérifie les propriétés suivantes :

– pour tout $m, m' \in \mathbb{Z}_{n^s}$ et pour tout $r, r' \overset{R}{\leftarrow} \mathbb{Z}_n^*$, on a :

$$\mathsf{Encrypt}(m; r) \cdot \mathsf{Encrypt}(m; r') = \mathsf{Encrypt}(m + m \mod n^s; r \cdot r' \mod n),$$

où $\cdot$ est une opération multiplicative dans $\mathbb{Z}_n^*$ et $+$ est une opération additive dans $\mathbb{Z}_{n^s}$;

– pour tout entier $c \in \phi(n^2)$, on a :

$$\mathsf{Encrypt}(m,r)^c = \mathsf{Encrypt}(m \cdot c \mod n^s; r^c \mod n) \mod n^2.$$

Déchiffrement : pour tout chiffré c, on a alors $c = (1+n)^m \cdot r^{n^s} \mod n^{s+1}$.
Soit d un multiple de $\lambda(n)$ non nul. On a :

$$\begin{aligned} c^d &= (1+n)^{m\cdot d}(r^{n^s})^d \mod n^{s+1} \\ &= (1+n)^{m\cdot d} r^{n^s \cdot d} \mod n^{s+1} \\ &= (1+n)^{m\cdot d} \mod n^{s+1} \end{aligned}$$

Pour déchiffrer m, il s'agit donc de trouver le logarithme discret de c en base $1+n$ dans $\mathbb{Z}_{n^s}$. On a vu que $(1+n)^m = 1 + nm \mod n^2$. Nous donnons ci-dessous les étapes de l'algorithme de déchiffrement :

$$L((1+n)^i \mod n^{s+1}) = (i + \binom{i}{2} n + \cdots + \binom{i}{s} \cdot n^{s-1}) \mod n^s$$

Il s'agit d'extraire pas à pas les résidus modulo les puissances successives de n. Pour cela, on définit :

$$\begin{aligned} i_1 &= i \mod n \\ i_2 &= i \mod n^2 \\ &\vdots \\ i_j &= i \mod n^j \end{aligned}$$

On sait déjà extraire i_1 grâce au déchiffrement de Paillier.
À une étape donnée, on suppose avoir i_j et il s'agit de trouver i_{j+1}.

$$i_j = i_{j-1} + k \times n^{j-1}, \text{ pour } 0 \leqslant k < n.$$

On applique la fonction L et on obtient :

$$L((1+n)^i \mod n^{j+1}) = i_j + \binom{i_j}{2} n + \cdots + \binom{i_j}{j} n^{j-1} \mod n^j$$

$$\text{Pour } j > t > 0, \text{ on a, } \binom{i_j}{t+1} n^t = \binom{i_{j-1}}{t+1} n^t \mod n^j$$

En réécrivant i_j à l'aide de l'équation de la deuxième équation précédente, on a :

$$\binom{i_j}{t+1} n^t = \binom{i_{j-1} + k \times n^{j-1}}{t+1} n^t \mod n^j$$

Les termes $k \times n^{j-1}$ multiplié par une puissance non nul de n s'annulent modulo n^j.
Ce qui permet d'aboutir à l'équation suivante :

$$L((1+n)^i \mod n^{j+1}) = (i_{j-1} + k \times n^{j-1} + \binom{i_{j-1}}{2} n + \cdots + \binom{i_{j-1}}{j} n^{j-1}) \mod n^j$$

$\mathsf{Setup}(\lambda)$: génère $\mathsf{params} \leftarrow \mathcal{KG}_{\mathsf{LD}}$;
`retourner` $\mathsf{params} = \langle g, q, \mathbb{G} \rangle$;

$\mathsf{KeyGen}(\mathsf{params})$: retourne $\mathsf{pk} = (\mathsf{params}, y = g^x)$ et $\mathsf{sk} = (x)$ où $x \xleftarrow{R} \mathbb{Z}_q$;

$\mathsf{Encrypt}(\mathsf{pk}, m)$: pour chiffrer un message $m \in \mathbb{G}$;
calculer $c = (c_0, c_1) = (g^r, y^r \cdot m)$ pour $r \xleftarrow{R} \mathbb{Z}_q$;

$\mathsf{Decrypt}(\mathsf{sk}, c)$: `si` $c \in \mathbb{G}^2$, décomposer c en (c_0, c_1) et `retourner` $m = c_1/c_0^x$;
`sinon retourner` un symbole d'erreur

FIG. 1.2 – Cryptosystème ElGamal

En combinant les équations et (4), on obtient l'équation finale qui permet de récupérer i_j :

$$i_j = L((1+n)^i \mod n^{j+1}) - (i_{j-1} + \binom{i_{j-1}}{2} n + \cdots + \binom{i_{j-1}}{j} n^{j-1}) \mod n^j$$

Si on connaît les i_j pour toute valeur de j, on peut calculer m tel que $i = m \cdot d \mod n^s$ en calculant $d^{-1} \mod n^s$, puisque $m \cdot d \cdot d^{-1} = m \mod n^s$.

Cryptosystème ElGamal

Le cryptosystème ElGamal a été introduit en 1985 dans l'article [65]. On donne une description du schéma de chiffrement figure 1.2.

La fonction de chiffrement est à sens unique sous l'hypothèse Diffie-Hellman calculatoire et la sécurité sémantique de ce schéma repose sur le problème décisionnel associé. On remarque que ce schéma de chiffrement est homomorphe, puisque :

$$\forall m, m' \in \mathcal{M} \qquad \mathsf{Encrypt}(m; r) \cdot \mathsf{Encrypt}(m'; r') = \mathsf{Encrypt}(m \cdot m; r + r' \mod q)$$

Un point important pour ce schéma est que le message doit être encodé par un élément du groupe. Il est donc nécessaire de définir une fonction bijective d'encodage encode à valeurs dans le groupe $\mathbb{G}$ inversible de manière efficace. Le plus souvent cet encodage est assez coûteux et possède deux autres inconvénients majeurs : il détruit les propriétés homomorphiques du schéma et est incompatible avec le choix de paramètres permettant l'optimisation des calculs. Une approche subsidiaire de ce problème d'encodage consiste à définir une variante du schéma avec un fonction de hachage masquant le message, mais la preuve de sécurité nécessite la modélisation de cette fonction par un oracle aléatoire.

1.4 Application aux réseaux de mélangeurs

Le concept de *réseaux de mélangeurs* (ou *mix-networks* en anglais) a été inventé par David Chaum [50] dans les années 1980 pour l'anonymat dans les systèmes de votes électroniques. Sa structure est très commode pour construire des systèmes de votes, car il

permet d'obtenir directement les propriétés de vérifiabilité et de consistance indispensable à ces derniers :

- la première propriété signifie que chaque votant peut vérifier que son vote a été pris en compte ou déléguer cette capacité de vérification, elle permet également de s'assurer que tout le monde peut vérifier le résultat de l'élection ;
- la seconde propriété signifie qu'il n'existe pas de sous-ensemble de participants capable d'interrompre le bon déroulement du processus.

Dans cette partie, nous décrivons les deux types de protocoles *mix-networks* rencontrés : les *réseaux de mélangeurs* avec déchiffrement et les *réseaux de mélangeurs* avec rechiffrement. Nous donnons un exemple pour cette dernière catégorie, dont nous reparlerons un peu plus loin, au chapitre 5.

1.4.1 Réseaux avec déchiffrement

La structure globale consiste en une phase de chiffrement et une phase de concaténation de plusieurs mix de déchiffrés partiels. Plus exactement, on considère k *mélangeurs* (ou *mixnets*), notés Mix_i pour $i = 1, \cdots, k$, chacun possédant une paire de clés publique et secrète $(\mathsf{PK}_i, \mathsf{SK}_i)$. La phase initiale de chiffrement consiste à chiffrer successivement chacun des l votes B_j pour $j = 1, \cdots, l$ avec les clés $\mathsf{PK}_1, \cdots, \mathsf{PK}_k$. Chacun des *mélangeurs* Mix_i reçoit l chiffrés partiels de la forme $C_j = E_i(E_{i-1} \cdots E_1(B_j))$. Le mélangeur Mix_i déchiffre et applique une permutation secrète aux éléments $C_j = E_{i-1}(E_{i-2} \cdots E_1(B_j)))$ qu'il envoie au *mélangeur* suivant Mix_{i-1}. Afin de garantir la propriété de vérifiabilité, chaque *mélangeur* produit une preuve qu'il renvoie les permutés des l déchiffrés .
On remarque que quelques précautions doivent être prises :

- pour préserver l'anonymat de la source, les chiffrés doivent tous avoir la même taille ;
- afin de garantir que le dernier *mélangeur* renvoie la réponse correcte, sa clé secrète peut être partagée entre plusieurs serveurs ;
- afin d'empêcher toute attaque qui exploiterait la malléabilité entre les bulletins chiffrés, les votes chiffrés ne sont publiés qu'après que tous les participants ont voté. Dans ce cas, on peut se contenter de la sécurité sémantique qui est garantie si au moins un des *mélangeurs* permute bien les bulletins chiffrés.

1.4.2 Réseaux avec rechiffrement

Le rôle d'un *réseau de mélangeurs* avec rechiffrement [4] est seulement de mélanger l'entrée. Cependant, on ne peut pas se contenter de brouiller l'entrée car l'ensemble des chiffrés associé à un message reste alors inchangé, et on peut dans ce cas retrouver la source associée à un chiffré. On ajoute donc une étape qui se charge de randomiser une nouvelle fois les chiffrés. Chacun des *mélangeurs* $\text{Mix}_1, \cdots, \text{Mix}_k$ applique une permutation secrète et rechiffre le résultat. On décrit ci-dessous l'un des schémas de rechiffrement qui est couramment utilisé, le schéma ElGamal ; nous aborderons plus en détail la construction de ces schémas un peu plus loin, à la partie II section 5.2.

Exemple : rechiffrer avec ElGamal
Nous avons décrit le cryptosystème ElGamal figure 1.2. Supposons que nous sommes en possession d'un chiffré $c = (c_0, c_1) = (g^r, y^r \cdot m)$. Pour rechiffrer c en c', on choisit $s \xleftarrow{R} \mathbb{Z}_q$,

[4] Nous considérons pour le moment que les termes rechiffrement/rerandomisation, rechiffré/rerandomisé, rechiffrer/rerandomiser sont équivalents.

et on définit $c' = (c'_0, c'_1) = (c_0 \cdot g^s, c_1 \cdot y^s)$ qui est un élément aléatoire dans l'ensemble des chiffrés associés au message m.

Réseaux avec rechiffrement ElGamal :

1. Initialisation : étant donnée la clé publique pk, cette première phase, indicée k consiste à chiffrer les bulletins de vote B_j pour $j = 1, \cdots, l$, puis à envoyer les chiffrés notés $c_{j,k}$ obtenus ;
2. la $i_{\text{ième}}$ étape consiste à appliquer une permutation sur l'entrée et à rechiffrer les bulletins chiffrés $c_{j,i+1}$, pour $j = 1, \cdots, l$;
3. l'étape finale consiste à déchiffrer les l chiffrés $c_{j,2}$ (la clé secrète est distribuée entre plusieurs serveurs).

Vérifiabilité et consistance

Chaque *mélangeur* doit fournir une preuve qu'il existe bien une permutation telle que la sortie $c_{j,i}$ est le rechiffré de l'entrée permutée $c_{\pi(j),i}$. Pour le schéma de rechiffrement ElGamal, si $c = (c_0, c_1) = (g^r, y^r \cdot m)$ et $c' = (c'_0, c'_1) = (g^s, y^s \cdot m')$ sont deux chiffrés, alors, les deux conditions suivantes sont équivalentes :

1. c' est le rechiffré de c ;
2. $\left(g, y, \frac{c'_0}{c_0}, \frac{c'_1}{c_1}\right) = \left(g, y, g^{s-r}, \frac{m'}{m} \cdot y^{s-r}\right)$ est un t-uplet DDH.

Nous rappelons que le langage $\mathcal{L}_{\mathsf{DH}_{\mathbb{G}}}$ est défini par $\mathcal{L}_{\mathsf{DH}_{\mathbb{G}}} \stackrel{\text{déf}}{=} \{(g, y, c, c') \mid \log_g c = \log_y c'\}$; on a alors :

$$\mathcal{L}_{\mathsf{DH}_{\mathbb{G}}} = \{(g, y, c, c') \mid \exists s \in \mathbb{Z}_q \;\; c = g^s \text{ et } c' = y^s)\}$$

Le protocole de Chaum et Pederson [53] permet à un prouveur $\mathcal{P}$ de montrer l'appartenance d'un quadruplet au langage $\mathcal{L}_{\mathsf{DH}_{\mathbb{G}}}$ sans laisser fuir d'information sur l'exposant s. Dans le prochain chapitre, nous précisons les propriétés que doit vérifier ce type de système de preuve. Pour l'instant, nous donnons seulement les interactions entre le prouveur $\mathcal{P}$ et le vérifieur $\mathcal{V}$:

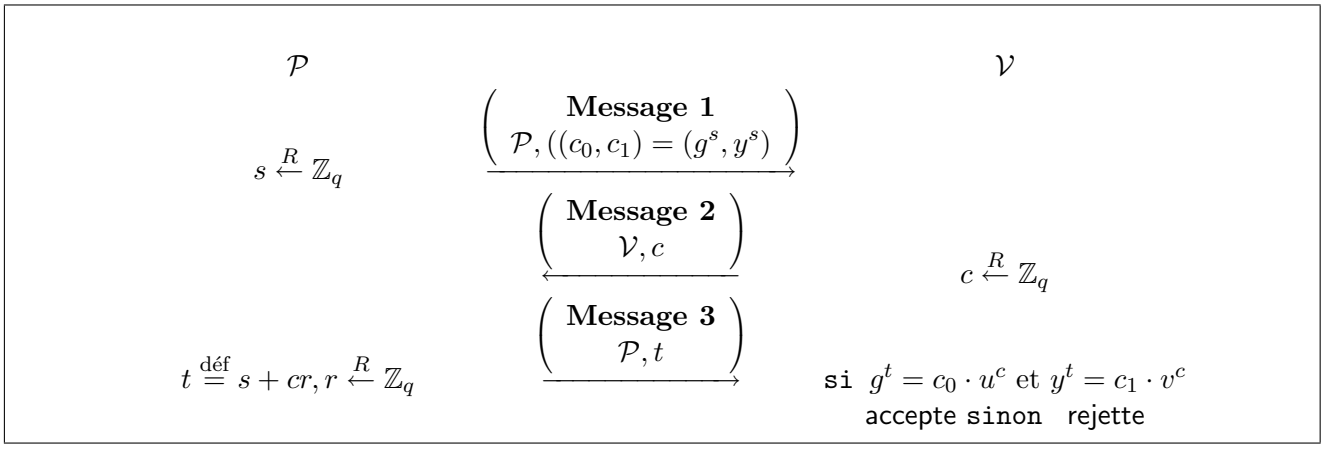


FIG. 1.3 – Preuve d'appartenance au langage $\mathcal{L}_{\mathsf{DH}_{\mathbb{G}}}$ de Chaum-Pederson (à divulgation nulle de connaissance contre un vérifieur honnête).

Chapitre 2

Protocoles interactifs pour l'anonymat

Dans ce chapitre, l'interaction est à l'honneur ; nous présentons les protocoles interactifs que nous utiliserons par la suite : les *preuves interactives* et le concept de *transfert transparent*. Les *preuves interactives* sont un ingrédient majeur dans la construction de nombreux protocoles cryptographiques. Même si le concept a été introduit en 1989, leur puissance ne cesse d'être appréciée. Le concept de *transfert transparent*, quant à lui, pose une problématique ancienne et plutôt simple mais sa résolution suscite encore aujourd'hui de nombreuses questions ouvertes. Commençons par rappeler le principe général de ces deux concepts, avant de montrer comment elles peuvent être utilisées pour le chiffrement de groupe au chapitre 5 et l'authentification anonyme au chapitre 8.

Sommaire

2.1 Preuves interactives

2.1.1 Schéma de *Mise en gage*

Un protocole de *Mise en gage* permet à un émetteur et son interlocuteur d'échanger des messages en s'engageant sur une valeur sans la révéler de telle sorte à ce que cet engagement ne soit plus modifiable *a posteriori*. La donnée d'une trappe permet à un interlocuteur de retrouver la valeur sur laquelle l'émetteur s'est engagée. Plus formellement :

Définition 6 *Un schéma de* Mise en gage *est déterminé par la donnée de trois algorithmes* $\langle \mathcal{Z}_c, \mathcal{C}, \mathcal{T} \rangle$*, chacun défini de la manière suivante :*

- $\mathcal{Z}_c$ *prend en entrée un paramètre de sécurité* λ*. Cet algorithme retourne des paramètres publics* cpk *;*
- $\mathcal{C}$ *prend en entrée un message et les paramètres* cpk*. Cet algorithme renvoie un engagement sur le message* m*, noté* C *et une information supplémentaire d'ouverture, notée* ρ *;*

– *$\mathcal{T}$ prend en entrée l'information ρ, un message m et l'engagement C, $\mathcal{T}(\rho, m, C)$ renvoie 1 si C est bien un engagement sur m et 0 sinon.*

Ces algorithmes vérifient les propriétés suivantes :

1. **dissimulation** (ou *hiding* en anglais) : sans la donnée de l'information ρ, un vérificateur, en possession de C ne doit apprendre aucune information sur le message. Plus généralement, les ensembles des engagements sur deux valeurs différentes sont indistinguables. On parle de dissimulation *calculatoire statistique* ou *parfaite* selon la puissance de calcul donnée à l'attaquant et sa probabilité de succès.
2. **engagement** (ou *binding* en anglais) : pour toute paire (m, cpk) et tout engagement C tel que $(C, \rho) \leftarrow \mathcal{C}(m, \mathsf{cpk})$, il doit être difficile de trouver un couple m', ρ' tels que $m' \neq m$ vérifiant $\mathcal{T}(\rho, m, C) = \mathcal{T}(\rho', m', C) = 1$. On parle d'engagement *calculatoire*, *statistique* ou *parfait* selon la puissance de calcul donnée à l'attaquant et sa probabilité de succès.

On peut vouloir être capable d'extraire la valeur secrète. Dans ce cas, l'algorithme $\mathcal{Z}_c$ renvoie une trappe supplémentaire τ_x, et on définit un algorithme supplémentaire, appelé extracteur et que l'on note $\mathcal{D}_c$; cet extracteur prend en entrée la trappe τ_x une paire (C, ρ) et renvoie m si C est un engagement sur la valeur m avec pour information auxiliaire ρ, autrement cet algorithme renvoie un symbole d'erreur, $\perp$. On parle alors d'engagement "extractable".

De plus, $\mathcal{Z}_c$ peut prendre en entrée une trappe τ_{eq}. Nous définissons dans ce cas un algorithme supplémentaire, l'"équivocateur" $\mathcal{Q}_{eq}$ qui étant donnés τ_{eq}, m, m' avec $m' \neq m$ et $(C, \rho) \leftarrow \mathcal{C}(\mathsf{cpk}, m)$ retourne ρ' tel que $\mathcal{T}(\rho, m, C) = \mathcal{T}(\rho', m', C) = 1$. On parle d'engagement équivoque.

2.1.2 Preuves interactives d'appartenance à un langage

Une preuve interactive est un protocole à deux parties qui permet à un prouveur de convaincre un vérifieur de la validité d'une assertion, par exemple "{x} appartient au langage L". Les deux entités sont modélisées pas des machines de Turing probabilistes (la fonction de transition dépend d'un aléa) possédant un ruban aléatoire d'entrée et un ruban de communication. On dit deux machines sont en interaction si elles partagent le même ruban de communication, de telle sorte que chacune puisse lire le dernier message écrit par l'autre machine ; on suppose de plus que les deux machines peuvent écrire des messages sur leur propre ruban.

Si $X_{\mathcal{V}}$ et $X_{\mathcal{P}}$ sont les données secrètes initiales de $\mathcal{V}$ et $\mathcal{P}$ respectivement et si Y est l'entrée initiale commune, on note $\langle \mathcal{V}(X_{\mathcal{V}}), \mathcal{P}(X_{\mathcal{V}}) \rangle(Y)$. On écrira $\mathcal{V}_{\mathcal{P}(x)}(x)$ accepte , en cas de succès et $\mathcal{V}_{\mathcal{P}(x)}(x)$ rejette en cas d'échec.

Définition 7 *Soient $\mathcal{P}$ un algorithme exécuté par une machine de Turing probabiliste interactive et $\mathcal{V}$ un algorithme exécuté par une machine de Turing probabiliste* polynomiale *sans entrées auxiliaires. On suppose que les deux machines partagent un ruban d'entrée et un ruban de communication. On dit que $\langle \mathcal{P}, \mathcal{V} \rangle$ constitue un système de preuve interactive pour le langage $\mathcal{L}$ si les deux propriétés suivantes sont vérifiées :*

1. ***consistance*** *(ou* completness*) : $\mathcal{V}$ accepte les entrées qui sont des éléments de L avec probabilité écrasante :*

$$\forall x \in L \quad \mathsf{Prob}[\langle \mathcal{V}, \mathcal{P}(x) \rangle(x) : \mathcal{V}_{\mathcal{P}(x)}(x) \ accept] > 1 - \nu(\lambda)$$

2. ***significativité*** *(ou* soundness*) : tout prouveur (éventuellement malhonnête) avec une entrée qui n'est pas dans le langage est accepté avec probabilité négligeable ; plus formellement :*

$$\forall \mathcal{P}' \ \forall x \notin L \quad \mathsf{Prob}[\langle \mathcal{V}, \mathcal{P}'(x) \rangle(x) : \mathcal{V}_{\mathcal{P}'(x)}(x) \ \textit{accept}] < \nu(\lambda),$$

ici les calculs de $\mathcal{P}'$ dépendent de l'historique des messages (ou transcripts *en anglais) échangés avec le vérifieur.*

Les probabilités sont sur les aléas du vérifieur $\mathcal{V}$ et des prouveurs $\mathcal{P}, \mathcal{P}'$.

On suppose que la puissance de calcul du prouveur est illimitée, contrairement à celle du vérifieur qui, elle, est bornée. Cela dit, pour certaines applications cryptographiques courantes cette hypothèse n'est pas réaliste, par exemple lorsque le prouveur veut convaincre le vérifieur qu'il connaît une ou la solution d'un problème difficile. On définit dans ce cas un type particulier de protocoles, appelés preuves interactives de connaissance.

2.1.3 Preuves de connaissance

Le principe des preuves de connaissance est un peu différent : la validité d'une assertion se traduit par l'exhibition d'un témoin x. Le prouveur est assimilé à une machine de Turing possédant une entrée auxiliaire secrète, "un témoin" qui lui permet de répondre aux questions du vérifieur. Et le succès d'un tel protocole se traduit par la validité d'une assertion et "$\mathcal{P}$ connaît un témoin de x" que ce dernier a donné comme entrée commune. Cependant, pour les protocoles de connaissance le prouveur ne peut plus calculer ce témoin car il est supposé polynomialement borné. Nous supposons alors que les machines interactives $\mathcal{P}$ et $\mathcal{V}$ disposent d'un ruban d'entrée auxiliaire commun. L'acceptation du protocole sur cette entrée traduit que le prouveur (polynomialement borné) connaît un témoin associé à cette valeur.

Considérons une relation $\mathcal{R}$ décidable en temps polynomial. Deux éléments (x, w) sont en relation si $(x, w) \in \mathcal{R}$ et on dit que w est un témoin pour x.

Définition 8 *Une preuve de connaissance associée à une relation $\mathcal{R}$ est une paire de machines de Turing probabilistes $\langle \mathcal{P}, \mathcal{V} \rangle$ qui satisfait les deux propriétés suivantes :*

1. ***consistance*** *(ou* completness*) : un prouveur honnête $\mathcal{P}$ est accepté avec grande probabilité :*

$$\forall (x, w) \in \mathcal{R} \quad \mathsf{Prob}[\langle \mathcal{V}, \mathcal{P}(w) \rangle(x) : \mathcal{V}_{\mathcal{P}(x,w)}(x) \ \textit{accepte}] > 1 - \nu(\lambda)$$

2. ***significativité*** *(ou* soundness*) : si $\mathcal{V}$ accepte un prouveur malhonnête $\mathcal{P}'$, il existe une extracteur $\mathcal{E}_{\mathcal{P}'}$ capable de contrôler le ruban de $\mathcal{P}'$, et qui à la suite de son interaction avec $\mathcal{P}'$ parvient à extraire le secret. Plus formellement, $\forall \mathcal{P}' \ \exists \mathcal{E}_{\mathcal{P}'} \ \forall x \ \forall w'$*

$$\left| \mathsf{Prob}[\langle \mathcal{P}'(*), \mathcal{V} \rangle(x) : \mathcal{V}_{\mathcal{P}'(*)}(x) \ \textit{accepte}] \Longrightarrow \mathsf{Prob}[(x, \mathcal{E}_{\mathcal{P}'}(x)) \in \mathcal{R}] \right| < \nu(\lambda)$$

Les probabilités sont sur les aléas du vérifieur $\mathcal{V}$, des prouveurs $\mathcal{P}, \mathcal{P}'$ et de l'extracteur effectif $\mathcal{E}_{\mathcal{P}}$.

La propriété de *zero-knowledge* est une des propriétés les plus intéressantes car elle résout une question paradoxale et primordiale en cryptographie : comment prouver la

connaissance d'un secret sans le révéler, et plus précisément sans transmettre au vérifieur la capacité de prouver l'assertion en question ? Ces protocoles sont appelés preuve de connaissance à divulgation nulle de connaissance et ont été introduites en 1985 par Goldwasser *et al.* dans leur article fondateur [74]. Afin de définir cette notion, considérons l'existence d'un simulateur utilisant le vérifieur $\mathcal{V}$ comme boîte noire pour simuler l'exécution du protocole interactif $\langle \mathcal{P}, \mathcal{V} \rangle$. On dit qu'un protocole interactif $\langle \mathcal{P}, \mathcal{V} \rangle$ est à divulgation nulle de connaissance ou *zero-knowledge*, si pour tout vérifieur il existe un simulateur S capable de reproduire une simulation indistinguable de celle associée aux communications réelles entre le prouveur et le vérifieur. On parlera selon le contexte d'indistinguabilité calculatoire, statistique ou parfaite.

2.2 Transfert inconscient, $\mathcal{OT}$

Considérons la situation suivante : un serveur possède une liste de n chaînes de caractères $x_1, \cdots, x_n$ et le client veut récupérer l'élément x_i. Le but est d'établir un protocole éventuellement interactif entre le serveur et le client[1] de manière à ne révéler au serveur aucune information sur l'indice i. Une solution simple serait d'envoyer toute la base de données mais cette solution ne satisfait ni le serveur qui ne veut pas transmettre les entrées x_j, pour $j \neq i$, ni les contraintes d'efficacité : plus exactement, on cherche à obtenir un protocole avec une complexité de communication strictement inférieure à n, la taille de la base de données.

Ce problème a été introduit par Rabin en 1981 [110] sous une forme légèrement différente, où le serveur envoie l'entrée demandée avec probabilité $\frac{1}{2}$, sans savoir si le client a bien reçu la réponse à sa requête. On fait souvent référence à ce protocole en l'appelant protocole de Rabin. On utilise souvent une variante de ce protocole, que l'on note $\binom{1}{2}$-$\mathcal{OT}$. Cette autre variante a plusieurs applications intéressantes : multi-partis ou de fonctions d'évaluation prouvées sûres (*secure evaluation functions*). Nous pouvons définir la primitive $\binom{1}{2}$-$\mathcal{OT}$ de la manière suivante : l'émetteur possède deux bits b_0, b_1[2], et le destinataire un bit t. Le but est de permettre à ce dernier de recevoir b_t sans compromettre la confidentialité du bit t. En 1987, Claude Crépeau [62] a montré l'équivalence entre le protocole de Rabin et la primitive $\binom{1}{2}$-$\mathcal{OT}$.

D'autres variantes de protocoles de *transfert transparent* ont été défini : dans un protocole $\binom{1}{n}$-$\mathcal{OT}$ (resp. $\binom{k}{n}$-$\mathcal{OT}$), le destinataire désire recevoir une entrée (resp. k entrées) de la base de données contenant n entrées.

Certains résultats importants méritent d'être cités : Goldreich montre dans le volume 2 de son livre que l'existence de permutations à trappe implique l'existence de protocoles $\binom{1}{2}$-$\mathcal{OT}$. Il s'agit dans un premier temps, de construire un protocole dans le modèle "honnête mais curieux", c.à.d. qu'on ne considère que les adversaires passifs qui ne peuvent pas modifier les données secrètes, puis en utilisant des preuves à divulgation nulle de connaissance, on rend ce protocole résistant aux attaques actives.

Naor et Pinkas [101] ont introduit une version adaptative du protocole $\binom{k}{n}$-$\mathcal{OT}$, où le serveur s'engage sur une base de données contenant les messages durant une phase d'initialisation, puis à l'issue d'au plus k interactions entre le serveur et l'utilisateur, ce dernier reçoit au plus k messages de son choix. Il est important de voir qu'à la $i_{\text{ième}}$ interaction, le choix de l'utilisateur peut dépendre des messages reçus aux $i-1$ interactions précédentes. Les contraintes d'efficacité sont similaires aux précédentes : pour la phase

[1] ou un émetteur et un récepteur de manière plus générale.

[2] on peut étendre ce protocole à des messages de l bits en faisant l invocations en parallèle.

d'initialisation, la complexité de communication doit être linéaire en n et λ, le paramètre de sécurité et pour chacune des phases, elle doit être au moins logarithmique en n (afin de permettre l'encodage de l'index choisi).

Naor et Pinkas introduisent également un modèle, qui considère des attaques actives du client et du serveur de manière indépendante : la sécurité du client est assurée si le serveur ne parvient pas à distinguer les simulations pour deux choix différents d'index ; les garanties pour le serveur sont plus fortes : tout utilisateur malhonnête peut être assimilé à un utilisateur engagé dans une exécution d'un protocole idéal, c.à.d. un protocole implémenté par une autorité de confiance. Pour cela, on doit pouvoir simuler le destinataire de manière efficace, et de fait Naor et Pinkas ont appelé ce modèle *half simulation model*. Les auteurs ont signalé une vulnérabilité dans ce modèle : le serveur peut éventuellement changer sa réponse au cours de l'exécution en fonction de l'index choisi par l'utilisateur. Ces attaques, appelées *selective failure attacks* ne sont pas prises en compte par le *half simulation model*. En 2007, Camenisch *et al.* [43] proposent une solution en étendant le modèle de Naor et Pinkas.

2.3 Private Information Retrieval, $\mathcal{PIR}$

La problématique d'interrogation confidentielle de bases de données ou *Private Information Retrieval* ($\mathcal{PIR}$) est la même que la précédente, celle définie pour le transfert inconscient excepté que les garanties de sécurité (confidentialité de la base de données) ne sont pas nécessaires pour le serveur. Si cette condition est explicitement requise, on parle alors de $\mathcal{PIR}$ symétrique (ou *Symmetric* $\mathcal{PIR}$ en anglais), une primitive équivalente à $\binom{1}{n}$-$\mathcal{OT}$ d'un point de vue conceptuel. On distingue deux catégories de protocoles de $\mathcal{PIR}$ selon les moyens donnés à l'attaquant, ceux pour lesquels l'attaquant possède une puissance de calcul illimitée pour retrouver l'index demandé, on les appelle *Information Theoretic* $\mathcal{PIR}$; et ceux pour lesquels l'attaquant est limité à une puissance de calcul polynomiale ; ces derniers protocoles sont appelés *Computational* $\mathcal{PIR}$.

Kushilevitz *et al.* [58] ont montré qu'il n'était pas possible d'obtenir une complexité sous-linéaire en la taille de la base de données, pour les *Information Theoretic* $\mathcal{PIR}$. Ils ont alors proposé un modèle avec réplication des bases de données, dont la généralisation peut se traduire ainsi : on considère 2^d copies de la base de données. Supposons que chacune d'entre elles est représentée comme un hypercube de dimension d, contenant $(\sqrt[d]{n})^d$ éléments. Il existe alors un protocole de $\mathcal{PIR}$ avec une complexité de communication $\sqrt[d]{n}$, où $d \geqslant 1$.

Les protocoles *Computational* $\mathcal{PIR}$ conduisent à une complexité de communication plus faible mais des calculs plus coûteux (au moins polynômiaux en n). En contrepartie, la sécurité du protocole est assurée sous la condition de la validité d'une hypothèse calculatoire. Un des avantages majeurs est qu'une seule base de données suffit à obtenir une complexité de communication sous-linéaire [91]. Il existe également une construction générique qui à partir d'un cryptosystème homomorphe sémantiquement sûr permet d'obtenir un protocole $\mathcal{PIR}$ garantissant l'indistinguabilité des index des entrées de la base de données. D'autres conjectures permettent d'assurer l'existence de protocoles $\mathcal{PIR}$. Nous donnons figures 2.1, 2.2 un aperçu des résultats généraux de le complexité de communication pour ces deux types de protocoles :

[3]Nous donnons une description de ce schéma BGN (Boneh, Goh et Nissim) figure 3.3 au chapitre qui suit.

techniques utilisées	réf.	complexité de communicaT.
racine $k_{\text{ième}}, k \geqslant 4$	[58]	$k \cdot n^{\frac{1}{\log k}}$
interpolation poly	[58]	$(k^2 \cdot \log_2 k) \cdot n^{\frac{1}{k}}$
récurrence	[6]	$2^{k^2} \cdot n^{\frac{1}{2k-1}}$
alg. linéaire	[9], [84]	$k^3 \cdot n^{\frac{1}{2k-1}}$

FIG. 2.1 – Résultats généraux pour les *Information Theoretic* $\mathcal{PIR}$, où n est la longueur de la base de données, k le nombre de bases de données, et où λ est un paramètre de sécurité.

hypothèse calc.	réf.	complexité ce comunicaT.
résiduosité quadratique	[91]	$\mathcal{O}(n^\epsilon)$, $\epsilon > 0$
perm. à trappe	[92]	$n - O(n)$
haute résiduosité	[49]	$\mathcal{O}(n^\epsilon \log_2 n)$, $\epsilon > 0$
BGN^3	[32]	$\mathcal{O}(\lambda \sqrt[3]{n})$

FIG. 2.2 – Résultats généraux pour les *Computational* $\mathcal{PIR}$ pour une base de données contenant n est éléments, où λ est un paramètre de sécurité.

Chapitre 3

Dans les groupes bilinéaires

Dans cette thèse, nous décrivons plusieurs constructions et proposons un schéma à base de couplages (voir partie II paragraphe 4.3.3). Dans cette partie, nous nous intéressons aux couplages en tant qu'objet mathématique, puis nous préciserons leur utilisation en cryptographie. Ces applications bilinéaires sont propices à une utilisation en cryptographie car elles transposent le problème du logarithme discret sur une courbe dans un corps fini. De fait, l'intérêt qui leur est porté ne cesse d'évoluer et de se renforcer au fil des années. Au départ, ces applications ont été un outil pour compromettre la sécurité de systèmes basée sur le logarithme discret pour certaines courbes [96], [68]. Aujourd'hui, elles suscitent un grand intérêt car elles permettent de concevoir de nouvelles primitives avec de multiples fonctionnalités. Aussi, nous ne pouvons manquer d'introduire les outils mathématiques permettant de définir ces applications bilinéaires. Nous présenterons seulement les éléments clés permettant d'aboutir à leur construction. Cette partie s'appuie essentiellement sur le livre de Neal Koblitz [89]. D'autres ouvrages plus complets peuvent être consultés, comme celui de Menezes [95], de Blake, Seroussi et Smart [24], ou de Enge [66].

Sommaire

3.1 Courbes elliptiques sur les corps finis

3.1.1 Définitions

Soit K un corps fini. Considérons l'équation de Weierstrass sous sa forme affine :

$$E : Y^2 + a_1XY + a_3Y - (X^3 + a_2X_2^2 + a_4X + a_6) = 0 \text{ où les } a_i \in K.$$

Une courbe elliptique sur K est une cubique non singulière (de déterminant non nul) définie par l'ensemble des solutions (x, y) d'une équation de Weierstrass, auxquelles on ajoute le point à l'infini $\mathcal{O}$. Plus formellement, on a :

$$E(K) \stackrel{\text{déf}}{=} \{(x, y) \in K^2 \mid y^2 + a_1xy + a_3y - (X^3 + a_2x_2^2 + a_4x + a_6) = 0\} \cup \{\mathcal{O}\}.$$

Si la caractéristique de K est différente de 2 ou de 3, la courbe peut alors être définie par l'équation :

$$E : Y^2 = X^3 + aX + b, \text{ pour } a, b \in K,$$

et le discriminant Δ est alors égal à $\Delta \stackrel{\text{déf}}{=} 16(4a^3 + 27b^2)$.

Nous notons $K[E] = K[X, Y]/(E)$ l'anneau des coordonnées des fonctions polynomiales et $K(E) = K(X)[Y]/(E) = \{a(X) + b(X)Y : a, b \in K(X)\}$ le corps des fonctions rationnelles de E dans $K \cup \{\infty\}$. La valeur ∞ est atteinte lorsque la fonction a un pôle en un point. Par la suite, nous prendrons $K = \mathbb{F}_q$, où q est une puissance d'un nombre premier p, c.à.d. $q = p^k$.

3.1.2 Loi de groupe

Les courbes elliptiques bénéficient d'une structure de groupe abélien possédant une interprétation géométrique : la somme de deux points est obtenue en appliquant la méthode dite de *corde et tangente*. Cette structure est construite à partir de l'idée générale suivante : "la somme des points d'intersection d'une droite et de la courbe est le point à l'infini". Plus exactement, pour tout point P et Q de la courbe, avec P ayant pour coordonnées affines (x, y), on a :

1. $\mathcal{O} + P = P$ et $P + \mathcal{O} = P$;
2. $\mathcal{O} = -\mathcal{O}$;
3. $-P$ a pour coordonnées $(x, -y)$;
4. si $P = -Q$, alors $P + Q = \mathcal{O}$;

5. si P et Q sont deux points distincts et différents du point à l'infini et si $Q \neq -P$, alors le troisième point de la courbe R (où la multiplicité est prise en compte) de la droite (PQ) si $P \neq Q$, ou de la tangente à la courbe en P si $P = Q$ est tel que : $P + Q = -R$;
6. la loi + est associative dans E. Ce résultat non trivial provient essentiellement du théorème de Riemann-Roch [81], [118].

Afin d'utiliser les courbes pour construire des cryptosystèmes, il est primordial de connaître précisément le nombre de points de la courbe. En effet, sa connaissance permet, en outre de se prémunir des attaques contre le problème du logarithme discret. Considérons l'application de Frobenius définie sur $E(\bar{\mathbb{F}}_q)$:

$$\pi : (x, y) \mapsto (x^q, y^q).$$

Remarquons que π est un homomorphisme de groupe qui laisse invariants les points à coordonnées dans $\mathbb{F}_q$, autrement dit on a : $Ker(\pi - id) = E(\mathbb{F}_q)$ et donc $|E(\mathbb{F}_q)| = |Ker(\pi - id)|$. On note t la trace de cette application.

Considérons l'endomorphisme de $E(\bar{\mathbb{F}}_q)$ défini par $[m] : P \mapsto mP$. Le noyau de $[m]$ est défini par $E[m] \stackrel{\text{déf}}{=} \{P(x, y) \in E(\bar{\mathbb{F}}_q) : \ mP = 0\}$. On dit que $E[m]$ est l'ensemble des points de m-torsions. Il a été montré que si la caractéristique de $\mathbb{F}_q$ ne divise pas m alors $|E[m]| = m^2$. Un corollaire de ce résultat est le suivant :

Théorème 2 *Soit E une courbe elliptique définie sur $\mathbb{F}_q$. Alors $E(\mathbb{F}_q) \simeq \mathbb{Z}/n_1\mathbb{Z} \times \mathbb{Z}/n_2\mathbb{Z}$, avec $n_1 \mid n_2$ ou $E(\mathbb{F}_q) \simeq \mathbb{Z}/n_1\mathbb{Z}$.*

Le nombre de points de la courbe est donné par le théorème suivant :

Théorème 3 (Théorème de Hasse) *Soit E une courbe elliptique définie sur $\mathbb{F}_q$. On a :*

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

Sa connaissance est très utile et possède de nombreuses applications. En théorie des codes correcteurs d'erreur par exemple, elle permet d'apprécier la qualité du code que l'on peut créer à partir d'une courbe. Depuis l'invention des premiers cryptosystèmes basés sur les courbes elliptiques, plusieurs algorithmes de calcul du nombre de points d'une courbe ont été proposés. Nous renvoyons à la thèse de Gaudry [70] pour les algorithmes de comptage de points. Afin d'expliciter la construction de couplages sur une courbe, nous avons besoin de quelques outils mathématiques. Commençons par la notion de diviseur : un diviseur de $E(\mathbb{F}_q)$ est une somme formelle $\mathcal{D}$ finie de points de la courbe E. On écrit :

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

où les n_P sont des éléments de $\mathbb{Z}$ presque tous nuls. On définit le degré de $\mathcal{D}$, $\deg \mathcal{D}$ et le support de $\mathcal{D}$, $\mathsf{supp}(\mathcal{D})$ respectivement par :

$$\deg \mathcal{D} = \textstyle\sum_{P \in E(\mathbb{F}_q)} n_P \quad \text{et} \quad \mathsf{supp}(D) = \{P \mid n_P \neq 0\}.$$

Nous définissons le groupe des diviseurs sur E que l'on note $\mathcal{D}iv$. Il s'agit d'un groupe abélien libre engendré par les points de la courbe. Par la suite, nous considérons le sous-groupe des diviseurs de degré 0, que l'on note $\mathcal{D}iv_0$.

Diviseurs d'une fonction : Étant donnée une fonction f de $E(\mathbb{F}_q)$ (vue comme une fonction rationnelle de $\mathbb{F}_q(x,y)$), on définit le diviseur de f, $\mathsf{div}(f)$ par :

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

où ord_P est l'ordre du zéro admis au point P par f.

Soit f une fonction de $E(\mathbb{F}_q)$, nous pouvons construire un diviseur de degré 0 (en considérant la différence des sommes formelles des zéros et des pôles de f). **Exemple :** considérons par exemple la courbe $E : y^2 = x^3 + 1$ et $f(x,y) = ax + by + c$. Supposons que la droite d'équation $f(x,y) = 0$ passe par deux points distincts P et Q tels que $P \neq -Q$ et $P, Q \in E(\mathbb{F}_q)$. Alors cette droite intersecte la courbe E en un troisième point $Q \in E(\mathbb{F}_q)$. La fonction $f(x,y)$ admet trois zéros P, Q et R et un pôle d'ordre 3 à l'infini. On obtient alors : $\mathsf{div}(f) = P + Q + R - 3\mathcal{O}$.

Soient E une courbe définie sur $\mathbb{F}_q$ et $\mathcal{B}$ un diviseur sur E. S'il existe une fonction $f \in \bar{\mathbb{F}_q}(E)$ telle que $\mathcal{B} = \mathsf{div}(f)$[1], on dit que $\mathcal{B}$ est un diviseur principal ; nous le notons (f). Nous avons la caractérisation suivante : un diviseur $\mathcal{B} = \sum_{P \in E(\mathbb{F}_q)} n_P(P)$ est principal si et seulement si

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

On dit que $\mathcal{B}$ et $\mathcal{A}$ sont deux diviseurs équivalents si la différence $\mathcal{B} - \mathcal{A}$ est un diviseur principal.

Remarque 2 *Remarquons que tout diviseur $\mathcal{B}$ de $\mathcal{D}\mathrm{iv}_0$, tel que $\mathcal{B} = \sum_{P \in E(\mathbb{F}_q)} n_P(P)$ et $\sum_{P \in E(\mathbb{F}_q)} n_P = 0$ est équivalent à un diviseur de la forme $\mathcal{B}' = (Q) - (\mathcal{O})$, pour un certain point de la courbe $Q \in E$. Dans ce cas, on a : $Q = \sum_{P \in E(\mathbb{F}_q)} n_P P$.*

Pour toute fonction $f \in K(E)$ et $P \in E$ n'appartenant pas au support de f, nous pouvons évaluer f en P (en substituant les indéterminés par les coordonnées de P pour toute fonction rationnelle de f). Nous pouvons également évaluer f en un diviseur $\mathcal{D}$ de support disjoint à f, en définissant :

$$f(\mathcal{D}) = \prod_{P \in \mathsf{supp}(D)} f(P)^{n_P}$$

.

3.1.3 Couplages sur une courbe elliptique

Nous pouvons désormais définir le couplage de Weil : il s'agit d'une fonction bilinéaire du groupe de torsion $E[n]$ vers le groupe multiplicatif des racines $n_{\text{ième}}$ de l'unité dans $\bar{\mathbb{F}_q}^*$. Étant donnés deux points P et Q de $E[n]$, tels que $\mathsf{pgcd}(p,n) = 1$, où p est la caractéristique de $\mathbb{F}_q$, il s'agit de déterminer deux fonctions rationnelles f_P et f_Q telles que $(f_P) = n(P) - n(\mathcal{O})$ et $f_Q = n(Q) - n(\mathcal{O})$ puis d'évaluer la fonction e_n définie par

$$e_n(P,Q) = \frac{f_P(Q)}{f_Q(P)}$$

Considérons $\mathcal{B}_P$ et $\mathcal{B}_Q$ des diviseurs équivalents aux diviseur $(P) - (\mathcal{O})$ et $(Q) - (\mathcal{O})$ respectivement. On sait que $n\mathcal{B}_P$ et $n\mathcal{B}_Q$ sont des diviseurs principaux (puisque $n(P) -$

[1] cela signifie que $\mathcal{B}$ est un diviseur d'une fonction de $E(\mathbb{F}_q)$.

$n(\mathcal{O})$ et $n(P) - n(\mathcal{O})$ sont des diviseurs principaux), il existe donc une fonction f_P (resp. f_Q) telle que $(f_P) = n\mathcal{B}_P$ (resp. $(f_Q) = n\mathcal{B}_Q$). Le couplage de Weil des points P et Q est alors défini par :

$$\begin{aligned} e_n : \quad & E[n] \times E[n] \to \mu_n \\ & (P, Q) \mapsto e_n(P, Q) = \tfrac{f_P(\mathcal{B}_Q)}{f_Q(\mathcal{B}_P)} \end{aligned}$$

Notons que ce quotient est défini à condition que le dénominateur ne s'annule pas. Afin d'alléger les notations, nous noterons la fonction bilinéaire définie ci-dessus par e au lieu de e_n. On peut montrer que cette fonction est bien définie, c.à.d. que la valeur de $e(P, Q)$ ne dépend pas du choix des diviseurs $\mathcal{B}_P$ et $\mathcal{B}_Q$ modulo la classe d'équivalence. En effet, soient $\hat{\mathcal{B}}_P$ et $\hat{\mathcal{B}}_Q$ deux diviseurs équivalents à $\mathcal{B}_P$ et $\mathcal{B}_Q$ respectivement et soit $\hat{f}_P$ une fonction telle que $(\hat{f}_P) = n\hat{\mathcal{B}}_P$. Alors il existe une fonction g telle que $\hat{\mathcal{B}}_P = \mathcal{B}_P + (g)$ et $\hat{f}_P = f_P \cdot g^n$. On a alors :

$$\begin{aligned} e(P, Q) &= \frac{\hat{f}_P(\mathcal{A}_Q)}{f_Q(\hat{\mathcal{B}}_P)} = \frac{f_P(\mathcal{B}_Q) \cdot g(\mathcal{A}_Q)^n}{f_Q(\mathcal{B}_P) \cdot f_Q(g)} && (3.1) \\ &= \frac{f_P(\mathcal{B}_Q)}{f_Q(\mathcal{B}_P)} \cdot \frac{g(n\mathcal{A}_Q)}{f_Q((g))} = \frac{f_P(\mathcal{B}_Q)}{f_Q(\mathcal{B}_P)} \cdot \frac{g((f_Q))}{f_Q((g))} && (3.2) \\ &= \frac{f_P(\mathcal{B}_Q)}{f_Q(\mathcal{B}_P)} \text{ par réciprocité, on a } f((g)) = g((f)). && (3.3) \end{aligned}$$

Pour tous points P et Q de $E[n]$, on a les propriétés suivantes (voir [118] pour une démonstration de ces propriétés) :

1. *bilinéarité :* $e(P_1 + P_2, Q) = e(P_1, Q) \cdot e(P_2, Q)$ et $e(Q, P_1 + P_2) = e(Q, P_1) \cdot e(Q, P_2)$;
2. *identité :* $\forall\ P \in E[n]$, on a : $e(P, P) = 1$;
3. *racine $n_{ième}$:* $\forall\ P, Q \in E[n]$, on a : $e(P, Q)^n = 1$;
4. *le couplage est alterné :* $\forall\ P, Q \in E[n]$, on a : $e(P, Q) = e(Q, P)^{-1}$;
5. *le couplage est non dégénéré :* si pour $P \in E[n]$, $\quad e(P, Q) = 1$ pour tout $Q \in E[n]$, alors $P = \mathcal{O}$.

En cryptographie, nous nous intéressons à un type particulier de fonctions bilinéaires, les fonctions bilinéaires admissibles. Plus exactement, une fonction bilinéaire e de $\mathbb{G}_1 \times \mathbb{G}_2$ dans $\mathbb{G}_T$, où $\mathbb{G}_1, \mathbb{G}_2$ et $\mathbb{G}_T$ sont trois groupes d'ordre premier p, est dite admissible si elle vérifie les propriétés suivantes :

1. bilinéarité : pour tout $P_1 \in \mathbb{G}_1$, $P_2 \in \mathbb{G}_2$ et pour tout entier $a, b \in \mathbb{Z}$, on a : $e(P_1^a, P_2^b) = e(P_1, P_2)^{a \cdot b}$;
2. non-dégénérescence : pour tout générateur P_1 et P_2 de $\mathbb{G}_1$ et $\mathbb{G}_2$ respectivement, on a : $e(P_1, P_2) \neq 1$;
3. évaluable efficacement : pour tout $P_1 \in \mathbb{G}_1$, $P_2 \in \mathbb{G}_2$, on peut calculer $e(P_1, P_2)$ en temps polynomial.

Le couplage de Weil, introduit ci-dessus est un type particulier de fonctions bilinéaires, qui n'est pas admissible. En effet, la propriété 2 fait de ce couplage une application dégénérée. Pour certaines courbes, il est possible de modifier ce couplage pour le rendre admissible, c'est la cas des courbes supersingulières (courbes pour lesquelles $t \mid p$, où t est la trace du Frobenius et p la caractéristique du corps sur lequel est définie la courbe). Le degré MOV (relativement à un certain entier), que nous définissons ci-dessous, permet d'identifier ces courbes.

Définition 9 *On appelle degré MOV d'une courbe elliptique E de $\mathbb{F}_q$ relativement à un entier r le plus entier k tel que $E[r] = E(\mathbb{F}_{q^k})[r]$.*

Considérons une courbe supersingulière E de degré MOV k relativement à un entier m, où m est un entier premier qui divise $|E(\mathbb{F}_q)|$. Menezes, Okamoto et Vanstone ont montré que si $m \neq p$ et $m \neq q-1$ alors le degré de MOV k relativement à m est inférieur ou égal à 6. Supposons qu'il existe un automorphisme ψ de $E(\mathbb{F}_q^k)$, de telle sorte que si P est un générateur de $E(\mathbb{F}_q)[m]$, P et $\psi(P)$ sont aussi des générateurs de $E[m]$. On modifie le couplage de Weil, en définissant la fonction suivante :

$$\begin{aligned} \hat{e} : \quad & E[m] \times E[m] \to \mathbb{F}_{q^k}^{\star} \\ & (S,T) \mapsto \hat{e}(S,T) = e(S,\psi(T)) \end{aligned}$$

Par la suite, nous prendrons soin d'utiliser des fonctions bilinéaires possédant certaines propriétés. Outre les considérations d'efficacité (la fonction $\hat{e}$ doit être calculable en temps polynomial en le paramètre de sécurité), la fonction doit être non dégénérée. Pour obtenir de telles fonctions, nous considérons les points suivants :

1. on se donne deux groupes $\mathbb{G}_1$ et $\mathbb{G}_2$, où $\mathbb{G}_1$ est un sous-groupe de points de $E(\mathbb{F}_q)$ et $\mathbb{G}_2$ est un sous-groupe de $\mathbb{F}_{q^k}$, avec k le degré de MOV de la courbe. En général, on prend $\mathbb{G}_1$ contenant environ 2^{256} éléments et q^k d'environ 3072 bits ;
2. étant donnés P et Q dans $\mathbb{G}_1$, on doit pouvoir calculer $\hat{e}(P,Q)$ en temps polynomial.

Pour le deuxième point, Miller [97] a proposé un algorithme polynomial pour calculer les couplages de Weil et Tate en temps polynomial. Nous renvoyons à l'article de Miller [98] qui donne les étapes de cet algorithme. La complexité de cet algorithme est proportionnelle à $\log n \times \tau_{q,k}$, où $\tau_{q,k}$ est la complexité de la multiplication dans le corps $\bar{\mathbb{F}}_{q^k}$ et q, k sont respectivement la taille du corps de base et le degré de MOV.

Nous rappelons comment générer une fonction bilinéaire associée à des groupes d'un ordre connu n, où n est un entier sans facteurs carré et non divisible par 3. Cette construction a été proposé par Boneh et Franklin [31] :

1. trouver le plus petit l tel que $p = ln - 1$ est premier et $p = 2 \mod 3$;
2. on considère le groupe des points de la courbe super-singulière d'équation $E : y^2 = x^3 + 1$ défini sur $\mathbb{F}_p$. Le groupe formé des points de la courbe possède un sous-groupe d'ordre n, qu'on note $\mathbb{G}$;
3. soit $\mathbb{G}_T$ le sous-groupe de $\mathbb{F}_{p^2}$ d'ordre n. Afin d'obtenir une fonction bilinéaire $e : \mathbb{G} \times \mathbb{G} \mapsto \mathbb{G}_T$, on applique le couplage de Weil (version modifiée) à la courbe.

Un autre couplage, le couplage de Tate, a également été proposé. Il s'évalue de manière plus rapide sous certaines conditions. Deux articles plus récents, un premier de Koblitz et Menezes [90] et un second de Granger *et al.* [109] compare ces deux couplages et aboutissent à des conclusions différentes. Des travaux récents [10] tentent de clarifier les ambiguïtés relatives aux performances de ces deux couplages.

3.2 Utiliser les couplages

Très souvent, en cryptographie, on utilise les couplages tels qu'une boîte noire, mais la conception de cryptosystèmes à base de couplages ne peut être faîte indépendamment de l'aspect mathématique et algorithmique. Plus précisément, considérons une fonction bilinéaire admissible

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \mapsto \mathbb{G}_T$$

Type	Hacher dans $\mathbb{G}_2$	Petite représ.	$\exists$ Homomorp.	GénéraT. Poly
1 (petite car.)	Ok	No	Ok	No
1 (grande car.)	Ok	No	Ok	Ok
2	No	Ok	Ok	Ok
3	Ok	Ok	No	Ok

FIG. 3.1 – Propriétés associées aux différents types de couplage, où Ok (resp. No) signifie que la condition figurant en haut du tableau est vérifiée (resp. n'est pas vérifiée) pour le type de courbe indiqué à gauche du tableau.

De manière générale, nous faisons certaines (et parfois toutes) des hypothèses suivantes :
- on peut "hacher" des éléments dans $\mathbb{G}_2$;
- les éléments de groupe ont une petite représentation ;
- il existe un homomorphisme de groupe de $\mathbb{G}_2$ vers $\mathbb{G}_1$ calculable en temps polynomial ;
- on peut générer des paramètres $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ en temps polynomial par rapport à λ, où λ est le paramètre de sécurité ;
- on dispose de groupes de taille suffisante pour que le problème du logarithme discret soit difficile, c.à.d. que le nombre d'opérations moyennes pour le résoudre doit au moins être 2^λ (nombre d'exponentiations).

Mais en pratique, ce n'est pas si simple car ces hypothèses ne sont en fait jamais toutes simultanément vérifiées. Ce point est argumenté dans [69][2]. Plus précisément, considérons $\mathbb{G}_1$ un sous-groupe de $\mathbb{F}_q$, $\mathbb{G}_2$ un sous-groupe de points de la courbe $E(\mathbb{F}_{q^k})$ et $\mathbb{G}_T$ un sous-groupe de $\mathbb{F}^*_{q^k}$ (on suppose que ces groupes sont tous de même ordre l). Les paramètres clés auxquels nous nous intéressons sont la taille du corps de base q, le degré de l'extension k et l'ordre des groupes l. Nous considérons figure 3.1 les hypothèses énumérées précédemment pour les trois choix de groupe figurant ci-dessous :

1. $\mathbb{G}_1 = \mathbb{G}_2$;
2. $\mathbb{G}_1 \neq \mathbb{G}_2$, et il existe un homomorphisme de $\mathbb{G}_2$ vers $\mathbb{G}_1$ calculable en temps polynomial ;
3. $\mathbb{G}_1 \neq \mathbb{G}_2$, et il n'existe pas d'homomorphisme calculable de manière efficace entre $\mathbb{G}_1$ et $\mathbb{G}_2$.

Comme on peut le constater, l'existence d'un algorithme polynomial pour la génération des paramètres n'est pas automatique. En pratique, on peut s'arranger pour choisir des paramètres offrant une flexibilité suffisante et un niveau de sécurité approprié. Mais malheureusement, dans ce cas, les schémas obtenus comme nous le verrons, sont rarement efficaces. Par conséquent, nous devons la plupart du temps faire un compromis entre les contraintes d'implémentations et la flexibilité du schéma imposant alors un type particulier de courbes. Par exemple, selon les données du *National Institute of Standards and Technology*, NIST recommande au moins 128 bits de sécurité ; dans ce cas, on doit considérer une taille de groupe de 2^{256} bits. Ces tailles dictent donc les performances d'un schémas. Une estimation de cet impact est donnée en [15], où les auteurs disent moins coûteux de faire quatre exponentiations dans un groupe de taille 160 bits plutôt qu'une exponentiation dans un groupe de taille 256.

[2]Cet article a été rédigé dans le cadre du projet européen `ECRYPT`.

3.3 Hypothèses calculatoires dans les groupes bilinéaires

Désormais, par commodité, nous passons en notation multiplicative pour les définitions des hypothèses calculatoires et la description des schémas à base de couplages. Pour les différentes hypothèses algorithmiques que nous introduisons ici, certaines d'entre elles dépendent d'un paramètre, nous dirons alors que l'hypothèse est non statique. Nous définissons $\mathcal{KG}_{\mathsf{bil}}$ un algorithme (probabiliste) de génération de paramètres publics qui étant donné un paramètre de sécurité λ, renvoie la description d'une structure bilinéaire constituée d'une fonction bilinéaire e, deux groupes $\mathbb{G}$ et $\mathbb{G}_T$ de même ordre et un générateur de $\mathbb{G}$, noté g de telle sorte que la fonction $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ soit une fonction bilinéaire admissible. On parlera d' "instance bilinéaire", par abus de langage et par la suite nous écrirons $\mathsf{params} \stackrel{\text{déf}}{=} \langle g, p, \mathbb{G}, \mathbb{G}_T, e\rangle$. De manière générale, nous n'exigeons pas que la structure bilinéaire soit asymétrique. Nous préciserons ce point si cette hypothèse est nécessaire.

3.3.1 Les problèmes non paramétrés ou "statiques"

Le problème décisionnel du sous-groupe

Cette hypothèse a été introduite par Boneh, Goh et Nissim [32] en 2005. Elle a donné naissance à un cryptosystème homomorphe (additivement et une fois multiplicativement) avec de nombreuses applications. Nous le décrivons figure 3.3. Étant donné un paramètre de sécurité λ, la génération est transposée dans un groupe d'ordre un entier n composé ; plus exactement, l'algorithme probabiliste $\mathcal{KG}_{\mathsf{bil}}$ génère un t-uplet de paramètres $\mathsf{params} \stackrel{\text{déf}}{=} \langle g, n, \mathbb{G}, \mathbb{G}_T, e\rangle$, avec $\mathbb{G}$ et $\mathbb{G}_T$ deux groupes d'ordre n tel que $n = p \cdot q$, avec p, q tous deux premiers et g un générateur de $\mathbb{G}$ et $e : \mathbb{G} \times \mathbb{G} \mapsto \mathbb{G}_T$ une fonction bilinéaire admissible.

Nous considérons le problème suivant : étant donnés la description d'une "structure bilinéaire composée" définie par params telle que $\mathsf{params} \stackrel{\text{déf}}{=} \langle g, n, \mathbb{G}, \mathbb{G}_T, e\rangle$ et un élément $u \in \mathbb{G}$, décider si u est d'ordre q dans $\mathbb{G}$. On définit l'avantage de $\mathcal{A}$ à distinguer la distribution uniforme de $\mathbb{G}$ de celle du sous-groupe $\mathbb{G}_q$ de $\mathbb{G}$, d'ordre q :

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{subD}}_{\mathcal{A}}(\lambda) &= \Pr\left[\mathcal{A}(\mathsf{params}, U) = 1 : \begin{array}{l}\mathsf{params} \leftarrow \mathcal{KG}_{\mathsf{bil}},\ \text{où}\\ \mathsf{params} \stackrel{\text{déf}}{=} \langle g, n, \mathbb{G}, \mathbb{G}_T, e\rangle; U \stackrel{R}{\leftarrow} \mathbb{G}_q\end{array}\right]\\
&- \Pr\left[\mathcal{A}(\mathsf{params}, U) = 1 : \begin{array}{l}\mathsf{params} \leftarrow \mathcal{KG}_{\mathsf{bil}}(\lambda),\ \text{où}\\ \mathsf{params} \stackrel{\text{déf}}{=} \langle g, n, \mathbb{G}, \mathbb{G}_T, e\rangle; U \stackrel{R}{\leftarrow} \mathbb{G}\end{array}\right]
\end{aligned}$$

On dit que l'hypothèse décisionnelle du *sous-groupe* est (λ, t, ϵ)-valide s'il n'existe pas d'algorithme $\mathcal{A}$ en temps t tel que l'avantage $\mathsf{Adv}^{\mathsf{subD}}_{\mathcal{A}}(\lambda)$ soit au moins ϵ.

Le problème Linéaire décisionnel, DLIN

Cette hypothèse a été introduite en 2004 par Boneh, Boyen et Shacham [30]. À l'origine, elle est définie dans un groupe $\mathbb{G}$ cyclique d'ordre q premier. Cette hypothèse restant valide même si résoudre le problème DDH est facile, nous l'introduisons dans cette section ; elle est d'ailleurs très souvent utilisée dans les schémas avec couplages. Pour la définir d'une façon générique, nous considérons l'algorithme $\mathcal{KG}_{\mathsf{LD}}$ qui génère un t-uplet $\mathsf{params} \stackrel{\text{déf}}{=} \langle g, q, \mathbb{G}\rangle$, avec $\mathbb{G}$ d'ordre premier q et g un générateur de $\mathbb{G}$. Soient u, v, w des éléments aléatoires de $\mathbb{G}$ et U, V, W des éléments de $\mathbb{G}$ tels que $U = u^a, V = v^b, W = w^c$, décider si $a + b = c$

mod q. On définit l'avantage de $\mathcal{A}$ à décider le problème linéaire décisionnel par :

$$\mathsf{Adv}^{\mathsf{dlin}}_{\mathcal{A}}(\lambda) = \left| \Pr\left[\mathcal{A}(\mathsf{params}, u, v, w, u^a, v^b, w^{a+b}) = 1 : \begin{array}{l} \mathsf{params} \leftarrow \mathcal{KG}_{\mathsf{LD}}(\lambda), \text{ où} \\ \mathsf{params} \stackrel{\text{déf}}{=} \langle g, q, \mathbb{G}\rangle; \\ u, v, w \stackrel{R}{\leftarrow} \mathbb{G}; a, b \stackrel{R}{\leftarrow} \mathbb{Z}_q^* \end{array} \right] - \Pr\left[\mathcal{A}(\mathsf{params}, u, v, w, u^a, v^b, w^c) = 1 : \begin{array}{l} \mathsf{params} \leftarrow \mathcal{KG}_{\mathsf{DL}}(\lambda), \text{ où} \\ \mathsf{params} \stackrel{\text{déf}}{=} \langle g, q, \mathbb{G}\rangle; \\ u, v, w \stackrel{R}{\leftarrow} \mathbb{G}; a, b, c \stackrel{R}{\leftarrow} \mathbb{Z}_q^* \end{array} \right] \right|$$

De même, on dit que l'hypothèse linéaire décisionnelle est (λ, t, ϵ)-valide s'il n'existe pas d'algorithme $\mathcal{A}$ en temps t tel que l'avantage $\mathsf{Adv}^{\mathsf{dlin}}_{\mathcal{A}}(\lambda)$ soit au moins ϵ. Nous définissions le langage des t-uplets linéaires par l'ensemble $\mathcal{L}_{\mathsf{DLIN}_{\mathbb{G}}}$

$$\mathcal{L}_{\mathsf{DLIN}_{\mathbb{G}}} \stackrel{\text{déf}}{=} \{(u, v, w, u^a, v^b, w^c) \mid a, b \stackrel{R}{\leftarrow} \mathbb{Z}_q,\ c = a + b \mod q\}.$$

Le problème du Diffie-Hellman Bilinéaire Calculatoire, CBDH

Cette hypothèse a été introduite par Boneh et Franklin [31] en 2001. Elle est très populaire et reste valide dans les groupes munis d'une structure bilinéaire asymétrique. Étant donné un paramètre de sécurité λ, l'algorithme $\mathcal{KG}_{\mathsf{bil}}$ génère les paramètres $\mathsf{params} \stackrel{\text{déf}}{=} (u, v, q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, avec u et v des générateurs de $\mathbb{G}_1$ et $\mathbb{G}_2$ respectivement. Désormais, on suppose que l'ordre des groupes, q est premier. Le problème du Diffie-Hellman bilinéaire calculatoire, $\mathsf{CBDH}_{\mathbb{G}_1,\mathbb{G}_2}$ est le suivant : étant donné $u, v, A = u^a, B = u^b, C = v^c$, calculer $e(u, v)^{abc} \in \mathbb{G}_T$. Nous définissons la probabilité de succès de l'attaquant par :

$$\mathsf{Succ}^{\mathsf{cbdh}}_{\mathcal{A}}(\lambda) = \Pr\left[\mathcal{A}(\mathsf{params}, u^a, u^b, v^c) = e(u, v)^{abc} : \begin{array}{l} \mathsf{params} \leftarrow \mathcal{KG}_{\mathsf{bil}}(\lambda) \text{ où} \\ \mathsf{params} \stackrel{\text{déf}}{=} \langle u, v, q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e\rangle; \\ a, b, c \stackrel{R}{\leftarrow} \mathbb{Z}_q^* \end{array} \right]$$

On dit que l'hypothèse $\mathsf{CBDH}_{\mathbb{G}_1,\mathbb{G}_2}$ est (λ, t, ϵ)-valide si pour tout attaquant $\mathcal{A}$ en temps t, la probabilité de succès de l'attaquant $\mathcal{A}$ est au plus ϵ.

Le problème du Diffie-Hellman Bilinéaire décisonnel, DBDH

La version décisionnelle du problème précédent peut également être difficile. L'algorithme $\mathcal{KG}_{\mathsf{bil}}$ génère les paramètres $\mathsf{params} \stackrel{\text{déf}}{=} \langle q, g, \mathbb{G}, \mathbb{G}_T, e\rangle$. Le problème DBDH est le suivant : soient $u, v, A = u^a, B = u^b, C = v^c \in \mathbb{G}$, et $V \in \mathbb{G}_T$, décider si $V = e(u, v)^{abc}$ ou un élément aléatoire de $\mathbb{G}_T$. On définit l'avantage de $\mathcal{A}$ à résoudre le problème DBDH par :

$$\mathsf{Adv}^{\mathsf{dbdh}}_{\mathcal{A}}(\lambda) = \left| \Pr\left[\mathcal{A}(\mathsf{params}, u^a, u^b, v^c, V) = 1 : \begin{array}{l} \mathsf{params} \leftarrow \mathcal{KG}_{\mathsf{bil}}(\lambda), \text{ où} \\ \mathsf{params} \stackrel{\text{déf}}{=} \langle q, g, \mathbb{G}, \mathbb{G}_T, e\rangle; \\ a, b, c \stackrel{R}{\leftarrow} \mathbb{Z}_q; V = e(u, v)^{abc} \end{array} \right] - \Pr\left[\mathcal{A}(\mathsf{params}, u^a, u^b, v^c, V) = 1 : \begin{array}{l} \mathsf{params} \leftarrow \mathcal{KG}_{\mathsf{bil}}(\lambda), \text{où} \\ \mathsf{params} \stackrel{\text{déf}}{=} \langle q, g, \mathbb{G}, \mathbb{G}_T, e\rangle; \\ a, b, c \stackrel{R}{\leftarrow} \mathbb{Z}_q; V \stackrel{R}{\leftarrow} \mathbb{G}_T \end{array} \right] \right|$$

On dit que l'hypothèse $\mathsf{DBDH}_{\mathbb{G}}$ est (λ, t, ϵ)-valide s'il n'existe pas d'algorithme $\mathcal{A}$ en temps t capable de résoudre le problème $\mathsf{DBDH}_{\mathbb{G}}$ avec un avantage au moins ϵ.

3.3.2 Les problèmes paramétrés ou "non statiques"

Le problème l-BDHI

L'hypothèse *Diffie-Hellman bilinéaire inverse non statique*, l-BDHI est un variante plus forte que le CBDH qui a été introduite par Boneh et Boyen [25] en 2004. Nous ne nous servirons pas directement de cette hypothèse mais d'une variante définie juste après. Nous choisissons, tout de même de l'introduire car elle est à la source d'un grand nombre de constructions dans le modèle standard avec des tailles de paramètres relativement courtes. Le problème est le suivant : étant donné un paramètre de sécurité λ, l'algorithme $\mathcal{KG}_{\mathsf{bil}}$ génère des paramètres publics $\mathsf{params} \stackrel{\text{déf}}{=} \langle g, q, \mathbb{G}, \mathbb{G}_T, e\rangle$. Considérons une séquence $g, g^x, \cdots, g^{x^l} \in \mathbb{G}^{l+1}$, et un élément $V \in \mathbb{G}_T$, où l est un paramètre et $x \stackrel{R}{\leftarrow} \mathbb{Z}_q$; résoudre le problème l-$\mathsf{BDHI}_{\mathbb{G},\mathbb{G}_T}$ signifie décider si $V = e(g,g)^{1/x}$ ou si V est un élément aléatoire dans $\mathbb{G}_T$. On définit la probabilité de succès de l'attaquant $\mathcal{A}$ à résoudre le problème l-$\mathsf{BDHI}_{\mathbb{G}}$ par :

$$\begin{aligned}\mathsf{Adv}^{l\text{-bdhi}}_{\mathcal{A}}(\lambda) \;=\; & \Big|\Pr\Big[\mathcal{A}(\mathsf{params}, g, g^x, \cdots, g^{x^l}, V) = 1 : \begin{array}{l}\mathsf{params} \leftarrow \mathcal{KG}_{\mathsf{bil}}(\lambda), \text{ où}\\ \mathsf{params} \stackrel{\text{déf}}{=} \langle q, g, \mathbb{G}, \mathbb{G}_T, e\rangle;\\ x \stackrel{R}{\leftarrow} \mathbb{Z}_q; V = e(u,v)^{1/x}\end{array}\Big] \\ - \;& \Pr\Big[\mathcal{A}(\mathsf{params}, g, g^x, \cdots, g^{x^l}, V) = 1 : \begin{array}{l}\mathsf{params} \leftarrow \mathcal{KG}_{\mathsf{bil}}(\lambda), \text{ où}\\ \mathsf{params} \stackrel{\text{déf}}{=} \langle q, g, \mathbb{G}, \mathbb{G}_T, e\rangle;\\ x \stackrel{R}{\leftarrow} \mathbb{Z}_q; V \stackrel{R}{\leftarrow} \mathbb{G}_T\end{array}\Big]\Big|\end{aligned}$$

On dit que l'hypothèse l-$\mathsf{BDHI}_{\mathbb{G}}$ est (λ, t, ϵ)-valide s'il n'existe pas d'algorithme $\mathcal{A}$ en temps t capable de résoudre le problème l-$\mathsf{BDHI}_{\mathbb{G}}$ avec une probabilité de succès au moins égale à ϵ. Le problème décisionnel associé se définit de manière similaire.

Le problème l-ABDHE

Le problème Diffie-Hellman bilinéaire à la puissance ajoutée *non statique*, l-ABDHE a été introduit en 2006 par Gentry [71] pour démontrer la sécurité d'un schéma à base de couplages (dans le modèle standard) décrit figure 4.3. Nous décrivons la simulation associée au paragraphe 4.2.2. Elle peut être définie de la manière suivante : étant donné un paramètre de sécurité λ, l'algorithme $\mathcal{KG}_{\mathsf{bil}}$ génère des paramètres publics $\mathsf{params} \stackrel{\text{déf}}{=} \langle g, q, \mathbb{G}, \mathbb{G}_T, e\rangle$. Considérons une séquence $g', g'^{x^{l+2}}, g, g^x, g^{x^2}, \cdots, g^{x^l}, g^{x^{l+2}}, \cdots, g^{x^{2\cdot l}}$ et un élément V de $\mathbb{G}_T$, où l est un paramètre, g un élément aléatoire de $\mathbb{G}$ et x un scalaire aléatoire dans $\mathbb{Z}_q$; résoudre le problème l-$\mathsf{ABDHE}_{\mathbb{G},\mathbb{G}_T}$ signifie décider si $V = e(g^{x^{l+1}}, g')$ ou si V est un élément aléatoire de $\mathbb{G}_T$. En réalité, le schéma de Gentry repose sur une version plus faible du problème précédent pour laquelle la séquence consiste en les éléments $g', g'^{x^{l+2}}, g, g^x, \cdots, g^{x^l}$. Nous appelons ce problème, le problème Diffie-Hellman bilinéaire à la puissance ajoutée tronqué *non statique*, l-T-ABDHE. On définit l'avantage de l'attaquant

$\mathcal{A}$ à résoudre le problème l-T-ABDHE$_{\mathbb{G},\mathbb{G}_T}$ par :

$$\mathsf{Adv}_{\mathcal{A}}^{l\text{-t-abdhe}}(\lambda) =$$

$$\left| \Pr\left[\mathcal{A}(\mathsf{params}, g', g'^{x^{l+2}}, g^x, \cdots, g^{x^l}, V) = 1 : \begin{array}{l} \mathsf{params} \leftarrow \mathcal{KG}_{\mathsf{bil}}(\lambda), \text{ où} \\ \mathsf{params} \stackrel{\text{déf}}{=} \langle q, g, \mathbb{G}, \mathbb{G}_T, e\rangle; \\ g' \stackrel{R}{\leftarrow} \mathbb{G}; x \stackrel{R}{\leftarrow} \mathbb{Z}_q; V = e(g^{x^{l+1}}, g') \end{array} \right] \right.$$

$$\left. - \Pr\left[\mathcal{A}(\mathsf{params}, g', g'^{x^{l+2}}, g^x, \cdots, g^{x^l}, V) = 1 : \begin{array}{l} \mathsf{params} \leftarrow \mathcal{KG}_{\mathsf{bil}}(\lambda), \text{ où} \\ \mathsf{params} \stackrel{\text{déf}}{=} \langle q, g, \mathbb{G}, \mathbb{G}_T, e\rangle; \\ g' \stackrel{R}{\leftarrow} \mathbb{G}; x \stackrel{R}{\leftarrow} \mathbb{Z}_q; V \stackrel{R}{\leftarrow} \mathbb{G}_T \end{array} \right] \right|$$

On dit que l'hypothèse l-T-ABDHE$_{\mathbb{G},\mathbb{G}_T}$ est (λ, t, ϵ)-valide s'il n'existe pas d'algorithme $\mathcal{A}$ en temps t capable de résoudre le problème l-T-ABDHE$_{\mathbb{G},\mathbb{G}_T}$ avec une probabilité de succès d'au moins ϵ.

Gentry a défini une variante plus faible que le problème l-BDHI$_{\mathbb{G},\mathbb{G}_T}$ qui se réduit de manière presque évidente au problème l-T-ABDHE$_{\mathbb{G},\mathbb{G}_T}$, nous ne rentrerons pas en détail dans la définition de nouvelles hypothèses, nous reportons le lecteur à l'article [71] pour des précisions à ce sujet.

Le problème l-SP-DBDH

Nous introduisons un nouveau problème, le problème des *puissances successives*, *Successive-Power Version*, l-SP-DBDH, qui peut être vu comme une variante du problème DBDH. Comme pour le problème l-BDHI$_{\mathbb{G},\mathbb{G}_T}$, il donne accès à des puissances successives, comme pour le problème précédent. Plus précisément, nous définissons ce problème ainsi : étant donnés les paramètres $\mathsf{params} \stackrel{\text{déf}}{=} \langle g, q, \mathbb{G}, \mathbb{G}_T, e\rangle$ générés par $\mathcal{KG}_{\mathsf{bil}}$ et une séquence $, g^x, g^y, g^z, g^{z/x}, \cdots, g^{z/x^l}$, où l est un paramètre et $x, y \stackrel{R}{\leftarrow} \mathbb{Z}_q,\ z \stackrel{R}{\leftarrow} \mathbb{Z}_q^*$, décider si $V = e(g,g)^{x\cdot y\cdot z}$ ou si V est un élément aléatoire de $\mathbb{G}_T$. On définit l'avantage de l'attaquant à casser l'hypothèse l-SP-DBDH$_{\mathbb{G}}$ par :

$$\mathsf{Adv}_{\mathcal{A}}^{l\text{-spdbdh}} =$$

$$\left| \Pr\left[\mathcal{A}(\mathsf{params}, g^x, g^y, g^z, g^{z/x}, \cdots, g^{z/x^l}, V) = 1 : \begin{array}{l} \mathsf{params} \leftarrow \mathcal{KG}_{\mathsf{bil}}(\lambda), \text{ où} \\ \mathsf{params} \stackrel{\text{déf}}{=} \langle g, q, \mathbb{G}, \mathbb{G}_T, e\rangle; \\ x, y, z \stackrel{R}{\leftarrow} \mathbb{Z}_q^*; V = e(u,v)^{xyz} \end{array} \right] \right.$$

$$\left. - \Pr\left[\mathcal{A}(\mathsf{params}, g^x, g^y, g^z, g^{z/x}, \cdots, g^{z/x^l}, V) = 1 : \begin{array}{l} \mathsf{params} \leftarrow \mathcal{KG}_{\mathsf{bil}}(\lambda), \text{où} \\ \mathsf{params} \stackrel{\text{déf}}{=} \langle g, q, \mathbb{G}, \mathbb{G}_T, e\rangle; \\ x, y, z \stackrel{R}{\leftarrow} \mathbb{Z}_q^*; V \stackrel{R}{\leftarrow} \mathbb{G}_T \end{array} \right] \right|$$

Dans l'article [85], nous avons montré avec David Pointcheval que pour tout attaquant générique, une telle séquence ne fournit pas plus d'information exploitable que celle apportée par les trois premiers éléments. Nous renvoyons à l'article pour la preuve de ce résultat. Supposons la difficulté de ce problème admise, on dit alors que l'hypothèse l-SP-DBDH$_{\mathbb{G}}$ est (λ, t, ϵ)-valide, s'il n'existe pas d'algorithme $\mathcal{A}$ en temps t capable de résoudre le problème l-SP-DBDH$_{\mathbb{G}}$ avec un avantage au moins égal à ϵ.

3.3.3 Les problèmes co-CDH et Cco-CDH

Le problème co-CDH

Définition 10 *Soient g_1 et g_2 deux générateurs de $\mathbb{G}_1$ et $\mathbb{G}_2$ respectivement. Pour $u = g_1^x \in \mathbb{G}_1$, nous définissons le co-Diffie-Hellman* $\mathsf{co\text{-}CDH}_{g_1,g_2}(u)$*, l'élément $v = g_2^x \in \mathbb{G}_2$.*

Nous définissons alors le problème $\mathsf{co\text{-}CDH}_{\mathbb{G}_1,\mathbb{G}_2}$:

étant donnés $g_1, u \in \mathbb{G}_1$ et $g_2 \in \mathbb{G}_2$, retourner la valeur $v = \mathsf{co\text{-}CDH}_{g_1,g_2}(u)$.

La probabilité de succès à casser le problème $\mathsf{co\text{-}CDH}_{\mathbb{G}_1,\mathbb{G}_2}$ est définie par :

$$\mathsf{Succ}^{\mathsf{cocdh}}_{\mathbb{G}_1,\mathbb{G}_2}(\mathcal{A}) = \left|\Pr\left[\mathcal{A}(g_1,g_2,g_1^x) = g_2^x : \begin{array}{l}\langle g_i, q, \mathbb{G}_i\rangle \leftarrow \mathcal{KG}_{\mathsf{LD}}(\lambda), \text{ pour} \\ i = 1,2 \text{ et } x \overset{R}{\leftarrow} \mathbb{Z}_p\end{array}\right]\right|$$

Remarquons que lorsque $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$, le problème $\mathsf{co\text{-}CDH}_{\mathbb{G},\mathbb{G}}$ est exactement le traditionnel problème *Diffie-Hellman calculatoire* dans $\mathbb{G}$, qui peut être difficile dans ce contexte. Cependant, la version décisionnelle est facile compte tenu de la structure bilinéaire associée. Nous pouvons définir le langage $\mathsf{co\text{-}DH}_{\mathbb{G}_1,\mathbb{G}_2}$, le langage des quadruplets $(a,b,c,d) \in \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_1 \times \mathbb{G}_2$, tels que $d = \mathsf{co\text{-}CDH}_{a,b}(c)$.

Le problème du co-CDH commun, Cco-CDH

Remarquons que étant donnés deux éléments, on peut facilement construire un quadruplet $\mathsf{co\text{-}DH}_{\mathbb{G}_1,\mathbb{G}_2}$. Cependant, trouver deux quadruplets satisfaisant certaines contraintes peut parfois être difficile. Nous considérons alors un nouveau problème, le problème du co-CDH commun, que l'on note Cco-CDH (*Common* co-CDH *problem* en anglais). Nous le définissons ainsi : étant donnés des paramètres params générés par $\mathcal{KG}_{\mathsf{bil}}$ et des éléments $g, h \overset{R}{\leftarrow} \mathbb{G}$, et $V \in \mathbb{G}_T$, retourner $k_0 \neq k_1 \in \mathbb{Z}_q$, $K_0, K_1 \in \mathbb{G}_T$ et un élément $c \in \mathbb{G}$ commun tels que :

$$(gh^{k_0}, V, c, K_0), (gh^{k_1}, V, c, K_1) \in \mathsf{co\text{-}DH}_{\mathbb{G},\mathbb{G}_T}$$

La probabilité de succès de $\mathcal{A}$ à casser le problème du $\mathsf{co\text{-}CDH}_{\mathbb{G},\mathbb{G}_T}$ commun est définie par :

$$\mathsf{Succ}^{\mathsf{cocdh}}_{\mathbb{G},\mathbb{G}_T}(\mathcal{A}) =$$

$$\Pr\left[\begin{array}{ll}\mathcal{A}(\mathsf{params}, g, h, V) = (c, k_0, k_1, K_0, K_1) \text{ t.q.} & \mathsf{params} \leftarrow \mathcal{KG}_{\mathsf{bil}}(\lambda), \text{ où} \\ k_0 \neq k_1 \wedge (gh^{k_0}, V, c, K_0) \in \mathsf{co\text{-}DH}_{\mathbb{G},\mathbb{G}_T} \quad : & \mathsf{params} \overset{\text{déf}}{=} \langle g, q, \mathbb{G}, \mathbb{G}_T, e\rangle; \\ \wedge (gh^{k_1}, V, c, K_1) \in \mathsf{co\text{-}DH}_{\mathbb{G},\mathbb{G}_T} & g, h \overset{R}{\leftarrow} \mathbb{G}; V \overset{R}{\leftarrow} \mathbb{G}_T\end{array}\right]$$

On dit que l'hypothèse $\mathsf{Cco\text{-}CDH}_{\mathbb{G},\mathbb{G}_T}$ est (λ, t, ϵ)-valide s'il n'existe pas d'algorithme $\mathcal{A}$ en temps t capable de résoudre le problème $\mathsf{Cco\text{-}CDH}_{\mathbb{G},\mathbb{G}_T}$ avec une probabilité de succès au moins égale à ϵ.

Avec David Pointcheval, nous avons montré dans l'article [85] que l'hypothèse $\mathsf{Cco\text{-}CDH}_{\mathbb{G},\mathbb{G}_T}$ est valide pour tout attaquant générique.

3.4 Chiffrement homomorphes dans les groupes bilinéaires

Schéma de chiffrement linéaire

Le schéma a été introduit en 2004 par Boneh, Boyen et Shacham [30]. Il est défini dans un groupe $\mathbb{G}$ cyclique d'ordre premier. La sécurité sémantique de ce schéma repose sur

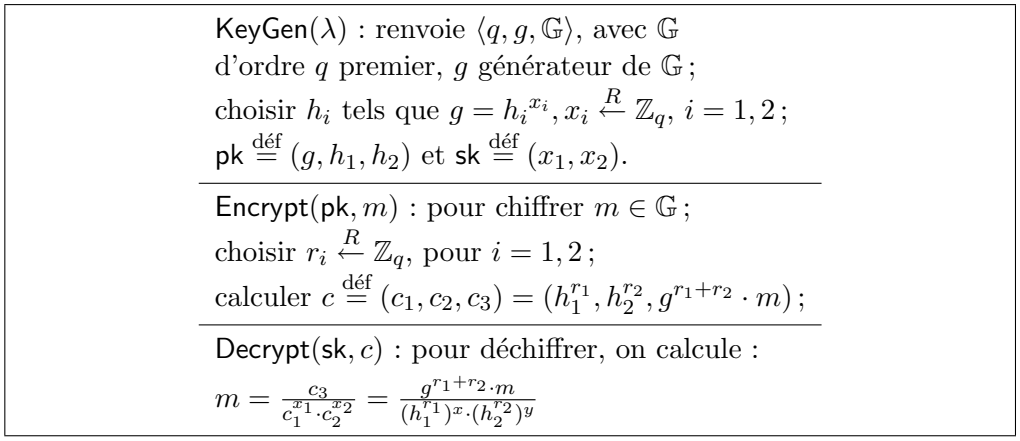


FIG. 3.2 – Schéma de chiffrement linéaire [30]

l'hypothèse décisionnelle linéaire dans $\mathbb{G}$. Nous donnons une description de ce cryptosystème figure 3.2. On peut voir que la fonction de chiffrement $\mathsf{Encrypt}$ est un homomorphisme multiplicatif de $\mathbb{G}$; en effet pour tout $m, m' \in \mathcal{M}$, on a :

$$\begin{aligned}
&\mathsf{Encrypt}(\mathsf{pk}, m; r_1, r_2) \cdot \mathsf{Encrypt}(\mathsf{pk}, m'; r'_1, r'_2) \\
&= (g_1^{r_1}, g_2^{r_2}, z^{r_1+r_2} \cdot m) \cdot (g_1^{r'_1}, g_2^{r'_2}, z^{r'_1+r'_2} \cdot m') \\
&= \mathsf{Encrypt}(\mathsf{pk}, m \cdot m'; r_1 + r'_1, r_2 + r'_2)
\end{aligned}$$

Chiffrement BGN

Ce système de chiffrement, introduit par Boneh, Goh et Nissim [32] est basé sur l'hypothèse du sous-groupe décisionnelle. Il s'agit d'une combinaison du cryptosystème de Paillier [105] et du cryptosystème de Okamoto-Uchiyama [104]. Nous donnons une description figure 3.3.

[1] h est un générateur du sous-groupe de $\mathbb{G}$ d'ordre p et si g ou h sont égaux à 1, on répète cette procédure jusqu'à obtenir deux générateurs non triviaux.

[2] Pour retrouver m, on peut utiliser la méthode de Pollard qui a une complexité en $\mathcal{O}(\sqrt{q})$.

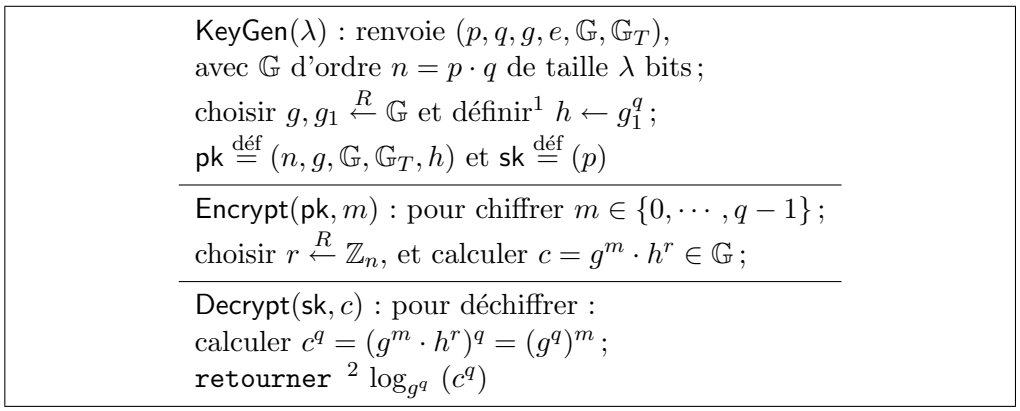


FIG. 3.3 – Schéma de chiffrement de Boneh, Goh et Nissim [32]

Propriétés homomorphiques : étant donné le chiffré de deux messages m et m', c et c' respectivement, il est possible de construire le chiffré de $m + m'$ en calculant $c \cdot c' \cdot h^r$ pour un élément r aléatoire dans $\mathbb{Z}_n$. De plus, on peut multiplier (une seule fois seulement) deux chiffrés $c = g^m \cdot h^r$ et $c' = g^{m'} \cdot h^{r'}$ de m et m' respectivement pour obtenir le chiffré de $m \cdot m'$, on calcule pour cela :

$$\begin{aligned} C &= e(c, c') \\ &= e(g^m \cdot h^r, g^{m'} \cdot h^{r'}) \\ &= G^{m \cdot m'} \cdot H^{mr + m'r' + a \cdot q \cdot r \cdot r'}, \end{aligned}$$

avec $G = e(g, g)$ et $H = e(g, h)$ et un certain entier a tel que $h = g^{a \cdot q}$.

L'aléa $\tilde{r} = mr + m'r' + a \cdot q \cdot r \cdot r'$ est uniformément distribué dans $\mathbb{Z}_n$. On a donc un chiffré de $m \cdot m' \mod n$ dans $\mathbb{G}_T$ de base G et H.

3.5 Chiffrement à base d'identité

La cryptographie à base d'identités permet de chiffrer un message pour un utilisateur, en utilisant son identité[3] comme clé publique, de telle sorte que le destinataire soit le seul capable de déchiffrer le message qui lui est destiné. Ce concept a été introduit en 1984 par Adi Shamir [115] dans le but de simplifier la gestion des clés publiques et leur certification : auparavant chaque utilisateur faisait appel à une autorité de certification hors ligne, lui délivrant une paire de clés certifiées pour une utilisation *a posteriori*. Désormais, un tel scénario simplifie considérablement la procédure de certification, en permettant une identification implicite.

3.5.1 Schéma de chiffrement à base d'identité, $\mathcal{IBE}$

Commençons par définir la primitive d'$\mathcal{IBE}$:

Définition 11 (Schéma de chiffrement basée sur l'identité) *Un schéma de chiffrement basée sur l'identité $\mathcal{IBE}$ est défini par la donnée de quatre algorithmes polynomiaux*

$$\langle \mathsf{Setup}_{\mathsf{IBE}}, \mathsf{Extract}_{\mathsf{IBE}}, \mathsf{Encrypt}_{\mathsf{IBE}}, \mathsf{Decrypt}_{\mathsf{IBE}} \rangle \text{ définis par :}$$

$\mathsf{Setup}_{\mathsf{IBE}}(\lambda)$ **:** *prend en entrée un paramètre de sécurité λ et renvoie les paramètres publics* mpk *et une clé maître secrète* msk *;*

$\mathsf{Extract}_{\mathsf{IBE}}(\mathsf{msk}, \mathsf{ID})$ **:** *prend en entrée la clé maître secrète* msk *et l'identité* ID *d'un utilisateur et renvoie la clé de déchiffrement de l'utilisateur associé à l'identité* ID*, c.à.d.* $\mathsf{usk}_{\mathsf{ID}}$*.*

$\mathsf{Encrypt}_{\mathsf{IBE}}(\mathsf{mpk}, \mathsf{ID}, m; r)$ **:** *est un algorithme probabiliste qui prend en entrée les paramètres publics* mpk*, l'identité du destinataire* ID*, un message m et un aléa r. Cet algorithme renvoie un chiffré c du message m associé à l'identité* ID*.*

$\mathsf{Decrypt}_{\mathsf{IBE}}(\mathsf{usk}_{\mathsf{ID}}, c)$ **:** *prend en entrée la clé secrète d'un utilisateur* $\mathsf{usk}_{\mathsf{ID}}$ *et un chiffré c. Cet algorithme renvoie le message m associé au chiffré c sous l'identité* ID *; et sinon, si c n'est pas un chiffré de m sous une identité cet algorithme un symbole d'erreur.*

[3] qui peut être une simple adresse *e-mail*, un nom, un identifiant numérique, ...

Comme pour le chiffrement classique, on définit la notion de sécurité sémantique qui formalise la confidentialité du message chiffré : elle modélise la difficulté de déchiffrer un message sans connaître la clé secrète associée au destinataire du message.

Définition 12 (Sécurité sémantique pour les schémas $\mathcal{IBE}$) *Étant donné un paramètre de sécurité λ, on considère un schéma $\mathcal{IBE} \stackrel{déf}{=} \langle \mathsf{Setup_{IBE}}, \mathsf{Extract_{IBE}}, \mathsf{Encrypt_{IBE}}, \mathsf{Decrypt_{IBE}} \rangle$. On associe à tout attaquant polynomial $\mathcal{A}$, l'expérience $\mathsf{Exp}^{\mathsf{ind\text{-}cpa}\text{-}b}_{\mathcal{IBE},\mathcal{A}}$ ci-dessous, pour $b = 0, 1$:*

Expérience $\mathsf{Exp}^{\mathsf{ind\text{-}cpa}\text{-}b}_{\mathcal{IBE},\mathcal{A}}(\lambda)$
$\mathsf{IDSet} \leftarrow \emptyset$; $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup_{IBE}}(\lambda)$;
$(\mathsf{ID}^*, m_0, m_1, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathcal{O}\mathsf{Extract_{IBE}}()}(\mathit{FIND}, \mathsf{mpk})$;
si $m \notin \mathcal{M}$ *or* $|m_0| \neq |m_1|$;
retourner 0 sinon
$C^* \leftarrow \mathsf{Encrypt_{IBE}}(\mathsf{mpk}, \mathsf{ID}, m_b)$;
$b' \leftarrow \mathcal{A}_2^{\mathcal{O}\mathsf{Extract_{IBE}}()}(\mathit{GUESS}, C^*, \mathsf{state})$;
si $\{\mathsf{ID}_0, \mathsf{ID}_1\} \cap \mathsf{IDSet} = \emptyset$;
retourner b' *sinon retourner* 0

$\mathcal{O}\mathsf{Extract_{IBE}}(\mathsf{ID})$
$\mathsf{IDSet} \leftarrow \mathsf{IDSet} \cup \{\mathsf{ID}\}$;
$\mathsf{usk_{ID}} \leftarrow \mathsf{Extract_{IBE}}(\mathsf{msk}, \mathsf{ID})$;
retourner $\mathsf{usk_{ID}}$

On dit qu'un schéma $\mathcal{IBE}$ est sémantiquement sûr résistant aux attaques à clairs choisis si, l'avantage de tout attaquant polynomial $\mathcal{A}$, est négligeable ; cet avantage est défini par :

$$\mathsf{Adv}^{\mathsf{ind\text{-}cpa}\text{-}b}_{\mathcal{IBE},\mathcal{A}}(\lambda) = \left| \Pr[\mathsf{Exp}^{\mathsf{ind\text{-}cpa}\text{-}1}_{\mathcal{IBE},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{ind\text{-}cpa}\text{-}0}_{\mathcal{IBE},\mathcal{A}}(\lambda) = 1] \right|$$

Par la suite, lorsque l'attaquant n'a pas accès aux requêtes d'extraction, nous parlerons de sécurité sémantique faible.

On peut également définir une expérience où l'attaquant a accès à un oracle de déchiffrement sauf pour le chiffré challenge sous l'identité challenge. On parle alors de résistance aux attaques à chiffrés choisis ; on définit alors l'avantage de tout attaquant $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{ind\text{-}cca}\text{-}b}_{\mathcal{IBE},\mathcal{A}}(\lambda)$ de manière similaire. Si cet avantage est négligeable pour tout attaquant polynomial $\mathcal{A}$, on dit que le schéma $\mathcal{IBE}$ est sémantiquement sûr résistant aux attaques à chiffrés choisis (adaptatives, si le choix des requêtes de l'attaquant dépend du challenge et des réponses aux requêtes précédentes).

Après l'introduction du concept d'$\mathcal{IBE}$ et l'apparition du schéma de Boneh-Franklin [31], plusieurs schémas à base d'identités sont apparus, tous prouvés sûrs dans le modèle de l'oracle aléatoire : la même année, en 2001, Cocks [59] présente un schéma $\mathcal{IBE}$ dont la sécurité repose sur la résiduosité quadratique. Ce schéma utilise une fonction de hachage modélisée par un oracle aléatoire dans l'analyse de sécurité.

Le premier système $\mathcal{IBE}$ sans oracle aléatoire a été proposé par Boneh et Boyen [25] mais dans un modèle de sécurité plus faible que celui initialement introduit par Boneh et Franklin dans le premier article. Dans ce modèle, introduit par Canetti *et al.* [46] en 2003, l'attaquant sélectionne tout au début de l'expérience l'identité "challenge". Par la suite, on parlera de sécurité sémantique à identités orientées (*Selective*-ID en anglais).

3.5.2 Schémas de chiffrement hybride à base d'identité, $\mathcal{IB}$-$\mathcal{KEM}$

En 2000, dans [116], Shoup a proposé une formalisation le procédé de chiffrement hybride, que nous avons défini au paragraphe 1.1.2. Plus récemment, Bentahar *et al.* [23] ont

étendu ce concept au chiffrement à base d'identités, et ont proposé quelques constructions génériques sémantiquement sûres.

D'un point de vue théorique, les deux concepts sont équivalents, puisqu'il existe une transformation d'un schéma $\mathcal{IBE}$ en un schéma un $\mathcal{IB}$-$\mathcal{KEM}$ et *vice-versa*[4]. Mais en pratique, il est très souvent plus intéressant d'utiliser un schéma $\mathcal{IB}$-$\mathcal{KEM}$ car cette primitive offre une plus grande flexibilité qui rend les schémas plus efficaces. Par exemple, on peut prendre un paramètre de sécurité qui nous arrange sans changer la taille de l'espace des messages. Et la plupart du temps, pour les schémas de chiffrement à clé publique, on effectue des pré-calculs de couples clés/chiffrés, puis on applique un algorithme de chiffrement symétrique efficace.

D'autre part, très souvent, une notion de sécurité plus faible pour les schémas $\mathcal{IBE}$ permet d'aboutir à un schéma $\mathcal{IB}$-$\mathcal{KEM}$ avec une notion de sécurité plus forte. D'ailleurs, [23] présente une conversion générique d'un schéma $\mathcal{IBE}$, IND-CPA en un schéma $\mathcal{IB}$-$\mathcal{KEM}$, IND-CCA.

Remarque 3 *Notons que le résultat de [23] n'est pas très surprenant puisqu'il est assez analogue aux résultats pour le chiffrement hybride : on sait que la combinaison d'un $\mathcal{KEM}$,* IND-CCA *avec un $\mathcal{DEM}$* one-time CCA *conduit à un schéma de chiffrement* IND-CCA *[61]. Mais plusieurs travaux ont montré que ces deux conditions ne sont pas nécessaires : un des contre-exemples les plus populaires est le schémas $\mathcal{KEM}$ de Kurosawa et Desmedt [83], qui n'est pas résistant aux attaques à chiffrés choisis, et pourtant le schéma de chiffrement associé est* IND-CCA*-sûr. D'autres travaux récents s'intéressent aux conditions nécessaires et suffisantes à une conversion générique d'un schéma [82], ou plus spécifiquement à la construction de $\mathcal{KEM}$,* IND-CCA *[8].*

Commençons par rappeler les définitions d'un schéma $\mathcal{IB}$-$\mathcal{KEM}$ avant de décrire quelques-unes des constructions utiles pour la suite :

Définition 13 (Mécanisme d'Encapsulation de clés basé sur l'identité, $\mathcal{IB}$-$\mathcal{KEM}$) *Un schéma $\mathcal{IB}$-$\mathcal{KEM}$ est défini par la donnée de quatre algorithmes polynomiaux :*

$\langle \mathsf{Setup}_{\mathsf{IBK}}, \mathsf{Extract}_{\mathsf{IBK}}, \mathsf{Encaps}_{\mathsf{IBK}}, \mathsf{Decaps}_{\mathsf{IBK}} \rangle$ *définis par* :

$\mathsf{Setup}_{\mathsf{IBK}}(\lambda)$ **:** *prend en entrée un paramètre de sécurité λ et renvoie une clé maître publique et secrète* mpk/msk *;*

$\mathsf{Extract}_{\mathsf{IBK}}(\mathsf{msk}, \mathsf{ID})$ **:** *prend en entrée la clé secrète* mpk*, et l'identité* ID *d'un utilisateur et renvoie la clé de déchiffrement de l'utilisateur associée à l'identité* ID*,* $\mathsf{usk}_{\mathsf{ID}}$ *;*

$\mathsf{Encaps}_{\mathsf{IBK}}(\mathsf{mpk}, \mathsf{ID}; r)$ **:** *est un algorithme probabiliste qui prend en entrée les paramètres publics* mpk*, l'identité* ID *du destinataire et un aléa ; il renvoie une paire (K, c), où K est la clé de session éphémère et c l'encapsulation associée ;*

$\mathsf{Decaps}_{\mathsf{IBK}}(\mathsf{usk}_{\mathsf{ID}}, c)$ **:** *est un algorithme déterministe qui prend en entrée la clé secrète d'un utilisateur,* $\mathsf{usk}_{\mathsf{ID}}$ *et un chiffré c. Cet algorithme renvoie la clé associée à c ou un symbole d'erreur, si le chiffré n'est pas valide.*

De plus, nous définissons formellement la fonction $\mathsf{Decaps}_{\mathsf{IBK}}(\mathsf{ID}, c)$*, qui prend en entrée l'identité d'un utilisateur* ID *et un chiffré c. Cet algorithme extrait d'abord la clé de déchiffrement associée à l'identité* ID*, et décapsule le chiffré c à l'aide de* usk*.*

[4] si on utilise un $\mathcal{DEM}$ approprié

Nous définissons ci-dessous la notion de sécurité sémantique définie pour les schémas $\mathcal{IB}$-$\mathcal{KEM}$. Celle-ci se déduit assez facilement de la notion définie pour le chiffrement à base d'identités classique. Mais à présent, la sécurité sémantique traduit la confidentialité de la clé. Comme pour les schémas $\mathcal{IBE}$, cette notion peut être selon le cas être combinée à de multiples autres requêtes aux oracles, traduisant alors les résistances aux attaques à messages/chiffrés choisis et à identités orientées/non-orientées. Le jeu de sécurité, dans le modèle de sécurité le plus fort, c.à.d. attaques à chiffrés choisis et attaques à identités non orientées (*full-identity attacks* en anglais) avec accès aux requêtes d'extraction est le suivant :

$\mathsf{Expérience}\ \mathsf{Exp}^{\mathsf{ind\text{-}cca}\text{-}b}_{\mathcal{IB}\text{-}\mathcal{KEM},\mathcal{A}}(\lambda)$
$\mathsf{IDSet} \leftarrow \emptyset ; \mathsf{CSet} \leftarrow \emptyset ;$
$(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\lambda) ;$
$(\mathsf{ID}^*, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathcal{O}\mathsf{Extract}_{\mathsf{IBK}}(),\mathcal{O}\mathsf{Decaps}_{\mathsf{IBK}}()}(\mathtt{FIND}, \mathsf{mpk}) ;$
$(K_0, C') \leftarrow \mathsf{Encaps}_{\mathsf{IBK}}(\mathsf{mpk}, \mathsf{ID}^*) ;$
et $(K_1, C^*) \leftarrow \mathsf{Encaps}_{\mathsf{IBK}}(\mathsf{mpk}, \mathsf{ID}^*) ;$
$b' \leftarrow \mathcal{A}_2^{\mathcal{O}\mathsf{Extract}(),\mathcal{O}\mathsf{Decaps}()}(\mathtt{GUESS}, K_b, C^*, \mathsf{state}) ;$
$\mathtt{si}\ \{\mathsf{ID}^*\} \cap \mathsf{IDSet} = \emptyset ;$
et $\mathtt{si}\ \{(\mathsf{ID}^*, C^*)\} \cap \mathsf{CSet} = \emptyset ;$
$\mathtt{retourner}\ b'\ \mathtt{sinon\ retourner}\ 0 ;$

$\mathcal{O}\mathsf{Extract}_{\mathsf{IBK}}(\mathsf{ID})$
$\mathsf{IDSet} \leftarrow \mathsf{IDSet} \cup \{\mathsf{ID}\} ;$
$\mathsf{usk}_{\mathsf{ID}} \leftarrow \mathsf{Extract}_{\mathsf{IBK}}(\mathsf{msk}, \mathsf{ID}) ;$
$\mathtt{retourner}\ \mathsf{usk}_{\mathsf{ID}}$

$\mathcal{O}\mathsf{Decaps}_{\mathsf{IBK}}(\mathsf{ID}, c)$
$\mathsf{CSet} \leftarrow \mathsf{CSet} \cup \{(\mathsf{ID}, c)\} ;$
$\mathsf{usk}_{\mathsf{ID}} \leftarrow \mathsf{Extract}_{\mathsf{IBK}}(\mathsf{msk}, \mathsf{ID}) ;$
$K \leftarrow \mathsf{Decaps}(\mathsf{usk}_{\mathsf{ID}}, c) ;$
$\mathtt{retourner}\ K$

Nous définissons l'avantage de l'attaquant $\mathcal{A}$ dans le jeu précédent pour un schéma $\mathcal{IB}$-$\mathcal{KEM}$ par la probabilité que $\mathcal{A}$ parvienne à décider si le chiffré challenge est l'*encapsulation* de la clé renvoyée ou d'une clé aléatoire.

Plus formellement, un schéma $\mathcal{IB}$-$\mathcal{KEM}$ est sémantiquement sûr si l'avantage ci-dessous est négligeable en la paramètre de sécurité λ :

$$\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{IBK},\mathcal{A}}(\lambda) \quad = \quad \left|\Pr\left[\mathsf{Exp}^{\mathsf{ind\text{-}cca}\text{-}1}_{\mathcal{IBE},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{ind\text{-}cca}\text{-}0}_{\mathcal{IBE},\mathcal{A}}(\lambda) = 1\right]\right|$$

Remarque 4 *Pour l'application que nous présenterons plus loin, nous aurons en fait seulement besoin d'une version plus faible de la notion de* sécurité sémantique *définie ci-dessus, où l'adversaire choisit tout au début de l'expérience l'identité sur laquelle il s'engage (identités orientées), et où $\mathcal{A}$ n'a pas accès aux oracles* Decaps *et* Extract*. Par la suite, nous appellerons cette notion,* sécurité sémantique faible.

3.6 Quelques constructions

3.6.1 Un peu d'histoire

Depuis sa conception en 1984, la primitive $\mathcal{IBE}$ n'a cessé d'évoluer et aujourd'hui encore, la construction d'un schéma efficace reste un problème ouvert. On se prépare même à la standardisation de ces schémas et plusieurs candidats sont actuellement en cours d'étude.

Dans cette partie, nous présenterons quelques uns des schémas intéressants qui sont apparus et qui ont été déterminant dans l'histoire de ces schémas. Pour ces description, nous adopterons le formalisme de Xavier Boyen [35] qui a proposé une nomenclature des schémas $\mathcal{IBE}$: on remarque en effet qu'ils se divisent en trois catégories selon la méthode algébrique utilisée pour leur construction, les schémas "Full Domain Hash", les schémas "commutative blinding" ou encore les schémas "inversion-based". De fait, les schémas peuvent être comparés selon un grand nombre de critères : le niveau de sécurité garanti, le modèle de preuve adopté (modèle standard ou oracle aléatoire), la difficulté de l'hypothèse utilisée, la finesse de la réduction, la taille des paramètres, et de manière complémentaire la flexibilité du schéma et sa capacité à s'interfacer avec d'autres primitives. Nous insisterons d'avantage sur les points en connexion avec l'anonymat lors de la description de ces schémas dans la prochaine partie.

3.6.2 Classification de Boyen [35]

Les schémas "Full Domain Hash" En 2001, Boneh et Franklin [31] proposent le premier schéma $\mathcal{IBE}$ avec une preuve de sécurité formelle dans le modèle de l'oracle aléatoire. On détaillera cette construction au prochain chapitre. Leur schéma utilise le couplage de Weil et la sécurité du schéma repose sur le DBDH. L'utilisation de couplages pour la conception de schémas $\mathcal{IBE}$ semble dès lors une approche prometteuse.

Les schémas "commutative blinding" Un des schémas déterminants dans la conception de schémas $\mathcal{IBE}$ avec une preuve de sécurité dans le modèle standard est le schéma Boneh et Boyen [27]. Ce schéma est sémantiquement sûr et à identités orientées[5]. Nous décrivons figure 3.4 la simple version $\mathcal{IBE}$ du schéma (BB1a) proposé par Boneh et Boyen [25], qui présentent en réalité une variante où l'identité est vu comme un vecteur. Ce schéma a l'avantage d'être simple et assez flexible. On pourra se reporter à l'article [34] qui propose une analyse détaillé des atouts de ce schéma. La même année, les mêmes auteurs proposent un autre schéma prouvé sûr dans le modèle standard [26] mais qui est malheureusement trop inefficace pour être utilisé en pratique. Le premier schéma efficace dans le modèle standard a été proposé par Waters en 2005 [120]. Il propose de fait une modification du schéma de Boneh et Boyen permettant d'obtenir un schéma IND-CCA. Mais la contribution importante de ce papier est une preuve originale où le simulateur interrompt le jeu si un certain prédicat n'est pas vérifié. La difficulté dans cette preuve est d'une part de montrer la correction de cette approche, et d'autre part d'évaluer la probabilité de succès en cas d'interruption du simulateur. Récemment, Bellare et Ristenpart [15] ont proposé une simplification de cette preuve et une amélioration de ses performances.

Remarque 5 *Pour le schéma $\mathcal{IBE}$ de Waters [120], la fonction F_{mpk} est une modification de la fonction F_{mpk} définie figure 3.4) du schéma de l'article [25] : elle utilise une fonction de hachage $\mathcal{H} : \{0,1\}^n \mapsto \mathbb{G}$, des éléments $h_i \overset{R}{\leftarrow} \mathbb{G}$ qui sont publics, $F_{\mathsf{mpk}} : \{0,1\}^\star \mapsto \mathbb{G}$ est alors définie par : $F_{\mathsf{mpk}}(\mathsf{ID}) = h_i \prod_{i=1} h_i^{\mathsf{ID}_i}$, où l'identité est une chaîne de caractères. On voit alors que la clé publique contient tous les h_i, c.à.d. $n+4$ éléments, elle augmente donc de taille par rapport au schéma figure 3.4,*

[5] Précisons qu'il existe une transformation d'un schéma IND-CPA à identités orientées en un schéma IND-CPA à identités non orientées, avec une perte d'un facteur N, où N est le nombre d'identités mais aussi une borne inférieure sur la taille du groupe.

$\mathsf{Setup}(\lambda)$: $\rangle g, p, \mathbb{G}, \mathbb{G}_T, e\langle \leftarrow \mathcal{G}(\lambda)$; $g_2, h \overset{R}{\leftarrow} \mathbb{G}$ et $\alpha \overset{R}{\leftarrow} \mathbb{Z}_p$;
$\mathsf{mpk} = (g, g_1 = g^{\alpha}, g_2, h)$ et $\mathsf{msk} = {g_2}^{\alpha}$;
définir $F_{\mathsf{mpk}}(x) = h \cdot {g_1}^{x}$, avec $x \in \mathbb{Z}_p$;
`retourner` mpk ;

$\mathsf{Extract}(\mathsf{msk}, \mathsf{ID})$:
choisir $r \overset{R}{\leftarrow} \mathbb{Z}_p$;
`retourner` $(\mathsf{usk}_1, \mathsf{usk}_2) = ({g_2}^{\alpha} \cdot F_{\mathsf{mpk}}(\mathsf{ID})^{r}, g^{r})$;

$\mathsf{Encrypt}(\mathsf{mpk}, \mathsf{ID}, m)$:
choisir $r \overset{R}{\leftarrow} \mathbb{Z}_p$;
calculer $C = (e(g_1, g_2)^{s} \cdot m, g^{s}, F_{\mathsf{mpk}}(\mathsf{ID})^{s})$;
`retourner` $C = (c_1, c_2, c_3)$;

$\mathsf{Decaps}(\mathsf{usk}_{\mathsf{ID}}, C)$:
(1) décomposer C en (c_1, c_2, c_3) et calculer :
(2) $K = e(c_3, \mathsf{usk}_2)/e(c_2, \mathsf{usk}_1)$
`retourner` $m = c_1/K$

FIG. 3.4 – Schéma $\mathcal{IBE}$ de Boneh-Boyen.

Les schémas "inversion-based" Un autre type de schéma, initié par les travaux de Mitsunari, Sakai et Kasahara [99] dans le contexte du traçage de traîtres, a permis d'aboutir à des schémas plus efficaces avec une preuve de sécurité dans le modèle standard. La construction de ces schémas s'appuie sur un algorithme d'extraction novateur ; la clé d'extraction est calculée en deux étapes : on calcule d'abord l'inverse d'une somme de la clé maître et de l'image de l'identité dans un sous-groupe de $\mathbb{Z}_p$, $\mathbb{Z}_q^{\star}$. Puis on multiplie cet élément par un générateur de $\mathbb{G}_q$. Le premier schéma à utiliser ces idées a été proposé par Sakai et Kasahara [111, 56] et beaucoup d'autres ont suivi. Au départ la sécurité de ces schémas a été difficile à analyser et aucun d'entre eux n'a été formellement prouvé. Chen et Cheng [55] sont les premiers à avoir proposé une preuve de ce schémas dans le modèle de l'oracle aléatoire en adaptant la preuve du schéma de Boneh et Boyen (BB1b) [25]. Les hypothèses de sécurité utilisées pour les preuves de ces schémas sont particulièrement fortes et sont en général non statiques. En contrepartie, elles permettent d'obtenir des schémas efficaces et des preuves de sécurité plus fines [71].

Nous présentons figure 3.5 une variante du schéma de Sakai et Kasahara [56] dont Chen *et al.* [55] ont démontré la sécurité sémantique dans le modèle de l'oracle aléatoire. Ce schéma est défini dans un contexte asymétrique : la preuve requiert l'existence d'un homomorphisme ϕ de $\mathbb{G}_2$ dans $\mathbb{G}_1$. Pour cette description g_1 est g_2 sont des générateurs respectifs de $\mathbb{G}_1$ et $\mathbb{G}_2$ et on a : $g_1 = \phi(g_2)$. On introduit également deux fonctions de hachages $\mathcal{H}_1$ et $\mathcal{H}_2$ (modélisées par des oracles aléatoires dans la preuve de sécurité) : $\mathcal{H}_1 : \{0,1\} \mapsto \mathbb{Z}_p$ et $\mathcal{H}_2 : \mathbb{G}_T \mapsto \{0,1\}^n$.

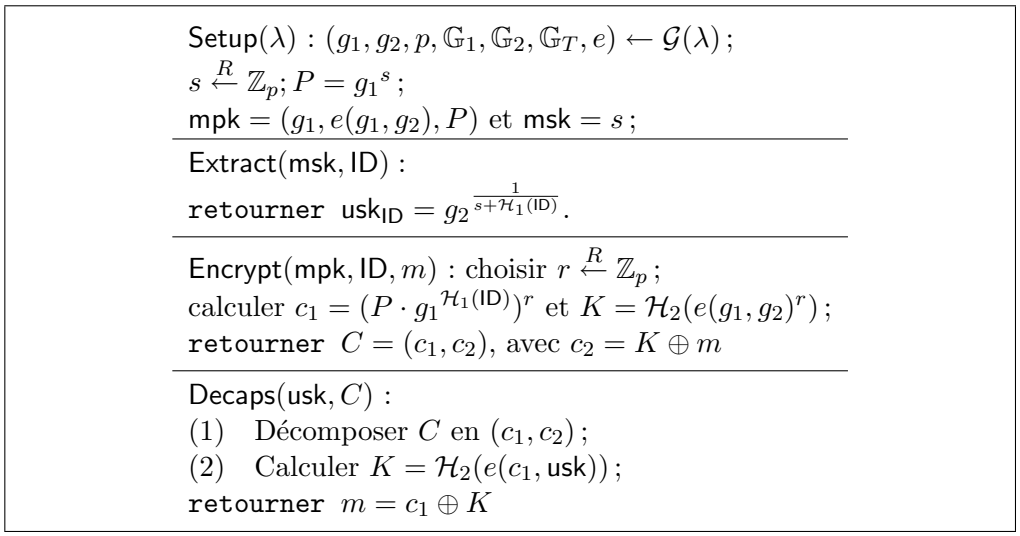


FIG. 3.5 – Variante du Schéma $\mathcal{IBE}$ de Sakai et Kasahara, proposée par [56].

Deuxième partie

Chiffrement anonyme

Chapitre 4

Chiffrement anonyme

L'article à la source de la notion d'anonymat pour le chiffrement probabiliste asymétrique a été publié en 2001 par Bellare, Boldyreva, Desai et Pointcheval [11]. Auparavant, la notion de sécurité classique pour le chiffrement formalisait l'*indistinguabilité* [73] et/ou la *non-malléabilité* [64] des messages clairs. Désormais, le destinataire est au centre des préoccupations : on cherche à préserver son anonymat. L'idée de la notion de *key privacy* introduite par Bellare *et al.* est de garantir qu'un chiffré ne laisse fuir aucune information sur la clé publique utilisée pour chiffrer. Cette notion peut paraître contre-intuitive, en particulier pour le chiffrement asymétrique. Néanmoins, si on se place dans un environnement multi-acteurs (par exemple), le but est de garantir la confidentialité de la clé publique associée à un chiffré parmi un ensemble de clés disponibles. Dans ce chapitre, nous commençons par rappeler la définition de cette notion de sécurité, avant de montrer comment elle s'étend au chiffrement à base d'identités. Nous présenterons quelques schémas à base d'identités vérifiant cette notion d'anonymat. Par ailleurs, rappelons que dans ce contexte, un centre de gestion de clés est capable de dériver toutes les clés secrètes associées aux identités. Dans l'article [85] en commun avec David Pointcheval, nous avons étendu cette notion d'anonymat en considérant la PKI comme attaquant potentiel. Dans cet article, nous avons également proposé un schéma à base d'identités satisfaisant la notion KwrtA, une notion complémentaire de la notion d'anonymat classique.

Sommaire

4.1 Anonymat pour le chiffrement

Comme pour la sécurité sémantique, la notion de *key privacy* se traduit par l'indistinguabilité de deux expériences, chacune correspondant à une valeur d'un bit b choisi par le challenger.

4.1.1 Notion de *key-privacy* [11]

Considérons un schéma de chiffrement constitué des quatre algorithmes $\mathsf{Setup}, \mathsf{KeyGen}$, $\mathsf{Encrypt}, \mathsf{Decrypt}$ et un attaquant $\mathcal{A}$. Choisissons un bit b aléatoire. Étant donné un paramètre de sécurité λ, le challenger fait appel à l'algorithme Setup pour la génération de paramètres publics params. Il génère ensuite deux paires clé publique/clé secrète $(\mathsf{pk}_i, \mathsf{sk}_i)$ pour $i = 0, 1$ *via* l'algorithme KeyGen avec params en entrée. Pendant une première phase, $\mathcal{A}$ reçoit les clés $(\mathsf{pk}_i, \mathsf{sk}_i)$ et renvoie un message. Le challenger lui chiffre ce message avec la clé pk_b. Dans une seconde phase, l'attaquant reçoit ce chiffré et retourne un bit b' ; il gagne s'il devine à laquelle des deux clés est associé le chiffré. Si l'attaquant ne parvient pas à gagner avec une probabilité significativement plus grande que $\frac{1}{2}$, même en ayant accès aux oracles de déchiffrement $\mathcal{O}\mathsf{Decrypt}_i$ pour $i = 0, 1$ (durant les deux phases) pour les clés sk_0 et sk_1, le schéma de chiffrement est dit *anonyme* résistant aux attaques à chiffrés choisis.

Nous définissons ci-dessous l'expérience $\mathsf{Exp}^{\mathsf{ind\text{-}key\text{-}cca}\text{-}b}_{\mathcal{E},\mathcal{A}}(\lambda)$ (pour $b = 0, 1$) décrite précédemment :

$\mathsf{Expérience}\ \mathsf{Exp}^{\mathsf{ind\text{-}key\text{-}cca}\text{-}b}_{\mathcal{E},\mathcal{A}}(\lambda)$
$\mathsf{CSet} \stackrel{\text{déf}}{=} \emptyset$; $(\mathsf{params}) \leftarrow \mathsf{Setup}(\lambda)$;
$(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{KeyGen}(\mathsf{params})$;
$(m, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathcal{O}\mathsf{Decrypt}_0(),\mathcal{O}\mathsf{Decrypt}_1()}(\mathtt{FIND}, \mathsf{params}, \mathsf{pk})$;
$c \stackrel{R}{\leftarrow} \mathsf{Encrypt}(\mathsf{pk}_b, m)$;
$b' \leftarrow \mathcal{A}_2^{\mathcal{O}\mathsf{Decrypt}_0(),\mathcal{O}\mathsf{Decrypt}_1()}(\mathtt{GUESS}, c, \mathsf{state})$;
`si` $c \notin \mathsf{CSet}$
`retourner` b' `sinon retourner` 0 ;

$\mathcal{O}\mathsf{Decrypt}_i(c)$
  $\mathsf{CSet} \leftarrow \mathsf{CSet} \cup \{c\}$;
  $m \leftarrow \mathsf{Decrypt}(\mathsf{sk}_i, c)$;
  `retourner` m

On définit l'avantage de l'attaquant $\mathcal{A}$ à casser l'anonymat du schéma de chiffrement $\mathcal{E}$ sous les attaques à chiffrés choisis adaptatives par :

$$\mathsf{Adv}^{\mathsf{ind\text{-}key\text{-}cca}}_{\mathcal{E},\mathcal{A}}(\lambda) = \left|\Pr[\mathsf{Exp}^{\mathsf{ind\text{-}key\text{-}cca\text{-}1}}_{\mathcal{E},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{ind\text{-}key\text{-}cca\text{-}0}}_{\mathcal{E},\mathcal{A}}(\lambda) = 1]\right|$$

4.1.2 Pertinence de l'anonymat pour le chiffrement

L'intérêt que nous portons à la primitive de chiffrement anonyme se justifie tant d'un point de vue pratique que théorique. Dans cette section, nous explicitons cette attention sous ces deux angles :

Les protocoles anonymes

Tout d'abord, le concept d'anonymat est crucial dans différentes applications courantes, comme par exemple les schémas de monnaie électronique [51, 52] (qui permettent d'utiliser des pièces authentifiées par la banque de manière anonyme) ; les systèmes de transactions anonymes proposés par [41] dans le but de permettre le contrôle de l'information relative à un utilisateur et susceptible d'être diffusée ; ou encore les ventes aux enchères. Pour ce dernier exemple, Sako [112] a proposé un schéma en adaptant la notion d'anonymat à la notion de confidentialité, où chaque proposition d'enchères est associée à un message connu chiffré avec une clé spécifique à l'offre. Cette approche permet de masquer la correspondance enchère/clé de chiffrement sans avoir à masquer d'identité quelconque. Nous verrons au chapitre 8 un autre exemple intéressant d'application de la notion d'anonymat, dans les protocoles d'authentification : en effet, lors d'un échange de clés authentifié, un serveur peut très bien établir des liens entre les connexions et le profil des utilisateurs ; il est tout à fait légitime que ce dernier veuille bénéficier d'un service d'authentification sans qu'on ne puisse pour autant l'identifier.

Spécifications des schémas anonymes

Il n'existe pas de lien direct entre les notion de sécurité sémantique et les schémas anonymes : il se peut très bien que l'on puisse, pour un schéma donné, garantir la confidentialité des messages clairs sous des attaques à chiffrés choisis sans que ce schéma soit anonyme pour autant. Par contre, nous disposons de tous les outils pour déterminer si un schéma donné vérifie la notion de *key privacy*. De plus, les preuves de sécurité pour l'anonymat sont tout à fait similaire à celles obtenues pour la sécurité sémantique. L'article de Bellare *et al.* témoigne de ce fait : ils formalisent la notion d'*anonymat* pour le chiffrement asymétrique, montrent que certains des schémas que nous connaissons déjà vérifient cette propriété : le cryptosystème ElGamal, le cryptosystème de Cramer-Shoup, une variante du schéma RSA-OAEP [11] ; juste pour exemple, les preuves d'*anonymat* et de *sécurité sémantique* pour le schéma de chiffrement ElGamal sont très proches. Les auteurs proposent également une construction dans le modèle standard reposant sur les permutations à trappe sous la seule hypothèse que le chiffrement RSA est à *sens unique*.

4.2 Chiffrement anonyme basé sur l'identité

4.2.1 Définition

La notion d'anonymat pour les schémas $\mathcal{IBE}$ a été définie par Abdalla *et al.* en 2005 dans l'article [1]. L'idée est assez simple : supposons que l'adversaire choisisse un message m et deux identités ; étant donné un chiffré de m sous l'une des deux identités, l'adversaire ne parvient pas à décider à laquelle des deux identités le chiffré challenge correspond. Plus formellement, considérons un schéma $\mathcal{IBE}$ déterminé par la donnée des algorithmes $\langle \mathsf{Setup}, \mathsf{Extract}, \mathsf{Encrypt}, \mathsf{Decrypt} \rangle$. On définit la notion d'anonymat pour les schémas $\mathcal{IBE}$ par le jeu suivant :

Expérience $\mathsf{Exp}^{\mathsf{anon\text{-}cpa}\text{-}b}_{\mathcal{IBE},\mathcal{A}}(\lambda)$
$\mathsf{IDSet} \leftarrow \emptyset; (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\lambda)$;
$(\mathsf{ID}_0, \mathsf{ID}_1, m, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathcal{O}\mathsf{Extract}()}(\mathtt{FIND}, \mathsf{mpk})$;
$b \xleftarrow{R} \{0,1\}$;
$c^* \leftarrow \mathsf{Encrypt}(\mathsf{mpk}, \mathsf{ID}_b, m)$;
$b' \leftarrow \mathcal{A}_2^{\mathcal{O}\mathsf{Extract}()}(\mathtt{GUESS}, c^*, \mathsf{state})$;
`si` $\{\mathsf{ID}_0, \mathsf{ID}_1\} \cap \mathsf{IDSet} = \{\emptyset\}$;
`retourner` b' `sinon retourner` 0

$\mathcal{O}\mathsf{Extract}(\mathsf{ID})$
$\mathsf{IDSet} \leftarrow \mathsf{IDSet} \cup \{\mathsf{ID}\}$
$\mathsf{usk}_{\mathsf{ID}} \leftarrow \mathsf{Extract}(\mathsf{msk}, \mathsf{ID})$
`retourner` $\mathsf{usk}_{\mathsf{ID}}$

On définit l'avantage de l'attaquant $\mathcal{A}$ à casser l'anonymat du schéma $\mathcal{IBE}$ sous les attaques à messages clairs choisis par :

$$\mathsf{Adv}^{\mathsf{anon\text{-}cpa}}_{\mathcal{IBE},\mathcal{A}}(\lambda) = \left|\Pr[\mathsf{Exp}^{\mathsf{anon\text{-}cpa\text{-}1}}_{\mathcal{IBE},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{anon\text{-}cpa\text{-}0}}_{\mathcal{IBE},\mathcal{A}}(\lambda) = 1]\right|$$

On dit qu'un schéma $\mathcal{IBE}$ est anonyme résistant aux attaques à messages choisis ou ANON-CPA si pour tout attaquant polynomial $\mathcal{A}$, l'avantage de $\mathcal{A}$ est une fonction négligeable en le paramètre de sécurité. Bien entendu, de la même manière que pour la sécurité sémantique, nous pouvons donner accès à l'oracle de déchiffrement : si l'avantage reste négligeable, on dit que le schéma $\mathcal{IBE}$ est anonyme résistant aux attaques à chiffrés choisis ou ANON-CCA. Si cet accès est possible durant les deux phases, l'attaquant est dit adaptatif. Par ailleurs, si les deux identités sont choisies tout au début de l'expérience, on parle d'attaques à identités orientées et on écrit sID-ANON-CPA ou sID-ANON-CCA, selon que l'accès à l'oracle de déchiffrement est autorisé ou non. Enfin, si nous pouvons simuler les requêtes d'extraction, nous parlons d'anonymat fort ; sinon nous palerons d'anonymat faible. Comme nous l'avons déjà précisé, nous nous intéresserons à la version hybride $\mathcal{IB}$-$\mathcal{KEM}$ de quelques uns des schémas $\mathcal{IBE}$ déjà connus ; nous obtenons des schémas plus efficaces avec des tailles de chiffrés plus courtes. Avec un mécanisme d'encapsulation de messages $\mathcal{DEM}$ approprié, nous définissons la notion d'anonymat pour les schémas $\mathcal{IB}$-$\mathcal{KEM}$ par le jeu suivant :

Expérience $\mathsf{Exp}^{\mathsf{anon-cca}-b}_{\mathcal{IB}\text{-}\mathcal{KEM},\mathcal{A}}(\lambda)$
$\mathsf{IDSet} \leftarrow \emptyset; \mathsf{CSet} \leftarrow \emptyset$;
$(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\lambda)$;
$(\mathsf{ID}_0, \mathsf{ID}_1, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathcal{O}\mathsf{Extract}(),\mathcal{O}\mathsf{Decaps}()}(\mathtt{FIND}, \mathsf{mpk})$;
$(K^*, C^*) \leftarrow \mathsf{Encaps}_{\mathsf{IBK}}(\mathsf{mpk}, \mathsf{ID}_b)$;
$b' \leftarrow \mathcal{A}_2^{\mathcal{O}\mathsf{Extract}(),\mathcal{O}\mathsf{Decaps}()}(\mathtt{GUESS}, K^*, C^*, \mathsf{state})$;
`si` $\{\mathsf{ID}_0, \mathsf{ID}_1\} \cap \mathsf{IDSet} = \emptyset$;
et `si` $\{(\mathsf{ID}_b, C^*)\} \cap \mathsf{CSet} = \emptyset$ pour $b = 0, 1$;
`retourner` b' `sinon retourner` 0

Les accès aux oracles d' extraction et de décapsulation de clés sont déterminés par les appels aux fonctions suivantes :

$\mathcal{O}\mathsf{Extract}(\mathsf{ID})$
$\mathsf{IDSet} \leftarrow \mathsf{IDSet} \cup \{\mathsf{ID}\}$;
$\mathsf{usk}_{\mathsf{ID}} \leftarrow \mathsf{Extract}(\mathsf{msk}, \mathsf{ID})$;
`retourner` $\mathsf{usk}_{\mathsf{ID}}$

$\mathcal{O}\mathsf{Decaps}(\mathsf{ID}, c)$
$\mathsf{CSet} \leftarrow \mathsf{CSet} \cup \{(\mathsf{ID}, c)\}$;
$\mathsf{usk}_{\mathsf{ID}} \leftarrow \mathsf{Extract}(\mathsf{msk}, \mathsf{ID})$;
$K \leftarrow \mathsf{Decrypt}(\mathsf{usk}_{\mathsf{ID}}, c)$;
`retourner` K

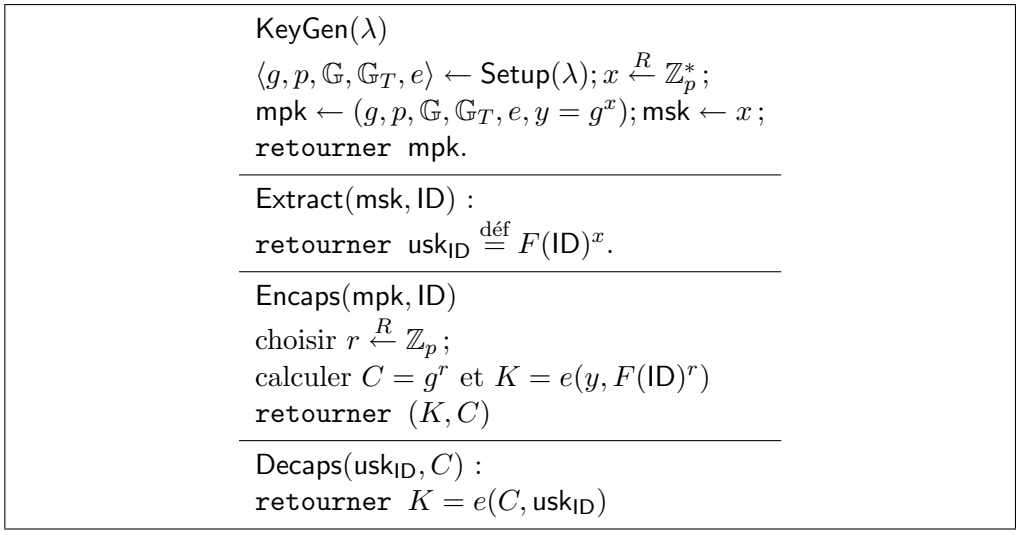


FIG. 4.1 – Schéma $\mathcal{IB}$-$\mathcal{KEM}$ de Boneh-Franklin, où la fonction F est modélisée par un oracle aléatoire.

On définit l'avantage de l'attaquant $\mathcal{A}$ dans le jeu précédent par :

$$\mathsf{Adv}^{\mathsf{anon-cca}}_{\mathcal{IB}\text{-}\mathcal{KEM},\mathcal{A}}(\lambda) = \left|\Pr[\mathsf{Exp}^{\mathsf{anon-cca}-1}_{\mathcal{IB}\text{-}\mathcal{KEM},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{anon-cca}-0}_{\mathcal{IB}\text{-}\mathcal{KEM},\mathcal{A}}(\lambda) = 1]\right|.$$

Si cet avantage est une fonction négligeable en le paramètre de sécurité λ, on dit que le schéma $\mathcal{IB}$-$\mathcal{KEM}$ est anonyme résistant aux attaques à chiffrés choisis.

4.2.2 Quelques constructions intéressantes

Dans cette partie, nous décrivons quelques schémas $\mathcal{IB}$-$\mathcal{KEM}$ dérivés des schémas $\mathcal{IBE}$ anonymes proposés ces dernières années. Pour leur description, nous aurons besoin d'une fonction F qui peut éventuellement dépendre des paramètres publics. Il s'agit d'une fonction qui envoie les identités dans le groupe.

Schéma de Boneh-Franklin[31]

Le premier schéma $\mathcal{IBE}$ qui a été proposé est celui de Boneh et Franklin [31]. Il s'agit d'un des schémas les plus populaires de par son efficacité et sa simplicité. L'un de ses inconvénients majeurs est la nécessité de calculs de hachés à valeurs dans le groupe, ce qui est coûteux mais restreint aussi le type de courbes pouvant être utilisées. Pour ce schéma, on définit F_{ID} à valeurs dans $\mathbb{G}$, indépendante de la clé publique ; elle est modélisée par un oracle aléatoire dans l'analyse de sécurité. Nous décrivons le schéma $\mathcal{IB}$-$\mathcal{KEM}$ dérivé figure 4.1. Ce schéma est assez proche du schéma de chiffrement ElGamal. La *sécurité sémantique* (sans accès aux oracles de décapsulation mais avec accès aux requêtes Extract) repose sur le problème $\mathsf{DBDH}_{\mathbb{G},\mathbb{G}_T}$ dans le modèle de l'oracle aléatoire. Notons que ce schéma est anonyme sous l'hypothèse $\mathsf{DBDH}_{\mathbb{G},\mathbb{G}_T}$ (la clé K est un élément aléatoire et uniformément distribuée dans $\mathbb{G}_T$).

Schéma de Boyen-Waters [36]

Ce schéma a été proposé par Boyen et Waters en 2006. Il est déterminant puisqu'il s'agit de la première construction (anonyme) "commutative blinding" avec une preuve de sécurité dans le modèle standard. L'un des points clés de cette construction s'appuie sur la remarque suivante : pour tous les schémas $\mathcal{IBE}$ se passant de l'oracle aléatoire, l'algorithme d'extraction est probabiliste ; par conséquent, afin de rendre le déchiffrement possible, l'aléa issu de le phase d'extraction se doit d'être compensé par un complément d'information. Dans les schémas existants, cela se traduit par une redondance dans le chiffré qui permet d'assurer la correction mais fournit en contrepartie un test public d'identités pour le chiffré. L'idée de Boyen et Waters est de ne rendre exploitable cette redondance que si un problème algorithmique difficile peut être résolu ; ici, il s'agit du problème linéaire décisionnel $\mathsf{DLIN}_{\mathbb{G}}$ (voir définition 3.3.1). Nous donnons une description du schéma anonyme de Boyen-Waters figure 4.2, où F_{mpk} est une fonction qui dépend de la clé publique, définie sur l'ensemble des identités ; elle est définie par : $F_{\mathsf{mpk}}(x) = g_0 \cdot g_1^x$. Cette construction est sémantiquement sûre (à identité orientée) sous l'hypothèse $\mathsf{DBDH}_{\mathbb{G}}$. Nous renvoyons le lecteur à l'article [28] qui présente également un schéma plus général, où l'identité est vue comme un vecteur, sémantiquement sûr (à identités non orientées) et anonyme.

Schéma de Gentry [71]

Ce schéma a été proposé en 2006, il comporte de nombreux avantages, et en particulier un nombre de paramètres publics constant. Cette construction est sémantiquement sûre résistante aux attaques à chiffrés choisis sous l'hypothèse l-T-$\mathsf{ABDHE}_{\mathbb{G},\mathbb{G}_T}$, où l est un paramètre. La preuve de ce résultat est assez intuitive. L'hypothèse utilisée n'est pas très naturelle et paramétrée par le nombre de requêtes maximum autorisées à l'attaquant. Par souci de clarté et de cohérence pour nos futures comparaisons, nous présentons seulement le schéma sémantiquement sûr figure 4.3. Nous définissons la fonction F_{mpk} par $F_{\mathsf{mpk}}(\mathsf{ID}) = g_1 \cdot g^{-\mathsf{ID}} = g^{\alpha - \mathsf{ID}}$. Dans l'article [71], Gentry présente un schéma résistant aux attaques à chiffrés choisis en ajoutant trois éléments publics ; la stratégie de la preuve de Cramer-Shoup s'adapte parfaitement aux idées utilisées pour la preuve du schéma de Gentry résistant aux attaques à chiffrés choisis. Pour plus de précisions sur le schéma $\mathsf{IND\text{-}CCA}$, nous renvoyons à l'article de Gentry [71]. Nous préférons présenter la preuve en détail car la preuve de notre schéma (voir paragraphe 4.3.3) utilise la même stratégie.

Théorème 4 *Soit $q = q_{\mathsf{ID}} + 1$. Supposons la validité de l'hypothèse (t, ϵ)-q-T-$\mathsf{ABDHE}_{\mathbb{G},\mathbb{G}_T}$. Le schéma figure 4.3 est $(t', \epsilon', q_{\mathsf{ID}})$-sémantiquement sûr (à identités non orientées) résistant aux attaques à clairs choisis et anonyme (avec accès à q_{ID} requêtes d'extraction adaptatives) avec $t' = t - \mathcal{O}(\tau \cdot q^2)$ et $\epsilon' = \epsilon + \frac{2}{p}$, où τ est le temps requis pour une exponentiation dans le groupe $\mathbb{G}$.*

Étant donné un paramètre de sécurité λ, nous considérons une expérience $\mathsf{Exp}^{\mathsf{ind\text{-}cpa\text{-}anon\text{-}}c\text{-}b}_{\mathcal{IB}\text{-}\mathcal{KEM},\mathcal{A}}(\lambda)$ définie par une séquence de jeux hybrides en modifiant le format du challenge. L'avantage de l'attaquant $\mathcal{A}$ est alors défini par :

$$\mathsf{Adv}^{\mathsf{ind-cpa-anon}}_{\mathcal{IB}\text{-}\mathcal{KEM},\mathcal{A}}(\lambda) \quad = \quad \left| \Pr\left[\mathsf{Exp}^{\mathsf{ind\text{-}cpa\text{-}anon\text{-}1\text{-}1}}_{\mathcal{IB}\text{-}\mathcal{KEM},\mathcal{A}}(\lambda) = (1,1)\right] - \Pr\left[\mathsf{Exp}^{\mathsf{ind\text{-}cpa\text{-}anon\text{-}0\text{-}0}}_{\mathcal{IB}\text{-}\mathcal{KEM},\mathcal{A}}(\lambda) = (1,1)\right] \right|$$

Nous supposons que son attaque est valide si cet avantage est significativement plus grand que $\frac{1}{4}$.

$\mathsf{Setup}(\lambda)$:
$\mathsf{params} \leftarrow \mathcal{KG}\mathsf{bil}(\lambda)$, où $\mathsf{params} = (g, p, \mathbb{G}, \mathbb{G}_T, e)$;
choisir $\omega, t_1, t_2, t_3, t_4 \xleftarrow{R} \mathbb{Z}_p$; $g_0, g_1 \xleftarrow{R} \mathbb{G}$;
$\mathsf{mpk} = (\mathsf{params}, \Omega = e(g,g)^{t_1 \cdot t_2 \cdot \omega}, g_0, g_1, v_1 = g^{t_1}, v_2 = g^{t_2}, v_3 = g^{t_3}, v_4 = g^{t_4})$
$\mathsf{msk} = (\omega, t_1, t_2, t_3, t_4)$; `retourner` mpk ;

$\mathsf{Extract}(\mathsf{msk}, \mathsf{ID})$: choisir $r_1, r_2, t_i \xleftarrow{R} \mathbb{Z}_p$, pour $i = 1, 2, 3, 4$
calculer :
$\mathsf{usk}_0 = g^{r_1 t_1 t_2 + r_2 t_3 t_4}$
$\mathsf{usk}_1 = g^{-\omega t_2} F_{\mathsf{mpk}}(\mathsf{ID})^{-r_1 t_2}$ $\quad$ $\mathsf{usk}_2 = g^{-\omega t_1} F_{\mathsf{mpk}}(\mathsf{ID})^{-r_1 t_1}$
$\mathsf{usk}_3 = F_{\mathsf{mpk}}(\mathsf{ID})^{-r_2 t_4}$ $\quad$ $\mathsf{usk}_4 = F_{\mathsf{mpk}}(\mathsf{ID})^{-r_2 t_3}$
`retourner` $\mathsf{usk}_{\mathsf{ID}} = (\mathsf{usk}_1, \mathsf{usk}_2, \mathsf{usk}_3, \mathsf{usk}_4)$;

$\mathsf{Encaps}(\mathsf{mpk}, \mathsf{ID})$: choisir $s, s_i \xleftarrow{R} \mathbb{Z}_p$, pour $i = 1, 2, 3, 4$
$K = \Omega^s$ $\quad$ $c_0 = F_{\mathsf{mpk}}(\mathsf{ID})^s$
$c_1 = v_1^{s-s_1}$ $\quad$ $c_2 = v_2^{s_1}$
$c_3 = v_3^{s-s_2}$ $\quad$ $c_4 = v_4^{s_2}$
`retourner` $(K, C) = (K, (c_0, c_1, c_2, c_3, c_4))$;

$\mathsf{Decaps}(\mathsf{usk}_{\mathsf{ID}}, C)$:
(1) décomposer C et $\mathsf{usk}_{\mathsf{ID}}$ en $(c_0, c_1, c_2, c_3, c_4)$ et $(\mathsf{usk}_1, \mathsf{usk}_2, \mathsf{usk}_3, \mathsf{usk}_4)$ resp. ;
(2) calculer :

$$
\begin{aligned}
K^{-1} &= \Omega^{-s} = e(g,g)^{-\omega t_1 t_2 s} \\
&= e(F_{\mathsf{mpk}}(\mathsf{ID}), g)^{s(r_1 t_1 t_2 + r_2 t_3 t_4)} \times e(g,g)^{-\omega t_1 t_2 s} \\
&\quad \times e(F_{\mathsf{mpk}}(\mathsf{ID}), g)^{-r_1 t_1 t_2 s} \times e(F_{\mathsf{mpk}}(\mathsf{ID}), g)^{-r_2 t_3 t_4 s} \\
&\quad \times e(F_{\mathsf{mpk}}(\mathsf{ID}), g)^{-r_2 t_3 t_4 (s - s_2)} \times e(F_{\mathsf{mpk}}(\mathsf{ID}), g)^{-r_2 t_3 t_4 s_2} \\
&= e(F_{\mathsf{mpk}}(\mathsf{ID}), g)^{s(r_1 t_1 t_2 + r_2 t_3 t_4)} \times e(g,g)^{-\omega t_1 t_2 (s - s_1)} \\
&\quad e(F_{\mathsf{mpk}}(\mathsf{ID}), g)^{-r_1 t_1 t_2 (s - s_1)} \times e(g,g)^{-\omega t_1 t_2 s_1} e(F_{\mathsf{mpk}}(\mathsf{ID}), g)^{-r_1 t_1 t_2 s_1} \\
&= e((F_{\mathsf{mpk}}(\mathsf{ID})^s, g^{r_1 t_1 t_2 + r_2 t_3 t_4}) \times e(v_1^{s-s_1}, g^{-\omega t_2} (F_{\mathsf{mpk}}(\mathsf{ID}))^{-r_1 t_2}) \\
&= e(c_0, \mathsf{usk}_0) \times e(c_1, \mathsf{usk}_1) \times e(c_2, \mathsf{usk}_2) \times e(c_3, \mathsf{usk}_3) \times e(c_4, \mathsf{usk}_4)
\end{aligned}
$$

`retourner` K

FIG. 4.2 – Schéma $\mathcal{IB}$-$\mathcal{KEM}$ de Boyen-Waters, où la fonction F_{mpk} est définie par $F_{\mathsf{mpk}}(x) = g_1 \cdot g^{-x}$.

$\mathsf{Setup}(\lambda)$:
$\mathsf{params} \leftarrow \mathcal{KG}_{\mathsf{bil}}(\lambda)$;
$\mathsf{params} \stackrel{\text{déf}}{=} (g, p, \mathbb{G}, \mathbb{G}_T, e)$; $g, h \stackrel{R}{\leftarrow} \mathbb{G}$ et $\alpha \stackrel{R}{\leftarrow} \mathbb{Z}_p$;
$\mathsf{mpk} = (g, g_1 = g^\alpha, h)$ et $\mathsf{msk} = \alpha$; `retourner` (mpk);

$\mathsf{Extract}(\mathsf{msk}, \mathsf{ID})$:
choisir $r \stackrel{R}{\leftarrow} \mathbb{Z}_p$;
`retourner` $(\mathsf{usk}_1, \mathsf{usk}_2) = (r, (hg^{-r})^{1/(\alpha - \mathsf{ID})})$

$\mathsf{Encaps}(\mathsf{mpk}, \mathsf{ID})$:
choisir $r \stackrel{R}{\leftarrow} \mathbb{Z}_p$;
calculer $C = (F_{\mathsf{mpk}}(\mathsf{ID})^s, e(g,g)^s)$ et $K = e(g,h)^s$;
`retourner` (K, C) ;

$\mathsf{Decaps}(\mathsf{usk}_{\mathsf{ID}}, C)$: calculer $K = e(g,h)^s$;

$$\begin{aligned} K &= e(g^s, h) \cdot e(g,g)^{-sr} \cdot e(g,g)^{sr} \\ K &= e(g^s, hg^{-r}) \cdot e(g,g)^{sr} \\ K &= e(g^{\alpha - \mathsf{ID}}, hg^{-r})^{s/(\alpha-\mathsf{ID})} \cdot e(g,g)^{s \cdot r} \\ K &= e(c_1, \mathsf{usk}_2) \cdot {c_2}^{\mathsf{usk}_1}. \end{aligned}$$

`retourner` $K = e(c_1, \mathsf{usk}_2) \cdot {c_2}^{\mathsf{usk}_1}$.

FIG. 4.3 – Schéma $\mathcal{IB}$-$\mathcal{KEM}$ de Gentry, où la fonction F_{mpk} est définie par $F_{\mathsf{mpk}}(\mathsf{ID}) = g_1 \cdot g^{-\mathsf{ID}}$.

Nous montrons la sécurité sémantique et l'anonymat par une expérience hybride. Nous définissons trois jeux hybrides Jeu 1, Jeu 2, Jeu 3 pour lesquels le challenge diffère. Le but est de montrer que les challenges sont indistinguables sauf s'il existe un attaquant polynomial contre le problème q-T-ABDHE$_{\mathbb{G},\mathbb{G}_T}$.

1. Jeu 1 : on renvoie la paire chiffré/clé challenge $(C = (c_1, c_2), K)$ comme définie par la construction.
2. Jeu 2 : on renvoie la paire chiffré/clé challenge $(C = (c_1, \tilde{c_2}), K)$, où $\tilde{c_2}$ est aléatoire dans $\mathbb{G}_T$ et indépendant de K.
3. Jeu 3 : on renvoie la paire chiffré/clé challenge $(C = (\tilde{c_1}, \tilde{c_2}), K)$, où $\tilde{c_1}, \tilde{c_2}$ sont indépendants de K.

Démonstration: Soit $(g, g', {g'}^{\alpha^{q+2}}, g_1 = g^\alpha, \cdots, g_q = g^{\alpha^q}, Z)$ une instance du problème q-T-ABDHE$_{\mathbb{G},\mathbb{G}_T}$, on note $g_{q+1} = g^{\alpha^{q+1}}$. Il s'agit de décider si Z est égale $e(g_{q+1}, g')$, ou si Z est un élément aléatoire de $\mathbb{G}_T$.

Initialisation : on commence par choisir un polynôme aléatoire de $\mathbb{Z}_p[x]$, $P(x)$ de degré $q_{\mathsf{ID}} = q - 1$. On définit $h = g^{P(\alpha)}$, qui peut être calculé à partir des éléments $g_1 = g^\alpha, \cdots, g_q = g^{\alpha^q}$ connu du simulateur ; h est uniformément distribué dans $\mathbb{G}$, comme c'est la cas pour le schéma. On pose $\mathsf{mpk} = (g, g_1, h)$;

Réponse aux requêtes Extract : on supposera par la suite que $\mathsf{ID} \neq \alpha$, car sinon on a facilement un attaquant $\mathcal{B}$ capable de résoudre le problème q-T-ABDHE$_{\mathbb{G},\mathbb{G}_T}$. On répond à une requête pour une identité $\mathsf{ID} \neq \alpha$ de la manière suivante :

$$\mathsf{usk}_{1,\mathsf{ID}} = P(\mathsf{ID}) \text{ et } \mathsf{usk}_{2,\mathsf{ID}} = g^{f_{\mathsf{ID}}(\alpha)}, \text{ où } f_{\mathsf{ID}}(x) = \frac{P(x) - P(\mathsf{ID})}{x - \mathsf{ID}}.$$

Comme P est un polynôme uniformément distribué de degré $q-1$, les valeurs $P(\mathsf{ID})$ ont une distribution identique à celle définie par la construction (ou bien ID est l'une des q requêtes, ou bien $\mathsf{ID} \in \{\alpha, \mathsf{ID}_0, \mathsf{ID}_1\}$) ;

Challenge : l'attaquant retourne deux identités ID_0 et ID_1 (pour lesquelles aucune requête Extract n'a été demandée). On supposera $\mathsf{ID}_0, \mathsf{ID}_1 \neq \alpha$, sinon on a facilement un attaquant contre le problème q-T-ABDHE$_{\mathbb{G},\mathbb{G}_T}$. On commence par choisir un bit b. On définit un jeu auxiliaire, Jeu A où on calcule (C, K) de la manière suivante :

1. on définit d'abord un polynôme f_{ID_b} tel que $f_{\mathsf{ID}_b}(x) = \frac{x^{q+2} - {\mathsf{ID}_b}^{q+2}}{x - \mathsf{ID}_b}$: il s'agit d'un polynôme de degré $q+1$ de coefficient dominant 1. Notons $f_0, \cdots, f_j, \cdots, f_q$ ces autres coefficients ;
2. on définit alors le chiffré et la clé challenge associée par :

$$c_1 = {g'}^{\alpha^{q+2} - {\mathsf{ID}_b}^{q+2}} \qquad c_2 = Z \cdot e(g', \prod_{j=0}^{q} g^{f_j \alpha^j}) \qquad K = e(g', h)^{f_{\mathsf{ID}_b}(\alpha)}$$

Remarquons que nous pouvons calculer le chiffré et la clé associée puisque leur calcul fait seulement appel à la connaissance des termes ${g'}^{\alpha^{q+2}}$ et $g^{f_j \cdot \alpha^j}$ pour $j < q+1$.

– si $Z = e(g^{\alpha^{q+1}}, g')$, alors en posant $s = (\log_g g') \cdot f_{\mathsf{ID}_b}(\alpha)$, on remarque $C = (c_1, c_2)$ est bien l'encapsulation correspondant à la clé K, puisque :

$$c_1 = g^{\frac{s \cdot (\alpha^{q+2} - {\mathsf{ID}_b}^{q+2})}{f_{\mathsf{ID}_b}(\alpha)}} = g^{s \cdot (\alpha - \mathsf{ID}_b)} \qquad c_2 = e(g,g)^s \text{ et } K = e(g,h)^s,$$

avec un aléa s parfaitement aléatoire (puisque $\log_g g'$ est parfaitement aléatoire). La distribution du challenge (C, K) du jeu A est identique à celle définie par le jeu 1.

– autrement, si $Z \neq e(g^{\alpha^{q+1}}, g')$, c_2 et K sont complètement indépendants, la distribution du challenge (C, K) du jeu A est identique à celle définie par le jeu 2.

À présent, on définit un nouveau jeu auxiliaire, Jeu B pour lequel on redéfinit le challenge : on choisit un bit c aléatoire puis on calcule $\mathsf{usk}_{1,\mathsf{ID}_b}$ et $\mathsf{usk}_{2,\mathsf{ID}_b}$ exactement comme on le ferait lors d'une réponse à une requête d'extraction pour ID_b. Le chiffré challenge et la clé associée (C, K_c) sont alors définis de la manière suivante :

$$c_1 = {g'}^{\alpha^{q+2} - {\mathsf{ID}_b}^{q+2}} \qquad c_2 \overset{R}{\leftarrow} \mathbb{G}_T \qquad K_c = e(c_1, \mathsf{usk}_{2,\mathsf{ID}_b}) \cdot \left(Z \cdot e(g', \prod_{j=0}^{q} g^{f_j \alpha^j}) \right)^{\mathsf{usk}_{1,\mathsf{ID}_b}}$$

De manière similaire, on a une caractérisation sur la distribution du challenge :

– si $Z = e(g^{\alpha^{q+1}}, g')$, alors en posant $s = (\log_g g') \cdot f_{\mathsf{ID}_b}(\alpha)$, nous remarquons que

$$c_1 = g^{\frac{s \cdot (\alpha^{q+2} - {\mathsf{ID}_b}^{q+2})}{f_{\mathsf{ID}_b}(\alpha)}} = g^{s \cdot (\alpha - \mathsf{ID}_b)} \qquad c_2 \overset{R}{\leftarrow} \mathbb{G}_T \text{ et } K_c = e(g,h)^s,$$

avec un aléa s parfaitement aléatoire. La distribution du challenge (C, K_c) du jeu B est identique à celle définie par le jeu 2.

– autrement, si $Z \neq e(g^{\alpha^{q+1}}, g')$, c_1 et K_c sont complètement indépendants, la distribution du challenge (C, K_c) du jeu B est identique à celle définie par le jeu 3.

Réponse aux requêtes Extract **:** on répond aux requêtes d'extraction comme à la phase 1 ;

Réponse : l'attaquant $\mathcal{A}$ retourne sa réponse b' et c'.

Analyse de probabilité :

1. si $Z = e(g^{\alpha^{q+1}}, g')$, alors la paire (C, K_c) a une distribution identique à celle d'une paire associée à l'identité ID_b définie par la construction. Dans ce cas, la probabilité qu'a $\mathcal{B}$ de gagner est au moins égale à la probabilité de deviner les bits b et c plus l'avantage de $\mathcal{A}$, c.à.d. $\epsilon' + \frac{1}{4}$. Dans ce cas,

$$\Pr[\mathsf{Exp}^{q\text{-t-abdhe-1}}_{\mathcal{IB}\text{-}\mathcal{KEM},\mathcal{B}}(\lambda)] \leqslant \mathsf{Adv}^{\mathsf{anon\text{-}ind\text{-}cpa}}_{\mathcal{IB}\text{-}\mathcal{KEM},\mathcal{A}}(\lambda) + \frac{1}{4} = \epsilon' + \frac{1}{4}$$

2. sinon, on a Z parfaitement aléatoire dans $\mathbb{G}_T$. Le chiffré est complètement indépendant de b et de c, sauf si $c_2 = e(c_1, g)^{\alpha - \mathsf{ID}_b}$, pour $b = 0$ ou $b = 1$. Sachant que :

$$\Pr[c_2 = e(c_1, g)^{\alpha - \mathsf{ID}_0} \cup c_2 = e(c_1, g)^{\alpha - \mathsf{ID}_1}] = \frac{2}{p}, \text{ on a :}$$

$$\Pr\left[\mathsf{Exp}^{q\text{-t-abdhe}-0}_{\mathcal{IB}\text{-}\mathcal{KEM},\mathcal{B}}(\lambda) = 1\right] \leqslant \frac{1}{4} - \frac{2}{p}$$

Finalement, on a :

$$\mathsf{Adv}^{q\text{-t-abdhe}}_{\mathcal{IB}\text{-}\mathcal{KEM},\mathcal{B}}(\lambda) \leqslant \epsilon' - \frac{2}{p}$$

□

Performance du schéma

Afin de permettre une meilleure comparaison avec les autres schémas au prochain paragraphe, on se limitera à l'étude des paramètres du schéma résistant aux attaques à clairs choisis.

Taille des paramètres : un des avantages majeurs est la taille des paramètres publics (3 éléments de $\mathbb{G}$ et 5 pour le schéma résistant aux attaques à chiffrés choisis) comparés aux autres schémas sans oracles aléatoires et résistant aux attaques à identités non orientées.

Finesse de la réduction : on remarque que le temps t de $\mathcal{B}$ dépend du temps nécessaire pour répondre aux requêtes d'extraction. On a donc $t = t' + \mathcal{O}(\tau \cdot q_{\mathsf{ID}} \times q)$, où q_{ID} est le nombre de requêtes, q est le degré du polynôme $f_{\mathsf{ID}_b}(\alpha)$ et où τ est le temps nécessaire à une exponentiation dans $\mathbb{G}$.

Comme le fait remarquer Gentry, on ne peut pas réellement dire que la réduction est plus fine que d'autres réductions associées à une hypothèse "plus" standard que l'hypothèse q-$\mathsf{T\text{-}ABDHE}_{\mathbb{G},\mathbb{G}_T}$, comme l'hypothèse $\mathsf{DBDH}_{\mathbb{G},\mathbb{G}_T}$, car d'une part la difficulté du problème sous-jacent dépend d'un paramètre qui est censé évoluer (le nombre de requêtes $\mathsf{Extract}$), et d'autre part on ne connaît pas très bien la difficulté relative de l'hypothèse utilisée. Dans le modèle générique, il semblerait que le problème q-$\mathsf{T\text{-}ABDHE}_{\mathbb{G},\mathbb{G}_T}$ est plus facile à résoudre que le problème $\mathsf{DBDH}_{\mathbb{G},\mathbb{G}_T}$ [29]. Mais en réalité, il peut très bien exister un algorithme sous-exponentiel (et non générique) pour résoudre ces problèmes. En supposant que ces deux problèmes sont de difficulté égale, Gentry montre que plus une réduction sous le problème q-$\mathsf{T\text{-}ABDHE}_{\mathbb{G}}$ est fine, plus le système obtenu est efficace. Voici l'idée du raisonnement : on fixe q le degré du polynôme. On suppose que l'attaquant $\mathcal{A}$ a un avantage ϵ' et une complexité égale à $t' = 2^{100}$. On note λ le paramètre de sécurité correspondant. On suppose

Schéma	BF	KS	BW	Gentry	Représentation (en bits)	
Taille de mpk					SS/ 80 bits	MNT/ 80 bits
$\mathbb{G}$	2	2	7	3	1	1
$\mathbb{G}_T$			1		1024	1026
Complexité Encaps					Coût relatif[1]	
Exp/Dble Exp. ds $\mathbb{G}$	1	0/1	4/1	0/1	1	1
Exp ds $\mathbb{G}_T$	1	1	1	2	4	36
Haché dans $\mathbb{G}$	1				1	36
Couplage	1				20	150
Complexité Decaps						
Exp ds $\mathbb{G}/\mathbb{G}_T$				0/1		
Couplage	1	1	5	1		

FIG. 4.4 – Performance des schémas $\mathcal{IB}$-$\mathcal{KEM}$ anonymes présentés en terme de taille des paramètres publics, complexité en temps des algorithmes d'encapsulation et de décapsulation.

que le nombre de requêtes q_{ID} est tel que $q_{\mathsf{ID}} < 2^{30}$. Alors la complexité de $\mathcal{B}$ est $t = t' + \mathcal{O}(\tau \cdot q_{\mathsf{ID}} \times q)$. Et puisque $\tau \cdot q_{\mathsf{ID}} \times q = 2^{90} \ll 2^{100}$, t' conditionne le paramètre de sécurité. Et si on suppose qu'une réduction perd un facteur q, on doit choisir le paramètre de sécurité λ de telle sorte à avoir $t \approx 2^{130}$. Nous obtenons donc un paramètre de sécurité plus important (et donc des exponentiations plus coûteuses).

Comparaison des schémas

Nous donnons figure 4.4 les performances des schémas $\mathcal{IB}$-$\mathcal{KEM}$ présentés et figure 4.5 les paramètres permettant d'apprécier la finesse des réductions pour la sécurité sémantique (face aux attaques à messages choisis) et l'anonymat. Pour le premier tableau, la partie droite donne une idée de la complexité en pratique pour les paramètres de sécurité $\lambda = 80$ bits[2] dans le cas des courbes de Type 1 (de grande caractéristique) décrites dans [31] et de type 2, non super-singulières introduites par Miyaji *et al.* [100], puis décrites et baptisées courbes MNT dans [33].

4.3 Extension de la notion d'anonymat

4.3.1 Notion d'anonymat KwrtA

Notons que même si la construction de schémas anonymes n'est pas une tâche facile, la notion d'anonymat n'est en soit pas si forte puisqu'elle ne capture pas l'anonymat vis à vis de l'autorité. Dans cette section, nous introduisons une nouvelle notion de sécurité en considérant l'autorité comme attaquant potentiel. Par la suite nous appellerons cette notion *Anonymat vis à vis de l'autorité*, (ou *Key Anonymity with respect to the Authority*) et nous parlerons d'*anonymat* au sens KwrtA. Notons que la notion d'anonymat classique ne

[1] Dans [34], on distingue les exponentiations en base fixe de celles en base quelconque, puisque ces dernières sont 5 fois plus coûteuses. Ici, on donne juste une approximation et on reporte le lecteur à [34] pour une expertise détaillée de la complexité.

[2] Ce choix requiert une taille de groupes au moins de 160 bits, et un corps fini de taille au moins 1024 bits.

Schéma	BF [31]	KS [56]	BW, [36]	Gentry, [71]
Hypothèse utilisée				
Sécurité sémantique Anonymat	$\mathsf{DBDH}_{\mathbb{G},\mathbb{G}_T}$ Model. de $\mathcal{H}$	(s-ID) q-$\mathsf{DBDHI}_{\mathbb{G},\mathbb{G}_T}$ Model. de $\mathcal{H}$	(s-ID) $\mathsf{DBDH}_{\mathbb{G},\mathbb{G}_T}$ $\mathsf{DLIN}_{\mathbb{G},\mathbb{G}_T}$	q-$\mathsf{T\text{-}ABDHE}_{\mathbb{G},\mathbb{G}_T}$ q-$\mathsf{T\text{-}ABDHE}_{\mathbb{G},\mathbb{G}_T}$
Finesse				
Facteur de sécurité séc. sém.	$q_{\mathcal{H}}$	$q_{\mathcal{H}}^2$	1	1

FIG. 4.5 – Hypothèses de sécurité et finesse de la réduction pour quelques-uns des schémas $\mathcal{IB}$-$\mathcal{KEM}$ anonymes. L'abréviation Model. de $\mathcal{H}$ signifie que la sécurité dépend du nombre de requêtes à l'oracle. Une estimation de $q_{\mathcal{H}}$ d'après [18] est 2^{50} en prenant $\lambda = 80$ pour paramètre de sécurité.

se généralise pas trivialement face à une autorité malhonnête : supposons que l'adversaire $\mathcal{A}$ génère les paramètres secrets et publics[3], il peut alors générer toutes les clés secrètes des utilisateurs. Pour une paire (K^*, c^*) donnée, l'attaquant peut évidemment vérifier à quelle identité cette paire correspond. Par conséquent, dans le jeu de sécurité, nous tronquons le challenge de manière à ne donner que c^* à l'attaquant. Cependant, pour justifier la pertinence de cette notion, certaines précautions doivent être prises. Considérons un exemple simple : nous soumettons un chiffré c^* à l'attaquant et ce dernier teste le déchiffrement pour une identité de son choix (il peut s'agir d'une identité quelconque ou d'une identité parmi deux ciblées). Il obtient une clé éphémère K. Pour le chiffrement $\mathcal{IB}$-$\mathcal{KEM}$, si la clé suit une distribution uniforme, notre nouvelle définition KwrtA se révèle alors tout à fait cohérente. Et puisqu'un schéma $\mathcal{IB}$-$\mathcal{KEM}$ est traditionnellement associé à un schéma $\mathcal{DEM}$, il convient de justifier la notion d'anonymat au sens KwrtA dans le contexte du chiffrement hybride à base d'identités. Considérons un chiffré global $C = C_0 \| C_1$, où C_0 est l'encapsulation d'une clé K (non connue de l'attaquant) sous une certaine identité et C_1 est le chiffré obtenu par application du $\mathcal{DEM}$ sur un certain message m avec la clé K. Supposons que le schéma $\mathcal{IB}$-$\mathcal{KEM}$ soit anonyme au sens KwrtA. Nous devons alors nous assurer que le chiffré C_1 ne laisse pas fuir d'information sur l'identité associée à C_0, c.à.d. que C_1 ne fournit pas de test additionnel permettant de sélectionner une identité associée au chiffré C_0. Si la clé éphémère encapsulée suit une distribution uniforme et si l'attaquant n'a pas de connaissance sur la distribution *a priori* du message m, il ne peut pas déterminer la validité d'une paire message/identité : pour un schéma $\mathcal{DEM}$ approprié, la clé obtenue en décapsulant C_0 est uniforme et le déchiffrement de C_1 sous K ne révèle pas d'information sur l'identité associé à C_0. Autrement dit, le déchiffrement de C_1 n'apporte dans ce cas aucun biais sur K et donc sur l'identité.

Définition

Il est important de préciser que pour cette nouvelle notion, l'adversaire doit générer des paramètres publics mpk valides. Dans certains cas, il s'agit bien d'une valeur aléatoire ; mais parfois, si une certaine forme de redondance est présente dans la clé publique, la sécurité de tout le système peut être compromise. La clé mpk doit donc vérifier des propriétés de validité, définies par l'algorithme suivant :

$\mathsf{Valid}_{\mathsf{IBK}}(\mathsf{mpk})$ **:** cet algorithme prend en entrée les paramètres publics mpk, et vérifie s'ils satisfont les propriétés requises définies par le schéma.

[3]Ces derniers paramètres sont soumis à certaines conditions que nous préciserons.

Pour le concept de mécanisme d'*encapsulation de clés* basé sur l'identité, $\mathcal{IB}$-$\mathcal{KEM}$, nous définissons la notion d'*anonymat* au sens KwrtA :

Définition 14 (Notions de sécurité d'anonymat au sens KwrtA) *Considérons un attaquant $\mathcal{A}$ et un schéma $\mathcal{IB}$-$\mathcal{KEM}$ défini par $\mathcal{IB}$-$\mathcal{KEM} \stackrel{déf}{=} \langle \mathsf{Setup}, \mathsf{Extract}, \mathsf{Encaps}, \mathsf{Decaps} \rangle$. Nous définissons la notion de sécurité d'*Anonymat vis à vis de l'autorité *par l'expérience suivante :*

Expérience $\mathsf{Exp}^{\mathsf{kwrta-anon}\text{-}b}_{\mathcal{IB}\text{-}\mathcal{KEM},\mathcal{A}}(\lambda)$
$(\mathsf{mpk}, \mathsf{ID}_0, \mathsf{ID}_1, \mathsf{state}) \leftarrow \mathcal{A}_1(\texttt{FIND}, \lambda)$;
t.q. $\mathsf{Valid}_{\mathsf{IBK}}(\mathsf{mpk})$;
$b \stackrel{R}{\leftarrow} \{0,1\}; (K^*, c^*) \leftarrow \mathsf{Encaps}_{\mathsf{IBK}}(\mathsf{mpk}, \mathsf{ID}_b)$;
$b' \leftarrow \mathcal{A}_2(\texttt{GUESS}, c^*, \mathsf{state})$;
`retourner` b' ;

On définit l'avantage de l'attaquant $\mathcal{A}$ dans le jeu précédent par :

$$\mathsf{Adv}^{\mathsf{kwrta-anon}}_{\mathcal{IB}\text{-}\mathcal{KEM},\mathcal{A}}(\lambda) = \left| \Pr[\mathsf{Exp}^{\mathsf{kwrta-anon}\text{-}1}_{\mathcal{IB}\text{-}\mathcal{KEM},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{kwrta-anon}\text{-}0}_{\mathcal{IB}\text{-}\mathcal{KEM},\mathcal{A}}(\lambda) = 1] \right|$$

On dit qu'un schéma $\mathcal{IB}$-$\mathcal{KEM}$ est anonyme au sens KwrtA si l'avantage de $\mathcal{A}$ à décider à laquelle des deux identités engagées le chiffré challenge correspond est négligeable en le paramètre de sécurité.

Comparaison avec l'anonymat classique

Pour cette nouvelle notion d'anonymat KwrtA, l'adversaire connaît la clé maître permettant de générer les clés privées des utilisateurs. Cela dit, sa décision ne doit dépendre que du chiffré challenge c^*. Par conséquent, les deux notions d'anonymat ne sont pas vraiment comparables. De plus, si l'adversaire génère des paramètres publics mpk , nous devons pouvoir vérifier que sa génération a été faite correctement.

4.3.2 Analyse des schémas $\mathcal{IB}$-$\mathcal{KEM}$ au sens KwrtA

Dans cette partie, nous étudions les propriétés d'anonymat au sens KwrtA de quelques-uns des schémas présentés au paragraphe 4.2.2. Rappelons que F_{mpk} est une fonction définie sur l'ensemble des identités, à valeurs dans $\mathbb{G}$, et dépendant éventuellement d'un paramètre mpk.

Schéma de Boneh-Franklin [31]

Dans ce schéma, $\mathsf{msk} = s \stackrel{R}{\leftarrow} \mathbb{Z}_p$ et $\mathsf{mpk} = g^s$ (voir figure 4.1). La fonction F est indépendante de mpk, elle est à valeurs dans $\mathbb{G}$ et est modélisée par un oracle aléatoire dans l'analyse de sécurité. Le chiffré $c = g^r \in \mathbb{G}$ correspond à la clé $K = e(F(\mathsf{ID}), \mathsf{mpk})^r = \mathsf{BDH}_g(\mathsf{mpk}, c, F(\mathsf{ID})) = e(\mathsf{usk}_{\mathsf{ID}}, c)$, où $\mathsf{usk}_{\mathsf{ID}} = F(\mathsf{ID})^s = \mathsf{co\text{-}CDH}_g F(\mathsf{ID})(\mathsf{mpk}) \in \mathbb{G}$.

Puisque le chiffré est totalement indépendant de l'identité, ce schéma est anonyme au sens KwrtA de manière inconditionnelle.

Schéma de Boneh-Boyen [25]

Nous avons décrit ce schéma $\mathcal{IBE}$ figure 3.4. Nous pouvons dériver le schéma $\mathcal{IB}$-$\mathcal{KEM}$ associé : avec $\alpha \xleftarrow{R} \mathbb{Z}_p$, $g, g_2, h \xleftarrow{R} \mathbb{G}$, on a : $\mathsf{mpk} \stackrel{\text{déf}}{=} (g, g_1 = g^\alpha, g_2, h)$, alors que msk vaut g_2^α. La fonction F_{mpk} est définie par $F_{\mathsf{mpk}}(\mathsf{ID}) = g_1^{\mathsf{ID}} \cdot h$. Le chiffré $c = (g^s, F_{\mathsf{mpk}}(\mathsf{ID})^s)$ correspond à la clé :

$$K = e(g_1, g_2)^s = e(c_1, \mathsf{usk}_2)/e(\mathsf{usk}_1, c_2),$$

si $\mathsf{usk}_{\mathsf{ID}}$ est définie par $(g^r, \mathsf{msk} \cdot F_{\mathsf{mpk}}(\mathsf{ID})^r)$, pour un certain $r \xleftarrow{R} \mathbb{Z}_p$.

Comme pour le schéma précédent, la sécurité sémantique repose sur l'hypothèse $\mathsf{DBDH}_{\mathbb{G},\mathbb{G}_T}$ dans le modèle standard, mais face les attaques à clairs choisis et à identités orientées, avec accès aux requêtes $\mathsf{Extract}$. Rappelons que le schéma de signature sous-jacent est résistant aux contrefaçons, face aux attaques à messages choisis sous l'hypothèse $\mathsf{CBDH}_{\mathbb{G},\mathbb{G}_T}$. Cependant, la redondance présente dans le chiffré permet de tester si ce chiffré est attribué à une identité donnée, ce schéma n'est donc pas anonyme : en effet, n'importe qui peut tester pour un candidat ID et un chiffré $c = (c_1, c_2)$, si

$$e(c_1, F_{\mathsf{mpk}}(\mathsf{ID})) \stackrel{?}{=} e(c_2, g)$$

Puisque cette attaque ne nécessite pas de clé K candidate, le schéma n'est *a fortiori* pas anonyme au sens KwrtA.

Schéma de Gentry [71]

Rappelons brièvement ce schéma, décrit figure 4.3. Pour définir les paramètres publics, on choisit $g, h \xleftarrow{R} \mathbb{G}$ et $\alpha \xleftarrow{R} \mathbb{Z}_p$ et on pose $\mathsf{mpk} = (g, g_1 = g^\alpha, h)$ et $\mathsf{msk} = \alpha$. La fonction F_{mpk} est définie par $F_{\mathsf{mpk}}(\mathsf{ID}) = g_1 \cdot g^{-\mathsf{ID}} = g^{\alpha - \mathsf{ID}}$. Le chiffré $c = (F_{\mathsf{mpk}}(\mathsf{ID})^s, e(g,g)^s)$ correspond à l'*encapsulation* de la clé $K = e(g,h)^s$. En posant $(\mathsf{usk}_1, \mathsf{usk}_2) = (r, (hg^{-r})^{1/(\alpha - \mathsf{ID})})$, pour un exposant r aléatoire dans $\mathbb{Z}_p$, on a :

$$\begin{aligned} K &= e(g,h)^s = e(g^s, h) \cdot e(g,g)^{-sr} \cdot e(g,g)^{sr} = e(g^s, hg^{-r}) \cdot e(g,g)^{sr} \\ &= e(g^{\alpha - \mathsf{ID}}, hg^{-r})^{s/(\alpha - \mathsf{ID})} \cdot e(g,g)^{sr} = e(c_1, \mathsf{usk}_2) \cdot {c_2}^{\mathsf{usk}_1}. \end{aligned}$$

Comme nous l'avons vu au paragraphe 4.2.2, ce schéma est sémantiquement sûr et anonyme contre les attaques à clairs choisis avec accès aux requêtes d'extraction sous l'hypothèse q-T-$\mathsf{ABDHE}_{\mathbb{G},\mathbb{G}_T}$.

Ce schéma n'est pas anonyme au sens KwrtA. En effet, la structure bilinéaire combinée à la redondance du chiffré permet de tester une identité ID' : connaissant α, un attaquant $\mathcal{A}$ peut tester si

$${c_2}^{\alpha - \mathsf{ID}'} = e(g,g)^{s(\alpha - \mathsf{ID}')} \stackrel{?}{=} e(c_1, g) = e(g^{s(\alpha - \mathsf{ID}')}, g)$$

Schéma de Boyen-Waters [36]

Nous avons déjà donné une description de ce schéma $\mathcal{IBE}$ figure 4.2. Rappelons juste la forme du chiffré. La fonction F_{mpk} est définie par $F_{\mathsf{mpk}}(\mathsf{ID}) = g_0 \cdot {g_1}^{\mathsf{ID}}$. Pour encapsuler une clé, on choisit un $s, s_1, s_2 \xleftarrow{R} \mathbb{Z}_p$ et on pose $K = \Omega^s$. Puis on calcule $c = (c_0, c_1, c_2, c_3, c_4)$, avec $c_0 = F_{\mathsf{mpk}}(\mathsf{ID})^s$, $c_1 = v_1^{s - s_1}$, $c_2 = v_2^{s_1}$, $c_3 = v_3^{s - s_2}$, et $c_4 = v_4^{s_2}$.

Rappelons que ce schéma est *sémantiquement sûr* sous l'hypothèse $\mathsf{DBDH}_{\mathbb{G},\mathbb{G}_T}$, et *anonyme* sous l'hypothèse linéaire décisionnelle. Étant donné que notre schéma n'est pas

comparable à celui-ci, nous ne donnerons pas plus de détails sur les preuves. Le lecteur est invité à consulter l'article [36] pour plus de précisions sur la preuve de sécurité de ce schéma. Par ailleurs, ce schéma n'est pas anonyme au sens *KwrtA* : en effet, la connaissance de la clé secrète msk permet de tester si le triplet c_0, c_1, c_2 et/ou c_0, c_3, c_4 est un triplet linéaire en base v_0, v_1, v_2 et v_0, v_3, v_4 respectivement.

Remarquons que pour tous les schémas décrits, les paramètres publics sont constitués d'éléments indépendants dans les groupes appropriés. Le test de validité $\mathsf{Valid}_{\mathsf{IBK}}$ est donc trivial.

4.3.3 Un candidat

On remarque qu'aucun des schémas précédents ne satisfait la notion d'anonymat *KwrtA* dans le modèle standard. Mais surtout, pour l'application de la prochaine partie, nous introduirons une nouvelle notion de sécurité : malheureusement, aucun des schémas ne satisfait ces deux notions à la fois.

Description

Dans ce paragraphe, nous présentons un schéma proposé an collaboration avec David Pointcheval dans l'article [85].

$\mathsf{Setup}_{\mathsf{IBK}}$ **:** l'algorithme Setup choisit deux générateurs aléatoires $g, h \in \mathbb{G}$, un exposant $\omega \in \mathbb{Z}_p$. On définit la clé secrète maître par $\mathsf{msk} = \omega$. Les paramètres publics sont définis par : $\mathsf{mpk} = (g, g_1 = g^\omega, h)$. La fonction F_{mpk} est définie par : $F_{\mathsf{mpk}}(\mathsf{ID}) = g_1 \cdot g^{\mathsf{ID}} = g^{\omega+\mathsf{ID}}$. Comme précédemment, les paramètres publics sont constitués d'éléments indépendants dans les groupes appropriés. Le test de validité $\mathsf{Valid}_{\mathsf{IBK}}(\mathsf{mpk})$ est donc trivial.

$\mathsf{Extract}_{\mathsf{IBK}}(\mathsf{mpk}, \mathsf{ID})$ **:** l'autorité extrait la clé privée correspondant à l'identité ID par : $\mathsf{usk}_{\mathsf{ID}} = h^{1/(\omega+\mathsf{ID})}$.

$\mathsf{Encaps}_{\mathsf{IBK}}(\mathsf{mpk}, \mathsf{ID})$ **:** pour générer une clé éphémère pour l'identité ID, l'algorithme choisit un exposant aléatoire $r \in \mathbb{Z}_p$, et crée une paire clé éphémère/chiffré en calculant : $c = F(\mathsf{ID})^r$, qui correspond à la clé $K = e(g, h)^r$.

$\mathsf{Decaps}_{\mathsf{IBK}}(\mathsf{usk}_{\mathsf{ID}}, c)$ **:** l'algorithme de déchiffrement extrait la clé éphémère du chiffré c en calculant : $K = e(\mathsf{usk}_{\mathsf{ID}}, c)$.

Propriétés et sécurité

Notre candidat satisfait différentes notions de sécurité : *sécurité sémantique, anonymat classique et anonymat au sens* *KwrtA*, et *non malléabilité d'identités*, une notion que nous préciserons au chapitre 8. Ce paragraphe est consacré à la preuve de ces résultats.

Correction

la procédure de déchiffrement se vérifie par un calcul simple :

$$K = e(\mathsf{usk}_{\mathsf{ID}}, c) = e(h^{1/(\omega+\mathsf{ID})}, g^{r(\omega+\mathsf{ID})}) = e(h, g)^r$$

Sécurité sémantique et Anonymat

Il est important de préciser que nous n'avons pas besoin d'être capable de simuler les oracles d'extraction de clés et de déchiffrement pour l'application de la prochaine partie. La notion de *sécurité sémantique faible*, (voir remarque 4) suffit.

Théorème 5 *La* sécurité sémantique faible *de notre schéma contre les* attaques à messages choisis *et contre les* attaques à identités orientées *repose sur le problème* $\mathsf{DBDH}_{\mathbb{G},\mathbb{G}_T}$ *dans le* modèle standard.

Démonstration:
Supposons par l'absurde qu'il existe un attaquant $\mathcal{A}$ capable de "casser" la *sécurité sémantique faible* du schéma paragraphe 4.3.3 en un temps t' avec un avantage non négligeable ϵ'. On construit alors un algorithme $\mathcal{B}$ capable de résoudre le problème $\mathsf{DBDH}_{\mathbb{G},\mathbb{G}_T}$ en temps t. Soit $(u, A = u^a, B = u^b, C = u^c, V)$ une instance du problème $\mathsf{DBDH}_{\mathbb{G},\mathbb{G}_T}$, où V est soit $e(u,u)^{abc}$ ou soit un élément aléatoire de $\mathbb{G}_T$. Soit ID^* l'identité retournée par l'attaquant $\mathcal{A}$.

Initialisation : on calcule d'abord :

$$g = A = u^a \quad h = C = u^c = g^{c/a} \quad g_1 = u^t \cdot A^{-\mathsf{ID}^*} = u^{t-a\mathsf{ID}^*}, \text{ et } c = B.$$

Ce qui définit de manière implicite $\mathsf{msk} = t/a - \mathsf{ID}^*$, pour un t choisi aléatoirement $t \overset{R}{\leftarrow} \mathbb{Z}_p$;

Challenge : d'après ce qui précède

$$F_{\mathsf{mpk}}(\mathsf{ID}^*) = g_1 g^{\mathsf{ID}^*} = u^t \cdot A^{-\mathsf{ID}^*} \cdot A^{\mathsf{ID}^*} = u^t,$$

et l'aléa r du chiffré challenge est déterminé par : $c = F_{\mathsf{mpk}}(\mathsf{ID}^*)^r = u^{tr} = u^b = B$ et $r = b/t$. Et donc, la clé encapsulée correspondante est définie par :

$$K = e(h,g)^r = e(u^c, u^a)^{b/t} = e(u,u)^{abc/t}.$$

On définit la sortie du challenger par $(V^{1/t}, c)$;

Réponse : l'attaquant $\mathcal{A}$ retourne sa réponse b' et $\mathcal{B}$ renvoie $b = b'$.

Analyse de probabilité :

1. Si $V = e(u,u)^{abc}$, le chiffré challenge est bien l'*encapsulation* de la clé K challenge associée, comme définie par la construction. Dans ce cas, la probabilité que $\mathcal{B}$ gagne est égale à la probabilité que $\mathcal{A}$ devine le bit b plus $\frac{1}{2}$:
2. et sinon, la paire challenge est constituée de deux éléments parfaitement aléatoires dans $\mathbb{G}_T \times \mathbb{G}$. Et la probabilité de gagner est $\frac{1}{2}$.

Un adversaire capable de casser la sécurité sémantique sans appel à l'oracle d'extraction permet de décider si V est bien le Diffie-Hellman bilinéaire de A, B et C en base $u = g$ ou bien une valeur aléatoire.

□

Pour prouver la notion de *sécurité sémantique forte*, c.à.d. avec accès aux requêtes d'extraction, mais aussi à identités non orientées, nous devons être capable de simuler ces requêtes, qui nécessitent des entrées supplémentaires. Mais d'abord, nous modifions le schéma de telle sorte à utiliser $\mathcal{H}(\mathsf{ID})$, à la place de ID, dans la description précédente, où $\mathcal{H}$ est un oracle aléatoire [16] à valeurs dans $\mathbb{Z}_p$.

Théorème 6 *La* sécurité sémantique *de notre schéma (où on utilise* $\mathcal{H}(\mathsf{ID})$ *à la place de* ID*) contre les* attaques à messages choisis *et contre les* attaques à identités non orientées *repose sur le problème* Successive-Power Version*, dans le modèle de l'*oracle aléatoire.

Démonstration:
Supposons par l'absurde qu'il existe un attaquant $\mathcal{A}$ capable de "casser" la sécurité sémantique du schéma présenté au paragraphe 4.3.3 que nous avons modifié, en un temps t' avec un avantage non négligeable ϵ'. Supposons que cet attaquant fait au plus q_{ID} requêtes d'extraction. On construit alors un algorithme $\mathcal{B}$ capable de résoudre le problème q-$\mathsf{SP\text{-}DBDH}_{\mathbb{G},\mathbb{G}_T}$ en temps t. Soient u, $A = u^a$, $B = u^b$, $C = u^c$, $C_i = C^{1/a^i}$, pour $i = 1, \ldots, q$, et $V \in \mathbb{G}_T$ des instances du problème q-$\mathsf{SP\text{-}DBDH}_{\mathbb{G},\mathbb{G}_T}$, où V est soit le Diffie-Hellman bilinéaire de A, B et C en base u, $e(u,u)^{abc}$, soit un élément aléatoire de $\mathbb{G}_T$.

Initialisation : on calcule d'abord $\{V_i = e(u,u)^{bc/a^i}\}_{i=0\ldots q}$, sachant que $V_0 = e(B,C)$, et $V_i = e(B,C_i)$, pour $i = 1, \ldots, q$. Puis, on choisit des éléments aléatoires $x_1, \ldots, x_q \overset{R}{\leftarrow} \mathbb{Z}_p^*$, et on pose $P(X) = \prod(tX + x_i)$, qui est un polynôme de degré q. On pose alors $g = A = u^a$ et $g_1 = u^t \cdot A^{-x^*}$, pour $t, x^* \overset{R}{\leftarrow} \mathbb{Z}_p$, ce qui définit de manière implicite $\mathsf{msk} = t/a - x^*$. Puis, on pose $h = C^{P(1/a)} = u^{cP(1/a)}$, qui peut être calculé facilement à partir de C, C_1, $\ldots$, C_q.

Phase 1 : réponse aux requêtes Extract **:** tout d'abord, on répondra à toutes les requêtes par un exposant $x^* + x_i$, ou x^* (pour une requête unique choisie aléatoirement) : on espère assigner x^* à $\mathcal{H}(\mathsf{ID}^*)$, l'identité sur laquelle l'adversaire s'est engagée. La probabilité qu'un tel événement se produise est $1/q$. Supposons qu'il a lieu.

– Par définition, on a $F_{\mathsf{mpk}}(\mathsf{ID}^*) = g_1 g^{\mathcal{H}(\mathsf{ID}^*)} = u^t \cdot A^{-x^*} \cdot A^{x^*} = u^t$;
– et pour toutes les autres identités, on a : $\mathcal{H}(\mathsf{ID}_j) = x_j$, et usk_j peut alors être calculée par :

$$h^{1/(\mathsf{mpk}+x^*+x_j)} = C^{P(1/a)/(\mathsf{mpk}+x^*+x_j)} = C^{P(1/a)/(t/a+x_j)} = C^{P_j(1/a)},$$

où P_j est un polynôme de degré $q-1$. Alors, nous pouvons calculer usk_j à partir de C, C_1, $\ldots$, C_{q-1}. On peut de cette manière simuler les oracles d'extraction ;

Challenge : comme précédemment, on définit le chiffré challenge par $c^\star = B = u^b = F_{\mathsf{mpk}}(\mathsf{ID}^*)^r$ pour $r = b/t$. La clé *encapsulée* correspondante est donc :

$$K^\star = e(g,h)^r = e(u^a, u^{cP(1/a)})^{b/t} = (e(u,u)^{abc})^{P(1/a)/t}.$$

Développons le polynôme P : $P(X) = \sum_{i=0}^{i=q} p_i X^i$, et alors

$$K^\star = e(u,u)^{abc \cdot p_0/t} \times \prod_{i=1}^{i=q} e(u,u)^{bc/a^{i-1} \cdot p_i/t} = \left(e(u,u)^{abc}\right)^{p_0/t} \times \prod_{i=1}^{i=q} V_{i-1}^{p_i/t}.$$

On retourne le challenge :

$$\left(V^{p_0/t} \times \prod_{i=1}^{i=q} V_{i-1}^{p_i/t}, c^\star\right).$$

Phase 2 réponse aux requêtes Extract **:** comme à la Phase 1 ;

Réponse : l'attaquant $\mathcal{A}$ retourne sa réponse b' et $\mathcal{B}$ renvoie $b = b'$.

Analyse de probabilité :

1. si $V = e(u,u)^{abc}$, la clé correspondante est donnée par : $V^{p_0/t} \times \prod_{i=1}^{i=q} V_{i-1}^{p_i/t}$. On remarque que $p_0 = \prod x_i \neq 0 \bmod p$: le chiffré challenge est bien l'*encapsulation* de la clé K challenge associée, comme défini par la construction. Dans ce cas, la probabilité que $\mathcal{B}$ gagne est égale à la probabilité qu'a $\mathcal{A}$ de deviner le bit b plus $\frac{1}{2}$;
2. sinon, la paire challenge $(K^\star, c^\star)$ est constituée de deux éléments parfaitement aléatoires dans $\mathbb{G}_T \times \mathbb{G}$: la probabilité de gagner est $\frac{1}{2}$.

Un adversaire capable de casser la *sécurité sémantique forte* sans appel à l'oracle d'extraction permet de décider si V est bien le Diffie-Hellman bilinéaire ou bien une valeur aléatoire. On casse alors le problème q-$\mathsf{SP\text{-}DBDH}_{\mathbb{G},e}$. □

Anonymat La notion d'anonymat usuelle repose sur la même hypothèse que la sécurité sémantique, c.à.d. le problème *Successive Power Version*. On peut prouver ce résultat en changeant le format du challenge dans la preuve précédente exactement comme pour la preuve présentée au paragraphe 4.2.2. D'autre part, comme le chiffré $c = F(\mathsf{ID})^r$ est un élément aléatoire dans $\mathbb{G}$, quelle que soit l'identité ID, le schéma est anonyme au sens *KwrtA* de manière inconditionnelle.

Théorème 7 *Notre schéma est* anonyme au sens KwrtA *de manière inconditionnelle.*

Démonstration: Soit $\mathcal{A}$ un attaquant contre l'*anonymat au sens KwrtA* de notre candidat. Nous supposons qu'il est en possession d'une clé maître msk que nous ne connaissons pas. Il retourne deux identités $\mathsf{ID}_0, \mathsf{ID}_1$ et des paramètres publics mpk. Nous vérifions la validité de $\mathsf{mpk} = (g, g_1, h)$. Si la clé n'est pas valide, l'attaquant a un avantage nul ; sinon, nous choisissons un bit b et calculons $\mathsf{Encaps}_{\mathsf{IBK}}(\mathsf{mpk}, \mathsf{ID}_b) = (c, K)$. Nous lui renvoyons le chiffré c, où $c = F_{\mathsf{mpk}}(\mathsf{ID}_b)^r = (g_1 \cdot g^{\mathsf{ID}_b})^r$ pour un r aléatoire dans $\mathbb{Z}_p$. Il est évident que c est un élément parfaitement aléatoire dans $\mathbb{G}$: la connaissance de msk ne suffit pas à deviner le bit b avec un avantage significativement meilleur que $1/2$. □

Chapitre 5

Anonymat révocable

Dans ce chapitre, nous exposons le concept d'anonymat pour le chiffrement dans un contexte multi-utilisateurs. Nous introduisons la primitive de chiffrement de groupe proposé par Kiayas, Tsiounis et Yung dans [88], permettant d'obtenir un tel mécanisme. Pour commencer, nous rappellerons brièvement le concept de signature de groupe, qui peut être vu comme le dual du concept de chiffrement de groupe, excepté que dans ce deuxième cas, nous voulons masquer l'identité du destinataire d'un message et non plus celle du signataire. Nous mettrons également l'accent sur les mécanismes permettant la révocation de membres.

Par ailleurs, prouver l'appartenance d'un membre au groupe reste un objectif fondamental : nous étendons le modèle de sécurité en considérant des tentatives frauduleuses d'enregistrement au sein du groupe. Pour parvenir à la conception d'une primitive intégrant ces multiples fonctionnalités, nous introduirons les schémas de rechiffrement.

Sommaire

5.1 Primitives de groupe

Nous commençons par présenter les définitions et propriétés des signatures de groupe et du chiffrement de groupe. Nous pourrons alors établir des connexions entre ces deux primitives duales, et mettre en lumière les particularités de l'anonymat pour le chiffrement.

5.1.1 Signature de groupe

Les signatures de groupe ont été introduites en 1991 par Chaum et Van Heyst [54]. Ce procédé permet à des usagers de signer des messages au nom du groupe sans que la signature produite ne révèle l'identité du signataire. Néanmoins, on doit être capable de vérifier la validité de la signature et l'appartenance du signataire au groupe. De plus, en cas de conflit, une autorité attitrée doit être capable de lever cet anonymat.

En premier lieu, la motivation proposée par Chaum et Van Heyst était assez restrictive : une entreprise souhaite permettre à ces employés d'accéder à un service de manière anonyme. Au fil des années, l'usage des signatures de groupe s'est généralisé à divers contextes, comme les enchères électroniques [103] ou la monnaie électronique [45], par exemple.

Définition

Un schéma de signature de groupe met en jeu des utilisateurs, une autorité chargée d'enregistrer les membres, et un tiers de confiance chargé de la révocation des membres. Dans certains cas, ces deux autorités ne font qu'une entité.

Définition 15 *Un schéma de signature de groupe est défini par la donnée de cinq algorithmes :* $\mathsf{GSetup}, \mathsf{GJoin}, \mathsf{GSign}, \mathsf{GVerify}$ *et* GOpen *chacun défini par :*

GSetup **:** *étant donné un paramètre de sécurité* λ, GSetup *renvoie les paramètres communs du système* params, *deux paires clé publique/clé secrète* $(\mathsf{mpk}, \mathsf{msk})$ *et* $(\mathsf{pk}_O, \mathsf{sk}_O)$ *respectivement attribuée au manager et à l'autorité d'ouverture.*

GJoin **:** *il s'agit d'un protocole interactif* $\langle J_{\mathsf{ID}}, J_{GM} \rangle$ *entre un utilisateur* ID *et le manager à l'issue duquel l'utilisateur obtient un certificat d'appartenance au groupe* `cert`[1].

[1] On suppose que le certificat contient la clé secrète $\mathsf{sk}_{\mathsf{ID}}$

GSign : *il s'agit d'un algorithme probabiliste qui prend un entrée un message m, une clé secrète* $\mathsf{sk}_{\mathsf{ID}}$ *et un certificat* `cert`*. Cet algorithme retourne la signature du message,* σ*.*

GVerify : *étant donnés les paramètres publiques* mpk*, un message m et une signature σ, cet algorithme (déterministe) retourne* 1 *si σ est la signature du message m produite par l'un des membres et* 0 *sinon.*

GOpen : *étant donnés la clé* sk_O*, les paramètres* mpk*, un message m et une signature valide de m, σ, cet algorithme renvoie l'identité du signataire.*

Propriétés

Deux modèles de sécurité ont été proposés pour les schémas de signature de groupe. Le premier modèle, introduit en 2003 par Bellare *et al.* [13] (BMW) s'applique aux signatures statiques ; ce modèle a été étendu par Bellare *et al.* [21] aux signatures dynamiques (BSZ). Le premier modèle regroupe trois propriétés : la consistance, l'anonymat et la traçabilité ; en fait, les propriétés d'anonymat et de non-reliabilité sont unifiées en un seul jeu et la notion de traçabilité regroupe à la fois la notion de traçabilité classique, la résistance aux falsifications et aux coalitions. Pour le second modèle, on distingue quatre propriétés : la consistance, l'anonymat, la traçabilité et la non-diffamation. Nous présentons ci-dessous les différentes propriétés existantes de manière informelle. Pour mieux comprendre leur impact dans chacun des modèles, nous reportons le lecteur aux articles respectifs les ayant introduites :

correction et consistance : l'algorithme GVerify renvoie toujours 1 pour toute signature produite par l'algorithme GSign pour une clé secrète valide et 0 sinon ;

traçabilité : pour toute signature valide, l'autorité de révocation doit être capable de retrouver l'identité du signataire

résistance à la falsification : tout attaquant polynomial non enregistré ne doit pas être capable de produire une signature valide ;

anonymat : tout attaquant polynomial en possession d'une signature ne doit pas être capable de retrouver l'auteur de celle-ci ;

anonymat fort (ou non-traçabilité) : tout attaquant polynomial en possession de deux signatures, ne doit pas être capable de déterminer si elles ont été produites par le même signataire ;

non-reliabillité : tout attaquant polynomial en possession de deux paires message/signature ne doit pas être capable de déterminer si elles ont été produites par le même signataire ;

non-inculpation : tout membre enregistré ne doit pas être capable de produire une signature au nom d'un autre membre du groupe ;

non-diffamation : ni les membres, ni le manager ne doivent être capable de produire une signature au nom d'un autre membre du groupe ;

résistance aux coalitions : toute coalition de membres ne doit pas être capable de produire une signature ouvrable sur un membre n'appartenant pas à la coalition.

Les premiers schémas de signatures proposés par Chaum et Van Heyst possédaient des tailles de paramètres publics proportionnelles au nombre de membres, ce qui rendait ces schémas peu utilisables en pratique. Mais depuis, d'autres schémas beaucoup plus efficaces ont été proposés [30, 37, 78], offrant aussi une plus grande flexibilité : désormais,

tous les schémas considèrent des groupes dynamiques [7, 42, 44], à l'opposé de leurs prédécesseurs [54, 57, 40], pour lesquels la distribution des clés est simultanée pour tous les membres. L'apparition de nouveaux mécanismes anonymes, (en partie liée à l'utilisation des groupes bilinéaires pour la conception de protocole) a énormément contribué à ces améliorations. On pourra citer pour exemple les systèmes de preuves à la Groth-Sahai [79], permettant de vérifier la validité d'une assertion sur des données chiffrées sans jamais compromettre l'anonymat du signataire.

5.1.2 Chiffrement de groupe

Le chiffrement de groupe a été introduit en 2007 par Kiayas, Tsiounis et Yung dans [88]. Ce concept permet aux utilisateurs de masquer leurs préférences, comme par exemple dans certains systèmes flexibles permettant de désigner une autorité de confiance ; dans le cas des signature de groupe, il s'agit de l'autorité d'ouverture.

Introduction au chiffrement de groupe

Dans l'article [88], les auteurs donnent les propriétés définissant cette primitive ; ils formalisent également un modèle de sécurité à l'exemple des travaux publiés sur les signatures de groupes statiques [13] et dynamiques [21]. Ils exhibent également une construction générique utilisant comme brique de base un schéma de chiffrement sémantiquement sûr et anonyme, résistant aux attaques à chiffrés choisis adaptatives.

Une des applications théoriques intéressantes, que nous approfondissons dans cette partie, est l'identification des chiffrés invalides. Plus précisément, le système doit être correct[2] et de manière complémentaire, le système doit être consistant : tout chiffré valide doit pouvoir être associé à une identité enregistrée. Dans le chiffrement de groupe, cette propriété de consistance est garantie par l'usage de preuves interactives d'appartenance, ce qui aboutit à des protocoles inefficaces. Par ailleurs, un utilisateur peut très bien faire passer de l'information de manière malhonnête au moyen de l'aléa de la fonction de chiffrement. En effet, afin de garantir la sécurité sémantique, toute fonction de chiffrement asymétrique est probabiliste et le pouvoir de chiffrer suffit à faire passer de l'information supplémentaire : par exemple, que se passe-t-il si l'attaquant utilise la clé publique d'un destinataire qui n'est pas dans le groupe ? Dans le but de masquer toute information que l'attaquant serait susceptible de vouloir préserver, nous suggérons de rechiffrer les chiffrés. Nous présentons d'abord les primitives de chiffrement de groupe et de rechiffrement, avant d'introduire une nouvelle primitive vérifiant des propriétés de consistance que nous préciserons.

Définitions et propriétés

Dans cette partie, nous définissons la primitive de chiffrement de groupe et formalisons les propriétés associées. Ce modèle se déduit de manière assez naturelle à partir de celui défini pour les signatures de groupe.

Pour les définitions qui suivent, on notera $(\mathsf{out1}|\mathsf{out2}) \leftarrow \langle \mathcal{P}(\mathsf{in1}), \mathcal{V}(\mathsf{in2})\rangle(\mathsf{in})$ l'exécution d'un protocole interactif entre $\mathcal{P}$ et $\mathcal{V}$ possédant chacun les entrées $\mathsf{in1}$ et $\mathsf{in2}$ respectives, avec in pour entrée commune ; $\mathcal{P}$ et $\mathcal{V}$ retournent $\mathsf{out1}$ et $\mathsf{out2}$ respectivement. On définit les flag $\mathsf{accepte}$, $\mathsf{termine}$ ou $\mathsf{rejette}$ pour indiquer que l'un des deux interlocuteurs accepte, termine ou rejette ; en outre, ces flag indiquent la fin du processus.

[2] Dans notre cas, nous parlerons de correction parfaite.

Définition 16 (Schéma de chiffrement de groupe) *Un schéma de chiffrement de groupe pour une relation publique $\mathcal{R}$ décidable en temps polynomial est défini par une collection de procédures et de protocoles :*

$$\langle \mathsf{GSetup}, \mathsf{KeyGen}_{GM}, \mathsf{KeyGen}_{OA}, \mathsf{GJoin}, \langle \mathcal{G}_{\mathcal{R}}, \mathcal{R}, \mathtt{sample}_{\mathcal{R}} \rangle, \mathsf{GEncrypt}, \mathsf{GDecrypt}, \mathsf{GOpen}, \mathsf{GJudge} \rangle$$

définis par :

GSetup : *étant donné un paramètre de sécurité λ,* GSetup *renvoie les paramètres communs du système* params *;*

$\mathsf{KeyGen}_{\mathbf{GM}}$ **(resp.** $\mathsf{KeyGen}_{\mathbf{OA}}$**) :** *prend en entrée les paramètres* params *et renvoie une paire de clés publique/secrète* $(\mathsf{mpk}, \mathsf{msk})$ *(resp.* $(\mathsf{pk}_O, \mathsf{sk}_O)$*) ;*

GJoin : *il s'agit d'un protocole interactif $\langle J_{\mathsf{ID}}, J_{GM} \rangle$ entre un utilisateur* ID *(*ID *peut être vu comme un identifiant ou un pseudonyme) et le manager à l'issue duquel l'utilisateur obtient une paire de clés* $(\mathsf{pk}_{\mathsf{ID}}, \mathsf{sk}_{\mathsf{ID}})$ *et le certificat*[3] *associé* `cert`*. La clé publique de l'utilisateur* $\mathsf{pk}_{\mathsf{ID}}$ *et le certificat associé* `cert` *sont ajoutés dans une liste d'enregistrement $\mathcal{L}$ initialement vide. On définit cette liste de la manière suivante :*

$$\mathcal{L} \stackrel{déf}{=} \{\mathsf{pk}_{\mathsf{ID}} \mid \exists \mathtt{cert} \;\; ((\mathsf{pk}_{\mathsf{ID}}, \mathsf{sk}_{\mathsf{ID}}, \mathtt{cert}) \mid \mathsf{pk}_{\mathsf{ID}}, \mathtt{cert}) \leftarrow \langle J_{\mathsf{ID}}(), J_{GM}(\mathsf{msk})(\mathsf{mpk}) \rangle\}$$

$\langle \mathcal{G}_{\mathcal{R}}, \mathcal{R}, \mathtt{sample}_{\mathcal{R}} \rangle$: *l'algorithme $\mathcal{G}_{\mathcal{R}}$ prend en entrée le paramètre de sécurité λ et renvoie une clé publique et privée $\mathsf{pk}_{\mathcal{R}}, \mathsf{sk}_{\mathcal{R}}$ associées à la relation $\mathcal{R}$. L'algorithme de test $\mathcal{R}$ est polynomial, il prend un entrée une paire d'éléments (x, w) et renvoie 1 si et seulement si (x, w) appartient à la relation $\mathcal{R}$ associée au paramètre public $\mathsf{pk}_{\mathcal{R}}$. La relation $\mathcal{R}$ prend en entrée la clé publique et une paire (x, w) et renvoie 1 si les éléments x et w vérifie la relation $\mathcal{R}$ et 0 sinon. La procédure $\mathtt{sample}_{\mathcal{R}}$ prend en entrée la clé publique $\mathsf{pk}_{\mathcal{R}}$ et la clé secrète $\mathsf{sk}_{\mathcal{R}}$, puis renvoie un couple (x, w) tel que $\mathcal{R}(x, w)$ renvoie 1. Si cette procédure est publique alors $\mathsf{sk}_{\mathcal{R}}$ est une chaîne de caractères vide. La propriété de vérifiabilité est facultative et dans certains cas, la relation $\mathcal{R}$ peut tout simplement être la relation triviale où le témoin w associé à x peut être une chaîne de caractères quelconques ;*

GEncrypt : *étant donnés une identité* ID*, un message w, cet algorithme chiffre w pour le destinataire* ID *en possession de la clé* $\mathsf{pk}_{\mathsf{ID}}$ *et du certificat* `cert` *associé : il calcule*

$$c \leftarrow \mathsf{GEncrypt}(\mathsf{params}, \mathsf{mpk}, \mathsf{pk}_O, \mathsf{pk}_{\mathsf{ID}}, w, L),$$

où L est une étiquette contenant de l'information publique auxiliaire. Enfin, l'algorithme renvoie (x, c, L) ;

GDecrypt : *étant donnés la clé* $\mathsf{sk}_{\mathsf{ID}}$*, le chiffré c (destiné à* $\mathsf{pk}_{\mathsf{ID}}$*) et l'étiquette L associée, cet algorithme retourne le message w ;*

GOpen : *étant donnés la clé* sk_O*, un chiffré c et l'étiquette L associée, cet algorithme retourne* $\mathsf{pk}_{\mathsf{ID}}$ *;*

GJudge : *il s'agit d'un protocole interactif entre un prouveur $\mathcal{P}$ et un vérifieur $\mathcal{V}$, noté*

$$\langle \mathcal{P}(\mathsf{pk}_{\mathsf{ID}}, \mathtt{cert}, w, \mathsf{coin}_c), \mathcal{V} \rangle (\mathsf{params}, \mathsf{mpk}, \mathsf{pk}_O, \mathsf{pk}_{\mathsf{ID}}, x, c, L),$$

où coin_c *est l'aléa qui a été utilisé pour produire c. Le but du prouveur est de convaincre le vérifieur qu'il existe un utilisateur enregistré capable de déchiffrer le*

[3] Il s'agit d'une signature de $\mathsf{pk}_{\mathsf{ID}}$ avec une clé secrète connue du manager.

chiffré envoyé par le prouveur. À l'issue de ce protocole, le vérifieur est capable de savoir s'il existe bien un utilisateur enregistré, qui en déchiffrant c obtient w, ou une valeur w′ telle qu'il existe une fonction f telle que $(f(w'), x) \in \mathcal{R}$. *Pour simplifier, on supposera que f est la fonction identité. En cas de succès, le prouveur renvoie "*$\mathsf{termine}$*" indiquant la terminaison du protocole et le vérifieur retourne le message "*vrai*".*

Correction

De manière intuitive, la propriété de correction peut se formaliser ainsi : si un chiffré est valide, c.à.d. correctement formé et correspond à une clé publique enregistrée (générée de manière honnête), l'algorithme GOpen renvoie la clé publique du destinataire, et si le prouveur est honnête, le vérifieur accepte le prouveur avec probabilité écrasante. Plus formellement, le schéma est *correct* si l'expérience ci-dessous renvoie 1 avec probabilité écrasante :

$\mathsf{Expérience}\ \mathsf{Exp}^{\mathsf{correct}}_{\mathcal{GE},\mathcal{A}}(\lambda)$
$\mathsf{params} \leftarrow \mathsf{GSetup}(\lambda)\,;$
$\langle \mathsf{pk}_{\mathcal{R}}, \mathsf{sk}_{\mathcal{R}} \rangle \leftarrow \mathcal{G}_{\mathcal{R}}(\lambda)\,;\ (x, w) \leftarrow \mathtt{sample}_{\mathcal{R}}(\mathsf{pk}_{\mathcal{R}}, \mathsf{sk}_{\mathcal{R}})\,;$
$(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{KeyGen}_{\mathrm{GM}}(\mathsf{params})\,;\ (\mathsf{pk}_O, \mathsf{sk}_O) \leftarrow \mathsf{KeyGen}_{\mathrm{OA}}(\mathsf{params})\,;$
$\langle \mathsf{pk}_{\mathsf{ID}}, \mathsf{sk}_{\mathsf{ID}}, \mathtt{cert} \mid \mathsf{pk}_{\mathsf{ID}}, \mathtt{cert} \rangle \leftarrow \langle J_{\mathsf{ID}}, J_{\mathrm{GM}}(m\mathsf{sk}) \rangle (\mathsf{mpk})\,;$
$\mathtt{si}\ \mathsf{pk}_{\mathsf{ID}} \notin \mathcal{L}\ \mathtt{retourner}\ 0\ \mathtt{sinon}$
$c \leftarrow \mathsf{GEncrypt}(\mathsf{mpk}, \mathsf{pk}_O, \mathsf{pk}_{\mathsf{ID}}, \mathtt{cert}, w, L)\,;$
$(\mathsf{out1} \mid \mathsf{out2}) \leftarrow \langle \mathcal{P}(\mathsf{pk}_{\mathsf{ID}}, \mathtt{cert}, w, \mathsf{coin}_c), \mathcal{V} \rangle (\mathsf{params}, \mathsf{mpk}, \mathsf{pk}_O, \mathsf{pk}_{\mathcal{R}}, x, c, L)\,;$
$\mathtt{si}\ \mathsf{out1} = \mathsf{termine}\ \mathtt{et}\ \mathtt{si}\ \mathsf{pk}_{\mathsf{ID}} = \mathsf{GOpen}(\mathsf{sk}_O, \tilde{c}, L)\ \mathtt{et}$
$\mathtt{si}\ \mathsf{out2} = \mathsf{accepte}\ \mathtt{retourner}\ 1\ \mathtt{sinon}\ \mathtt{retourner}\ 0.$

Remarque 6 *Dans [88], l'algorithme* GOpen *prend en entrée un préfixe* $\tilde{c}$ *de c, cette distinction est essentielle pour atteindre une notion d'anonymat plus forte, avec accès à l'oracle de révocation.*

Sécurité

Le jeu de sécurité se déroule de la manière suivante : l'attaquant génère les clés publiques des deux autorités ; il a la possibilité d'introduire un membre dans le groupe. Il retourne une clé publique. Nous supposons que l'attaquant a accès à un oracle de déchiffrement $\mathcal{O}\mathsf{GDecrypt}_{\mathsf{ID}}()$ qui déchiffre pour le clé du destinataire, $\mathsf{sk}_{\mathsf{ID}}$ du chiffré challenge (sauf pour le chiffré challenge c). Il choisit ensuite une relation $\mathcal{R}$ et une paire $(x, w) \in \mathcal{R}$. Suivant la valeur du bit b, on lui renvoie le chiffré de w sous la clé $\mathsf{pk}_{\mathsf{ID}}$ ou un chiffré d'un message w', complètement aléatoire dans l'espace de tous les messages possibles que l'on note $\mathcal{M}$. À l'issue de cette phase, on envisage deux possibilités suivant la valeur du bit b : ou bien $\mathcal{A}$ est engagé dans une preuve de validité du chiffré avec l'émetteur réel, ou bien, il est engagé dans une preuve de validité du chiffré avec un simulateur. L'attaquant gagne s'il devine le bit b avec un avantage significativement plus grand que $\frac{1}{2}$.

Pour définir cette expérience, on définit un oracle supplémentaire, $\mathcal{O}\mathsf{Prove}^{b}_{\mathcal{P},\mathcal{P}'}(* \mid *)$ qui étant donné un bit b, simule l'exécution du prouveur :

- $\mathcal{P}$, qui prend en entrée $\mathsf{mpk}, \mathsf{pk}_O, \mathsf{pk}_{\mathsf{ID}}, \mathtt{cert}, \mathsf{pk}_{\mathcal{R}}, x, w, c, L, \mathsf{coin}_c$ comme défini par le modèle (si $b = 1$), ou
- $\mathcal{P}'$ qui prend en entrée $\mathsf{mpk}, \mathsf{pk}_O, \mathsf{pk}_{\mathsf{ID}}, \mathtt{cert}, \mathsf{pk}_{\mathcal{R}}, x, c, L$ (si $b = 0$).

Un schéma $\mathcal{GE}$ est sûr s'il existe un protocole $\mathcal{P}'$ tel que pour tout attaquant polynomial $\mathcal{A}$, l'expérience, ci-dessous retourne 1 avec probabilité écrasante :

$\mathsf{Expérience}\ \mathsf{Exp}^{\mathsf{ind\text{-}cca}}_{\mathcal{GE},\mathcal{A}}(\lambda)$
$\mathsf{params} \leftarrow \mathsf{GSetup}(\lambda)\,;\ (\mathsf{mpk}, \mathsf{pk}_O, \mathsf{state}) \leftarrow \mathcal{A}(\mathsf{params})\,;$
$\langle \mathsf{pk}_{\mathsf{ID}}, \mathsf{sk}_{\mathsf{ID}}, \mathtt{cert} \mid \mathsf{state}' \rangle \leftarrow \langle J_{\mathsf{ID}}, \mathcal{A}(\mathsf{state}) \rangle (\mathsf{mpk})\,;$
$(\mathsf{state}'', x, w, L, \mathsf{pk}_{\mathcal{R}}) \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{GDecrypt}_{\mathsf{ID}}()}(\mathsf{state}')$
$\mathtt{si}\ \ (x, w) \notin \mathcal{R}\ \mathtt{retourner}\ 0\ \mathtt{sinon}$
$b \overset{R}{\leftarrow} \{0,1\}\,;\ \mathtt{si}\ \ b = 1,\ w_b \leftarrow w\ \mathtt{sinon}\ \ w_b \overset{R}{\leftarrow} \mathcal{M}$
$c \leftarrow \mathsf{GEncrypt}(\mathsf{pk}_{\mathsf{ID}}, w_b, L)\,;$
$b' \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{Prove}^b_{\mathcal{P},\mathcal{P}'}(* \mid *), \mathcal{O}\mathsf{GDecrypt}_{\mathsf{ID}}()}(\mathsf{state}'', c)\,;$
$\mathtt{si}\ \ b = b'\ \mathtt{retourner}\ 1\ \mathtt{sinon}\ \mathtt{retourner}\ 0$

Remarque 7 *Cette notion est une extension de la notion de sécurité sémantique. On remarque qu'elle prend en compte les corruptions (simultanées) du manager et de l'autorité d'ouverture puisqu'il se peut que l'attaquant connaisse les secrets* msk *et* sk_O*. Cependant, elle ne donne aucune information sur des éventuelles coalitions entre utilisateurs : une attaque est jugée valide seulement si la clé publique challenge est valide. Dans le cas contraire, cette notion ne nous apporte aucune information.*

Anonymat

L'anonymat pour le chiffrement de groupe n'est pas tout à fait similaire à la notion d'anonymat pour les signatures de groupe. Casser l'anonymat du chiffrement de groupe signifie en quelque sorte casser l'anonymat du système de chiffrement associé à l'autorité de révocation. Dans ce jeu, l'attaquant a un contrôle partiel du système : il ne connaît pas la clé de révocation mais a accès à un oracle d'ouverture, noté $\mathcal{O}\mathsf{GOpen}$; il a accès à un oracle d'enregistrement lui permettant d'ajouter deux membres de son choix, puis un oracle de déchiffrement. Le jeu de sécurité se déroule ainsi : tout d'abord, l'attaquant enregistre deux identités ID_0 et ID_1 de son choix, et renvoie une relation $\mathcal{R}$, un témoin w valide pour cette relation et une étiquette L. L'attaquant a accès à un oracle de déchiffrement $\mathcal{O}\mathsf{GDecrypt}_i()$ associé au destinataire ID_i pour $i = 0, 1$. Pour calculer la challenge, on choisit un bit b, puis on calcule le chiffré de w sous la clé pk_b avec en entrée la clé mpk, le message w et l'étiquette L. Dans cette expérience, $\mathcal{A}$ a également accès à un oracle, que l'on note $\mathcal{O}_{\mathcal{P}}$: cet oracle prend en entrée $\mathsf{mpk}, \mathsf{pk}_O, \mathsf{pk}_{\mathcal{R}}, \mathsf{pk}_b, \mathtt{cert}, x, w, c, L, \mathsf{coin}_c$, où coin_c est l'aléa utilisé pour produire le chiffré et simule le prouveur sur ces entrées exactement comme défini par le modèle.

L'attaquant gagne s'il devine le bit b. Afin de modéliser la propriété de séparabilité (c.à.d. les pouvoirs du manager et de l'autorité d'ouverture sont partagés), on peut permettre à l'attaquant de générer les paramètres du manager.

On définit l'oracle partiel d'enregistrement RegU de la manière suivante :

RegU **:** prend en entrée la clé publique du manager et simule l'enregistrement de deux utilisateurs désirant devenir membres du groupe. Il a accès à une chaîne de caractères contenant les deux clés publiques et secrètes puis les certificats associés.

Le schéma de chiffrement de groupe est anonyme résistant aux attaques à chiffrés choisis adaptatives avec accès à l'oracle d'ouverture si l'expérience suivante retourne 1 avec probabilité écrasante :

$\mathsf{Expérience}\ \mathsf{Exp}^{\mathsf{anon\text{-}cca}}_{\mathcal{GE},\mathcal{A}}(\lambda)$
$\mathsf{params} \leftarrow \mathsf{Setup}(\lambda)$;
$(\mathsf{pk}_O, \mathsf{sk}_O) \leftarrow \mathsf{KeyGen}_{\mathrm{OA}}(\mathsf{params})$;$(\mathsf{mpk}, \mathsf{state}) \leftarrow \mathcal{A}(\mathsf{params}, \mathsf{pk}_O)$;
$(\mathsf{state}') \leftarrow \mathcal{A}^{\mathsf{RegU}(\mathsf{mpk},\mathsf{ID}_0,\mathsf{ID}_1),\mathcal{O}\mathsf{GOpen}()}(\mathsf{state})$;
$(\mathsf{state}'', x, w, L, \mathsf{pk}_{\mathcal{R}}) \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{GOpen}(),\mathcal{O}\mathsf{GDecrypt}_0(),\mathcal{O}\mathsf{GDecrypt}_1()}(\mathsf{state}')$;
`si` $(x, w) \notin \mathcal{R}$ `retourner` 0 `sinon`
$b \xleftarrow{R} \{0,1\}; c \leftarrow \mathsf{GEncrypt}(\mathsf{params}, \mathsf{mpk}, \mathsf{pk}_O, \mathsf{pk}_b, w, L)$;
$b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathcal{P}}(),\mathcal{O}\mathsf{GOpen}(),\mathcal{O}\mathsf{GDecrypt}_0(),\mathcal{O}\mathsf{GDecrypt}_1()}(\mathsf{state}'', c)$;
`si` $b = b'$ `retourner` 1 `sinon retourner` 0

Consistance

Jusqu'ici nous nous sommes intéressés à ce qui se passait vis à vis d'un émetteur malhonnête. À présent, du point de vue d'un prouveur malhonnête, on veut garantir que tout chiffré valide est associé à au moins une clé d'un des membres. Rappelons que le but du prouveur est justement de convaincre le vérifieur que le chiffré correspond à un message donné pour la relation $\mathcal{R}$ et que ce chiffré est associé à au moins une des identités enregistrées. Pour formaliser la consistance, on définit le jeu de sécurité suivant : l'attaquant crée de manière adaptative la liste des membres. Il choisit une relation $\mathcal{R}$ et crée un chiffré pour une identité de son choix. Nous définissons l'oracle d'enregistrement GReg de la manière suivante :

GReg **:** prend en entrée la clé secrète du manager et simule l'algorithme J_{GM} lors de l'exécution du protocole GJoin. L'attaquant ajoute les clés publiques et les certificats correspondants dans une liste auxiliaire $\mathcal{L}'$, initialement vide. La liste $\mathcal{L}'$ contient donc toutes les paires clés publiques/certificats des utilisateurs enregistrés par l'attaquant.

L'attaquant gagne s'il parvient à convaincre le vérifieur que le chiffré satisfait la relation $\mathcal{R}$ et si l'une des deux conditions suivantes est vérifiée :

1. l'autorité de révocation identifie un utilisateur qui n'est pas membre du groupe ;
2. la clé retournée n'est pas dans la liste $\mathcal{L}'$.

Nous définissons un algorithme supplémentaire Valid qui prend en entrée une paire $\mathsf{pk}, \mathtt{cert}$ et renvoie 1 si `cert` est un certificat associé à la clé publique pk et 0 sinon.

Nous définissons la liste $\mathcal{L}_c^{x,L,\mathsf{pk},\mathsf{pk}_{\mathcal{R}},\mathsf{mpk},\mathsf{pk}_O}$ de la manière suivante :

$$\mathcal{L}_c^{x,L,\mathsf{pk},\mathsf{pk}_{\mathcal{R}},\mathsf{mpk},\mathsf{pk}_O} \stackrel{\text{déf}}{=} \{\mathsf{GEncrypt}(\mathsf{mpk}, \mathsf{pk}_O, \mathtt{cert}, w, L) \mid w \ : \ (x, w) \in \mathcal{R}, \mathsf{Valid}(\mathsf{pk}, \mathtt{cert})\}$$

On dit qu'un schéma est consistant si pour tout attaquant (polynomial), l'expérience ci-dessous retourne 1 avec probabilité écrasante :

$\mathsf{Expérience}\ \mathsf{Exp}^{\mathsf{sound}}_{\mathcal{GE},\mathcal{A}}(\lambda)$
$\mathsf{params} \leftarrow \mathsf{Setup}(\lambda)$;
$(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{KeyGen}(\mathsf{params})$; $(\mathsf{pk}_O, \mathsf{sk}_O) \leftarrow \mathsf{KeyGen}_{\mathrm{OA}}(\mathsf{params})$;
$(\mathsf{pk}_{\mathcal{R}}, x, c, L, \mathsf{state}) \leftarrow \mathcal{A}^{\mathsf{GReg}(\mathsf{msk},*)}(\mathsf{params}, \mathsf{mpk}, \mathsf{pk}_O, \mathsf{sk}_O)$;
$\langle \mathsf{state} \mid \mathsf{out2} \rangle \leftarrow \langle \mathcal{A}(\mathsf{state}), \mathcal{V} \rangle(\mathsf{params}, \mathsf{mpk}, \mathsf{pk}_O, \mathsf{pk}_{\mathcal{R}}, x, c, L)$;
$\mathsf{pk} \leftarrow \mathsf{GOpen}(\mathsf{sk}_O, \tilde{c}, L)$;
`si` $(\mathsf{pk} \notin \mathcal{L}'$ `ou` $c \notin \mathcal{L}_c^{x,L,\mathsf{pk},\mathsf{pk}_{\mathcal{R}},\mathsf{mpk},\mathsf{pk}_O})$
`et` $\mathsf{out2} = \mathsf{accepte}$ `retourner` 1 ;
`sinon retourner` $\perp$;

Autrement dit, l'attaquant gagne s'il parvient se faire accepter avec un chiffré tel que, étant donnés les paramètres du système, après application de l'algorithme d'ouverture secret, ou bien nous obtenons une clé qui n'est pas enregistrée, ou bien nous obtenons une clé non valide.

Remarque 8 *Dans les expériences* $\mathsf{Exp}^{\mathsf{anon-cca}}_{\mathcal{GE},\mathcal{A}}$ *et* $\mathsf{Exp}^{\mathsf{sound}}_{\mathcal{GE},\mathcal{A}}$*, les requêtes* RegU *et* GReg *permettent d'autoriser les attaques concurrentes ou l'exécution séquentielle de l'enregistrement de membres.*

5.1.3 Construction générique

Dans l'article [88], Kiayias *et al.* montrent que les primitives cryptographiques suivantes permettent d'obtenir des conditions nécessaires et suffisantes pour la construction d'un schéma de chiffrement de groupe vérifiant les propriétés introduites au paragraphe précédent :

- un schéma de signature $\langle \mathsf{Setup}_s, \mathcal{G}_s, \mathcal{S}_s, \mathcal{V}_s \rangle$ résistant aux attaques à messages choisis adaptatives ;
- un schéma de chiffrement $\langle \mathsf{Setup}_e, \mathsf{KeyGen}_e, \mathsf{Encrypt}, \mathsf{Decrypt} \rangle$ sémantiquement sûr et anonyme résistant aux attaques à chiffrés choisis adaptatives ;
- deux preuves d'appartenance à divulgation nulle de connaissance $\langle \mathcal{P}_{\mathsf{pk}}, \mathcal{V}_{\mathsf{pk}} \rangle$ et afin de faciliter l'extraction d'un témoin, un schéma de *mise en gage* $\langle \mathcal{Z}_c, \mathcal{C}_c, \mathcal{T}_c \rangle$ est utilisé ; ce schéma doit vérifier les propriétés de *dissimulation et d'engagement* définies au paragraphe 2.1.1.

Description

GSetup : génère les paramètres en faisant une exécution séquentielle de $\mathcal{Z}_c, \mathsf{Setup}_s$; cet algorithme retourne la clé cpk et des paramètres globaux params. ;

KeyGen : $\mathsf{KeyGen}_{\mathrm{GM}}$ correspond à $\mathcal{G}_s$ et $\mathsf{KeyGen}_{\mathrm{OA}}$ correspond à Setup_e : cet algorithme retourne les clés mpk et msk retournées par $\mathcal{G}_s$ puis les paramètres globaux du schéma chiffrement ;

GJoin : chaque utilisateur ID désirant devenir membre exécute au préalable l'algorithme KeyGen_e pour obtenir une paire de clés $\mathsf{pk}_{\mathsf{ID}}, \mathsf{sk}_{\mathsf{ID}}$; puis il s'engage dans un protocole interactif avec le manager afin de prouver l'appartenance de la clé $\mathsf{pk}_{\mathsf{ID}}$ au langage $\mathcal{L}$. Dans ce cas, le "langage des clés valides" est défini par :

$$\mathcal{L} \stackrel{\text{déf}}{=} \{\mathsf{pk} | \exists \mathsf{sk}, r \ \langle \mathsf{pk}, \mathsf{sk} \rangle \leftarrow \mathsf{KeyGen}_e(\mathsf{params}; r)\}$$

Le manager répond par le certificat $\mathtt{cert} \leftarrow \mathcal{S}_s(\mathsf{msk}, \mathsf{pk})$ et la clé publique associée pk ;

GEncrypt : étant donnés une clé publique $\mathsf{pk}_{\mathsf{ID}}$, un message w associé à une valeur x tel que $(x, w) \in \mathcal{R}$, une étiquette L, cet algorithme calcule :

$c_4 = \mathcal{C}_c(\mathsf{cpk}, \mathtt{cert})$
$c_3 = \mathcal{C}_c(\mathsf{cpk}, \mathsf{pk}_{\mathsf{ID}})$
$c_2 = \mathsf{Encrypt}(\mathsf{pk}_O, \mathsf{pk}_{\mathsf{ID}}, L_2)$, où $L_2 = c_3 \| c_4 \| L$,
$c_1 = \mathsf{Encrypt}(\mathsf{pk}_{\mathsf{ID}}, w, L_1)$, où $L_1 = c_2 \| c_3 \| c_4 \| L$
Cet algorithme retourne $c = (c_1, c_2, c_3, c_4)$.

GDecrypt : étant donnés la clé $\mathsf{sk}_{\mathsf{ID}}$, le chiffré c destiné à $\mathsf{pk}_{\mathsf{ID}}$ et l'étiquette L associée, cet algorithme retourne $\mathsf{Decrypt}(\mathsf{sk}_{\mathsf{ID}}, c_1, c_2 \| c_3 \| c_4 \| L)$;

GOpen : étant donnés la clé sk_O, le suffixe d'un chiffré $\tilde{c} \stackrel{\text{déf}}{=} (c_2, c_3, c_4)$ et l'étiquette L associée, cet algorithme retourne $\mathsf{Decrypt}(\mathsf{sk}_O, c_2, c_3 \| c_4 \| L)$;

GJudge : il s'agit d'un protocole interactif $\langle \mathcal{P}, \mathcal{V} \rangle$ prouvant la validité du chiffré $c = (c_1, c_2, c_3, c_4)$. Plus précisément, le but du prouver est de prouver au vérifieur par une preuve d'appartenance à divulgation nulle de connaissance que le t-uplet $(\mathsf{params}, \mathsf{mpk}, \mathsf{pk}_O, \mathsf{pk}_{\mathcal{R}}, x, c = (c_1, c_2, c_3, c_4), L)$ vérifie :

$$\exists(\mathsf{coin}_1, \mathsf{coin}_2, \mathsf{coin}_3, \mathsf{coin}_4, \mathsf{pk}_{\mathsf{ID}}, \mathtt{cert}, w) : \mathcal{C}_c(\mathsf{cpk}, \mathsf{pk}_{\mathsf{ID}}; \mathsf{coin}_3) = c_3 \wedge$$
$$\mathcal{C}_c(\mathsf{cpk}, \mathtt{cert}; \mathsf{coin}_4) = c_4 \wedge \mathcal{V}_s(\mathsf{pk}_s, \mathtt{cert}; \mathsf{coin}_3) = 1 \wedge$$
$$\mathsf{Encrypt}(\mathsf{pk}_{\mathsf{ID}}, w, (c_2 \| c_3 \| c_4 \| L); \mathsf{coin}_1) = c_1 \wedge$$
$$\mathsf{Encrypt}(\mathsf{pk}_O, \mathsf{pk}_{\mathsf{ID}}, (c_3 \| c_4 \| L); \mathsf{coin}_2) = c_2 \wedge (x, w) \in \mathcal{R}$$

Exemple

Cette construction modulaire, inefficace constitue malgré tout une preuve d'existence de la primitive ; elle justifie en quelque sorte les propriétés qui la définissent. Les auteurs présentent un schéma de chiffrement assez complexe, nous renvoyons à l'article pour la description et les preuves, nous commentons juste les points de la construction modulaire qui nous intéressent ici :

1. **une relation de vérifiabilité :** les auteurs proposent un schéma de chiffrement vérifiable pour la relation du logarithme discret. Le système $\langle \mathcal{G}_{\mathcal{R}}, \mathcal{R}, \mathtt{sample}_{\mathcal{R}} \rangle$ est défini de la manière suivante : $\mathcal{G}_{\mathcal{R}}$ renvoie une clé publique et une clé secrète, $\mathsf{pk}_{\mathcal{R}} = \langle \mathbb{G}, g, q \rangle$, où $\mathbb{G}$ est un groupe cyclique d'ordre premier q, g un générateur du groupe $\mathbb{G}$ et $\mathsf{sk}_{\mathcal{R}} = \emptyset$. Enfin, $\mathtt{sample}_{\mathcal{R}}$ sélectionne un élément x dans $\mathbb{Z}_q^\star$ et renvoie $(x, y = g^x)$, ce qui définit implicitement la relation ;
2. **un schéma de chiffrement** IND-CCA **et** ANON-CCA : l'article [88] propose un schéma de chiffrement sémantiquement sûr et anonyme résistant aux attaques à chiffrés choisis. La construction de ce schéma est inspirée des cryptosystèmes de Paillier et de Cramer-Shoup. La sécurité sémantique repose sur le problème décisionnel de la haute résiduosité, et l'anonymat sur un nouveau problème, qui peut être vu comme l'extension du problème DDH dans un sous-groupe cyclique particulier χ_{n^2} (χ_{n^2} est le sous-groupe des $n_{\text{ième}}$ résidus quadratique dans $\mathbb{Z}_{n^2}^\star$-tel que l'ordre ne soit pas friable). Nous n'aborderons pas plus en détail cette construction. Pour plus de précisions, nous renvoyons à l'article [88].
3. **une preuve de validité de la clé publique** : cette preuve est liée au schéma de chiffrement ; l'utilisateur doit prouver qu'il est en possession d'une clé publique générée par l'algorithme KeyGen_e. Pour leur schéma, il s'agit de montrer que la clé publique est un triplet appartenant au sous-groupe χ_{n^2}.

5.1.4 Modéliser la validité

D'un point de vue théorique, le *chiffrement de groupe* comporte de nombreuses similarités avec les *signatures de groupe*, et notamment on doit pouvoir être capable d'identifier la validité du chiffré, autrement dit qu'il existe bien un utilisateur enregistré associé à ce chiffré, ou le cas échéant on doit être capable de détecter que le traçage n'est pas correct. Dans ce dernier cas, nous pouvons distinguer deux cas :

- ou bien il n'existe pas de clés enregistrées associées au chiffré ;
- ou bien le juge rejette la preuve associée à la paire chiffré/clé publique (ou bien la paire signature/clé publique).

Pour le chiffrement de groupe, nous avons vu que la propriété de consistance ou *soundness* est modélisée par un jeu où le but de l'attaquant est ou bien de faire accepter au vérifieur un chiffré destiné à un joueur non enregistré, ou bien de faire accepter un chiffré pour lequel l'algorithme de traçage renvoie une clé qui ne correspond à aucun membre. Afin d'apporter de telles garanties, leur construction nécessite des preuves interactives qui rendent ces schémas inefficaces dans le modèle standard. On remarque que la clé publique du destinataire est chiffrée : elle pourrait très bien être utilisée pour envoyer de l'information de manière illégale ; la notion de consistance définie au paragraphe 5.1.2 ne considère pas ces attaques. Dans le prochain chapitre, nous étendons la notion de consistance dans le contexte du rechiffrement dans le but de "brouiller" l'information que l'attaquant souhaiterait transmettre, notre but étant de modéliser les attaques liées à l'information subliminale exploitable par l'attaquant.

Notre objectif est d'obtenir un résultat de la forme :
- ou bien l'algorithme de traçage renvoie une clé publique enregistrée ;
- ou bien le chiffré ne laisse transmettre aucune information.

D'un point de vue théorique, ce résultat semble constituer un compromis raisonnable entre les deux objectifs suivants :
- obtenir un schéma de rechiffrement anonyme sûr face aux attaques actives, qui a été posé comme problème ouvert dans l'article [108], et
- obtenir un schéma de rechiffrement de groupe efficace à anonymat révocable et sans canaux subliminaux, une question encore inexplorée dans le contexte du chiffrement.

5.2 Schémas de rechiffrement

Dans cette partie, nous introduisons les définitions d'un schéma de rechiffrement et les constructions qui ont été proposées dans [76] et [108].

5.2.1 Introduction

Dans l'article [76], Golle *et al.* proposent un schéma *universellement rechiffrable*[4] ou *rerandomizable*. Leur motivation est la construction de réseaux de mélangeurs "universels", offrant plusieurs avantages parmi lesquels :
- une configuration moins contraignante : les serveurs n'ont plus besoin de secrets partagés ;
- la garantie de la propriété de *perfect-forward-anonymity* : même si tous les serveurs sont corrompus, l'anonymat des paquets permutés est conservée ;

Commençons par définir les schémas de rechiffrement universel :

5.2.2 Définitions et propriétés

Définition 17 (Schéma de rechiffrement) *Un schéma de rechiffrement* $\mathcal{R}\mathsf{e}\mathcal{E}\mathsf{nc}$ *est défini par la donnée de cinq algorithmes* $\langle \mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{ReRand}, \mathsf{Decrypt} \rangle$ *:*
- Setup *: prend en entrée un paramètre de sécurité* λ *et renvoie les paramètres communs du système,* params *;*
- KeyGen *: prend en entrée les paramètres du système et renvoie une paire clé publique/clé secrète* pk/sk *;*

[4] ne nécessite pas la clé publique pour rechiffrer.

- $\mathsf{Encrypt}$ *: est un algorithme probabiliste qui prend en entrée un message* m*, une clé publique* pk *et renvoie un chiffré* $c \in \mathcal{C}$*.*
- ReRand *: est un algorithme probabiliste qui prend en entrée un chiffré* c *et renvoie un chiffré* $c \in \mathcal{C}$*.*
- $\mathsf{Decrypt}$ *: est un algorithme déterministe qui prend en entrée un chiffré* c*, une clé secrète* sk *et renvoie* m *si* $c = \mathsf{Encrypt}(\mathsf{pk}, m)$ *pour la clé publique associée* pk *et* $\perp$ *sinon.*

Nous introduisons les deux ensembles suivants $\mathcal{D}_{\mathsf{ReRand}}$ et $\mathcal{D}_{\mathsf{Encrypt}}$, munis de deux distributions indistinguables :

$$\mathcal{D}_{\mathsf{ReRand}} = \{C' \mid \exists\ C \in \mathcal{C},\ r \in \mathcal{R}\ \ C' = \mathsf{ReRand}(C; r)\} \text{ et}$$

$$\mathcal{D}_{\mathsf{Encrypt}} = \{C \mid \exists\ m \in \mathcal{M},\ r' \in \mathcal{R}\ \ C' = \mathsf{Encrypt}(m; r')\}.$$

La notion de sécurité cruciale en vue de garantir la confidentialité est la sécurité sémantique. Elle s'étend de manière assez naturelle pour les schémas de rechiffrement classique : supposons que l'adversaire choisisse deux chiffrés c_0, c_1 associés à la clé pk, le challenger lui transmet alors le "rechiffré" de l'un des deux chiffrés. Le but de l'attaquant est de deviner lequel des deux messages lui a été donné.

Afin de mettre en évidence la particularité du chiffrement universel, nous définissons une notion, que nous notons USS, pertinente pour les schémas de rechiffrement et complémentaire de la notion de sécurité sémantique. Cette notion a été définie par Golle *et al.* dans l'article [76] ; l'idée est la suivante : on donne à l'attaquant deux clés publiques, l'attaquant retourne deux messages et deux aléas associés, il produit deux chiffrés en utilisant les clés publiques et les aléas respectifs. On"rechiffre universellement" ces deux chiffrés. L'attaquant gagne s'il retrouve la correspondance chiffré/rechiffré pour les deux chiffrés qu'il a généré (de manière honnête). Plus précisément, étant donné un paramètre de sécurité λ, on définit ci-dessous l'expérience, $\mathsf{Exp}^{\mathsf{uss}\text{-}b}_{\mathcal{A},\mathcal{R}e\mathcal{E}\mathsf{nc}}(\lambda)$, définie pour un bit b aléatoire. Cette expérience est associée à un schéma de rechiffrement $\mathcal{R}e\mathcal{E}\mathsf{nc}$ et à un attaquant $\mathcal{A}$ polynomial :

$\mathsf{Expérience}\ \mathsf{Exp}^{\mathsf{uss}\text{-}b}_{\mathcal{A},\mathcal{R}e\mathcal{E}\mathsf{nc}}(\lambda)$
$(\mathsf{params}) \leftarrow \mathsf{Setup}(\lambda)\,;$
$(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow \mathsf{KeyGen}(\mathsf{params})\,;\ (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{KeyGen}(\mathsf{params})\,;$
$(m_0, m_1, r_0, r_1, \mathsf{state}) \leftarrow \mathcal{A}_1(\mathsf{params}, \mathsf{pk}_0, \mathsf{pk}_1)$
`si` $m_0, m_1 \notin \mathcal{M}$ `ou` $r_0, r_1 \notin \mathcal{R}\,;$
`retourner` '0' ; `sinon`
$c_0 \leftarrow \mathsf{Encrypt}(\mathsf{pk}_0, m_0; r_0); c_1 \leftarrow \mathsf{Encrypt}(\mathsf{pk}_1, m_1; r_1)\,;$
$r'_0, r'_1 \xleftarrow{R} \mathcal{R}\,;$
$c'_0 \leftarrow \mathsf{ReRand}(c_0; r'_0), c'_1 \leftarrow \mathsf{ReRand}(c_1; r'_1)\,;$
$b' \leftarrow \mathcal{A}_2(c'_b, c'_{1-b}, \mathsf{state})\,;$
`retourner` b'

On dit qu'un schéma de rechiffrement universel $\mathcal{R}e\mathcal{E}\mathsf{nc}$ est universellement sémantiquement sûr sous rechiffrement résistant aux attaques à clairs choisis ou USS, si dans l'expérience ci-dessus, l'attaquant $\mathcal{A}$ devine b avec un avantage inférieur ou égale ϵ, où ϵ est une fonction négligeable en le paramètre de sécurité ; on définit l'avantage par :

$$\mathsf{Adv}^{\mathsf{uss}\text{-}0}_{\mathcal{R}e\mathcal{E}\mathsf{nc},\mathcal{A}} = \left|\Pr[\mathsf{Exp}^{\mathsf{uss}\text{-}1}_{\mathcal{R}e\mathcal{E}\mathsf{nc},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{uss}\text{-}0}_{\mathcal{R}e\mathcal{E}\mathsf{nc},\mathcal{A}}(\lambda) = 1]\right|$$

5.2.3 Construction d'un schéma de rechiffrement

Construction générique

Par la suite, on note $\bar{v} = (v_1, v_2)$ un élément de $\mathbb{G} \times \mathbb{G}$. On étend le produit de $\mathbb{G}$ dans $\mathbb{G} \times \mathbb{G}$ en définissant pour $\bar{u} = (u_1, u_2)$ et $\bar{v} = (v_1, v_2)$ la multiplication dans $\mathbb{G} \times \mathbb{G}$: on multiplie composante par composante chacun des éléments de $\bar{u}$ et $\bar{v}$. Autrement dit, on a : $\bar{u} \cdot \bar{v} = (u_1 \cdot u_2, v_1 \cdot v_2) \in \mathbb{G} \times \mathbb{G}$.

Nous présentons ci-dessous, le construction de Golle *et al.* présentée dans [76] :

MSetup : étant donné un paramètre de sécurité λ, cet algorithme renvoie les paramètres publics ; Pour notre exemple, il s'agit de la description d'un groupe cyclique $\mathbb{G}$ d'ordre q, $\mathsf{params} \stackrel{\text{déf}}{=} \langle \mathbb{G}, q, g \rangle$

MKeyGen : étant donnés les paramètres publics, cet algorithme renvoie une paire $(\mathsf{pk}, \mathsf{sk}) = (y = g^x, x)$ pour $x \stackrel{R}{\leftarrow} \mathbb{Z}_q$;

MEncrypt : prend en entrée un message m et un aléa $(u, v) \in \mathcal{R} \times \mathcal{R}$. Cet algorithme renvoie le chiffré c de m défini par $c = ((a_0, a_1), (b_0, b_1)) = (\mathsf{Encrypt}(m), \mathsf{Encrypt}(1))$, avec c associé un facteur aléatoire (u, v), avec

$$c = ((a_0, a_1), (b_0, b_1))$$

Si la fonction Encrypt est la fonction de chiffrement ElGamal, on a :

$$c = ((my^u, g^u), (y^v, g^v))$$

MReRand : prend en entrée un chiffré $c = ((a_0, a_1), (b_0, b_1))$ et un aléa $(u', v') \in \mathcal{R} \times \mathcal{R}$. Cet algorithme renvoie un chiffré

$$c' = ((a_0 \cdot {b_0}^{u'}, a_1 \cdot {b_1}^{u'}), ({b_0}^{v'}, {b_1}^{v'}))$$

MDecrypt : prend en entrée un chiffré $c = ((a_0, a_1), (b_0, b_1))$ et la clé secrète x. Cet algorithme vérifie d'abord que $(a_0, a_1), (b_0, b_1) \in \mathbb{G} \times \mathbb{G}$, et renvoie $\perp$ sinon. Autrement, si $m_1 = b_0/b_1^x = 1$, il renvoie $m_0 = a_0/a_1^x$ et sinon il renvoie $\perp$.

Remarque 9 *On remarque que la procédure de rechiffrement consiste à rerandomiser l'aléa u additivement et l'aléa v multiplicativement ; le résultat est un chiffré uniformément distribué dans l'espace des chiffrés avec pour aléa $r = u + v \cdot u'$ et $s = v \cdot v'$ respectivement.*

Exemple dérivé de Cramer-Shoup

Nous présentons ci-dessous une construction proposée dans [108] : cet article propose une construction universellement rechiffrable résistante aux attaques à chiffrés choisis sous rerandomization ; il s'agit d'une modification du schéma de Cramer-Shoup. Il s'appuie sur la même idée que l'exemple précédent : il s'agit d'appliquer la procédure de dédoublement au schéma Cramer-Shoup.

MSetup : étant donné un paramètre de sécurité λ, cet algorithme renvoie la description d'un groupe multiplicatif $\mathbb{G}$ d'ordre q et un générateur g de ce groupe.

MKeyGen : étant donnés les paramètres publics, cet algorithme choisit deux éléments aléatoires g_1, g_2 de $\mathbb{G}$ et $a_1, a_2, b_1, b_2 \stackrel{R}{\leftarrow} \mathbb{Z}_q$ et définit :

$$\mathsf{sk} = (a_1, a_2, b_1, b_2) \text{ et } \mathsf{pk} = (g_1, g_2, A \stackrel{\text{déf}}{=} g_1^{a_1} g_2^{a_2}, B \stackrel{\text{déf}}{=} g_1^{b_1} g_2^{b_2}).$$

MEncrypt : étant donnés une clé publique d'un destinataire pk et un message $m \in \mathbb{G}$, cet algorithme :

1. choisit $(r, s) \overset{R}{\leftarrow} \mathbb{Z}_q \times \mathbb{Z}_q^\star$ et définit pour $i = 1, 2 \quad V_i \overset{\text{déf}}{=} g_i^r$ et $W_i \overset{\text{déf}}{=} g_i^s$;
2. retourne $(\bar{V}, mA^r, B^r, \bar{W}, A^w, B^w)$, avec $\bar{V} = (V_1, V_2)$ et $\bar{W} = (W_1, W_2)$.

MReRand : étant donné un chiffré $c = (\bar{c}_1, c_2, c_3, \bar{c}_4, c_5, c_6)$, cet algorithme choisit deux éléments $u, v \overset{R}{\leftarrow} \mathbb{Z}_q \times \mathbb{Z}_q^\star$, et retourne le chiffré :

$$c = (\bar{c}_1 \cdot \bar{c}_4^u, c_2 \cdot c_5^u, c_3 \cdot c_6^u, \bar{c}_4^v, c_5^v, c_6^v)$$

MDecrypt(sk, c) : étant donné un chiffré c, cet algorithme

1. décompose c en $(\bar{c}_1, c_2, c_3, \bar{c}_4, c_5, c_6)$;
2. teste l'intégrité du chiffré, en vérifiant :

$$c_3 \overset{?}{=} \prod_{i=1}^{2} V_i^{b_i}; \quad c_5 \overset{?}{=} \prod_{i=1}^{2} W_i^{a_i}; \quad c_6 \overset{?}{=} \prod_{i=1}^{2} W_i^{b_i}$$

3. `si` $c_{4,1} = c_{4,2} = 1$ `retourner` $\perp$, `sinon retourner` $c_2/(\prod_{i=1}^{2} c_{1,i}^{a_i})$

Remarque 10 *Pour $h \in \mathbb{G}$ et un chiffré $c = (\bar{c}_1, c_2, c_3, \bar{c}_4, c_5, c_6)$ valide, on considère la loi $\otimes$ dans l'espace des chiffrés $\mathcal{C} \subset \mathbb{G}$, définie par :*

$$h \otimes c = (\bar{c}_1, h \cdot c_2, c_3, \bar{c}_4, c_5, c_6)$$

On remarque alors que : $\mathsf{MDecrypt}(\mathsf{sk}, h \otimes c) = h \cdot \mathsf{MDecrypt}(\mathsf{sk}, c)$

Il est évident que pour les deux schémas précédents, si nous donnons accès à l'oracle déchiffrement, le schéma de rechiffrement n'est plus sémantiquement sûr (l'algorithme de déchiffrement appliqué aux deux chiffrés renvoie le même message). De fait, pour les schémas de rechiffrement, on considère une notion affaiblie pour la sécurité sémantique face aux attaques à chiffrés choisis, appelé replayable CCA.

5.2.4 Sécurité "replayable CCA"

Cette notion de sécurité a été introduite en 2003 par Canetti *et al.* [47] comme une version affaiblie de la notion d'attaques à chiffrés choisis pour le chiffrement. Pour définir la sécurité sémantique, les définitions considèrent de manière exclusive les attaques à messages choisis ou les attaques à chiffrés choisis, donnant accès à l'oracle de déchiffrement. Cette dernière notion s'avère être parfois trop forte pour s'appliquer à certains schémas. Pour illustrer cela, considérons un exemple simple : supposons qu'Alice envoie un chiffré c à Bob. Carl qui voit le chiffré c construit un autre chiffré c' tel que $\mathsf{Decrypt}(c) = \mathsf{Decrypt}(c')$. Eve qui voit c' ne peut pas dire si c' est un élément obtenu à partir de l'algorithme de chiffrement chiffré ou s'il s'agit d'une "copie". D'autre part, si on donne à l'attaquant la possibilité de déchiffrer c' le schéma n'est plus sémantiquement sûr, compte tenu de la malléabilité induite par la procédure de "copie". L'idée de la notion de *replayable CCA* est d'exclure toute forme de malléabilité à l'exception de la "copie". Plus formellement, on considère l'expérience suivante $\mathsf{Exp}^{\mathsf{ind\text{-}rcca\text{-}}b}_{\mathcal{ReE}\mathsf{nc},\mathcal{A}}$, cette notion est définie pour un bit b aléatoire par l'expérience suivante :

Définition

Notion de sécurité RCCA :

Expérience $\mathsf{Exp}^{\mathsf{ind\text{-}rcca}\text{-}b}_{\mathcal{R}\mathsf{e}\mathcal{E}\mathsf{nc},\mathcal{A}}(\lambda)$
$\mathsf{CSet} \leftarrow \emptyset$; $\mathsf{params} \leftarrow \mathsf{Setup}(\lambda)$;
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{params})$;
$(m_0, m_1, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathcal{O}\mathsf{Decrypt}_{m_0,m_1}()}(\mathsf{params}, \mathsf{pk})$;
`si` $m_0, m_1 \notin \mathcal{M}$ `ou` $m_0 = m_1$;
stopper ; `sinon`
$c^\star \leftarrow \mathsf{Encrypt}(\mathsf{pk}, m_b)$;
$b' \leftarrow \mathcal{A}_2^{\mathcal{O}\mathsf{Decrypt}_{m_0,m_1}()}(c^\star, \mathsf{state})$;
`retourner` b'

$\mathcal{O}\mathsf{Decrypt}_{m_0,m_1}(c)$
$\mathsf{CSet} \leftarrow \mathsf{CSet} \cup \{(c)\}$;
$m \leftarrow \mathsf{Decrypt}(\mathsf{sk}, c)$;
`si` $m \in \{m_0, m_1\}$;
`retourner` 'replay' ;
`sinon retourner` m

On dit qu'un schéma de rechiffrement universel $\mathcal{R}\mathsf{e}\mathcal{E}\mathsf{nc}$ est sémantiquement sûr sous rechiffrement avec rejeu résistant aux attaques à chiffrés choisis, ou IND-RCCA, si dans l'expérience ci-dessus, l'attaquant $\mathcal{A}$ devine b avec un avantage inférieure ou égale ϵ, où ϵ est une fonction négligeable en le paramètre de sécurité ; on définit l'avantage par :

$$\mathsf{Adv}^{\mathsf{ind\text{-}rcca}}_{\mathcal{R}\mathsf{e}\mathcal{E}\mathsf{nc},\mathcal{A}} = \left|\Pr[\mathsf{Exp}^{\mathsf{ind\text{-}rcca}-1}_{\mathcal{R}\mathsf{e}\mathcal{E}\mathsf{nc},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{ind\text{-}rcca}-0}_{\mathcal{R}\mathsf{e}\mathcal{E}\mathsf{nc},\mathcal{A}}(\lambda) = 1]\right|$$

En 2004, Groth [77] propose le premier schéma IND-RCCA contre tout attaquant générique, cette construction est malheureusement inefficace. À Crypto 2007, Prahbhakaran *et al.* [108] proposent le premier schéma rerandomizable résistant aux attaques RCCA dans le modèle standard, en modifiant le second schéma décrit paragraphe 5.2.3. Nous décrivons ce schéma en procédant par étapes dans le but de donner l'idée intuitive du cheminement. De plus, cette approche constitue en quelque sorte une justification de notre construction, que nous présenterons dans la section qui suit.

Construction "replayable CCA"

Étape 1 : dédoubler un "chiffré Cramer-Shoup"

Les paramètres publics sont $\mathsf{params} \overset{\text{déf}}{=} \langle q, \mathbb{G}, g_1, g_2\rangle$ où $\mathbb{G}$ est un groupe cyclique d'ordre un grand nombre premier q. g_1, g_2 sont deux générateurs aléatoires de $\mathbb{G}$. Les clés secrètes et publiques sont respectivement définies par :

$$\mathsf{sk} = (x_1, x_2, y_1, y_2, z) \quad \text{et} \quad \mathsf{pk} = (g_1, g_2, h, B, C, D), \text{ avec :}$$

$$B = {g_1}^z \quad C = g_1^{x_1} g_2^{x_2} \quad \text{et} \quad D = g_1^{y_1} g_2^{y_2}.$$

Pour chiffrer un message M quelconque, on encode ce message en élément m de $\mathbb{G}$, puis on calcule :

$$\mathsf{Encrypt}(\mathsf{pk}, m) = (g_1^r, g_2^r, mB^r, C^r D^{r\cdot\mu}),$$

où μ est l'encodage de M et non plus le haché des trois premiers éléments du chiffrés comme dans le schéma initial.

On applique la procédure de dédoublement comme dans [76], on obtient un chiffré de la forme :

$$\mathsf{Encrypt}(\mathsf{pk}, m) = (g_1^r, g_2^r, m \cdot B^r, C^r D^{r\cdot\mu}, g_1^s, g_2^s, B^s, C^s D^{s\cdot\mu}), \text{ avec } r, s \xleftarrow{R} \mathbb{Z}_q.$$

Remarquons que comme pour les schémas USS sûr, la seconde composante du chiffré permet de rerandomiser le chiffrer. Mais pour ce premier schéma, cette seconde composante contient des éléments publics chiffrés alors que pour appliquer ce paradigme à un "chiffré Cramer-Shoup", la seconde composante doit contenir un chiffré du message.

D'autre part, on remarque que le chiffré précédent présente deux "faiblesses" majeures auxquelles nous allons nous intéresser :

1. deux composantes de deux chiffrés différents du même message μ peuvent être combinées pour former un chiffré valide ;
2. le chiffré possède suffisamment de redondance si bien qu'un invariant subsiste après rerandomization : plus précisément, pour tout chiffré bien formé, on a :

$$\log_{g_1} g_2 = \log_{c_1} c_2 = \log_{c_5} c_6, \quad \log_{CD^\mu} c_4 = \log_B c_3/m \quad \text{et} \quad \log_{CD^\mu} c_8 = \log_B c_7$$

Étape 2 : restreindre la malléabilité du schéma

Nous nous intéresserons au second point lié aux canaux subliminaux dans la section qui suit. Le premier point est considéré dans [108]. Afin de restreindre les recombinaisons de chiffrés indépendants associés à un même message μ, les auteurs de cet article proposent de relier les deux composantes du chiffré avec un aléa x commun. Par ailleurs, afin de permettre le rechiffrement, cet aléa x doit être chiffré avec un schéma de rechiffrement possédant certaines propriétés. On obtient finalement le chiffré ci dessous :

$$c = ((g_1^r)^x, (g_2^r)^x, MB^r, C^r D^{r\cdot\mu}, (g_1^s)^x, (g_2^s)^x, B^s, C^s D^{s\cdot\mu}, u = \mathsf{MEncrypt}(x)),$$

où $\mathcal{R}\mathrm{e}\mathcal{E}\mathsf{nc} \stackrel{\text{déf}}{=} \langle \mathsf{MSetup}, \mathsf{MKeyGen}, \mathsf{MEncrypt}, \mathsf{MDecrypt} \rangle$ est un schéma de chiffrement malléable (pour permettre la rerandomization de l'aléa u) et rerandomizable à valeurs dans un sous-groupe de $\mathbb{Z}_p^\star$).

Enfin, on remarque que la première composante du chiffré est malléable (multiplicativement). Pour corriger ce point, on perturbe l'aléa de la première composante de manière additive. Cette dernière dérivation est non triviale, nous décrivons ci-dessous le schéma de rechiffrement final (universel et IND-RCCA sûr) obtenu dans [108]. Ce schéma utilise comme brique de base le schéma présenté section 5.2.3, dont les éléments appartiennent à un sous-groupe de $\mathbb{Z}_p^\star$, que nous noterons $\hat{\mathbb{G}}$.

Étape 3 : schéma final

Setup : étant donné un paramètre de sécurité, cet algorithme définit les paramètres publics du système :

- une structure de groupe multiplicative $\langle \mathbb{G}, p, g \rangle$, où $\mathbb{G}$ est un groupe d'ordre premier p et g un générateur de $\mathbb{G}$;
- une fonction injective g_{enc} (resp. f_{enc}) encodant des messages dans $\mathbb{G}$ (resp. dans $\mathbb{Z}_p$ ou $\mathbb{Z}_p^\star$),
- un schéma de chiffrement $\mathcal{R}\mathrm{e}\mathcal{E}\mathsf{nc}$ rerandomizable défini par :

$$\mathcal{R}\mathrm{e}\mathcal{E}\mathsf{nc} \stackrel{\text{déf}}{=} \langle \mathsf{MKeyGen}, \mathsf{MEncrypt}, \mathsf{MReRand}, \mathsf{MDecrypt} \rangle$$

homomorphe dans un sous-groupe d'ordre premier q de $\mathbb{Z}_p^\star$. On note ce groupe $\hat{\mathbb{G}}$ et $\otimes$ la loi de groupe dans $\hat{\mathbb{G}}$.

– et enfin un vecteur fixe $\bar{e} = (e_1, e_2, e_3, e_4)$.

KeyGen : étant donnés les paramètres publics, cet algorithme génère une paire clé secrète/clé publique sk, pk, où

$$\mathsf{sk} = (\bar{x}, \bar{y}, \bar{z}) \text{ et } \mathsf{pk} = (g_1, g_2, g_3, g_4, C, D, E),$$

où $\bar{x} = (x_1, x_2, x_3, x_4)$ et $\bar{y} = (y_1, y_2, y_3, y_4)$ et $\bar{z} = (z_1, z_2, z_3, z_4)$, les g_i sont des générateurs aléatoires et :

$$C = \prod_{i=1}^{4} g_i^{x_i}, \qquad D = \prod_{i=1}^{4} g_i^{y_i}, \qquad E = \prod_{i=1}^{4} g_i^{z_i}$$

Encrypt :

1. choisir $x \overset{R}{\leftarrow} \mathbb{Z}_p$ et $y \overset{R}{\leftarrow} \mathbb{Z}_p^\star$ et $\hat{u} \in \hat{G}$;
2. pour $i = 1, \cdots, 4$, définir $X_i = g_i^{(x+e_i)\cdot\hat{u}}$ et $Y_i = g_i^{y\cdot\hat{u}}$;
3. calculer $m = g_{\mathsf{enc}}(M)$ et $\mu = f_{\mathsf{enc}}(M)$;
4. `retourner`
$$(\bar{X}, mC^x, (DE^\mu)^x, \bar{Y}, C^y, (DE^\mu)^y, \hat{U}),$$
où $\bar{X} = (X_1, X_2, X_3, X_4)$ et $\bar{Y} = (Y_1, Y_2, Y_3, Y_4)$ et $\hat{U} = \mathsf{MEncrypt}(\hat{u})$

ReRand : 1. Décomposer c en $(c_1, c_2, c_3, c_4, c_5, c_6, c_7)$;
2. choisir $\hat{r} \overset{R}{\leftarrow} \hat{G}, s \overset{R}{\leftarrow} \mathbb{Z}_p$ et $t \overset{R}{\leftarrow} \mathbb{Z}_p^\star$;
3. calculer $\hat{U}' = \mathsf{MReRand}(\hat{r} \otimes c_7)$;
4. pour $i = 1, \cdots, 4$ calculer $X_i' = (X_i Y_i^s)^{\hat{r}}$ et $Y_i' = Y_i^{\hat{r}\cdot t}$;
on pose $\bar{X}' \overset{\text{déf}}{=} (X_1', X_2', X_3', X_4')$ et $\bar{Y}' \overset{\text{déf}}{=} (Y_1', Y_2', Y_3', Y_4')$
5. calculer $c_2' = c_2 \cdot c_5^s$ et $c_3' = c_3 \cdot c_6^s$;
6. $c_5' = c_5^t$ et $c_6 = c_6^t$;

`retourner` $(\bar{X}', c'_2, c'_3, \bar{Y}', c'_5, c'_6)$

Decrypt : 1. décomposer c en $(\bar{c}_1, c_2, c_3, \bar{c}_4, c_5, c_6, c_7)$,
avec $\bar{c}_1 = (X_1, X_2, X_3, X_4)$ et $\bar{c}_2 = (Y_1, Y_2, Y_3, Y_4)$
2. déchiffrer $\hat{u} = \mathsf{MDecrypt}(c_7)$. `si` $\hat{u} = \perp$ `retourner` $\perp$; `sinon`
3. pour $i = 1, \cdots, 4$ poser $\bar{X}_i = X_i^{1/\hat{u}} g_i^{-e_i}$ et $\bar{Y}_i = Y_i^{1/\hat{u}}$
4. poser $m = c_2 / \prod_{i=1}^4 \bar{X}_i^{x_i}$; calculer $M = g_{\mathsf{enc}}^{-1}(m)$ et $\mu = f_{\mathsf{enc}}(M)$;
5. vérifier l'intégrité du chiffré :
$$c_5 \overset{?}{=} \prod_{i=1}^{4} \bar{Y}_i^{x_i}, \qquad c_3 \overset{?}{=} \prod_{i=1}^{4} \bar{X}_i^{y_i + z_i\cdot\mu}, \qquad c_6 \overset{?}{=} \prod_{i=1}^{4} \bar{Y}_i^{y_i + z_i\cdot\mu}.$$
`si` $Y_1 = Y_2 = Y_3 = Y_4 = 1$ `ou si` l'une des égalités ci-dessus n'est pas vérifiée, `retourner` $\perp$; `sinon retourner` M.

Théorème 8 *Le schéma précédent est parfaitement rerandomizable et* RCCA*-sûr sous l'hypothèse* DDH *dans* $\mathbb{G}$ *et* $\hat{\mathbb{G}}$.

La technique de preuve de ce résultat est une extension non triviale des preuves basées sur le schéma de Cramer-Shoup : l'algorithme de chiffrement et de déchiffrement sont modifiés de manière à ne laisser fuir aucune information sur le message. Elle s'appuie essentiellement sur une représentation algébrique du schéma qui consiste à montrer une indépendance linéaire comme dans [11]. La preuve de notre schéma ne s'appuyant pas sur la preuve de cette construction, nous reportons le lecteur à l'article [108].

5.3 Extension de l'anonymat

Lors des descriptions des protocoles multi-acteurs, signature de groupe et chiffrement de groupe, nous avons pu voir que tout utilisateur qui désire devenir membre du groupe doit commencer par s'enregistrer. Du point de vue du simulateur, cet algorithme supplémentaire est très commode : il permet de vérifier la légitimité d'un membre interagissant avec un autre membre du groupe. Dans le cas des signatures, la nécessité de ce protocole d'enregistrement paraît évidente : la génération d'une signature au nom d'un membre du groupe suppose que l'émetteur connait un secret. Mais qu'en est-il pour le chiffrement ? Pour le chiffrement de groupe, il permet de désigner les membres du groupe et de vérifier que le chiffré est consistant, c.à.d. associé à l'un des membres ; dans le cas contraire, un symbole d'erreur est renvoyé. Nous étendons cette notion de consistance, en prenant en compte tous les chiffrés. Plus exactement, nous voulons être sûr que si le chiffré est "inconsistant"[5], il ne laisse passer aucune information subliminale : dans ce cas, nous montrerons qu'il s'agit d'un chiffré indistinguable d'un chiffré aléatoire. Les travaux présentés dans cette section ont été menés en collaboration avec David Pointcheval et Damien Vergnaud.

5.3.1 Primitive de groupe

Phase d'enregistrement

Tout d'abord, pour construire une primitive de groupe, une fonctionnalité supplémentaire est nécessaire afin d'empêcher les utilisateurs de fabriquer leur clé de manière illégale, tout en ayant la capacité de détecter toute utilisation frauduleuse de la clé ; nous avons considéré les deux possibilités suivantes :

1. nous pouvons opter pour un schéma de chiffrement à base d'identité. Dans ce cas, l'autorité qui délivre les clés connaît les clés secrètes de tous les utilisateurs, le problème du *Key-escrow* se pose alors ;
2. nous pouvons également construire un schéma avec une phase d'enregistrement si bien que chaque membre possédant une clé obtient une paire de clés (secrète et publique) et le certificat associé sans que l'autorité chargée de délivrer le certificat ne connaisse la clé secrète associée. Il s'agit en fait de l'analogue du protocole d'enregistrement pour les schémas de signatures certifiées dans le contexte du chiffrement. Dans la description des schémas de chiffrement de groupe de la section 5.1.2, le protocole Join permet d'obtenir cette fonctionnalité, il est implanté par un schéma de signature et par l'usage de preuves d'appartenance de la paire de clés au langage défini par les paires valides. Un des inconvénients majeurs est l'interactivité induite par ce protocole et l'utilisation coûteuse de preuves d'appartenance.

Pour notre schéma, nous établissons un compromis entre ces deux solutions. Plus exactement, lorsqu'un utilisateur désire devenir membre du groupe, il est en possession d'une paire de clés $(\mathsf{pk}, \mathsf{sk})$ délivrée par une *Infrastructure de clés publiques* (ou *Public Key Infrastructure*, PKI). Précisons que cette dernière autorité retourne une paire de clés certifiée à tous les membres, les autorités y compris, afin d'empêcher quiconque de tromper l'utilisateur s'enregistrant et de révoquer un utilisateur à tort.

Ensuite, l'utilisateur fournit une signature de sa clé publique[6] au manager du groupe ; il s'agit d'un certificat. Ce dernier vérifie la validité du certificat et définit une clé secrète

[5] le chiffré est bien formé mais il n'est pas associé à une clé publique enregistrée.

[6] sa clé publique est un encodage de son identité qui peut être vue comme un pseudonyme.

associée à l'identité de l'utilisateur. Il est important de préciser que cette dernière clé ne permet pas de casser la sécurité sémantique : on suppose que les messages sont chiffrés au préalable et le schéma global est employé comme "sur-couche" permettant d'obtenir une primitive de groupe.

Le médiateur

Revenons sur la procédure de rerandomization : celle-ci ne requiert pas la connaissance d'un secret particulier et n'importe qui peut rerandomiser un chiffré. Afin d'assurer que le schéma est correct et que tout chiffré est bien rechiffré en un chiffré indistinguable du même message, la procédure doit être appliquée honnêtement. Pour cela, on considère un modèle introduit par Simmons [119], où un médiateur, le gardien de prison supposé honnête autorise deux prisonniers à communiquer entre eux. Il autorise les deux individus à se mettre d'accord sur des paramètres communs puis à échanger des messages d'une certaine forme. À l'exemple de ce modèle, nous supposons que la phase de rechiffrement est effectuée par un médiateur honnête, mais ne possédant aucun secret particulier contrairement au manager ou l'autorité de révocation.

5.3.2 Modèles

Définition

Nous donnons à présent les définitions de notre nouveau concept, $\mathcal{M}$ATES (*Mediated Anonymous Traceable Encryption*) que nous avons conçu en collaboration avec David Pointcheval et Damien Vergnaud. Une description plus complète peut parallèlement être consultée [86]. On considère trois autorités : une autorité d'enregistrement (*issuer* ou *manager* en anglais) qui a pour charge l'ajout de nouveaux membres, une autorité d'ouverture (*opener* en anglais) qui est capable de révoquer l'anonymat des destinataires des chiffrés, et un médiateur qui rechiffre les entrées de manière systématique. Cette dernière autorité est supposée honnête mais curieuse : afin de garantir l'absence de canaux subliminaux, nous supposons que les membres ne peuvent pas s'associer au médiateur. De plus, afin d'empêcher les autorités d'enregistrement et d'ouverture de corrompre un utilisateur, nous supposons l'existence d'une infrastructure PKI délivrant une paire clé publique/clé privée pk/sk certifiées à chaque utilisateur (aux autorités et aux utilisateurs).

Définition 18 (Mediated Anonymous Traceable Encryption Schemes) *Un schéma* $\mathcal{M}$ATES *est défini par la donnée des algorithmes ou protocoles suivants :*

$$\langle \mathsf{Setup}, \mathsf{Join}, \mathsf{Encrypt}, \mathsf{ReRand}, \mathsf{Decrypt}, \mathsf{Trace}, \mathsf{Judge} \rangle,$$

définis de la manière suivante :

Setup : *cet algorithme est probabiliste et est exécuté par une autorité de confiance. Il prend en entrée un paramètre de sécurité* λ *et génère trois clés : une clé publique de groupe* mpk*, la clé secrète de l'autorité d'enregistrement* msk *et une clé secrète d'ouverture* sk_O *; Il génère également une structure de données* $\mathcal{L}$*, appelée* liste d'enregistrement *qui est initialement vide ;*

Join : *il s'agit d'un protocole interactif polynomial entre un membre possédant une paire de clés certifiées et l'autorité d'enregistrement :*

JoinM : *est un algorithme initié par un utilisateur en possession d'une paire de clés* $(\mathsf{pk}, \mathsf{sk})$ *délivrée par une autorité de confiance et prenant en entrée l'identité de l'utilisateur* ID*, la clé publique* mpk *et sa clé secrète* sk*. À l'issue de ce protocole, il reçoit une paire de clés secrète et publique associée à son identité,* $\mathsf{pk}_{\mathsf{ID}}$ *et* $\mathsf{sk}_{\mathsf{ID}}$ *;*

JoinG : *il s'agit d'un protocole initié par le manager en possession des clés* mpk *et* msk*. Il prend en entrée une identité* ID *de l'utilisateur désirant devenir membre du groupe. Il vérifie la validité de la clé* pk *et retourne un certificat associé qu'il ajoute dans une structure de données* $\mathcal{L}$*, appelée* liste d'enregistrement *des membres.*

Plus formellement, avec nos notations, nous avons :

$$((\mathsf{pk}_{\mathsf{ID}}, \mathsf{sk}_{\mathsf{ID}}, \mathcal{L}), (\mathsf{pk}_{\mathsf{ID}}, \mathcal{L})) \leftarrow \langle \mathsf{JoinM}(\mathsf{ID}, \mathsf{mpk}, \mathsf{sk}), \mathsf{JoinG}(\mathsf{ID}, \mathsf{msk}, \mathsf{pk}) \rangle.$$

Encrypt : *cet algorithme (probabiliste) prend en entrée un message* m*, l'identifiant* $\mathsf{pk}_{\mathsf{ID}}$ *du destinataire et la clé publique* mpk*. Il renvoie un chiffré* c *du message* m *associé à l'utilisateur* ID *;*

ReRand : *cet algorithme (probabiliste) prend en entrée la clé publique de groupe* mpk *et un chiffré* c *et renvoie le rechiffré* c' *de* c *;*

Decrypt : *cet algorithme (déterministe) prend en entrée la clé publique de groupe* mpk*, une clé secrète d'un membre* $\mathsf{sk}_{\mathsf{ID}}$ *et un chiffré* c *et renvoie un message si* c *est un chiffré de* m *associé à l'identité* ID *et un symbole d'erreur sinon ;*

Trace : *cet algorithme (déterministe) prend en entrée la clé publique de groupe* mpk*, la liste d'enregistrement* $\mathcal{L}$*, la clé d'ouverture* sk_O*, une identité et un chiffré* c*. Cet algorithme renvoie une* 0 *ou* 1 *suivant que le chiffré soit destiné à l'utilisateur* ID *ou non et une preuve* Π *destinée au Juge indiquant si le chiffré est bien associé à l'utilisateur* ID*. En particulier, pour tout message* m*, on a :* $\mathsf{Trace}(\mathsf{ReRand}(\mathsf{Encrypt}(m))) = (1; \Pi)$*. On écrira plus simplement* $\mathsf{Trace}(\mathsf{ReRand}(\mathsf{Encrypt}(m))) = 1$*.*

Judge : *cet algorithme prend en entrée la clé publique de groupe* mpk*, la liste d'enregistrement* $\mathcal{L}$*, un chiffré* c*, une identité* ID *et une preuve* Π*. Il vérifie que la preuve* Π *est correcte, c.à.d. que la preuve en entrée prouve que* ID *est le destinataire du chiffré* c*.*

Modèle de sécurité

Comme nous l'avons déjà précisé dans la description précédente, après l'exécution du protocole Join, la clé publique de l'utilisateur ainsi que son identité sont enregistrées dans la liste $\mathcal{L}$, mais la clé privée sk délivrée par la *PKI* est gardée secrète. Dans notre modèle, chaque membre possède sa clé secrète $\mathsf{sk}_{\mathsf{ID}}$, délivrée par le manager du groupe et éventuellement plusieurs clés publiques, mais il n'a pas accès aux autres clés secrètes.

En pratique, l'utilisation de cartes à puces pour le stockage des clés secrètes permet la réalisation de ce modèle. D'un point du vue théorique, cela signifie que nous excluons toute coalition de membres, les utilisateurs ne peuvent pas construire un algorithme de déchiffrement en combinant les clés secrètes entre elles. Nous introduisons l'oracle d'enregistrement suivant :

$\mathsf{Join}^*()$: cet oracle simule plusieurs réponses de requêtes au protocole Join, il retourne plusieurs clés publiques (la liste $\mathcal{L}$ est mise à jour) et retourne une paire de clés $(\mathsf{pk}_{\mathsf{ID}}, \mathsf{sk}_{\mathsf{ID}})$.

L'attaquant peut faire plusieurs requêtes Join mais il ne reçoit que les clés publiques et pour l'une des exécutions, il reçoit une clé secrète, supposée être la sienne.

Pour les notions de sécurité que nous allons définir, nous considérons seulement les attaques à clairs choisis, l'adversaire n'a pas accès à l'oracle de déchiffrement. On peut bien sûr étendre le modèle de sécurité afin de prendre en compte les attaques à chiffrés choisis sous rechiffrement[7], en adaptant la notion de sécurité RCCA. Rappelons que l'existence d'une primitive de groupe vérifiant les propriétés de sécurité sémantique (sous rechiffrement) et d'anonymat résistant aux attaques à chiffrés choisis avec rejeu est actuellement un problème ouvert, posé dans l'article [108].

Notion de sécurité sémantique

La principale notion de sécurité pour un schéma de chiffrement est la sécurité sémantique, qui est formalisée par l'indistinguabilité de deux expériences (pour $b = 0, 1$). Dans ce jeu, nous pouvons donner la clé d'ouverture sk_O à l'attaquant, ce qui permet de modéliser la coalition d'un participant avec l'autorité d'ouverture. Pour cette notion, nous nous restreignons aux clés publiques valides[8] car sinon personne ne peut déchiffrer le message. Nous nous intéressons donc seulement à la confidentialité des chiffrés associés à un membre enregistré. Plus formellement, on définit l'expérience $\mathsf{Exp}^{\mathsf{ind}\text{-}b}_{\mathcal{M}_{\mathrm{ATES}},\mathcal{A}}$ pour $b = 0, 1$:

$\mathsf{Expérience}\ \mathsf{Exp}^{\mathsf{ind}\text{-}b}_{\mathcal{M}_{\mathrm{ATES}},\mathcal{A}}(\lambda)$
$(\mathsf{mpk}, \mathsf{msk}, \mathsf{sk}_O, \mathcal{L}) \leftarrow \mathsf{GSetup}(\lambda)$;
$(\mathsf{pk}, m_0, m_1) \leftarrow \mathcal{A}^{\mathsf{Join}^*()}(\mathtt{FIND}, \mathsf{mpk}, \mathsf{sk}_O, \mathcal{L})$;
$C^* \leftarrow \mathsf{Encrypt}(\mathsf{mpk}, \mathsf{pk}, m_b)$;
$b' \leftarrow \mathcal{A}(\mathtt{GUESS}, C^*)$;
`si` $\mathsf{pk} \notin \mathcal{L}$ `retourner` 0 ;
`sinon retourner` b'

Nous définissons l'avantage de l'attaquant $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{M}_{\mathrm{ATES}},\mathcal{A}}$ à casser la sécurité sémantique (sous les attaques à clairs choisis) par son avantage à distinguer les expériences $\mathsf{Exp}^{\mathsf{ind}\text{-}b}_{\mathcal{M}_{\mathrm{ATES}},\mathcal{A}}$ pour $b = 0, 1$:

$$\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{M}_{\mathrm{ATES}},\mathcal{A}}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{ind}\text{-}1}_{\mathcal{M}_{\mathrm{ATES}},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{ind}\text{-}0}_{\mathcal{M}_{\mathrm{ATES}},\mathcal{A}}(\lambda) = 1].$$

Nous définissons également une version plus faible pour la sécurité sémantique, par laquelle l'attaquant n'a pas accès à la clé de révocation ; cette notion peut être utile lorsqu'on ne peut pas séparer les rôles des autorités d'enregistrement et de révocation. De manière similaire, l'avantage de l'attaquant est donné par la formule suivante :

$$\mathsf{Adv}^{\mathsf{weak-ind}}_{\mathcal{M}_{\mathrm{ATES}},\mathcal{A}}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{weak-ind}\text{-}1}_{\mathcal{M}_{\mathrm{ATES}},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{weak-ind}\text{-}0}_{\mathcal{M}_{\mathrm{ATES}},\mathcal{A}}(\lambda) = 1].$$

Notion d'anonymat

Rappelons que notre but est de garantir l'anonymat du destinataire : dans le contexte du chiffrement, l'anonymat se traduit par la garantie du maintien de la confidentialité de la clé publique. Nous formalisons cette propriété par l'indistinguabilité des deux expériences $\mathsf{Exp}^{\mathsf{anon}\text{-}b}_{\mathcal{M}_{\mathrm{ATES}},\mathcal{A}}$ (pour $b = 0, 1$). Comme pour la sécurité sémantique, on restreint l'adversaire à n'utiliser que des clés publiques valides (ou enregistrées).

[7] certaines précautions doivent être prises du fait de la malléabilité inhérente des schémas de rechiffrement.

[8] ou enregistrées.

Nous pouvons également donner accès à l' "oracle d'ouverture" : cette requête supplémentaire permet de modéliser la propriété d'*anonymat fort* (ou *full-anonymity* en anglais), mais comme notre candidat ne vérifie pas cette propriété nous nous concentrons sur la notion d'anonymat classique. Nous définissons ci-dessous l'expérience $\mathsf{Exp}^{\mathsf{anon}\text{-}b}_{\mathcal{M}_{\mathrm{ATES}},\mathcal{A}}$ pour $b = 0, 1$:

$\mathsf{Expérience}\ \mathsf{Exp}^{\mathsf{anon}\text{-}b}_{\mathcal{M}_{\mathrm{ATES}},\mathcal{A}}(\lambda)$
$(\mathsf{mpk}, \mathsf{msk}, \mathsf{sk}_O, \mathcal{L}) \leftarrow \mathsf{GSetup}(\lambda)$;
$(\mathsf{pk}_0, \mathsf{pk}_1, m) \leftarrow \mathcal{A}^{\mathsf{Join}^*()}(\mathtt{FIND}, \mathsf{mpk}, \mathcal{L})$;
$C^* \leftarrow \mathsf{Encrypt}(\mathsf{mpk}, \mathsf{pk}_b, m)$;
$b' \leftarrow \mathcal{A}(\mathtt{GUESS}, C^*)$;
`si` $\mathsf{pk}_0 \notin \mathcal{L}$;
  `ou` $\mathsf{pk}_1 \notin \mathcal{L}$ `retourner` 0 ;
`sinon retourner` b'

Nous définissons l'avantage de l'attaquant $\mathcal{A}$ à casser l'anonymat (sous les attaques à clairs choisis) par son avantage à distinguer les deux expériences $\mathsf{Exp}^{\mathsf{anon}\text{-}b}_{\mathcal{M}_{\mathrm{ATES}},\mathcal{A}}$ (pour $b = 0, 1$). Plus formellement, on a :

$$\mathsf{Adv}^{\mathsf{anon}}_{\mathcal{M}_{\mathrm{ATES}},\mathcal{A}}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{anon}\text{-}1}_{\mathcal{M}_{\mathrm{ATES}},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{anon}\text{-}0}_{\mathcal{M}_{\mathrm{ATES}},\mathcal{A}}(\lambda) = 1].$$

Absence de canaux subliminaux

Rappelons que notre objectif est de construire un schéma de chiffrement anonyme (c.à.d. qui masque l'identité du destinataire d'un chiffré), révocable (c.à.d. nous pouvons identifier le destinataire d'un chiffré *en temps polynomial*). Par conséquent, notre notion de sécurité formalise les propriétés suivantes : ou bien l'algorithme de traçage parvient à identifier une identité valide, ou bien aucune information ne peut être transmise au destinataire qui reste anonyme. Il s'agit de la propriété usuelle de traçabilité ou *traceability* pour les signatures de groupes. Notons par ailleurs que ce résultat est en parfaite liaison avec l'emploi d'une phase de rerandomization additionnelle à la primitive de chiffrement.

Cette propriété se formalise par l'indistinguabilité des deux expériences $\mathsf{Exp}^{\mathsf{subF}-b}_{\mathcal{M}_{\mathrm{ATES}},\mathcal{A}}$ (pour $b = 0, 1$) décrites ci-dessus : si l'attaquant parvient à faire passer de l'information dans le chiffré C_b (éventuellement même après rerandomization), alors il peut deviner le bit b.

Puisque nous voulons exclure cette possibilité pour les chiffrés qui ne sont associés à aucune identité enregistrée, nous définissons l'algorithme de test suivant :

$$\mathsf{Traceable}(\mathsf{mpk}, \mathcal{L}, \mathsf{ID}, \mathsf{sk}_O, C)$$

Si un t-uplet $(\mathsf{mpk}, \mathcal{L}, \mathsf{ID}, \mathsf{sk}_O, C)$ vérifie ce critère, cela signifie que

$$(\mathsf{ID}, \Pi) \leftarrow \mathsf{Trace}(\mathsf{mpk}, , \mathsf{sk}_O, C) \text{ et de plus } \mathsf{Judge}(\mathsf{mpk}, , C, \mathsf{ID}, \Pi) = \mathsf{Ok}.$$

Plus précisément, cela signifie que l'algorithme de traçage renvoie une identité valide avec une preuve convaincante. Comme précédemment, nous supposons que l'attaquant a accès à plusieurs requêtes $\mathsf{Join}^*()$ qui lui permettent d'ajouter plusieurs membres, mais il n'obtient seulement qu'une clé secrète, puisque nous excluons les coalitions de traîtres. Nous définissons l'expérience $\mathsf{Exp}^{\mathsf{subF}\text{-}b}_{\mathcal{M}_{\mathrm{ATES}},\mathcal{A}}$ pour $b = 0, 1$ par le jeu de sécurité suivant :

$\mathsf{Expérience}\ \mathsf{Exp}^{\mathsf{subF}\text{-}b}_{\mathcal{M}_{\mathrm{ATES}},\mathcal{A}}(\lambda)$
$(\mathsf{mpk}, \mathsf{msk}, \mathsf{sk}_O, \mathcal{L}) \leftarrow \mathsf{GSetup}(\lambda)$
$(C_0, C_1) \leftarrow \mathcal{A}^{\mathsf{Join}^*()}(\texttt{FIND}, \mathsf{mpk}, \mathcal{L})$
$C^* \leftarrow \mathsf{ReRand}(\mathsf{mpk}, C_b)$
$b' \leftarrow \mathcal{A}(\texttt{GUESS}, C^*)$
`si` $\mathsf{Traceable}(\mathsf{mpk}, \mathcal{L}, \mathsf{ID}, \mathsf{sk}_O, C_0)$
`ou` $\mathsf{Traceable}(\mathsf{mpk}, \mathcal{L}, \mathsf{ID}, \mathsf{sk}_O, C_1)$ `retourner` 0
`sinon retourner` b'

On définit l'avantage de l'attaquant $\mathcal{A}$ à construire un canal subliminal par son avantage à distinguer les deux expériences $\mathsf{Exp}^{\mathsf{subF}-b}_{\mathcal{M}_{\mathrm{ATES}},\mathcal{A}}$ (pour $b = 0$ ou $b = 1$) :

$$\mathsf{Adv}^{\mathsf{subF}}_{\mathcal{M}_{\mathrm{ATES}},\mathcal{A}}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{subF}\text{-}1}_{\mathcal{M}_{\mathrm{ATES}},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{subF}\text{-}0}_{\mathcal{M}_{\mathrm{ATES}},\mathcal{A}}(\lambda) = 1].$$

Relation entre les notions de sécurité

Supposons qu'il existe un attaquant contre la sécurité sémantique faible (voir paragraphe 5.3.2) ou l'anonymat pour un chiffré c challenge associé à une identité ID différentes de celles possédées par l'attaquant. Ce dernier peut alors faire passer de l'information en utilisant les deux messages ou les deux identités sur lesquels l'attaquant s'engage dans les deux jeux précédents. Remarquons que par définition, les jeux de sécurité pour ces deux notions (sémantique et anonymat) portent sur des chiffrés associés à une clé secrète différente de $\mathsf{sk}_{\mathsf{ID}}$, la clé secrète de l'utilisateur.

5.3.3 Description de notre schéma

$\mathsf{GSetup}(\lambda)$ **:** l'algorithme GSetup prend en entrée un paramètre de sécurité λ et définit un groupe cyclique $\mathbb{G}$ d'ordre premier q et un générateur g. On note $\bar{\mathbb{G}}$ le groupe $\mathbb{G}$ privé de son unité ($\bar{\mathbb{G}} = \mathbb{G}\backslash\{1\}$) ; il s'agit du groupe des éléments d'ordre exactement q dans $\mathbb{G}$. De plus, il choisit $x_1^{(0)}, x_1^{(1)}, \ldots, x_\ell^{(0)}, x_\ell^{(1)} \in \mathbb{Z}_p$, où la taille ℓ est la longueur de la clé publique, tandis que les identités sont de longueur μ.

Il choisit également un code de distance minimale 2, qui encode des mots de longueur μ bits (les identités) en des mots de longueurs ℓ (les clés publiques). Nous définissons $\mathcal{G}$ un algorithme probabiliste de génération de clés (qui peut éventuellement être secret vis à vis du manager) : $\mathsf{pk}_{\mathsf{ID}} = \mathcal{G}(\mathsf{ID})$.

Les clés maître et d'ouverture sont définies par : $\mathsf{msk} = \mathsf{sk}_O = (x_1^{(0)}, x_1^{(1)}, \ldots, x_\ell^{(0)}, x_\ell^{(1)})$, et la clé publique de groupe $\mathsf{mpk} = (\Omega_1^{(0)}, \Omega_1^{(1)}, \ldots, \Omega_\ell^{(0)}, \Omega_\ell^{(1)})$, où $\Omega_i^{(b)} = g^{1/x_i^{(b)}}$ pour $b \in \{0,1\}$ et $i \in \{1, \ldots, \ell\}$. Il retourne également une liste $\mathcal{L}$ initialement vide ;

Join : il s'agit d'un protocole interactif entre le manager et un utilisateur désirant devenir membre du groupe. Il est défini de la manière suivante :

1. $\mathsf{JoinM}(\mathsf{ID}, \mathsf{mpk}, \mathsf{sk})$: cet algorithme encode les identités en des clés publiques, $\mathsf{pk}_{\mathsf{ID}} = \mathcal{G}(\mathsf{ID}) = h_1 h_2 \ldots h_\ell$, où $h_i \in \{0,1\}$. Le membre utilise sa clé secrète sk (délivrée par une PKI) pour donner une signature de $\mathsf{pk}_{\mathsf{ID}}$;
2. $\mathsf{JoinG}(\mathsf{ID}, \mathsf{pk}, \mathsf{msk})$: la clé secrète de l'utilisateur est défini par : $\mathsf{sk}_{\mathsf{ID}} = (x_1^{(h_1)}, \ldots, x_\ell^{(h_\ell)})$. Le manager du groupe met à jour la liste d'enregistrement $\mathcal{L}$ en ajoutant l'identité ID, la clé publique $\mathsf{pk}_{\mathsf{ID}}$ et sa signature fournie par l'utilisateur.

$\mathsf{Encrypt}(\mathsf{mpk}, \mathsf{pk}_{\mathsf{ID}}, m)$ **:** pour chiffrer un message $m \in \mathbb{G}$ avec la clé $\mathsf{pk}_{\mathsf{ID}} = \mathcal{G}(\mathsf{ID}) = h_1 \dots h_\ell$,

1. choisir des éléments aléatoires $K_i, U_i \in \mathbb{G}$, pour $i = 1, \dots, \ell$, tels que $\prod_{i=1}^{\ell} K_i = m$ et $\prod_{i=1}^{\ell} U_i = 1$;
2. choisir deux séquences de scalaires aléatoires $t_i, s_i \in \mathbb{Z}_q$ pour $i = 1, \dots, \ell$;
3. pour $i = 1, \dots, \ell$, calculer

$$A_i = g^{t_i} \times K_i, \quad B_i = (\Omega_i{}^{(h_i)})^{t_i}, \quad C_i = g^{s_i} \times U_i, \quad D_i = (\Omega_i^{(h_i)})^{s_i}.$$

Si, pour un certain i, $C_i = 1$ ou $D_i = 1$, renouveler la procédure de chiffrement, sinon renvoyer le chiffré $(A_i,\ B_i,\ C_i,\ D_i)_{i=1,\cdots,\ell}$.

$\mathsf{ReRand}(\mathsf{mpk}, C)$ **:** pour rerandomiser un chiffré $C = (A_i, B_i, C_i, D_i)_{i=1,\dots,\ell} \in (\mathbb{G}^2 \times \bar{\mathbb{G}}^2)^\ell$,

1. choisir des éléments aléatoires $V_i, W_i \in \mathbb{G}$, pour $i = 1, \dots, \ell$, tels que $\prod_{i=1}^{\ell} V_i = \prod_{i=1}^{\ell} W_i = 1$;
2. choisir quatre séquences de scalaires aléatoires $r_i^{(0)}, r_i^{(1)}, u_i^{(0)}, u_i^{(1)} \in \mathbb{Z}_q$, pour $i = 1, \dots, \ell$;
3. choisir deux scalaires aléatoires $r, u \in \mathbb{Z}_q$, et calculer, pour $i = 1, \dots, \ell$, et pour $b = 0, 1$:

$$\begin{array}{rclrcl} A_i^{(b)} & \leftarrow & A_i \times C_i^r \times g^{r_i^{(b)}} \times W_i, & B_i^{(b)} & \leftarrow & B_i \times D_i^r \times (\Omega_i^{(b)})^{r_i^{(b)}} \\ C_i^{(b)} & \leftarrow & C_i^u \times g^{u_i^{(b)}} \times V_i, & D_i^{(b)} & \leftarrow & D_i^u \times (\Omega_i^{(b)})^{u_i^{(b)}}. \end{array}$$

$\mathsf{Decrypt}(\mathsf{mpk}, \mathsf{sk}_{\mathsf{ID}}, C)$ **:** pour déchiffrer un chiffré associé à un utilisateur ID, avec $\mathsf{pk}_{\mathsf{ID}} = (h_1, \dots, h_\ell)$, connaissant la clé secrète $\mathsf{sk}_{\mathsf{ID}} = (X_1, \dots, X_\ell) = (x_1^{(h_1)}, \dots, x_\ell^{(h_\ell)})$, calculer

$$\prod_{i=1}^{\ell} A_i^{(h_i)} \times (B_i^{(h_i)})^{-X_i}$$

$\mathsf{Trace}(\mathsf{msk}, \mathsf{sk}_O, C)$ **:** pour tracer un chiffré C pour une clé $\mathsf{pk}_{\mathsf{ID}} = h_1, \dots, h_\ell$, vérifier si

$$\prod_{i=1}^{\ell} C_i^{(h_i)} = \prod_{i=1}^{\ell} (D_i^{(h_i)})^{x_i^{(h_i)}}$$

Si une telle clé n'existe pas renvoyer un symbole d'erreur $\perp$; autrement, renvoyer la clé trouvée $\mathsf{pk}_{\mathsf{ID}}$ ainsi qu'une preuve non-interactive à divulgation nulle de connaissance Π, destinée à prouver la validité de l'égalité ci-dessus pour la clé publique maître $\mathsf{mpk} = (\Omega_1^{(0)}, \Omega_1^{(1)}, \dots, \Omega_\ell^{(0)}, \Omega_\ell^{(1)})$, où $\Omega_i^{(b)} = g^{1/x_i^{(b)}}$ pour $b \in \{0, 1\}$ et $i \in \{1, \dots, \ell\}$. Plus précisément, nous devons montrer l'existence de $y_1, \dots, y_\ell \in \mathbb{Z}_q$ tels que :

$$\prod_{i=1}^{\ell} C_i^{(h_i)} = \prod_{i=1}^{\ell} (D_i^{(h_i)})^{y_i} \qquad \text{and } g = (\Omega_i^{(h_i)})^{y_i} \quad \text{pour } i = 1, \dots, \ell.$$

$\mathsf{Judge}(\mathsf{mpk}, \mathcal{L}, C, \mathsf{ID}, \Pi)$: vérifier que la preuve Π est valide.

5.3.4 Analyse de sécurité

Avant de prouver la résistance aux attaques subliminales, nous montrons d'abord que toutes les équations sont bien vérifiées si les utilisateurs sont honnêtes.

Correction

Tout d'abord, remarquons que durant la phase de chiffrement, K_i et U_i vérifient : $\prod K_i = m$ et $\prod U_i = 1$. De manière similaire, durant la phase de rerandomization, les V_i et W_i sont tels que $\prod V_i = 1$ et $\prod W_i = 1$. Par conséquent, après la rerandomization, un chiffré a le format suivant : pour $i = 0, \cdots, \ell$ et pour $b = 0, 1$, où les s_i et les t_i sont des valeurs aléatoires choisies par l'émetteur, alors que r, u, $r_i^{(b)}$ et $u_i^{(b)}$ sont choisis par le médiateur, chargé de rerandomiser :

$$\begin{array}{rclrcl} A_i^{(b)} & \leftarrow & g^{t_i+rs_i} \times g^{r_i^{(b)}} \times K_i, & B_i^{(b)} & \leftarrow & (\Omega_i{}^{(h_i)})^{t_i+rs_i} \times (\Omega_i^{(b)})^{r_i^{(b)}}, \\ C_i^{(b)} & \leftarrow & g^{us_i} \times g^{u_i^{(b)}} \times U_i, & D_i^{(b)} & \leftarrow & (\Omega_i^{(h_i)})^{us_i} \times (\Omega_i^{(b)})^{u_i^{(b)}}. \end{array}$$

Par conséquent, en utilisant la clé secrète $\mathsf{sk}_{\mathsf{ID}} = (X_1, \ldots, X_\ell) = (x_1^{(h_1)}, \ldots, x_\ell^{(h_\ell)})$, puisque nous avons par définition
$\Omega_i^{(b)} = g^{1/x_i^{(b)}}$, pour tout i et b, alors $(\Omega_i^{(h_i)})^{x_i^{(h_i)}} = g$ pour tout i, et donc

$$\begin{aligned} \prod_{i=1}^{\ell} A_i^{(h_i)} \times (B_i^{(h_i)})^{-X_i} &= \prod_{i=1}^{\ell} g^{t_i+rs_i} \times g^{r_i^{(h_i)}} \times K_i \times \left((\Omega_i{}^{(h_i)})^{t_i+rs_i} \times (\Omega_i^{(h_i)})^{r_i^{(h_i)}} \right)^{-x_i^{(h_i)}} \\ &= \prod_{i=1}^{\ell} g^{t_i+rs_i+r_i^{(h_i)}} \times K_i \times \left(g^{t_i+rs_i+r_i^{(h_i)}} \right)^{-1} \\ &= \prod_{i=1}^{\ell} K_i = m. \end{aligned}$$

Concernant la procédure de traçage,

$$\begin{aligned} \prod_{i=1}^{\ell} C_i^{(k_i)} \times (D_i^{(k_i)})^{-x_i^{(k_i)}} &= \prod_{i=1}^{\ell} g^{us_i} \times g^{u_i^{(k_i)}} \times U_i \times \left((\Omega_i^{(h_i)})^{us_i} \times (\Omega_i^{(k_i)})^{u_i^{(k_i)}} \right)^{-x_i^{(k_i)}} \\ &= \prod_{i=1}^{\ell} g^{us_i+u_i^{(k_i)}} \times \left((\Omega_i^{(h_i)}/\Omega_i^{(k_i)})^{us_i} \times (\Omega_i^{(k_i)})^{us_i+u_i^{(k_i)}} \right)^{-x_i^{(k_i)}} \\ &= \prod_{i=1}^{\ell} g^{us_i+u_i^{(k_i)}} \times g^{-us_i \times (x_i^{(k_i)}/x_i^{(h_i)}-1)} \times g^{-us_i-u_i^{(k_i)}} \\ &= \prod_{i=1}^{\ell} g^{-us_i \times (x_i^{(k_i)}/x_i^{(h_i)}-1)} = \left(\prod g^{s_i \times (1-x_i^{(k_i)}/x_i^{(h_i)})} \right)^u \end{aligned}$$

Il est évident que si pour chaque index i, nous avons $k_i = h_i$, le produit est égal à 1. Cependant, nous remarquons que si toutes les valeurs secrètes $x_i^{(b)}$ sont connues, il est possible de construire une séquence s_i telle que le produit précédent est égal à 1, même pour $(k_i) \neq (h_i)$. Cependant, pour notre modèle de sécurité, seul le manager du groupe connaît $x_i^{(b)}$, pour un même i mais pour $b = 0, 1$. Par conséquent, un utilisateur, même

malhonnête (excepté l'autorité) ne peut pas construire un chiffré qui trace deux utilisateurs simultanément.

Sécurité Sémantique

Pour les preuves qui suivent, on notons pour tout bit b, $\bar{b}$ le bit complémentaire.

Nous considérons une variante du problème $\mathsf{DDH}_{\mathbb{G}}$; nous définissons un nouveau problème, 2-$\mathsf{DDH}_{\mathbb{G}}$ de la manière suivante : étant donné un t-uplet $(X = g^x, Y = g^y, Z = g^z, \bar{Y} = g^{\bar{y}}, \bar{Z} = g^{\bar{z}})$, décider si les deux égalités $z = xy$ et $\bar{z} = x\bar{y}$ sont simultanément vérifiées. Si c'est la cas, nous dirons que $(X = g^x, Y = g^y, Z = g^z, \bar{Y} = g^{\bar{y}}, \bar{Z} = g^{\bar{z}})$ est un t-uplet 2-DDH. De plus, si nous supposons l'hypothèse DDH valide, alors nous pouvons prouver la validité de l'hypothèse 2-$\mathsf{DDH}_{\mathbb{G}}$ par une preuve hybride, avec une perte d'un facteur 2. Pour la sécurité sémantique, nous avons le résultat suivant, similaire à celui obtenu pour le schéma ElGamal :

Théorème 9 *Le schéma* $\mathcal{M}$ATES *est sémantiquement sûr* au sens faible *(voir paragraphe 5.3.2) résistant aux attaques à clairs choisis sous l'hypothèse* DDH *dans* $\mathbb{G}$. *Si* ℓ *est la longueur des identités, on a pour tout attaquant polynomial* $\mathcal{A}$ *:*

$$\mathsf{Adv}^{\mathsf{weak-ind}}_{\mathcal{M}\text{ATES},\mathcal{A}}(\lambda) \leqslant 2 \times \ell \times \mathsf{Adv}^{\mathsf{2\text{-}dh}}_{\mathbb{G}}(\lambda),$$

où λ *est le paramètre de sécurité.*

Démonstration: soit $\mathcal{A}$ un adversaire contre la sécurité sémantique du schéma $\mathcal{M}$ATES. Nous allons construire un algorithme $\mathcal{B}$, qui a accès à $\mathcal{A}$ dans le but de casser le problème 2-DDH. Notre algorithme $\mathcal{B}$ reçoit une instance 2-DDH $(g, X = g^x, Y = g^y, Z = g^z, \bar{Y} = g^{\bar{y}}, \bar{Z} = g^{\bar{z}})$. Nous simulons l'algorithme $\mathsf{GSetup}()$ en prenant g pour générateur du groupe cyclique $\mathbb{G}$. Nous choisissons 2ℓ scalaires dans $\mathbb{Z}_q$, $x_i^{(b)}$ pour $i = 0, \ldots, \ell$ et $b = 0, 1$. Nous choisissons une position $\gamma \in \{1, \ldots, \ell\}$ et un bit α. Nous notons formellement $x_\gamma^{(\alpha)} = 1/x$. On définit la clé publique maître par :

$$\begin{aligned} \Omega_i^{(b)} &= g^{1/x_i^{(b)}} \text{ pour } b = 0, 1 \text{ et } i = 1, \ldots, \ell \\ \Omega_\gamma^{(\alpha)} &= X \end{aligned}$$

L'adversaire demande plusieurs clés publiques et une seule clé secrète (supposée être la sienne) *via* l'oracle $\mathsf{Join}^*()$. Nous choisissons une clé publique aléatoire $\mathsf{pk}_{\mathsf{ID}} = (k_1, \ldots, k_\ell)$ dans le code et renvoyons la clé secrète associée. Nous sommes capables de répondre correctement avec probabilité $\frac{1}{2}$, puisqu'avec la même probabilité, la simulation ne requiert pas la connaissance de x.
L'attaquant, $\mathcal{A}$ retourne deux messages m_0, m_1 et la clé publique d'un utilisateur $\mathsf{pk} = h_1 \ldots h_\ell$. Avec probabilité négligeable $\mathsf{pk} \in \mathcal{L}$, autrement l'avantage de l'attaquant contre la sécurité sémantique est négligeable.
Par construction, on sait que $\mathsf{pk}_{\mathsf{ID}}$ et pk diffèrent en au moins deux positions : avec probabilité $1/l$, elles diffèrent en la position γ. Si c'est la cas, la clé pk contient l'élément X de l'instance que l'on souhaite résoudre. Pour simplifier, supposons $\gamma = 1$.
On choisit un bit β aléatoire. Afin de calculer le chiffré challenge d'un message m_β, $C = (A_i, B_i, C_i, D_i)_{i=1,\ldots,\ell}$, nous choisissons une séquence aléatoire d'éléments $U_i \xleftarrow{R} \mathbb{G}$ tels que $\prod_{i=1}^{\ell} U_i = 1$, et une séquence d'éléments aléatoires K_i, telle que $m_\beta = \prod_{i=1}^{\ell} K_i$; on choisit $t_i, s_i \in \mathbb{Z}_q$ puis on applique successivement les procédures suivantes :

1. nous choisissons des éléments aléatoires $K_i, U_i \in \mathbb{G}$ pour $i = 1, \ldots, \ell$, tels que
$$\prod_{i=1}^{\ell} K_i = m \qquad \prod_{i=1}^{\ell} U_i = 1$$
2. pour $i = 2, \ldots, \ell$, nous choisissons deux séquences de scalaires aléatoires $t_i, s_i \in \mathbb{Z}_q$;
3. puis on calcule :

$$\begin{array}{rclcrcl} A_1 & = & Y \times K_1 & & B_1 & = & Z \\ A_i & = & g^{t_i} \times K_i & & B_i & = & (\Omega_i^{(h_i)})^{t_i} \text{ pour } i = 2, \ldots, \ell \\ \\ C_1 & = & \bar{Y} \times U_1 & & D_1 & = & \bar{Z} \\ C_i & = & g^{s_i} \times U_i & & D_i & = & (\Omega_i^{(h_i)})^{s_i} \text{ pour } i = 2, \ldots, \ell \end{array}$$

L'attaquant $\mathcal{A}$ retourne alors sa réponse β' pour le bit β, et l'algorithme retourne le bit $(\beta = \beta')$;
– si $(X, Y, Z, \bar{Y}, \bar{Z})$ est réellement un quadruplet 2-DDH, alors le chiffré challenge est défini comme par la construction. Dans ce cas, $\beta = \beta'$ avec un biais significatif si $\mathcal{A}$ parvient à casser la sécurité sémantique ;
– si $(X, Y, Z, \bar{Y}, \bar{Z})$ est un quadruplet aléatoire : puisque le chiffré challenge ne contient aucune information sur le bit β, $\beta = \beta'$ avec probabilité exactement $\frac{1}{2}$.

□

Anonymat

L'anonymat de notre schéma repose sur l'hypothèse DLIN. Nous avons obtenu le résultat suivant :

Théorème 10 *Le schéma* $\mathcal{M}$ATES *est anonyme résistant aux attaques à clairs choisis sous l'hypothèse* $\mathsf{DLIN}_{\mathbb{G}}$*, si les clés publiques sont choisies dans un code de distance minimale* 2*, et on a pour tout attaquant polynomial* $\mathcal{A}$ *:*

$$\mathsf{Adv}^{\mathsf{anon}}_{\mathcal{M}\text{ATES},\mathcal{A}}(\lambda) \leqslant 8\ell^2 \times \mathsf{Adv}^{\mathsf{dlin}}_{\mathbb{G}}(\lambda),$$

où λ *est le paramètre de sécurité.*

Démonstration: soit $\mathcal{A}$ un attaquant contre l'anonymat du schéma $\mathcal{M}$ATES. Nous allons construire un algorithme $\mathcal{B}$ qui résout le problème $\mathsf{DLIN}_{\mathbb{G}}(g, u, v)$ en utilisant l'algorithme $\mathcal{A}$. Considérons une instance de ce problème (u, v, g, U, V, Z). Nous notons formellement (les scalaires ne sont pas connus du simulateur) : $u = g^{\mu}$, $v = g^{\nu}$, $U = u^a$, $V = v^b$ et $Z = g^c$.
Nous simulons l'algorithme GSetup en utilisant g comme générateur du groupe $\mathbb{G}$. Nous choisissons 2ℓ scalaires aléatoires $x_i^{(b)} \in \mathbb{Z}_q$, pour $i = 1, \ldots, \ell$ et $b = 0, 1$. Nous choisissons également deux indices aléatoires $\gamma, \delta \in \{1, \ldots, \ell\}$ ainsi que des bits aléatoires $\alpha, \beta \in \{0, 1\}$. On note formellement $x_\gamma^{(\alpha)} = 1/\mu$ et $x_\delta^{(\beta)} = 1/\nu$ (qui ne sont donc pas connus) :

$$\begin{array}{rcl} \Omega_i^{(b)} & = & g^{1/x_i^{(b)}}, \quad \text{pour } i = 1, \ldots, \ell \text{ et } b = 0, 1 \\ \Omega_\gamma^{(\alpha)} & = & u \\ \Omega_\delta^{(\beta)} & = & v \end{array}$$

L'adversaire demande plusieurs clés publiques et une seule clé secrète (supposée être la sienne) *via* l'oracle $\mathsf{Join}^*()$. Nous pouvons simuler cet oracle avec probabilité $\frac{1}{4}$, puisqu'elle ne requiert pas la connaissance de μ et ν une fois sur quatre. Si ce n'est pas le cas, nous abandonnons la simulation. Alors $\mathcal{A}$ renvoie un message m et deux clés publiques dans $\mathcal{L}$: $\mathsf{pk}_0 = (h_1^0, \ldots, h_\ell^0)$ et $\mathsf{pk}_1 = (h_1^1, \ldots, h_\ell^1)$.
À présent, nous choisissons un bit B. Comme les identités sont des mots appartenant à un code de distance minimale 2, nous savons que pk_B et $\mathsf{pk}_{\mathsf{ID}}$ diffèrent en au moins deux positions et avec probabilité $1/\ell^2$, elles diffèrent aux positions γ et δ. $h_\gamma^B \neq h_\gamma^{\mathsf{ID}}$ et $h_\delta^B \neq h_\delta^{\mathsf{ID}}$, et dans ce cas, la clé pk_B contient nécessairement la base (u, v) : $h_\gamma^B = \alpha$ et $h_\delta^B = \beta$ (sinon $\mathsf{pk}_{\mathsf{ID}}$ contient l'une d'entre elles, ce qui signifie alors que la clé secrète requiert la connaissance de μ ou ν). Sans perte de généralité, nous pouvons supposer que $\gamma = 1$ et $\delta = 2$.
Nous calculons le chiffré challenge de m associé à la clé publique pk_B de la manière suivante :

– tout d'abord, nous avons besoin de deux instances $\mathsf{DLIN}_{\mathbb{G}}$ en base (g, u, v). Nous avons déjà $(U = u^a, V = v^b, Z = g^c)$. Nous pouvons utiliser cette instance pour en dériver une seconde : $\bar{U} = U^z u^x$, $\bar{V} = V^z v^y$ et $\bar{Z} = Z^z g^{x+y}$ pour des éléments x, y, z aléatoires dans $\mathbb{Z}_p$.
Nous remarquons alors que :
$(\bar{U} = u^{\bar{a}}, \bar{V} = v^{\bar{b}}, \bar{Z} = g^{\bar{c}})$ est un triplet linéaire (en base u, v, g) si et seulement (U, V, Z) est un triplet linéaire (en base u, v, g).
– nous calculons alors $C = (A_i, B_i, C_i, D_i)_{i=1,\ldots,\ell}$ en appliquant successivement les procédures suivantes :

1. nous choisissons des éléments aléatoires $K_i, U_i \in \mathbb{G}$ pour $i = 1, \ldots, \ell$, et $W, W', \bar{W}, \bar{W}' \in \mathbb{G}$, tels que
$$\prod_{i=1}^{\ell} K_i = m \qquad \prod_{i=1}^{\ell} U_i = 1 \qquad W \times W' = Z \qquad \bar{W} \times \bar{W}' = \bar{Z};$$
2. pour $i = 1, \ldots, \ell$, nous choisissons deux séquences de scalaires aléatoires $t_i, s_i \in \mathbb{Z}_q$;
3. puis on calcule :
$$\begin{array}{rclcrcl} A_1 &=& W \times K_1 & & B_1 &=& U \\ A_2 &=& W' \times K_2 & & B_2 &=& V \\ A_i &=& g^{t_i} \times K_i & & B_i &=& (\Omega_i^{(h_i^B)})^{t_i} \end{array}$$
$$\begin{array}{rclcrcl} C_1 &=& \bar{W} \times U_1 & & D_1 &=& \bar{U} \\ C_2 &=& \bar{W}' \times U_2 & & D_2 &=& \bar{V} \\ C_i &=& g^{s_i} \times U_i & & D_i &=& (\Omega_i^{(h_i^B)})^{s_i} \text{ pour } i = 3, \ldots, \ell \end{array}$$

En posant $W = g^w$, $W' = g^{w'}$, $\bar{W} = g^{\bar{w}}$ et $\bar{W}' = g^{\bar{w}'}$, nous obtenons alors $w + w' = c$ et $\bar{w} + \bar{w}' = \bar{c}$, et

$$\begin{array}{rclcrcl} A_1 &=& W \times K_1 = g^w \times K_1 = g^a \times (g^{w-a} \times K_1) & & B_1 &=& U = u^a = (\Omega_1^{(h_1^B)})^a \\ C_1 &=& \bar{W} \times U_1 = g^{\bar{w}} \times U_1 = g^{\bar{a}} \times (g^{\bar{w}-\bar{a}} \times U_1) & & D_1 &=& \bar{U} = u^{\bar{a}} = (\Omega_1^{(h_1^B)})^{\bar{a}} \\ A_2 &=& W' \times K_2 = g^{w'} \times K_2 = g^b \times (g^{w'-b} \times K_2) & & B_2 &=& V = v^b = (\Omega_2^{(h_2^B)})^b \\ C_2 &=& \bar{W}' \times U_2 = g^{\bar{w}'} \times U_2 = g^{\bar{b}} \times (g^{\bar{w}'-\bar{b}} \times U_2) & & D_2 &=& \bar{V} = v^{\bar{b}} = (\Omega_2^{(h_2^B)})^{\bar{b}} \end{array}$$

Nous choisissons une clé publique aléatoire $(k_1, \ldots, k_\ell)$ dans le code. Nous définissons la clé publique maître par $\Omega_i^{(\bar{k}_i)} = X_i$ et $\Omega_i^{(k_i)} = g^{1/x_i^{(k_i)}}$, pour un scalaire $x_i^{(k_i)}$ aléatoire dans $\mathbb{Z}_q$, pour $i = 1, \ldots, \ell$: et donc, pour tout i, $x_i^{(k_i)}$ est connu, tandis que $x_i^{(\bar{k}_i)} = 1/x_i$ n'est pas connu.
Lorsque l'attaquant demande à être enregistré en tant que membre du groupe, on lui renvoie la clé publique $(k_1, \ldots, k_\ell)$ et la clé secrète $(x_1^{(k_1)}, \ldots, x_\ell^{(k_\ell)})$. Lorsque $\mathcal{A}$ renvoie un chiffré $C = (A_i = g^{a_i}, B_i = g^{b_i}, C_i = g^{c_i}, D_i = g^{d_i})_{i=1,\ldots,\ell}$,
En posant

$$\begin{array}{rclcrcll} K'_1 & = & g^{w-a} \times K_1 & & U'_1 & = & g^{\bar{w}-\bar{a}} \times U_1 & \\ K'_2 & = & g^{w'-b} \times K_2 & & U'_2 & = & g^{\bar{w}'-\bar{b}} \times U_2 & \\ K'_i & = & K_i & & U'_i & = & U_i & \text{pour } i = 3, \ldots, \ell \end{array}$$

nous obtenons

$$\begin{array}{rclcrclcrcl} A_1 & = & g^a \times K'_1 & & B_1 & = & (\Omega_1^{(h_1^B)})^a & & C_1 & = & g^{\bar{a}} \times U'_1 & & D_1 & = & (\Omega_1^{(h_1^B)})^{\bar{a}} \\ A_2 & = & g^b \times K'_2 & & B_2 & = & (\Omega_2^{(h_2^B)})^b & & C_2 & = & g^{\bar{b}} \times U'_2 & & D_2 & = & (\Omega_2^{(h_2^B)})^{\bar{b}} \\ A_i & = & g^{t_i} \times K'_i & & B_i & = & (\Omega_i^{(h_i^B)})^{t_i} & & C_i & = & g^{s_i} \times U'_i & & D_i & = & (\Omega_i^{(h_i^B)})^{s_i}, \end{array}$$

pour $i = 3, \ldots, \ell$ et avec :

$$\prod_{i=1}^{\ell} K'_i = g^{w-a} \times g^{w'-b} \times \prod_{i=1}^{\ell} K_i = g^{c-a-b} \times m \quad \text{et} \quad \prod_{i=1}^{\ell} U'_i = g^{\bar{w}-\bar{a}} \times g^{\bar{w}'-\bar{b}} \times \prod_{i=1}^{\ell} U_i = g^{\bar{c}-\bar{a}-\bar{b}}.$$

Enfin, $\mathcal{A}$ retourne sa réponse B' pour B, et $\mathcal{B}$ renvoie le booléen $(B' = B)$:
– si $\mathcal{B}$ reçoit un triplet linéaire valide, alors les deux égalités suivantes sont vérifiées : $c = a + b$ et $\bar{c} = \bar{a} + \bar{b}$: $\prod_{i=1}^{\ell} K'_i = m$ et $\prod_{i=1}^{\ell} U'_i = 1$. Dans ce cas, $\mathcal{A}$ reçoit vraiment le chiffré de m associé à la clé pk_B, comme dans la construction. $\mathcal{B}$ peut gagner avec probabilité non négligeable si $\mathcal{A}$ devine correctement le bit B ;
– si $\mathcal{B}$ reçoit un triplet aléatoire, alors $\Delta = c - a - b \neq 0$, et $\bar{c} - \bar{a} - \bar{b} = (zc + x + y - za - x - zb - y) = z\Delta$: ce qui conduit à un chiffré aléatoire et indépendant du bit B. Et dans ce cas, $B = B'$ avec probabilité $\frac{1}{2}$.

□

Absence de canaux subliminaux

Cette nouvelle propriété repose sur l'hypothèse DDH. Nous présentons ci-dessous la simulation pour la preuve du schéma $\mathcal{M}$ATES :

Théorème 11 *Le schéma* $\mathcal{M}$ATES *est sans canaux subliminaux résistant aux attaques à clairs choisis si l'hypothèse* $\mathsf{DDH}_{\mathbb{G}}$ *est valide. Plus précisément, si* ℓ *est la longueur des identités, on a pour tout attaquant polynomial* $\mathcal{A}$ *:*

$$\mathsf{Adv}^{\mathsf{subF}}_{\mathcal{M}\text{ATES},\mathcal{A}}(\lambda) \leqslant 4 \times \mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(\lambda),$$

où λ *est le paramètre de sécurité.*

Démonstration: considérons un attaquant possédant sa propre clé secrète sk au moyen de laquelle il tente de transmettre de l'information. Plus exactement, son objectif est de créer un chiffré qui est associé à une autre clé que la sienne (notée pk). Nous allons montrer que si l'hypothèse $\mathsf{DDH}_{\mathbb{G}}$ est valide, alors tout chiffré "rerandomisé" qui n'est pas associé à la clé de l'utilisateur est indistinguable d'un chiffré parfaitement aléatoire. Ainsi, nous aurons le résultat que nous attendons : la procédure de rerandomization de deux chiffrés non associés à la clé pk permet d'obtenir des chiffrés indistinguables.
Comme précédemment, étant donnée une instance 2-DDH $(X = g^x, Y = g^y, Z = g^z, \bar{Y} = g^{\bar{y}}, \bar{Z} = g^{\bar{z}})$, il est possible de dériver ℓ autres instances similaires :

$$(X_i = g^{x_i}, Y_i = g^{y_i}, Z_i = g^{z_i}, \bar{Y}_i = g^{\bar{y}_i}, \bar{Z}_i = g^{\bar{z}_i}) \text{ pour } i = 1, \ldots, \ell.$$

Nous choisissons une clé publique aléatoire $(k_1, \ldots, k_\ell)$ dans le code. Nous définissons la clé publique maître par $\Omega_i^{(\bar{k}_i)} = X_i$ et $\Omega_i^{(k_i)} = g^{1/x_i^{(k_i)}}$, pour un scalaire $x_i^{(k_i)}$ aléatoire dans $\mathbb{Z}_q$, pour $i = 1, \ldots, \ell$: et donc, pour tout i, $x_i^{(k_i)}$ est connu, tandis que $x_i^{(\bar{k}_i)} = 1/x_i$ n'est pas connu.
Lorsque l'attaquant demande à être enregistré en tant que membre du groupe, on lui renvoie la clé publique $(k_1, \ldots, k_\ell)$ et la clé secrète $(x_1^{(k_1)}, \ldots, x_\ell^{(k_\ell)})$. Lorsque $\mathcal{A}$ renvoie un chiffré $C = (A_i = g^{a_i}, B_i = g^{b_i}, C_i = g^{c_i}, D_i = g^{d_i})_{i=1,\ldots,\ell}$,
Nous appliquons la procédure de rerandomization définie comme suit : pour $i = 1, \ldots, \ell$:

$$\begin{array}{rclcrcl} A_i^{(\bar{k}_i)} & = & A_i \times C_i^r \times Y_i \times W_i & & B_i^{(\bar{k}_i)} & = & B_i \times D_i^r \times Z_i \\ A_i^{(k_i)} & = & A_i \times C_i^r \times R_i \times W_i & & B_i^{(k_i)} & = & B_i \times D_i^r \times S_i \end{array}$$

$$\begin{array}{rclcrcl} C_i^{(\bar{k}_i)} & = & C_i^u \times \bar{Y}_i \times V_i & & D_i^{(\bar{k}_i)} & = & D_i^u \times \bar{Z}_i \\ C_i^{(k_i)} & = & C_i^u \times \bar{R}_i \times V_i & & D_i^{(k_i)} & = & D_i^u \times \bar{S}_i \end{array}$$

avec les aléas $V_i = g^{v_i}$ et $W_i = g^{w_i}$ et tels que $\prod V_i = 1$ et $\prod W_i = 1$, et les aléas $R_i = g^{r_i}$, $\bar{R}_i = g^{\bar{r}_i}$, avec $r, u \in \mathbb{Z}_q$; en revanche les valeurs $S_i = g^{s_i}$ et $\bar{S}_i = g^{\bar{s}'_i}$ suivent les distributions respectives suivantes :

- Jeu 1 : $S_i = g^{s_i} = R_i^{1/x_i^{(k_i)}}$, $\bar{S}_i = g^{\bar{s}'_i} = \bar{R}_i^{1/x_i^{(k_i)}}$, et donc $s_i = r_i/x_i^{(k_i)}$ et $\bar{s}_i = \bar{r}_i/x_i^{(k_i)}$. De plus, nous supposons que $(X, Y, Z, \bar{Y}, \bar{Z})$ est réellement un t-uplet 2-DDH : le chiffré rerandomisé est donc défini exactement comme pour la construction.
- Jeu 1* : comme pour le jeu précédent, $S_i = g^{s_i} = R_i^{1/x_i^{(k_i)}}$, $\bar{S}_i = g^{\bar{s}'_i} = \bar{R}_i^{1/x_i^{(k_i)}}$. Mais à présent, nous supposons que $(X, Y, Z, \bar{Y}, \bar{Z})$ est un t-uplet aléatoire. Si l'hypothèse DDH est valide, cette nouvelle distribution du chiffré rerandomisé est indistinguable de la distribution définie par la construction.
  Remarquons que dans ce jeu, le chiffré rerandomisé a la forme suivante :

$$\begin{array}{rclcrcl} A_i^{(\bar{k}_i)} & = & g^{y_i} & B_i^{(\bar{k}_i)} & = & g^{z_i} & C_i^{(\bar{k}_i)} = g^{\bar{y}_i} \quad D_i^{(\bar{k}_i)} = g^{\bar{z}_i} \\ A_i^{(k_i)} & = & g^{a_i + rc_i + r_i + w_i} & B_i^{(k_i)} & = & g^{b_i + rd_i + s_i} & C_i^{(k_i)} = g^{uc_i + \bar{r}_i + v_i} \quad D_i^{(k_i)} = g^{ud_i + \bar{s}_i}, \end{array}$$

  avec les aléas $V_i = g^{v_i}$ et $W_i = g^{w_i}$, tels que $\prod V_i = 1$ et $\prod W_i = 1$, et les scalaires $y_i, \bar{y}_i, z_i, \bar{z}_i, r_i, \bar{r}_i \xleftarrow{R} \mathbb{Z}_q$, et avec $r, u \in \mathbb{Z}_q$, en revanche, on a $s_i = r_i/x_i^{(k_i)}$ et $\bar{s}_i = \bar{r}_i/x_i^{(k_i)}$.
- Jeu 2 : ce jeu est défini exactement comme le jeu précédent, excepté que $(X, Y, Z, \bar{Y}, \bar{Z})$ est un t-uplet aléatoire et à présent, $r_i, \bar{r}_i, s_i, \bar{s}_i$ sont choisis aléatoirement dans $\mathbb{Z}_q$.

Montrons que le chiffré est de la même forme que le chiffré du jeu 1*. Le chiffré n'étant pas associé à la clé de l'adversaire, on a : $\sum(c_i - d_i \times x_i^{(k_i)}) \neq 0$. Posons

$$\delta = \frac{\sum(r_i - s_i \times x_i^{(k_i)})}{\sum(c_i - d_i \times x_i^{(k_i)})} \qquad \bar{\delta} = \frac{\sum(\bar{r}_i - \bar{s}_i \times x_i^{(k_i)})}{\sum(c_i - d_i \times x_i^{(k_i)})} \qquad r' = r + \delta \qquad u' = u + \bar{\delta}$$

et pour $i = 1, \ldots, \ell$,

$$\begin{array}{rclcrclcrcl} r'_i & = & r_i - \delta d_i & & s'_i & = & s_i - \delta d_i & & w'_i & = & w_i + \delta(d_i \times x_i^{(k_i)} - c_i) - (s_i \times x_i^{(k_i)} - r_i) \\ \bar{r}'_i & = & \bar{r}_i - \bar{\delta} d_i & & \bar{s}'_i & = & \bar{s}_i - \bar{\delta} d_i & & v'_i & = & v_i + \bar{\delta}(d_i \times x_i^{(k_i)} - c_i) - (\bar{s}_i \times x_i^{(k_i)} - \bar{r}_i) \end{array}$$

Alors,

$$\begin{array}{rcl} a_i + rc_i + r_i + w_i & = & a_i + r'c_i - \delta c_i + r_i + w'_i - \delta(d_i \times x_i^{(k_i)} - c_i) + (s_i \times x_i^{(k_i)} - r_i) \\ & = & a_i + r'c_i + s'_i \times x_i^{(k_i)} + w'_i \\ b_i + rd_i + s_i & = & b_i + r'd_i - \delta d_i + s_i = b_i + r'd_i + s'_i \\ uc_i + \bar{r}_i + v_i & = & u'c_i - \bar{\delta} c_i + \bar{r}_i + v'_i - \bar{\delta}(d_i \times x_i^{(k_i)} - c_i) + (\bar{s}_i \times x_i^{(k_i)} - \bar{r}_i) \\ & = & u'c_i + v'_i + \bar{s}'_i \times x_i^{(k_i)} \\ ud_i + \bar{s}_i & = & u'd_i - \bar{\delta} d_i + \bar{s}_i = u' + d_i \bar{s}'_i, \end{array}$$

ce qui permet de conclure la preuve d'absence de canal subliminal pour notre schéma. □

Extension

On remarque que la description précédente n'explicite pas comment les utilisateurs obtiennent leur clés publique et secrète. Afin de garantir la confidentialité des messages *vis à vis* du manager, le schéma $\mathcal{M}$ATES est utilisé comme simple sur-couche de chiffrement. Supposons que l'on veuille envoyer un message à un participant, on chiffre d'abord le message avec un schéma dont la fonction de chiffrement est à valeurs dans le groupe spécifié par le schéma, puis la fonction du schéma $\mathcal{M}$ATES est appliquée composante par composante sur le chiffré obtenu. Avec notre schéma, nous devons utiliser le schéma de chiffrement ElGamal qui produit deux composantes dans le groupe $\mathbb{G}$.

5.3.5 Conclusion

Nous avons proposé une nouvelle primitive avec un modèle permettant une modélisation de la consistance. Notre modèle n'exige pas d'interaction entre les participants. Nous avons exhibé un schéma efficace, sans couplage vérifiant les propriétés de sécurité sémantique, d'anonymat et de résistance aux attaques subliminales dans le modèle standard. Il reste encore plusieurs pistes à explorer : nous avons déjà envisagé de séparer les rôles du manager et de l'autorité d'ouverture en munissant le groupe d'une structure bilinéaire, mais nous ne pouvons garantir la sécurité du schéma dans le modèle standard. De plus, il serait également intéressant de rendre le schéma résistant aux coalitions, au moins de manière partielle ; une des pistes possibles est de sélectionner un code avec des propriétés bien spécifiques aux garanties voulant être obtenues, ces garanties restent malheureusement trop fortes pour certifier l'existence du code requis. Ce point reste encore imprécis.

Troisième partie

Anonymat et Authentification

Chapitre 6

Échange de clés par mot de passe

Les protocoles d'échange de clés ont pour but de permettre à deux ou plusieurs participants d'établir une clé de session commune utilisée pour échanger des messages de manière sécurisée et authentifiée.

Dans ce scénario, un attaquant a le contrôle total du canal et ses stratégies d'attaques sont multiples : il peut interagir avec le protocole de manière à gagner de l'information en falsifiant les messages, en les rejouant, en les supprimant, en interceptant certains d'entre eux au nom d'un autre participant ou en les envoyant à un autre destinataire que celui prévu initialement. Ces protocoles introduisent de nouvelles notions et la sécurité de nombreux protocoles proposés par le passé reposait sur des arguments heuristiques à défaut de modèle formel. De manière à unifier un modèle caractérisant les pouvoirs de l'attaquant dans ce scénario, Bellare *et al.* [17, 14] ont introduit un modèle de sécurité dans le cas à deux et trois parties. Ils modélisent les propriétés de confidentialité et d'intégrité des messages envoyés ainsi que la propriété d'authentification mutuelle.

Puisque les stratégies de l'attaquant sont multiples, afin de pouvoir communiquer de manière sécurisée, les deux parties doivent posséder de l'information secrète. Nous considérons le scénario à base de mot de passe pour les protocoles à deux ou trois parties, où un client cherche à obtenir une clé authentifiée de session. Dans ce chapitre, nous commençons par rappeler les définitions relatives au protocole d'échange de clés à base de mot de passe à deux parties, 2-PAKE. Nous introduisons le modèle de communication et les notions de sécurité pour ces protocoles. C'est dans ce modèle que nous effectuerons la preuve de notre protocole générique que nous présenterons au chapitre 8. Ce premier chapitre, consacré aux protocoles 2-PAKE nous permettra également de dériver de manière naturelle le modèle de sécurité pour le scénario à trois parties, plus délicat, que nous considérons au chapitre suivant.

Sommaire

6.1 Introduction

6.1.1 Les premiers pas

L'article incontournable et fondateur du concept d'échange de clés est l'article de Diffie-Hellman. Il propose une méthode permettant à deux utilisateurs de se mettre d'accord sur une clé de session pour échanger des messages de manière sécurisée. Par la suite, cette méthode a été très influente pour la construction de nombreux protocoles d'échange de clés. Le principe est le suivant : les deux participants U_1, U_2 choisissent chacun au hasard $x, y \in \mathbb{Z}_q$ respectivement. Chacun peut alors calculer le secret Diffie-Hellman $g^{x \cdot y}$ en recevant le message de l'autre partenaire. L'un des inconvénients majeurs de ce protocole est qu'il ne permet pas aux joueurs de savoir par qui a été envoyé la valeur reçue. Le protocole est donc vulnérable aux attaques par le milieu. Il existe différentes méthodes d'authentification ; soit des méthodes asymétriques, par lesquelles les joueurs obtiennent une paire clé publique/clé privée certifiées ; dans ce cas, on suppose l'existence d'une infrastructure de clés publiques (PKI), ou soit des méthodes symétriques, pour lesquelles les joueurs partagent une clé secrète aléatoire de haute entropie. Une troisième possibilité est d'envisager que chaque paire d'utilisateurs partage un secret de faible entropie, un mot de passe : c'est ce scénario que nous considérons dans cette thèse.

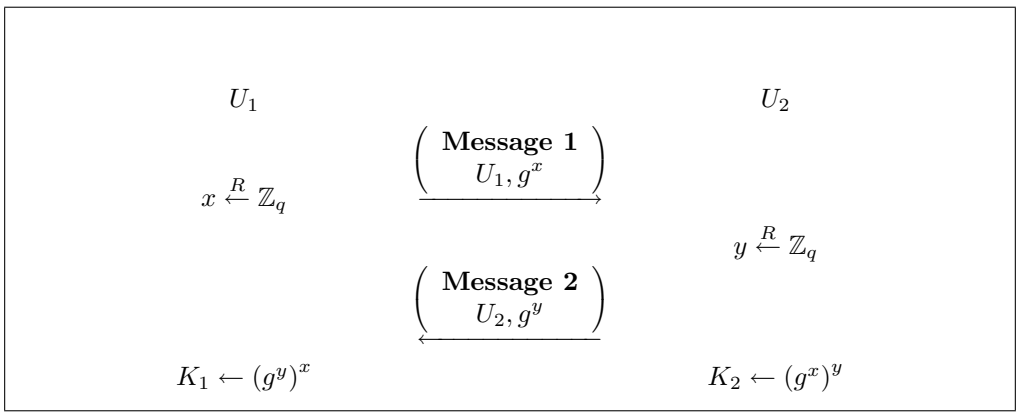


FIG. 6.1 – Protocole Diffie-Hellman. $\mathbb{G} = \langle g \rangle$ est un groupe cyclique d'ordre q.

6.1.2 Travaux antérieurs

Les variantes du protocole Diffie-Hellman avec authentification sont très nombreuses ; Bellovin et Merritt [22] ont été les premiers à considérer formellement ce concept. Leur protocole, EKE a été prouvé dans des modèles idéaux (*ideal cipher model* ou *random oracle model*) [16] ; nous reviendrons sur les variantes de ce protocole au prochain paragraphe. Cet article a été très influent par la suite dans la conception d'autres protocoles. Halevi et Krawczyk [80] ont construit un protocole prouvé sûr dans le modèle standard mais dans un contexte asymétrique qui suppose l'existence d'une PKI. Seul le protocole de Goldreich et Lindell [72] dans le modèle standard est sans hypothèses d'initialisation additionnelles ; leur construction suppose l'existence de permutations à trappe. Notons que même si leur protocole est inefficace,il permet d'établir une preuve de faisabilité. Boyko *et al.* [38] ont proposé un protocole à base de mot de passe efficace et prouvé sûr dans le modèle de l'oracle aléatoire. Bellare, Pointcheval et Rogaway [14] ont également considéré les protocoles 2-PAKE, pour lesquels ils ont proposé un modèle de sécurité formelle, permettant d'exhiber des preuves élégantes ; il s'agira par la suite d'un argument convaincant et gagnant en compétitivité face aux autres protocoles inefficaces dans le modèle standard ou dont la sécurité est heuristique.

6.1.3 EKE et variantes

L'un des protocoles d'échange de clés authentifié le plus célèbre est le protocole *Encrypted Key-Exchange, EKE* [22]. Ce protocole a été proposé comme variante du protocole Diffie-Hellman par Bellovin et Meritt ; il permet à un client d'obtenir une clé commune authentifiée avec un serveur. Comme son nom l'indique, ce protocole utilise comme composante additionnelle au protocole figure 6.1, une combinaison de primitives de chiffrement asymétrique et symétrique où le mot de passe est utilisé comme clé symétrique commune. Le principe est le suivant : notons U_1 et U_2 les deux participants ; U_1 génère une clé publique et l'envoie à U_2 chiffrée avec le mot de passe, qui déchiffre et chiffre une clé K avec la clé publique obtenue. Plusieurs variantes se sont développées par la suite. Nous distinguons deux formes différentes : une première, où le mot de passe est employé comme clé de chiffrement symétrique, il s'agit du protocole EKE initial de Bellovin et Meritt et une seconde, où le chiffré est obtenu en multipliant le message par un haché du mot de passe. Pour cette dernière version efficace, on définit une fonction destinée à masquer le mot de passe, où l'un des messages (ou les deux) issu de l'échange peut être chiffré ; il en découle plusieurs variantes [38], dont en particulier le protocole AuthA qui a été proposé par Bellare et Rogaway [20] comme candidat à la standardisation des protocoles d'échange de clés. Nous donnons figure 6.2 une description d'une variante EKE du second type, AuthA (le chiffré est obtenu en multipliant le message par un haché du mot de passe). Ce protocole, prouvé sûr dans le modèle de Bellare *et al.* par Bresson *et al.* [39], est à la base des deux protocoles que nous étudierons par la suite.

6.1.4 Attaques par dictionnaire

Du fait de la faible entropie du mot de passe, nous ne pouvons empêcher un attaquant de corrompre un utilisateur en le devinant. Les protocoles d'échange de clés à base de mots de passe peuvent être sujets à des attaques par recherche exhaustive, aussi connu sous le nom d'*attaques par dictionnaire* [22, 87, 93, 14, 38], dans lesquelles l'adversaire essaie de casser la sécurité du schéma en essayant toutes les valeurs possibles du mot de

passe. Nous distinguons les attaques en ligne (ou *on-line* en anglais) et hors ligne (ou *off-line* en anglais) suivant que l'attaquant interagisse ou non avec le système pour tester la validité du mot de passe en jeu. Puisque nous ne pouvons pas empêcher l'adversaire d'éliminer un mot de passe par session, notre but est de garantir que l'attaque *en ligne* est la seule possible ; en particulier nous devons nous assurer que les attaques *hors ligne* ne lui fournissent aucun biais. Plus précisément, notre but est de montrer que l'attaquant ne peut pas éliminer plus de n mots de passe après n sessions.

Par la suite, nous supposons que le mot de passe pw est choisi dans un dictionnaire $\mathcal{D}$ de taille bornée. On note N la taille du dictionnaire et nous supposons qu'elle est fixée à l'avance. De plus, afin d'éviter les attaques par partition, nous faisons l'hypothèse que le dictionnaire suit une distribution uniforme même si d'autres distributions peuvent être considérées.

Remarque 11 *Reprenons le protocole EKE de Bellovin et Merritt [22]. Nous pouvons remarquer que la clé publique (si elle contient une certaine forme de redondance, par exemple) peut être utilisée par l'attaquant* a posteriori *pour tester la validité d'un mot de passe. De plus, nous pouvons remarquer que le mot de passe est utilisé comme clé symétrique pour chiffrer à la fois la clé publique et le chiffré de la clé de session partielle ; une telle utilisation du mot de passe ajoute des restrictions d'implémentation, dictées entre autre par l'espace des clés de sessions et des chiffrés du schéma de chiffrement.*

6.2 Modélisation des protocoles d'échange de clés

6.2.1 Définitions

Participants

Nous supposons que l'ensemble des participants est fixé à l'avance : le serveur et un client. Ils peuvent participer à plusieurs exécutions du protocole ; ces exécutions peuvent être différentes et concurrentes. Dans le modèle que nous introduisons ici, chaque participant a accès à un nombre illimité d'instances au moyen desquelles il est autorisé à initier le protocole, ou à participer à son exécution. On note $\mathcal{I}$ l'ensemble de ces instances et U_i^s la s-ième instance du participant U_i. Lorsqu'il n'y a pas d'ambiguïté, nous noterons U_i plutôt que U_i^s. Nous définissons l'ensemble $\mathcal{C}$ des clients et S le serveur. Dans un protocole à deux parties, on a donc : $\mathcal{U} = \mathcal{C} \cup \{S\}$. Si U_i^s ou U_j^t sont engagés dans une "conversation" pour calculer une clé de session commune, on notera $\mathsf{sk}_{i,j}^{s,t} = \mathsf{sk}_{U_i^s} = \mathsf{sk}_{U_j^t}$ la clé de session calculée, en cas de succès[1] ; on dira que U_i^s et U_j^t sont partenaires dans cette session. On pourra se rapporter à la définition 20 pour la notion de *partenariat*.

Définition 19 (Échange de clés à base de mot de passe à deux parties (2-PAKE)) *Un protocole d'échange de clés à base de mot de passe à deux parties* 2-PAKE P*, est spécifié par trois algorithmes polynomiaux* $P \stackrel{déf}{=} \langle \mathsf{LL}, \mathsf{Client}, \mathsf{Serveur} \rangle$ *:*

- LL *: spécifie la distribution initiale des secrets à long terme. Cet algorithme prend en entrée un paramètre de sécurité* λ *;*
- Client *: spécifie l'exécution d'un client* C*. Cet algorithme prend en entrée une instance* C^s*, un état* s_C*, l'identité de l'émetteur* $U \in \mathcal{C}$ *et un message* m*. Il retourne le message que le client* C *est supposé renvoyer en réponse ;*

[1] Nous parlons de succès lorsque les deux clients ont acceptés avec une clé de session commune.

– Serveur *: spécifie l'exécution du serveur S. Cet algorithme prend en entrée une instance S^s, un état s_S, l'identité de l'émetteur et un message. Il retourne le message que le serveur S est supposé renvoyer comme réponse.*

Dans la définition ci-dessus, le format du message m en entrée est spécifique au protocole étudié.

LL est un algorithme probabiliste polynomial, qui retourne les secrets à long terme de tous les participants. Dans notre cas, celui de l'authentification d'un client à base de mots de passe *via* un canal ni confidentiel, ni authentifié, la clé consiste en un mot de passe pw pour chaque client (également donné au serveur).

6.2.2 Modélisation de la communication

flag d'identité de session ou sid/ flag partenaire de session ou pid

Dans le modèle de sécurité pour les protocoles 2-PAKE, nous modélisons les instances des joueurs par des requêtes accessibles à l'attaquant *via* des requêtes aux oracles. L'identifiant de session sid (ou *session ID* en anglais) permet de caractériser la communication entre deux participants pour une session ; cet identifiant est en général une suite d'échanges qui consiste en des combinaisons instances/messages envoyés. Sa définition varie selon les modèles : il peut dans certains cas être défini tout au début [17]. De manière générale, l'identifiant de session correspond à une partie de la communication échangée. Parfois, le *transcript* de la communication est souvent très commode pour cette caractérisation : il s'agit de la communication interceptée sur le canal public durant l'exécution d'une session.

flag d'acceptation/ flag de terminaison

Nous définissons les flag accepte et termine afin de modéliser les deux points suivants : une instance accepte *via* le flag accepte, lorsqu'elle est en possession d'une clé de session sk, une identité de session sid (session ID, en anglais) et un partenaire associé pid (se référer à la définition 20). Une instance termine *via* le flag termine lorsque son but est atteint et qu'il n'est pas en attente d'un autre message. Cette indication peut être commode lorsqu'une instance désire recevoir la confirmation que son partenaire existe bien : une possibilité pour lui consiste alors d'accepter et de ne terminer qu'une fois convaincu de l'existence de son partenaire. Nous définissons la notion de partenariat ci-dessous :

Définition 20 *On dira que deux instances U_i^s et U_j^t sont partenaires si et seulement si les quatre conditions suivantes sont vérifiées :*

1. *U_i^s et U_j^t sont dans un état d'acceptation ;*
2. *U_i^s et U_j^t partagent le même identifiant de session sid ;*
3. *Le participant U_i^s accepte avec la même clé que U_j^t et vice-versa ;*
4. *U_i^s est la seule instance en état d'acceptation avec U_j^t et vice-versa.*

Interactions adversaire/oracle : requêtes aux oracles

L'attaquant a accès à plusieurs instances en parallèle *via* des requêtes aux oracles. De fait, il peut contrôler la communication de plusieurs manières :

– si le canal est non sécurisé, il peut créer, falsifier, transmettre ou encore supprimer des messages ;

– si le canal est sécurisé et/ou authentifié, il peut suivant le modèle corrompre un adversaire et ses pouvoirs varient en fonction du moyen d'authentification utilisé (moyen authentification *symétrique* ou *asymétrique*).

Selon le cas, le modèle peut considérer un adversaire malhonnête et actif qui a le droit de corrompre les participants soit avant l'initialisation du protocole (l'attaquant est alors dit *statique*), soit après son initialisation (l'attaquant est alors dit adaptatif) selon le niveau de sécurité que l'on peut atteindre.

Nous nous concentrons sur deux modèles célèbres, *Find Then Guess* (FTG) et *Real or Random* (RoR). Ce dernier modèle est plus fort [12, 3] : on peut montrer qu'un protocole sûr dans le modèle RoR l'est également dans le modèle FTG avec une perte de facteur (q_{test} égal au nombre de requêtes Test ; ces requêtes seront définis juste après). Dans ces deux modèles, l'interaction de l'adversaire avec le protocole s'effectue au moyen de requêtes spécifiques aux oracles. Nous commençons par présenter le premier modèle. Une simple modification, nous permettra d'introduire le modèle RoR.

– $\mathsf{Execute}(U_i^r, S^t)$: cette requête modélise les attaques passives au cours desquelles l'adversaire demande une exécution honnête du protocole. Il reçoit le *transcript* total issu de la communication entre les instances U_i^r, S^t de son choix.
– $\mathsf{Send}(U_i^r; m)$: cette requête modélise le transfert à l'oracle U_i^r, où $U_i \in \mathcal{C} \cup \{S\}$ par l'adversaire d'un message m qu'il choisit. L'adversaire reçoit le message que U_i aurait produit s'il avait reçu le message m. Si le message n'est pas valide ou si U_i^r n'est pas dans en état de transition, noté "*waiting*", la requête est ignorée ;
– $\mathsf{Corrupt}(U_i)$: cette requête modélise la fuite des secrets long-termes. Pour les protocoles 2-PAKE, on distingue deux types de corruptions : les corruptions partielles, pour lesquelles l'attaquant apprend les secrets long termes de U_i (c.à.d. son mot de passe), mais il n'apprend rien sur les états internes de l'instance, et les corruptions *fortes*, pour lesquelles l'attaquant apprend le mot de passe et les états internes de U_i ;
– $\mathsf{Reveal}(U_i^r)$: si la clé de session est définie, la clé de session $\mathsf{sk}_{U_i^r}$ est renvoyée, sinon un symbole d'erreur est renvoyé. La requête Reveal modélise une utilisation abusive de la clé de session établie, c.à.d. une fuite d'information ; notre but est de vérifier que les clés de sessions ne révèlent aucune information sur le mot de passe ;
– $\mathsf{Test}(U_i^r)$: si l'instance U_i^r a accepté avec une clé de session $\mathsf{sk}_{U_i^r}$, on tire un bit b aléatoire. Si $b = 0$, on donne la clé $\mathsf{sk}_{U_i^r}$ à l'attaquant ; sinon, si aucune clé de session n'est définie pour l'instance U_i^r ou si $b = 1$, on lui renvoie une clé de session aléatoire de même taille que la précédente (toujours la même). Cette requête n'est accessible que si l'instance est fraîche, c.à.d. si la clé de session n'est pas trivialement connue de l'attaquant. Une instance est fraîche si elle n'est pas corrompue et si aucune Reveal ne lui a été posée. On initialise le flag fresh à vrai pour toutes les instances.

Dans le modèle RoR, l'attaquant est autorisé à poser plusieurs requêtes Test : à présent, on demande que les clés de session réelles soient non seulement indistinguables d'une clé aléatoire, mais également indépendantes les unes des autres. La requête est définie pour un bit b choisi par le challenger tout au début de l'expérience ; pour chaque requête, la réponse de l'oracle est indépendante du nombre de requêtes posées par l'attaquant : l'oracle répond toujours de la même manière ; ou bien toutes les clés de session obtenues comme réponse à l'oracle sont issues d'une exécution réelle du protocole ; ou bien ces clés sont toutes aléatoires selon la valeur du bit b. De plus, aussitôt qu'une requête Test a été posée à une instance donnée, aucune requête Reveal ne peut être posée à cette instance.

La propriété de *forward-secrecy* permet de modéliser le degré de compromission des secrets long termes [2] : un protocole vérifie la propriété de *forward-secrecy* si la connaissance des secrets à long terme (ici le mot de passe) ne compromet pas les clés de sessions établies par le passé.

6.2.3 Extensions du modèle

Oracles et Instances

Lorsqu'une requête est posée à un oracle avec un message m en entrée, où m provient d'une réponse à un oracle, on dit que m est un message généré par un oracle. Plus formellement, lorsqu'il n'y a pas d'ambiguïté sur le message, on écrira $\mathsf{OG}(U^s)$ lorsque U^s reçoit un message généré par un oracle. Plus généralement, on écrit $\mathsf{OG}(U^s, n)$, lorsque le n-ième message (dans le protocole en question) reçu par U^s est généré par un oracle.

Définition 21 *On dit que m est un message généré par un oracle s'il existe une instance U_j^s, avec $s \in \mathcal{I}$ et un participant $U_i \in \mathcal{U}$ tels que $m = \mathsf{Send}(U_j^s, U_i; m')$ pour un certain message m'.*

Nous redéfinissons la requête Send de manière à capturer l'identité de l'expéditeur :

$\mathsf{Send}(U_i^r, U_j; m)$ **:** cette requête modélise les attaques au cours desquelles l'attaquant envoie un message à l'instance U_i^r au nom du participant U. L'adversaire reçoit le message que U_i aurait produit s'il avait reçu le message m.

Extension de la notion de fraîcheur

La notion de sécurité pour les protocoles 2-PAKE (voir section 6.3) donne un accès partiel aux clés de sessions. Cela dit, dans certains cas, cette clé de session peut être trivialement connue de l'attaquant ; nous définissons donc la notion de fraîcheur qui caractérise les clés validant une attaque de l'adversaire. Au départ, nous supposons que toutes les sessions sont fraîches : le *flag* fresh est initialisé à vrai.

Signalons que les définitions existantes (voir définition de la (ou des) requête(s) Test paragraphe 6.2.2) supposent que si un des joueurs est corrompu, la clé de session est forcément non fraîche. Nous étendons ce modèle de sécurité en maintenant certaines clés de session fraîches même en cas de corruption d'un des participants : remarquons que si les "vrais" utilisateurs jouent le protocole, la clé de session peut être toujours protégée en pratique. Notre modèle de sécurité prend en en compte ce type d'attaque ; ce point sera mis, plus en évidence dans l'analyse de sécurité.

Plus formellement, nous considérons qu'une instance est "fraîche"pour une session, et nous attribuons la valeur vrai au flag fresh de cette instance si l'une des deux conditions suivantes est vérifiée :

1. ni l'instance en question, ni l'instance partenaire n'a été corrompue avant que la session ne débute, ou bien
2. l'attaquant interagit avec le protocole *via* des requêtes Execute ou de simples *transferts* de requêtes Send (tous les messages reçus sont générés par des oracles).

[2] le mot de passe, pour les protocoles 2-PAKE.

[2] Par extension, ce flag peut être attribué à une session, si celles-ci est exécutée par des instances fraîches.

Nécessité des requêtes Execute

Intuitivement, les requêtes Execute modélisent des observations passives des *transcripts* tandis que les requêtes Send modélisent des attaques actives contre certains des joueurs honnêtes. Ces requêtes peuvent être combinées les unes aux autres. Comme nous l'avons vu, les requêtes Execute peuvent, être simulées par une séquence de requêtes Send. Par conséquent, une séquence est décomptée dans le nombre d' "attaques passives", si toutes les requêtes sont simulées par des oracles.

Par exemple, nous avons supprimé ces requêtes dans le protocole 2-PAKE du chapitre 8 : en effet, si le *transcript* complet est constitué de messages générés au moyen d'une succession de requêtes non modifiées, l'utilisation d'une requête Execute dans les modèles existants [14] est alors tout à fait similaire. Mais si l'un des messages se trouve modifié, il s'agit alors d'une attaque *active*. Rappelons que pour les protocoles 2-PAKE, l'un des buts est de montrer que seules les attaques par dictionnaire *on-line* (inévitables) sont susceptibles de compromettre la sécurité. Autrement dit, une attaque passive n'apporte aucune information sur le mot de passe, et une attaque active ne doit permettre à l'adversaire que d'éliminer au plus un seul mot de passe.

En revanche, nous verrons que pour le protocole du prochain chapitre 7, les requêtes Execute sont importantes car elles peuvent être utilisées par l'attaquant pour avoir un biais sur le mot de passe : plus précisément, l'utilisation d'une succession de requêtes Send n'est pas tout à fait similaire car lors de la simulation, nous ne pouvons prédire le comportement de l'attaquant ; il se peut qu'il décide de ne pas transférer un message en cours d'exécution : nous nous devons donc de décompter les requêtes Execute et Send séparément.

6.3 Sécurité pour les protocoles PAKE

Une des particularités de notre modèle est la considération simultanée des propriétés suivantes :

- la sécurité sémantique pour la clé de session échangée ou *key privacy* vis à vis de participants extérieurs ou corrompus (ce qui permet de considérer la *forward-secrecy*) ;
- l'authentification *unilatérale* ou *mutuelle* suivant le protocole ;
- la protection du mot de passe du client vis à vis de l'attaquant ;
- l'anonymat du client vis à vis du serveur ; nous reviendrons sur ce point plus loin lors de la description de notre protocole et de sa preuve.

6.3.1 Sécurité sémantique

La sécurité sémantique de la clé de session est modélisée par la requête Test. Dans notre contexte, cette requête Test peut être demandée autant de fois que l'adversaire le souhaite. D'après ce qui précède, ces requêtes doivent être faîtes seulement sur des instances "fraîches". L'un des buts est de garantir que les clés de session établies entre C et S restent inconnues de tout autre participant : il s'agit de la propriété de *confidentialité de la clé* aussi appelée *sécurité sémantique*. En effet, nous voulons que l'adversaire ne puisse tirer aucune information significative des requêtes effectuées aux oracles sur les clés de session définies fraîches (c.à.d. non connues trivialement de l'adversaire, on pourra se reporter à notre définition de fraîcheur, définie paragraphe 6.2.3).

6.3.2 Authentification

L'un des buts pour les protocoles 2-PAKE est de garantir que les données reçues (resp. envoyées) proviennent bien de l'émetteur (*resp.* parviennent bien au destinataire) légitime. Nous considérons pour cela les notions d'authentification *unilatérale* ou *mutuelle*. Un protocole d'échange de clés vérifie la propriété d'authentification unilatéral s'il permet à une des deux entités de s'authentifier auprès de l'autre participant en étant correctement identifié. Un protocole vérifie la propriété d'*authentification mutuelle* s'il permet à deux parties de s'authentifier l'un à l'autre de telle sorte à ce que chacun soit assuré de l'identité de son interlocuteur. La notion d'authentification *mutuelle* s'est d'abord développée indépendamment des protocoles d'échange de clés. MacKenzie et Swaminathan [94] ont été les premiers à avoir intégré cette notion dans les protocoles d'échange de clés, en combinant un protocole à deux parties "rudimentaire" (AKE) avec cette notion. Boyko *et al.* [38] ont ensuite défini cette notion et prouvé sa validité dans le cas particulier des protocoles Diffie-Hellman. Dans [14], Bellare *et al.* adoptent une approche plus modulaire : dans un premier temps, ils construisent un protocole AKE dont ils prouvent la sécurité dans le modèle de l'oracle aléatoire, puis présentent une transformation générique permettant de garantir l'*authentification unilatérale* ou *mutuelle*.

Le modèle que nous proposons considère la notion d'authentification comme propriété intégrante du modèle : alors que les preuves existantes distinguer deux jeux indépendants pour la *sécurité sémantique* et l'*authentification*, notre modèle unifie la preuve en montrant ces deux propriétés dans un même jeu de sécurité.

6.3.3 Sécurité d'un protocole

Dans les protocoles 2-PAKE, le but est d'assurer l'authentification *unilatérale* ou *mutuelle* et la *confidentialité de la clé de session* vis à vis d'un utilisateur malhonnête. Les ressources de calcul de l'adversaire sont le temps d'exécution, noté t, et le nombre de requêtes aux oracles $\mathsf{Execute}, \mathsf{Send}, \mathsf{Reveal}$ respectivement notés $q_{\mathsf{execute}}, q_{\mathsf{send}}, q_{\mathsf{reveal}}$.

On estime qu'un adversaire $\mathcal{A}$ casse la sécurité AKE d'un protocole 2-PAKE P, s'il devine correctement le bit b. Plus précisément, considérons un attaquant $\mathcal{A}$ limité à une puissance de calcul polynomiale qui à l'issue de q_{test} requêtes Test retourne un bit b'. $\mathcal{A}$ casse la sécurité du protocole P si $\Pr[b = b'] > 1/2$. Nous définissons l'avantage AKE de l'attaquant $\mathcal{A}$ à casser la sécurité sémantique du protocole P par $\mathsf{Adv}^{\mathsf{ake}}_{P,\mathcal{A}}(\lambda) = 2 \cdot \Pr[\mathsf{Succ}] - 1$.

Définition 22 *On dit qu'un protocole* 2-PAKE*, P est sûr si pour tout attaquant polynomial $\mathcal{A}$ qui interagit avec le protocole P* via *au plus q instances, l'avantage $\mathsf{Adv}^{\mathsf{ake}}_{P,\mathcal{A}}$ est une fonction négligeable en le paramètre de sécurité,c.à.d.*

$$\mathsf{Adv}^{\mathsf{ake}}_{P,\mathcal{A}}(\lambda) < \frac{c \cdot q}{N} + \mathsf{negl}(\lambda),$$

où N est la taille du dictionnaire (contenant les de mots de passe et muni d'une distribution uniforme), c est une constante petite, idéalement proche de 1, et negl() *est une fonction négligeable en le paramètre de sécurité.*

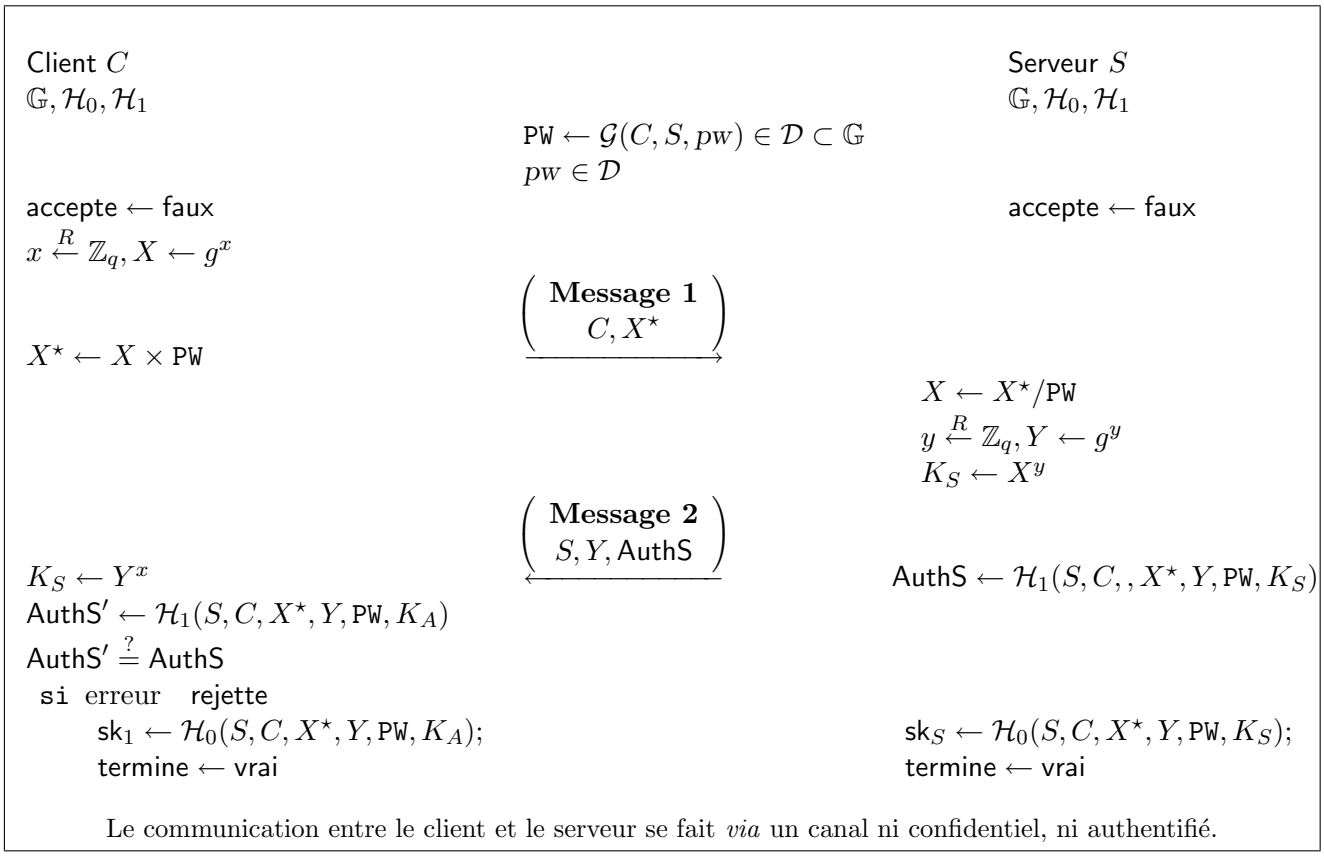


FIG. 6.2 – Variante AuthA du protocole EKE où un message est chiffré. $\mathbb{G} = \langle g \rangle$ est un groupe cyclique d'ordre q. $\mathcal{H}_i$ pour $i = 0, 1$ sont des fonctions de hachage de $\{0,1\}^*$ vers $\{0,1\}^{l_i}$ modélisés par des oracles aléatoires et le mot de passe $pw \overset{R}{\leftarrow} \mathcal{D}$, avec $\mathcal{D}$ un dictionnaire de taille N et uniformément distribué.

Chapitre 7

Scénarios à trois parties

Dans cette partie, nous nous intéressons aux protocoles d'échange de clés à trois parties, où un client souhaite accéder à un service proposé par un serveur. Dans ce domaine, la littérature est considérable et faire un compte-rendu détaillé des travaux antérieurs serait illusoire. Nous décrirons seulement un schéma qui a été très influent, et qui, même si très éloigné du nôtre, nous permet de mieux comprendre les critères de correction et de sécurité définis par la suite pour les protocoles PAKE à trois parties. Nous introduirons ensuite le modèle de sécurité pour les protocoles d'échange de clés à trois parties en étendant les définitions de [19] : il s'agit d'une extension du modèle présenté au chapitre précédent où le moyen d'authentification reste le mot de passe. Ce modèle étend les moyens de l'adversaire : les stratégies d'attaques valides se diversifient. Par ailleurs, cette extension considère comme auparavant, les propriétés de *sécurité sémantique* et résistance aux *attaques par dictionnaire* de manière simultanée. Enfin, nous présenterons une variante du protocole [2], dont nous prouverons la sécurité dans ce modèle.

Sommaire

7.1 Introduction

7.1.1 Exemple : Kerberos

L'incontournable système à la source, "*Kerberos*" a été mis au point dans les années 80 par le MIT. Il s'agit d'un des premiers protocoles où les participants communiquent *via* des canaux non sécurisés (ni confidentiels et ni authentifiés). Ce protocole met en jeu plusieurs entités :

1. un client possédant un clé secrète K_C, partagée avec un tiers de confiance, le centre de distribution de clés (KDC). Le client possède également un secret lui permettant de se faire reconnaître (généralement un mot de passe) ;
2. un serveur possédant une clé secrète K_S ;
3. un centre d'émission de tickets, TGS (pour *Ticket-Granting Service*), possédant une clé secrète K_{TGS} partagée avec le centre KDC. La TGS connaît la clé K_S du serveur ;
4. un centre de distribution de clés, KDC (pour *Key Distribution Center*), qui connait les clés secrètes K_C et K_{TGS}.

Description

Nous donnons une description du protocole *Kerberos* figure 7.1.

Le client C s'identifie et choisit un serveur S, il envoie ces informations au centre gestionnaire des clés des serveurs. L'identité envoyée est d'abord vérifiée et le centre KDC répond par un ticket T chiffré avec un identifiant et une clé de session $K_{C,\mathsf{TGS}}$, cette dernière étant également chiffrée avec la clé secrète de chiffrement de C. Le client retrouve la clé $K_{C,\mathsf{TGS}}$ qui lui permet de chiffrer son authentifiant, la date d'émission du ticket T, t_0 et sa durée de vie D. Le service d'émission de tickets retrouve ce ticket, et vérifie l'authenticité et l'intégrité des informations reçues. Un ticket d'accès au serveur $T_{C,S}$ est alors émis ainsi qu'une clé de session $K_{C,S}$ associée aux futures communications entre le client C et le serveur S choisi. Ces deux éléments sont chiffrés avec la clé du serveur et la clé partielle $K_{C,\mathsf{TGS}}$ respectivement.

Donnons l'idée intuitive des fonctionnalités apportées par les invariants d'une session[1] dans ce protocole *Kerberos* : la vérification de l'authenticité des requêtes se fait au moyen des clés de session possédées par chaque partie. Ces clés doivent être renouvelées à chaque exécution. Nous pouvons remarquer que le *transcript* d'une communication entre deux participants contient des informations sur l'identité du client, le ticket et une clé de session (tous ces éléments sont chiffrés). La notion d'identifiant de session (voir définition 6.2.2) est primordiale pour garantir la sécurité, elle permet d'assurer l'authenticité des messages envoyés par l'émetteur désigné. L'implication d'un tiers additionnel, la TGS, permet non seulement de limiter les risques que les clés des utilisateurs soient révélées, mais contribue également à rendre le système plus *transparent* vis-à-vis du client. Malheureusement, le gestionnaire des clés doit impérativement être implémenté par une autorité de confiance, car si le centre KDC est corrompu, alors toutes les clés des clients sont connues de l'attaquant. D'autre part, l'usage de *tickets* permet l'accès au serveur : si un adversaire s'en empare, il peut se faire passer pour le client. C'est pourquoi, ils ont une durée de validité limitée.

[1] Le terme "invariants" fait référence aux secrets qu'ils soient *long terme* ou non.

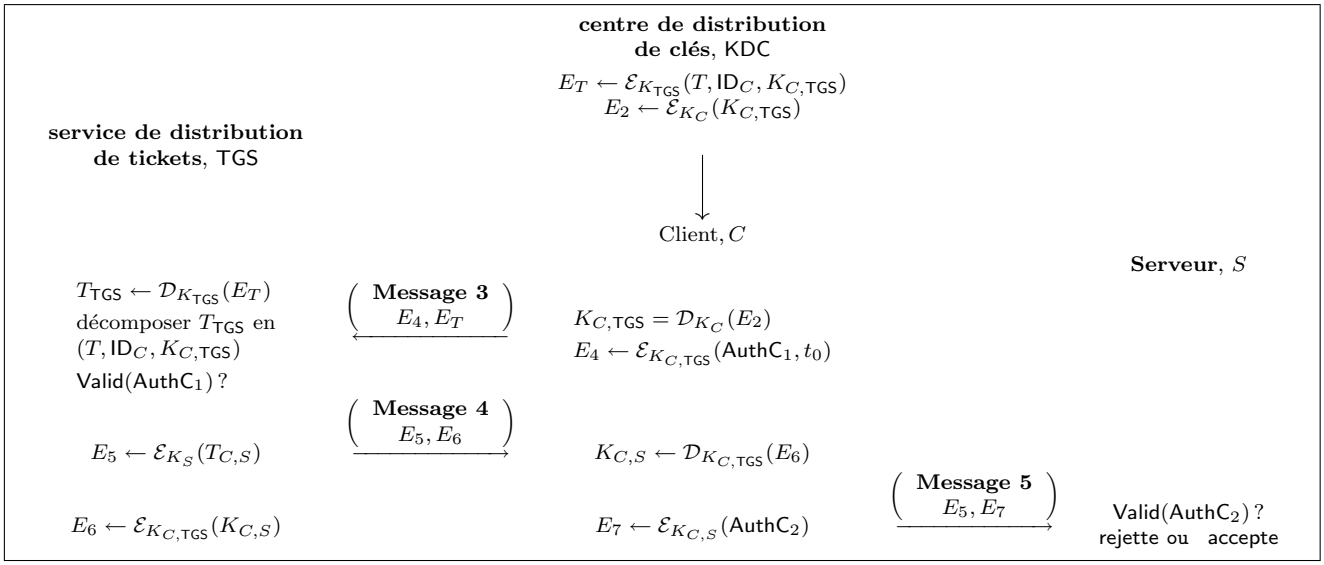


FIG. 7.1 – Protocole *Kerberos*.

7.1.2 Définitions

Nous considérons un protocole d'échange de clés à trois parties, où un client souhaite avoir accès à un service proposé par le serveur. Précisons qu'en pratique, lors de la procédure d'authentification, le client ne parle pas forcément directement au serveur d'authentification mais à un serveur d'application (que nous appellerons *passerelle*) qui permet d'établir le lien avec le serveur d'authentification. La passerelle ne connaît pas le mot de passe, il se charge juste de vérifier qu'il parle bien à un client dont le mot de passe est enregistré dans la base de données du serveur d'authentification. Dans un tel protocole, nos buts sont les suivants :

1. garantir la sécurité sémantique (ou *key privacy* en anglais) de la clé de session échangée vis-à-vis du serveur d'authentification (comme définie au chapitre précédent) mais aussi vis-à-vis de participants corrompus. Cette dernière garantie permet de considérer la *forward-secrecy* ;
2. garantir la protection du mot de passe du client *vis à vis* de la passerelle ;
3. rendre l'infrastructure passerelles/serveur la plus *transparente* possible *vis à vis* du client : ce dernier ne sait pas s'il parle à un serveur détenant les mots de passe ou à une passerelle.

Nous verrons au prochain chapitre comment ces propriétés contribuent à introduire l'anonymat du client vis à vis du serveur. Pour l'instant, nous nous intéressons juste aux particularités du modèle que nous introduisons et à la description du protocole GPAKE.

Une des contributions dans l'article [4] est la prise en compte de l'anonymat du client vis à vis du serveur. Nous étendons la définition de protocole 2-PAKE au scénario à trois parties dirigé par une *passerelle* (ou *getaway* en anglais) :

Définition 23 (Gateway-based Authenticated Key-Exchange) *Un protocole* Gateway-based Authenticated Key-Exchange, *P est spécifié par quatre algorithmes polynômiaux* $P = (\mathsf{LL}, \mathsf{Client}, \mathsf{Gateway}, \mathsf{Serveur})$ *:*

- LL *spécifie la distribution initiale des secrets long terme ; il prend en entrée un paramètre de sécurité* λ *;*
- Client *spécifie l'exécution du protocole réalisée par un client* C *; Il prend en entrée une instance* C^s*, un état* s_C*, l'identité de l'émetteur et un message. Il retourne le message que le client* C *doit renvoyer comme réponse ;*
- Gateway *spécifie l'exécution du protocole réalisée par une passerelle* G *; il prend en entrée une instance* G^s*, un état* s_G*, l'identité de l'émetteur et un message. Il retourne le message que la passerelle* G *doit renvoyer comme réponse ;*
- Serveur *spécifie l'exécution du protocole réalisée par le serveur* S *; il prend en entrée une instance* S^s*, un état* s_S*, l'identité de l'émetteur et un message. Il retourne le message que la passerelle* G *doit renvoyer comme réponse.*

LL est un algorithme probabiliste polynomial, qui retourne les secrets à long terme de tous les participants. Dans notre cas particulier, celui de l'authentification du client basé sur les mots de passe, et utilisant un canal symétrique confidentiel et authentifié entre les passerelles et le serveur, ces clés consistent en :

- un mot de passe pw pour chaque client (également donné au serveur) ;
- une clé symétrique SecureChan$-S-G$ pour chaque paire passerelle/serveur afin de permettre un canal sécurisé (permettant de garantir la confidentialité et l'intégrité des messages).

7.2 Modèle de sécurité et extensions

Pendant longtemps, les preuves des protocoles étaient approximatives : très souvent, on montrait qu'un schéma résistait à certaines attaques connues sans vraiment exhiber de preuves formelles. Par ailleurs, dans ce contexte, les stratégies de l'adversaire se diversifient de manière plus prononcée que les protocoles à deux parties. Il convient donc de définir parmi celles-ci, les attaques valides. La nécessité des notions de *partenariat* et de *fraîcheur* définies précédemment deviennent encore plus fondamentale à l'élaboration d'un modèle pour ce scénario. C'est en 1995, que Bellare et Rogaway [19] définissent un modèle formel pour les protocoles d'échange de clés à trois parties. Ce sont les premiers à définir une notion de correction pour ces protocoles. Ils définissent un modèle de preuves, qui constituent aujourd'hui une brique de base pour les protocoles prouvés sûrs.

D'un point de vue théorique, il peut être intéressant de remarquer que leur modèle formalise la distinction entre les notion de *distribution de clés* et d'*authentification d'entité*, une distinction, mise plus en évidence dans les protocoles à trois parties ; celles-ci se traduisent respectivement par la propriété d'*authentification unilatérale* ou *mutuelle* et la propriété de *confidentialité de clé*.

C'est dans une extension de ce modèle que sera effectuée la preuve de notre protocole GPAKE. La sous-section qui suit définit les propriétés caractérisant ce modèle de preuve, à l'exemple du chapitre précédent. Nous soulignerons pas à pas les extensions que nous avons pu y apportées.

7.2.1 Modèle de sécurité

Requêtes aux oracles

Comme dans [3], nous adoptons le modèle de sécurité Real-or-Random (RoR), ce qui signifie que l'adversaire interagit avec le protocole *via* des requêtes à des oracles dans un

ordre qu'il décide et pour des exécutions éventuellement concurrentes. À l'issue de ces interactions, nous aurons à montrer que l'attaquant n'est pas capable de distinguer des clés aléatoires de celles renvoyées, comme nous l'avons expliqué dans la chapitre précédent.

Les requêtes aux oracles sont les suivantes :

- $\mathsf{Execute}(C_i^r, G_j^s, S^t)$: cette requête modélise toujours les attaques passives : l'attaquant reçoit le *transcript* total issu de la communication entre les instances C_i^r, G_j^s et S^t de son choix lors d'une exécution honnête.
- $\mathsf{Send}(U_i^r, U; m)$: cette requête modélise les attaques actives au cours desquelles l'adversaire choisit un message m qu'il voudrait envoyé à U_i^r, où $U_i \in \mathcal{C} \cup \mathcal{G} \cup \{S\}$ au nom du participant U.
  Dans notre cas, les échanges entre les passerelles et le serveur se font *via* des canaux garantissant la confidentialité des communications et l'authentification de l'émetteur. De fait, si la passerelle G est corrompue (le $\mathsf{flag\ corrupt}_G$ est vrai), le destinataire n'acceptera pas un message envoyé par un émetteur non authentifié. Rappelons que nous supposons une authentification symétrique entre les passerelles et le serveur (voir définition 23).
- $\mathsf{Reveal}(U_j^s)$: si la clé de session est définie, la clé de session $\mathsf{sk}_{U_j^s}$ est renvoyée, sinon un symbole d'erreur est renvoyé.
- $\mathsf{Test}(U_j^s)$: les requêtes Test peuvent être posées au client ou à une passerelle. Dans le modèle RoR, nous rappelons que le challenger choisit tout au début un bit b. L'attaquant pose autant de requêtes qu'il le souhaite. Nous répondons à ces requêtes exactement comme définie paragraphe 6.2.2. Son attaque est valide s'il devine le bit b pour une session fraîche. Dans notre protocole GPAKE, cette notion de fraîcheur n'est pas tout à fait similaire, nous la précisons paragraphe 7.2.2.

Remarque 12 *Contrairement au modèle présenté à la section précédente, ici les requêtes* Execute *peuvent être simulées par une séquence de requêtes* Send*, mais une telle séquence sera décomptée dans le nombre de "sessions actives".*

Remarquons que là encore, la requête Reveal *modélise une utilisation abusive de la clé de session établie et donc une fuite d'information : nous devons vérifier que les clés de sessions ne laissent fuir aucune information sur le mot de passe.*

Corruption

On dit qu'un participant U est corrompu et le $\mathsf{flag\ corrupt}_U$ prend la valeur vrai si l'une des requêtes Corrupt lui est posée. L'attaquant apprend alors les secrets du participant à qui la requête est posée ; ces secrets varient selon l'identité du participant. Nous supposons que le flag corrupt est initialisé à faux pour tous les participants. Nous ne considérons que des corruptions partielles, où seuls les secrets long termes sont révélés à l'attaquant, mais pas les états internes. Ces différentes requêtes peuvent être posées par l'attaquant à un client, une passerelle et un serveur et sont définies de la manière suivante :

- $\mathsf{Corrupt}(C_i)$: cette requête modélise une corruption du client C_i où l'adversaire apprend le mot de passe du client C_i. On pose alors $\mathsf{corrupt}_{C_i} \leftarrow \mathsf{vrai}$;
- $\mathsf{Corrupt}(G_j)$: cette requête modélise une corruption d'une passerelle G_j où l'adversaire a accès en lecture et écriture au canal sécurisé entre la passerelle G_j et le serveur (nous avons supposé une authentification symétrique entre G_j et S). On pose alors $\mathsf{corrupt}_{G_j} \leftarrow \mathsf{vrai}$;
- $\mathsf{Corrupt}(S)$: cette requête modélise la corruption du serveur où l'adversaire apprend les mots de passe pw_C de tous les clients C stockés dans le serveur. Mais il a aussi

accès au canal sécurisé entre le serveur et toutes les passerelles (puisque nous avons supposé une authentification symétrique entre ces deux participants). On pose alors $\mathsf{corrupt}_C \leftarrow \mathsf{vrai}$ et $\mathsf{corrupt}_G \leftarrow \mathsf{vrai}$ pour tous les clients C et toutes les passerelles G.

Remarque 13 *Précisons dés à présent que nous supposerons que les corruptions sont statiques. Autrement dit, avant l'initialisation du protocole, nous savons quels sont le ou les participant(s) corrompus. Malheureusement, nous ne pouvons assurer une simulation consistante si une corruption intervient lors d'une exécution du protocole : si la* passerelle *est corrompue, elle peut très bien calculer une clé de session différente indépendante de celle du client ; ce qui est en contradiction avec la notion de partenariat.*

7.2.2 Extension de la notion de Fraîcheur

Rappelons que la notion de *fraîcheur* est indispensable pour définir les sessions validant une attaque : pour casser la *sécurité sémantique*, l'attaquant doit deviner le bit b associé à une session fraîche avec probabilité non négligeable. Par exemple, si l'on considère les cas énumérés ci-dessous, l'attaquant peut facilement deviner le bit b et gagner :

- Si le serveur est corrompu, l'adversaire peut jouer le rôle de la passerelle et du serveur contre le client : il choisit tout simplement les aléas utilisés par les deux parties pour établir une clé de session commune avec le client sans que ce dernier ne s'en aperçoive ;
- Si C est corrompu, l'adversaire peut jouer le rôle du client pour établir une clé de session commune avec la passerelle sans que celle-ci ne s'en aperçoive ;
- Si G est corrompu, l'adversaire peut jouer le rôle de la passerelle, sans que le serveur, ni le client ne s'en aperçoive.

À présent, supposons que tous les participants sont corrompus (même le serveur), mais que l'adversaire est passif pour une session, c.à.d. qu'il ne pose que des requêtes Execute ou fait simplement des transferts de requêtes Send, remarquons alors que cette session peut tout à fait rester fraîche.

En effet, même si les secrets long terme sont connus de l'adversaire, dans certains cas, le joueur corrompu peut être assuré qu'il parle bien à un joueur honnête. Dans ce cas, la clé de session peut très bien rester secrète ; ce qui nous permet d'étendre la notion de fraîcheur : une session peut rester fraîche même si un secret long terme est révélé.

Remarque 14 *Notons que si une corruption intervient après la fin d'une exécution, cela n'affecte pas le statut de fraîcheur, ce qui permet de considérer une notion plus forte de* forward-secrecy *que celle usuellement prise en compte.*

7.2.3 Quelques outils pour GPAKE

Hypothèse PCDDH

L'hypothèse Diffie-Hellman basé sur un mot de passe en base orientée, PCDDH est une variante de celle présentée dans [5]. Soit $\langle \mathbb{G}, g, q \rangle$ une structure associée à un groupe cyclique $\mathbb{G}$ d'ordre q et généré par g. L'expérience, $\mathsf{Exp}_b^{\mathsf{pcddh}}(\mathcal{A}, \mathcal{D})$ précise l'interaction entre le challenger du jeu Diffie-Hellman à base de mot de passe et en base choisie et l'attaquant. Nous supposons que $\mathcal{D}$ est un dictionnaire de taille N fixe contenant des éléments (les mots de passe) dans $\mathbb{G}$, tous aléatoires et indépendants. Nous munissons cet ensemble d'une distribution aléatoire. Le jeu définissant cette expérience est le suivant : au cours d'une première étape `FIND`, l'adversaire choisit une base X. Le challenger choisit un bit b, un mot de passe pw dans $\mathcal{D}$ et deux aléas s_0, s_1. L'attaquant reçoit alors le mot

de passe pw, deux éléments de $\mathbb{G}$: $X' = (X/pw)^{s_b}$ et $Y = g^{s_1}$. Il gagne s'il devine le bit b correctement.

L'adversaire doit alors deviner le bit b dans l'expérience suivante :

$$\begin{array}{l}\mathsf{Expérience}\ \mathsf{Exp}^{\mathsf{pcddh}-b}_{\mathcal{A},\mathcal{D}}(\lambda)\\ (X,s) \leftarrow \mathcal{A}_1(\texttt{FIND},\mathcal{D})\,;\\ pw \overset{R}{\leftarrow} \mathcal{D}; s_0, s_1 \overset{R}{\leftarrow} \mathbb{Z}_q\,;\\ X' \leftarrow (X/pw)^{s_b}, Y \leftarrow g^{s_0}\,;\\ \texttt{retourner}\ b' \leftarrow \mathcal{A}_2(\texttt{GUESS}, s, X', Y, pw)\,;\end{array}$$

Considérons un attaquant $\mathcal{A}$ possédant une puissance de calcul polynomiale. Soit $\mathcal{D}$ un dictionnaire de taille N et uniformément distribué. Nous définissons l'avantage de $\mathcal{A}$ dans l'expérience $\mathsf{Exp}^{\mathsf{pcddh}-b}_{\mathcal{A},\mathcal{D}}$ précédente pour $b = 0, 1$ par :

$$\mathsf{Adv}^{\mathsf{pcddh}}_{\mathcal{A},\mathcal{D}}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{pcddh}-1}_{\mathcal{D},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{pcddh}-0}_{\mathcal{A},\mathcal{D}}(\lambda) = 1].$$

L'hypothèse $\mathsf{PCDDH}_{\mathbb{G}}$ établit que l'avantage de tout attaquant polynomiale $\mathcal{A}$ est au plus $\frac{1}{N}$ plus une fonction négligeable de λ. Intuitivement, toute stratégie de l'attaquant pour deviner le bit b n'est pas significativement meilleur que celle qui consiste à deviner mot passe avec probabilité $\frac{1}{N}$.

Pour assurer la validité de notre hypothèse, il est important que les éléments du dictionnaire soient choisis de manière indépendante dans $\mathbb{G}$. En effet, le logarithme discret s_b de X' en base g doit être difficile à calculer. Notons que la variante calculatoire de ce problème, où l'objectif est de calculer $X' = (X/pw)^{s_0}$, pour une base X choisie par l'attaquant, étant donnés Y et pw avec probabilité supérieure à $1/N$, est impliquée par l'hypothèse $\mathsf{CDH}_{\mathbb{G}}$. Intuitivement, on peut voir que si les mots de passe sont choisis aléatoirement et indépendamment les uns des autres, le problème revient à deviner le mot de passe. Même si nous ne disposons pas de preuve formelle, cette hypothèse semble tout à fait raisonnable sous l'hypothèse $\mathsf{DDH}_{\mathbb{G}}$ [5].

Preuve NIZKPDL

Dans notre protocole, nous avons besoin de preuve non interactive de connaissance de logarithme discret sans fuite d'information, NIZKPDL. Dans notre cas, ce sera une signature de connaissance : on notera $\texttt{NIZKPDL}(m; g, h)$ la signature de connaissance du logarithme discret de h en base g sur le message m. C'est la signature de Schnorr [113, 114], prouvée sûre dans le modèle de l'oracle aléatoire [106, 107]. Grâce au *forking-lemma* [106, 107] lorsqu'une telle preuve valide est générée par l'adversaire, l'extraction est possible par un seul rejeu (*rewind*, en anglais).

Soit $\mathbb{G}$ un groupe cyclique d'ordre q généré par g. Nous utilisons une version non interactive du système de preuve introduit par Schnorr dans [113, 114], en lui appliquant la transformation de Fiat-Shamir [67] dans le modèle de l'oracle aléatoire [16, 106, 107] :

- Élément public : description du groupe $h = g^s$
- Témoin : l' exposant s
- Preuve de connaissance du témoin : le prouveur choisit un exposant aléatoire $\alpha \in \mathbb{Z}_q$ et pose $u = g^\alpha$. Il calcule $c = \mathcal{H}(X^\star, g, h, u)$ et $v = \alpha - cs \mod q$. Il envoie alors la preuve (c, v) au vérifieur.
- Vérification de la preuve : $c \overset{?}{=} H(X^\star, g, h, g^v h^c)$.

Le forking lemma [106, 107] montre qu'une exécution de ce protocole permet d'extraire le témoin en temps polynomial. Cependant, notons qu'après un *rewind*, nous devons nous assurer qu'aucune extraction supplémentaire n'est nécessaire, auquel cas un autre rejeu sera nécessaire. Et plusieurs extractions peuvent entraîner alors une complexité exponentielle. En ce qui nous concerne, notre analyse ne nécessite qu'un seul *rewind*.

7.3 Gateway-Based Password Authenticated key-Exchange

7.3.1 Description du protocole transparent GPAKE

À présent, nous sommes prêts pour la description de notre nouveau protocole GPAKE. Il s'agit d'une variante du protocole proposé par Abdalla, Chevassut, Fouque et Pointcheval dans l'article [2]. Nous commençons par décrire cette construction, qui peut être vue comme une variante du protocole EKE de Bellovin et Merritt.

Soit $\langle \mathbb{G}, g, q \rangle$ une description associée à un groupe cyclique $\mathbb{G}$ d'ordre q généré par l'élément g. Soit ℓ un paramètre de sécurité. Nous définissons plusieurs fonctions de hachage $\mathcal{G}$, $\mathcal{H}_1$, et $\mathcal{H}_2$:

$$\mathcal{G} : \mathcal{U}^2 \times \mathcal{D}' \mapsto \mathbb{G}, \quad \mathcal{H}_1 : \mathcal{U}^2 \times \mathbb{G}^2 \mapsto \{0,1\}^\ell, \quad \mathcal{H}_2 : \mathcal{U}^2 \times \mathbb{G}^2 \mapsto \{0,1\}^\ell.$$

Pour chaque mot de passe $pw \in \mathcal{D}'$, nous définissons $\mathtt{PW} \stackrel{\text{déf}}{=} \mathcal{G}(C, G, pw) \in \mathcal{D} \subset \mathbb{G}$, il s'agit du moyen d'authentification du client C auprès de la passerelle G.

Cette dernière a pour rôle de permettre au client d'établir un canal sécurisé qui ne laisse pas entrevoir au client l'implication de la passerelle dans la procédure d'authentification. Nous appelons cette propriété *transparence* vis à vis du client. Le client connaît le mot de passe pw, et la passerelle fait alors appel au serveur qui connaît les PW associés, la capacité de stockage étant moins limitée que pour la passerelle.

Par ailleurs, la génération de mots de passe PW aléatoires dans $\mathbb{G}$ (gérée par l'utilisation de la fonction $\mathcal{G}$) est essentielle pour pouvoir utiliser l'hypothèse $\mathsf{PCDDH}_{\mathbb{G}}$.

Le protocole consiste en quatre échanges : le client commence par choisir un scalaire x aléatoire dans $\mathbb{Z}_q$ et calcule $X = g^x$, qu'il chiffre en utilisant $\mathcal{G}(C, G, pw)$ comme masque ; le client obtient une valeur X^* qu'il envoie à la passerelle, accompagnée de son identité C. Après réception du message, la passerelle choisit $y \stackrel{R}{\leftarrow} \mathbb{Z}_q$ et calcule $Y = g^y$. Elle transfert X^*, Y au serveur. Après réception d'un message (X^*, Y) de la passerelle, le serveur calcule $X = X^*/\mathcal{G}(C, G, pw)$. Il choisit un scalaire s aléatoire dans $\mathbb{Z}_p$ et calcule la paire $(\overline{X} = X^s, \overline{Y} = Y^s)$ qu'il envoie à la passerelle. La passerelle reçoit alors une paire $(\overline{X}, \overline{Y})$ et calcule une clé de session partielle $K = \overline{X}^y$, un authentifiant $\mathsf{AuthG} = \mathcal{H}_2(C, G, X^*, \overline{Y}, K)$ et une clé de session $\mathsf{sk} \stackrel{\text{déf}}{=} \mathcal{H}_1(C, G, X^*, \overline{Y}, K)$ et renvoie $G, \overline{Y}$ et AuthG au client. Muni d'un triplet $(G, \overline{Y}, \mathsf{AuthG})$, le client calcule la "clé Diffie-Hellman" partielle $K = \overline{Y}^x$ et vérifie si $\mathsf{AuthG} \stackrel{?}{=} \mathcal{H}_2(C, G, X^*, \overline{Y}, K)$ et $\mathsf{sk} \stackrel{?}{=} \mathcal{H}_1(C, G, X^*, \overline{Y}, K)$. L'identifiant de session est défini par la concaténation des messages interceptés sur le canal, c.à.d. $C, G, X^*, \overline{Y}, K$. Notre protocole est composé de quatre échanges de messages entre le client, la passerelle et le serveur comme dans [2]. Une simple modification du protocole [2] décrit précédemment nous permettra de garantir l'anonymat du client vis à vis de la passerelle : après réception de la valeur $X^\star$ et de l'identité de l'émetteur C, la passerelle se contente de transférer au serveur les deux éléments $X^\star$ et C envoyés par le client sans calcul additionnel. La passerelle choisit alors l'exposant y et calcule $\overline{Y}$ et K en même temps. Afin de garantir la sécurité associée à la notion de *fraîcheur* dans le sens où nous l'avons définie, nous avons besoin de

deux preuves de connaissance (à divulgation nulle de connaissance) de logarithme discret des éléments h et $\overline{Y}$ en bases g et h respectivement (on pourra se référer au paragraphe 7.2.3 partie I consacré entre autre signatures de connaissance). Nous donnons une description complète du protocole GPAKE figure 7.2.

7.3.2 Sécurité de GPAKE

Résultat de sécurité

Grâce à l'hypothèse $\mathsf{PCDDH}_{\mathbb{G}}$, nous pouvons prouver la sécurité du protocole, dans un modèle plus fort en considérant une notion de *fraîcheur* plus large que celle usuellement considérée : cette notion couvre de manière presque évidente la *forward-secrecy*, en permettant certaines sessions de rester *fraîches*, même après une éventuelle corruption d'un ou plusieurs participants. Par ailleurs, nous conservons les sessions pour lesquelles l'adversaire ne fait que des *transferts* de messages comme sessions *fraîches*. Nous obtenons le résultat suivant que nous démontrons juste à la suite :

Théorème 12 (Sécurité) *Considérons le protocole 7.2 défini pour un groupe $\mathbb{G}$ d'ordre premier q, avec $\mathcal{D}$ un dictionnaire de taille N. Considérons un adversaire $\mathcal{A}$ capable d'initier des exécutions concurrentes du protocole, mais aussi de corrompre n'importe quel participant (de manière non-adaptative[2]) Si $\mathcal{A}$ initie moins de q_{send} sessions générés* via *des requêtes* Send*, nous avons alors :*

$$\mathsf{Adv}^{\mathsf{ake}}_{\mathcal{A}}(\lambda) \leqslant \frac{12 q_{\mathsf{send}}}{N} + \mathsf{negl}(\lambda).$$

Notons que les requêtes Execute ne déterminent pas le résultat précédent. En effet, la fuite d'information à l'issue de ces requêtes est négligeable.

Analyse de sécurité

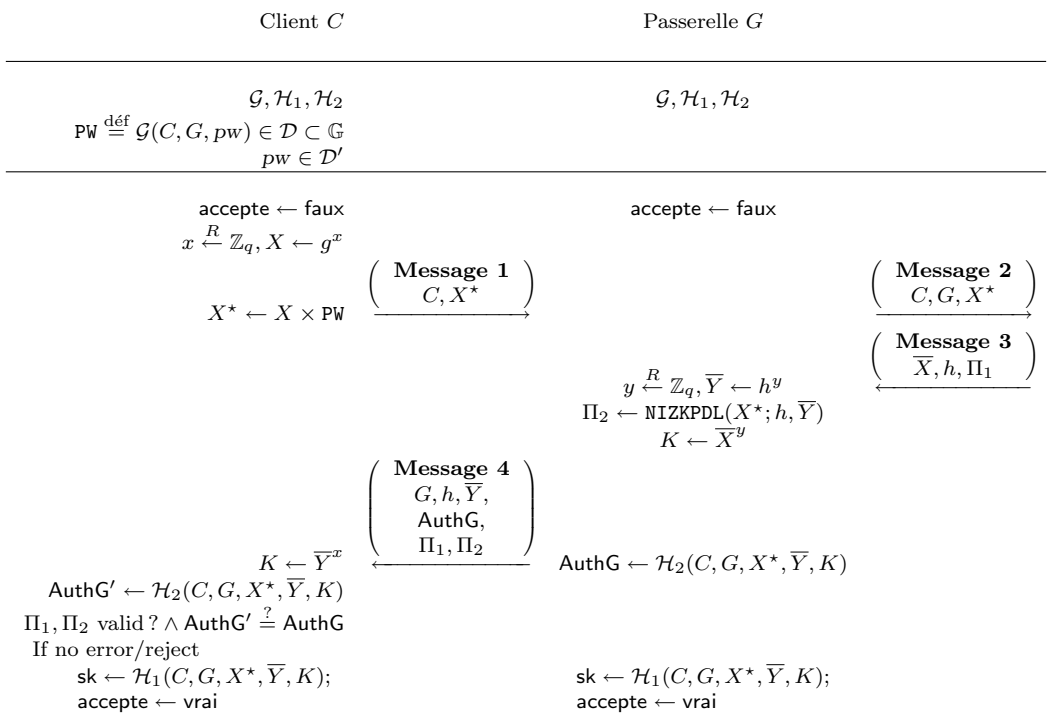
Démonstration: Dans la preuve du résultat précédent, nous nous intéressons à l'événement S_n qui a lieu si l'adversaire devine le bit b défini pour la requête Test, plus exactement si l'événement $b = b'$ a lieu. On considère une séquence de jeux, que nous allons modifier progressivement de manière à réduire l'attaque du jeu réel en un algorithme contre un problème calculatoirement difficile. On note Δ_n le distance entre deux jeux consécutifs $\mathbf{G}_n$ et $\mathbf{G}_{n+1}$. On commence par le jeu réel. On pourra remarquer que même si le canal entre les *passerelles* et le serveur est confidentiel, nous serons, malgré tout capable de simuler les communications. L'adversaire peut les mémoriser et ne vérifier qu'à postériori leur validité, et notamment lorsque l'une des passerelles sera corrompue.

Game $\mathbf{G}_0$: il s'agit du protocole réel, dans le modèle de l'oracle aléatoire. Par définition, on a :

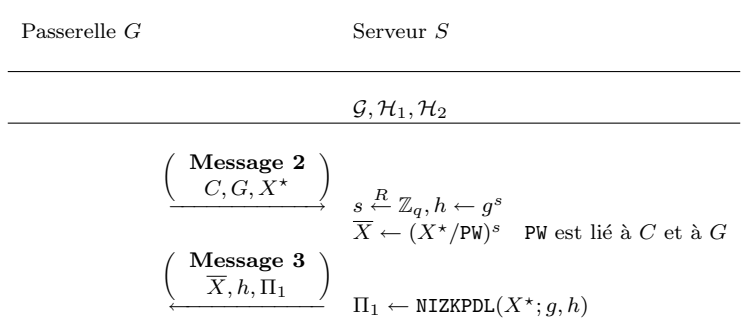
$$\mathsf{Adv}^{\mathsf{ake}}_{\mathsf{GPAKE},\mathcal{A}}(\lambda) = 2\Pr[\mathsf{S}_0] - 1.$$

Game $\mathbf{G}_1$: dans ce jeu, nous simulons les fonctions de hachage (modélisées par des oracles aléatoires) $\mathcal{G}$, $\mathcal{H}_1$ et $\mathcal{H}_2$, mais aussi deux fonctions de hachage additionnelles $\mathcal{H}'_1, \mathcal{H}'_2 : \mathcal{U}^2 \times \mathbb{G}^2 \to \{0,1\}^\ell$, qui apparaîtront plus tard, dans l'analyse de sécurité. Nous définissons les listes $\Lambda_{\mathcal{H}'}, \Lambda_{\mathcal{H}}$ et $\Lambda_{\mathcal{G}}$ initialement vides. La simulation des fonctions de

[2] Ce qui signifie qu'aucun participant ne peut être corrompu au milieu d'une session, mais seulement avant qu'elle ne débute.



Le communication entre le client et la passerelle se fait *via* un canal ni confidentiel, ni authentifié.



Le communication entre la passerelle et le serveur se fait *via* un canal sécurisé (assurant la confidentialité, l'intégrité et l'identification de l'origine des messages).

FIG. 7.2 – Notre nouveau protocole GPAKE

hachage est définie de la manière suivante :

- si le registre (n, q, r) apparaît dans la liste, on répond à la requête $\mathcal{H}_n(q)$ (resp. $\mathcal{H}'_n(q)$), $\Lambda_{\mathcal{H}}$ (resp. $\Lambda_{\mathcal{H}'}$), par r ; sinon on choisit un élément aléatoire $r \in \{0,1\}^{\ell}$ que nous renvoyons comme réponse, et nous ajoutons (n, q, r) à la liste $\Lambda_{\mathcal{H}}$ (resp. $\Lambda_{\mathcal{H}'}$) ;
- si le registre (q, r) apparaît dans la liste $\Lambda_{\mathcal{G}}$, on répond à une requête de hachage $\mathcal{G}$ par r ; sinon, on choisit un élément aléatoire dans $\mathbb{G}$ que nous ajoutons dans la liste $\Lambda_{\mathcal{G}}$.

Nous simulons les requêtes Send et Execute, de la même manière que l'auraient fait de réelles instances dans le protocole GPAKE. Nous simulons également les requêtes Reveal et Test comme définies par le modèle de sécurité :

Simulation des requêtes Send à C : on répond à une requête Send pour l'instance C de la manière suivante :

- on répond à une requête $\mathsf{Send}(C; \mathsf{INIT})$ en appliquant les règles suivantes :

  ▶**Rule C1**$^{(1)}$

  | choisir un exposant aléatoire $x \in \mathbb{Z}_q$, calculer $X = g^x$ et $X^\star = X \times \mathtt{PW}$.

  $(C, X^\star)$ constitue alors la réponse de cette requête et l'instance passe à un état d'acceptation.
- on répond à une requête $\mathsf{Send}(C, G; h, \overline{Y}, \mathsf{AuthG}, \Pi_1, \Pi_2)$ en appliquant les règles suivantes :

  ▶**Rule C2**$^{(1)}$

  | calculer $K = \overline{Y}^x$ et $\mathsf{AuthG}' = \mathcal{H}_2(C, G, X^\star, \overline{Y}, K)$;
  | si Π_1 et Π_2 sont valides et si $\mathsf{AuthG} = \mathsf{AuthG}'$,
  | on calcule alors $\mathsf{sk} = \mathcal{H}_1(C, G, X^\star, \overline{Y}, K)$, sinon on rejette.

  L'état du flag fresh_C est défini paragraphe 6.2.2. Finalement, C accepte si tout est correct, et termine dans tous les cas.
- Tous les autres cas sont ignorés.

Simulation des requêtes Send à G : on répond à une requête posée à une instance G de la manière suivante :

- on répond à une requête $\mathsf{Send}(G, C; C, X^\star)$ en transférant simplement le message $(C, G, X^\star)$.
- On répond à une requête $\mathsf{Send}(G, S; \overline{X}, h, \Pi_1)$ en appliquant les règles suivantes :

  ▶**Rule G1**$^{(1)}$

  | si Π_1 n'est pas valide, abandonner la simulation ; sinon
  | choisir un élément $y \in \mathbb{Z}_q$. Calculer $\overline{Y} = h^y$ et $K = \overline{X}^y$;
  | construire $\Pi_2 = \mathtt{NIZKPDL}(X^\star; h, \overline{Y})$, étant donné y.

  ▶**Rule G2**$^{(1)}$

  | calculer $\mathsf{AuthG} = \mathcal{H}_2(C, G, X^\star, \overline{Y}, K)$ et $\mathsf{sk} = \mathcal{H}_1(C, G, X^\star, \overline{Y}, K)$.

  L'état du flag fresh_G est défini paragraphe 6.2.2. Alors, G accepte et termine.
  On répond à cette requête par $(G, h, \overline{Y}, \mathsf{AuthG}, \Pi_1, \Pi_2)$.
- Tous les autres cas sont ignorés.

Simulation des requêtes Send à S : on répond à une requête Send posée à une instance S en appliquant les règles suivantes :

– pour une requête $\mathsf{Send}(S, G; C, G, X^\star)$, on applique les règles suivantes :

►**Rule S**$^{(1)}$

choisir un exposant $s \xleftarrow{R} \mathbb{Z}_q$ et calculer $h = g^s$;
calculer $\overline{X} = (X^\star/\mathtt{PW})^s$, où $\mathtt{PW}$ est le mot de passe de C ;
construire $\Pi_1 = \mathtt{NIZKPDL}(X^\star; g, h)$ (en choisissant s).

on répond alors à cette requête par $(\overline{X}, h, \Pi_1)$

– Tous les autres cas sont ignorés.

Simulation des autres requêtes :

– on répond à une requête $\mathsf{Execute}(C_i^r, G_j^s, S^t)$ en appliquant successivement la simulation des requêtes Send de la manière suivante :
$(C, X^\star) \leftarrow \mathsf{Send}(C; \mathsf{INIT})$, en faisant appel à $\mathbf{EC1}^{(1)}$ plutôt qu'à $\mathbf{C1}^{(1)}$;
$(C, G, X^\star) \leftarrow \mathsf{Send}(G, C; C, X^\star)$; $(\overline{X}, h, \Pi_1) \leftarrow \mathsf{Send}(S, G; C, G, X^\star)$,
en utilisant $\mathbf{ES}^{(1)}$ à la place de $\mathbf{S}^{(1)}$;
$(G, h, \overline{Y}, \mathsf{AuthG}, \Pi_1, \Pi_2) \leftarrow \mathsf{Send}(G, S; \overline{X}, h, \Pi_1)$, en faisant appel à $\mathbf{EG1}^{(1)}$ et $\mathbf{EG2}^{(1)}$ à la place de $\mathbf{G1}^{(1)}$ et $\mathbf{G2}^{(1)}$ respectivement ;
et $\leftarrow \mathsf{Send}(C, G; h, \overline{Y}, \mathsf{AuthG}, \Pi_1, \Pi_2)$, en faisant appel à $\mathbf{EC2}^{(1)}$ à la place de $\mathbf{C2}^{(1)}$.
On retourne alors le *transcript* $(C, G, X^\star), (\overline{X}, h, \Pi_1), (G, h, \mathsf{AuthG}, \Pi_1, \Pi_2)$.

– Une requête $\mathsf{Reveal}(U)$ retourne la clé de session si l'instance a déjà accepté et si aucune requête Test ne lui a été posée. Sinon, on renvoie $\perp$ comme réponse.

– Une requête $\mathsf{Test}(U)$ retourne la même valeur aléatoire $\mathsf{sk} \in \mathsf{SK}$, si $b = 0$. Sinon, elle renvoie $\mathsf{Reveal}(U)$.

Nous pouvons remarquer facilement que ce jeu est parfaitement indistinguable du protocole réel.

Game $\mathbf{G}_2$: de manière à simplifier l'analyse, on supprime les jeux pour lesquels des collisions apparaissent :

– collisions sur les sorties de $\mathcal{G}$
– collisions sur les sorties de $\mathcal{H}_1$ et $\mathcal{H}_2$
– collisions sur le *transcript* partiel $(C, G, X^\star, \overline{Y})$

$$\Delta_2 \leqslant \frac{q_{\mathsf{session}}^2 + q_{\mathcal{G}}^2}{2q} + \frac{q_{\mathcal{H}}^2}{2^\ell} = \mathsf{negl}(),$$

où q_{session} est le nombre de sessions et $q_{\mathcal{G}}$, $q_{\mathcal{H}}$ le nombre de requêtes s aux oracles $\mathcal{G}$ et $\mathcal{H}_1$ et $\mathcal{H}_2$ respectivement.

Game $\mathbf{G}_3$: dans ce jeu, nous considérons les attaques triviales dans lesquelles l'adversaire essaie de construire un authentifiant valide AuthG. Nous montrons dans un premier temps que s'il n'y a pas de requêtes $\mathcal{H}_2$ associé à AuthG, l'attaque échoue avec grande probabilité. D'autre part, on peut voir que si une telle requête existe, elle est unique, puisque nous avons supprimé les collisions de $\mathcal{H}_2$ (et $\mathcal{H}_1$) dans le jeu $\mathbf{G}_2$. Puis, nous modifions la simulation du client, lorsque personne n'est corrompue : ce dernier reçoit un message non généré par un oracle puis rejette l'authentifiant.

►**Rule C2**$^{(3)}$

– si $\neg\mathsf{corrupt}_C \wedge \neg\mathsf{corrupt}_G \wedge \neg\mathsf{OG}(C,4)$, on rejette.
– on identifie dans $\Lambda_{\mathcal{H}}$ l'élément K tel que $\mathsf{AuthG}' = \mathcal{H}_2(C,G,X^\star,\overline{Y},K)$.
  – s'il n'y en a pas, on rejette.
  – si Π_1 et Π_2 sont valides et si $K = \overline{Y}^x$,
    on calcule $\mathsf{sk} = \mathcal{H}_1(C,G,X^\star,\overline{Y},K)$, sinon on rejette.

$\mathbf{G}_3$ est identique au jeu $\mathbf{G}_2$ sauf si un authentifiant valide est rejeté. Dans ce cas, les deux événements suivants peuvent avoir eu lieu :
– $\mathsf{BadReject\text{-}NoAskH}$: la requête de hachage n'a pas été posée, mais l'authentifiant est valide ;
– $\mathsf{BadReject\text{-}Uncorrupted}$: personne n'est corrompue, mais l'adversaire tente de construire un authentifiant valide (sans l'aide du serveur d'authentification), puisqu'il ne peut poser aucune requête, ni intercepter la communication du serveur avec la passerelle, compte tenu de la présence du canal sécurisé.

Alors, $\Delta_3 \leqslant \Pr[\mathsf{BadReject\text{-}Uncorrupted}_3] + \Pr[\mathsf{BadReject\text{-}NoAskH}_3]$. Nous considérerons l'événement $\mathsf{BadReject\text{-}Uncorrupted}_3$ plus tard dans le preuve : en réalité, nous ne sommes pas encore capable d'évaluer la probabilité que cet événement ait lieu. Pour l'événement $\mathsf{BadReject\text{-}NoAskH}_3$, il est facile de voir que :

$$\Pr[\mathsf{BadReject\text{-}NoAskH}_3] \leqslant \frac{q_{\mathsf{send}}}{2^l} = \mathsf{negl}(),$$

où q_{send} est le nombre de requêtes Send posé à C.
Remarquons simplement que l'événement $\mathsf{BadReject\text{-}Uncorrupted}_3$ a lieu pour au plus un mot de passe par session. En effet, il se peut que le t-uplet $(g, X^\star/\mathtt{PW}, \overline{Y}, K)$ appartienne à $\mathcal{L}_{\mathsf{CDH}_{\mathbb{G}}}$, où K est issue de la requête $\mathcal{H}_2$ pour au plus une valeur de $\mathtt{PW}$. Cependant, puisque notre simulation utilise encore le mot de passe, nous ne pouvons pas encore affirmer que la probabilité est bornée par q_{send}/N : la vue de l'attaquant peut laisser fuir de l'information sur le mot de passe, ce qui peut l'aider à gagner plus facilement. Les prochains jeux ont justement pour but de simuler tous les oracles sans utiliser le mot de passe ; on pourra alors montrer que l'adversaire ne pourra avoir un biais sur la probabilité de deviner le mot de passe.

Game $\mathbf{G}_4$: dans ce jeu, nous considérons seulement les attaques générées *via* des requêtes $\mathsf{Execute}$. Pour ces sessions, nous modifions la simulation de ces requêtes en remplaçant les oracles $\mathcal{H}_1$ et $\mathcal{H}_2$ par les oracles secrets $\mathcal{H}'_1$ et $\mathcal{H}'_2$. Nous n'avons plus besoin de calculer la clé Diffie-Hellman K, puisque l'authentifiant et la clé de session en sont complètement indépendants. Voyons comment simuler les requêtes $\mathsf{Execute}$ dans ce jeu :

▶**Rule EG1**$^{(4)}$
on choisit un exposant $y \in \mathbb{Z}_q$ puis on calcule $\overline{Y} = h^y$;
on simule $\Pi_2 = \mathtt{NIZKPDL}(X^\star; h, \overline{Y})$, sans utiliser y ;

▶**Rule EG2**$^{(4)}$
on calcule $\mathsf{AuthG} = \mathcal{H}_2{}'(C,G,X^\star,\overline{Y})$ et $\mathsf{sk} = \mathcal{H}_1{}'(C,G,X^\star,\overline{Y})$;

▶**Rule EC2**$^{(4)}$
on calcule $\mathsf{sk} = \mathcal{H}'_1{}(C,G,X^\star,\overline{Y})$.

Les jeux $\mathbf{G}_4$ et $\mathbf{G}_3$ sont indistinguables sauf si $\mathcal{A}$ pose les requêtes de hachage $\mathcal{H}_1$ ou $\mathcal{H}_2$ sur certain(s) *transcript(s)* $C\|G\|X^\star\|\overline{Y}^\star\|K$ issu(s) de l'exécution du protocole. Puisque

nous avons restreint notre analyse aux attaques initiées *via* des requêtes Execute, nous appelons cet événement AskHE. Pour tout événement Ev, nous avons :

$$|\Pr[\mathsf{Ev}_4] - \Pr[\mathsf{Ev}_3]| \leqslant \Pr[\mathsf{AskHE}_4].$$

Si l'événement AskHE_4 a lieu, alors, pour certains *transcript(s)* obtenus *via* des requêtes Execute, $\mathcal{A}$ peut savoir si le simulateur a utilisé un oracle secret ou public. En outre, cela signifie que le t-uplet $(X^\star, \overline{Y}, K)$, où $K = \mathsf{CDH}_{\mathbb{G}}(X^\star/\mathtt{PW}, \overline{Y})$ apparaît dans la liste $\Lambda_{\mathcal{H}}$, ce qui signifie que $\mathcal{A}$, pourtant limité à une puissance de calcul polynomiale peut résoudre le problème CDH.
De manière à prouver ce résultat, nous considérons un jeu auxiliaire, dans lequel nous ne faisons aucune modification sur la vue de l'attaquant : nous modifions la simulation seulement si $\mathcal{A}$ interagit avec le protocole *via* des requêtes Execute. Il est important de préciser que le simulateur connaît le mot de passe, ce qui nous permet de simuler la requête INIT avec $X = Ag^\alpha$ et $\overline{Y} = Bg^\beta$, où $A = g^\alpha$ et $B = g^\beta$ sont deux entrées aléatoires du $\mathsf{CDH}_{\mathbb{G}}$.

Game $\mathbf{G}_{4.1}$:

►**Rule EC1**$^{(4.1)}$
choisir un exposant $\alpha \in \mathbb{Z}_q$,
calculer $X = Ag^\alpha$ et $X^\star = X \times \mathtt{PW}$.

►**Rule EG1**$^{(4.1)}$
Choisir un exposant aléatoire $\beta \in \mathbb{Z}_q$
et calculer $\overline{Y} = Bg^\beta$.
Simuler $\Pi_2 = \mathtt{NIZKPDL}(X^\star; h, \overline{Y})$, sans utiliser y.

Si l'événement AskHE a lieu, cela signifie que $\mathcal{A}$ parvient à extraire la valeur $\mathsf{CDH}_{\mathbb{G}}$ de A et B dans $\Lambda_{\mathcal{H}}$:

$$K = \mathsf{CDH}_{\mathbb{G}}(Ag^\alpha, Bg^\beta) = \mathsf{CDH}_{\mathbb{G}}(A, B) \times B^\alpha A^\beta g^{\alpha\beta}.$$

Nous obtenons alors $\mathsf{CDH}_{\mathbb{G}}(A, B) = K/B^\alpha A^\beta g^{\alpha\beta}$:

$$\Delta_4 \leqslant \Pr[\mathsf{AskHE}_4] \leqslant q_{\mathcal{H}} \times \mathsf{Succ}^{\mathsf{cdh}}(t') = \mathsf{negl}(),$$

où $t' = t + (2q_{\mathsf{execute}} + 3)\tau$ avec τ le temps de calcul d'une exponentiation dans le groupe et où q_{execute} est le nombre de requêtes Execute.
Notons que pour ce jeu, les mots de passe n'interviennent pas dans les sessions simulées *via* des requêtes Execute, ni pour le client, ni pour les *passerelles*, puisque ces deux participants ne calculent plus l'authentifiant, ni la clé de session. Cependant, le serveur utilise encore le mot de passe pour le calcul de h et $\overline{X}$.

Game $\mathbf{G}_5$: Dans ce jeu, nous simulons le serveur sans utiliser le mot de passe pour les sessions générées *via* des requêtes Execute :

►**Rule ES**$^{(5)}$
on choisit un exposant $s \in \mathbb{Z}_q$ et on calcule $h = g^s$;
on choisit un exposant $t \in \mathbb{Z}_q$ et on calcule $\overline{X} = g^t$;
on simule $\Pi_1 = \mathtt{NIZKPDL}(X^\star; g, h)$, sans utiliser s.

Les jeux $\mathbf{G}_5$ et $\mathbf{G}_4$ sont identiques sauf pour la simulation du serveur. Montrons que cette différence ne peut être détectée que s'il existe un attaquant contre le problème $\mathsf{DDH}_{\mathbb{G}}$.

Game $\mathbf{G}_{5.1}$: donnons-nous un triplet $\mathsf{CDH}_{\mathbb{G}}$, (A, B, C). La simulation des requêtes Execute est désormais définie de la manière suivante :

▶**Rule EC1**$^{(5.1)}$
on choisit des exposants aléatoires $x_1, x_2 \in \mathbb{Z}_q$ puis on calcule $X = g^{x_1} A^{x_2}$ et $X^\star = X \times \mathtt{PW}$;

▶**Rule ES**$^{(5.1)}$
on choisit y_1 et y_2 dans $\mathbb{Z}_q$ et on calcule $h = g^{y_1} B^{y_2}$;
et $\overline{X} = C^{x_2 y_2} A^{x_2 y_1} B^{x_1 y_2} g^{x_1 x_2}$;
on simule $\Pi_1 = \mathtt{NIZKPDL}(X^\star; g, h)$.

Il est facile de voir que cette simulation est parfaitement identique à celle du jeu $\mathbf{G}_4$.

Game $\mathbf{G}_{5.2}$: à présent, nous modifions juste l'entrée (A, B, C), en la remplaçant par une instance aléatoire : la simulation est alors parfaitement identique à celle du jeu $\mathbf{G}_5$ et alors $\Delta_5 \leqslant \mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(t') = \mathsf{negl}()$, où $t' = t + 8 q_{\mathsf{execute}} \tau$ et où q_{execute} est le nombre de requêtes Execute.

Game $\mathbf{G}_6$: dans ce jeu, nous restreignons notre analyse aux sessions pour lesquelles l'attaquant fait de simples transferts de messages issus de requêtes Send. Ce sont des exécutions passives, similaires à celles obtenues à partir de requêtes Execute, mais nous ne pouvons pas savoir dés le début si le *transcript* sera constitué de simples transferts ou non. Nous voulons montrer que l'adversaire ne peut pas connaître la clé de session, sauf s'il est capable casser le problème CDH.
Nous modifions la simulation de telle sorte à ce que que la clé soit calculée par des oracles secrets $\mathcal{H}'_1$ et $\mathcal{H}'_2$ et non plus par $\mathcal{H}_1$ et $\mathcal{H}_2$, comme précédemment :

▶**Rule G1**$^{(6)}$
on vérifie la validité de Π_1 ; si Π_1 n'est pas valide, on abandonne la simulation et sinon ;
– si $\neg\mathsf{corrupt}_C \wedge \neg\mathsf{corrupt}_G \wedge \mathsf{OG}(G, 1)$,
on choisit un exposant $y \xleftarrow{R} \mathbb{Z}_q$ et on calcule $\overline{Y} = h^y$;
– sinon, on choisit un exposant $y \xleftarrow{R} \mathbb{Z}_q$ et on calcule $\overline{Y} = h^y$ et $K = \overline{X}^y$;
on simule $\Pi_2 = \mathtt{NIZKPDL}(X^\star; h, \overline{Y})$, sans utiliser y.

▶**Rule G2**$^{(6)}$
– si $\neg\mathsf{corrupt}_C \wedge \neg\mathsf{corrupt}_G \wedge \mathsf{OG}(G, 1)$, on calcule
$\mathsf{AuthG} = \mathcal{H}_2{}'(C, G, X^\star, \overline{Y})$ et $\mathsf{sk} = \mathcal{H}_1{}'(C, G, X^\star, \overline{Y})$.
– sinon, on calcule
$\mathsf{AuthG} = \mathcal{H}_2(C, G, X^\star, \overline{Y}, K)$ et $\mathsf{sk} = \mathcal{H}_1(C, G, X^\star, \overline{Y}, K)$

▶**Rule C2**$^{(6)}$
– si $\neg\mathsf{corrupt}_C \wedge \neg\mathsf{corrupt}_G \wedge \neg\mathsf{OG}(C, 4)$, on rejette.
– si $\neg\mathsf{corrupt}_C \wedge \neg\mathsf{corrupt}_G \wedge \mathsf{OG}(G, 1)$,
on calcule $\mathsf{AuthG}' = \mathcal{H}'_2(C, G, X^\star, \overline{Y})$;
si Π_1 et Π_2 sont valides, et si $\mathsf{AuthG}' = \mathsf{AuthG}$,
on calcule $\mathsf{sk} = \mathcal{H}'_1(C, G, X^\star, \overline{Y})$; sinon on rejette.
– on identifie dans $\Lambda_{\mathcal{H}}$ l'élément K tel que $\mathsf{AuthG}' = \mathcal{H}_2(C, G, X^\star, \overline{Y}, K)$.
– s'il n'y en a pas, on rejette.
– si Π_1 et Π_2 sont valides et si $K = \overline{Y}^x$,
on calcule alors $\mathsf{sk} = \mathcal{H}_1(C, G, X^\star, \overline{Y}, K)$, sinon on rejette.

La probabilité de tout événement dans $\mathbf{G}_6$ et $\mathbf{G}_5$ est la même, sauf si l'adversaire pose les requêtes $\mathcal{H}_1$ ou $\mathcal{H}_2$ pour certain(s) transcript(s) $C\|G\|X^\star\|\overline{Y}\|K$, où K est le Diffie-Hellman de $X^\star/\mathtt{PW}$ et $\overline{Y}$. Puisque nous nous concentrons au cas passif (c.à.d. les sessions

obtenues *via* des transferts de messages), nous appelons cet événement AskHF_6. Pour tout événement Ev, nous avons :

$$|\Pr[\mathsf{Ev}_6] - \Pr[\mathsf{Ev}_5]| \leqslant \Pr[\mathsf{AskHF}_6].$$

Comme au jeu $\mathbf{G}_4$, on peut montrer que si l'événement AskHF_6 a lieu, nous pouvons résoudre le $\mathsf{CDH}_{\mathbb{G}}$. Pour cela, nous considérons un jeu auxiliaire dans lequel nous ne modifions pas la vue de l'adversaire, mais seulement la simulation de la passerelle et du client lorsque $\mathsf{OG}(G, 1)$.
Soulignons que le simulateur connaît le mot de passe utilisé pour la simulation du client. Dans notre cas, il n'est pas nécessaire de calculer la clé. Nous simulons $X^\star$ avec $Ag^\alpha \times \mathtt{PW}$ et Y par Bg^β, où $A = g^a$ et $B = g^b$ sont deux instances aléatoires du problème $\mathsf{CDH}_{\mathbb{G}}$.

Game $\mathbf{G}_{6.1}$:

►**Rule C1**$^{(6.1)}$
- si $\neg\mathsf{corrupt}_C \wedge \neg\mathsf{corrupt}_G$, on choisit un exposant aléatoire $\alpha \in \mathbb{Z}_q$,
  on calcule $X = Ag^\alpha$ et $X^\star = X \times \mathtt{PW}$.
- sinon, on choisit un exposant aléatoire $x \in \mathbb{Z}_q$,
  on calcule $X = g^x$ et $X^\star = X \times \mathtt{PW}$;

►**Rule G1**$^{(6.1)}$

on vérifie si la validité de Π_1 ;
si Π_1 n'est pas valide, on rejette et sinon ;
- si $\neg\mathsf{corrupt}_C \wedge \neg\mathsf{corrupt}_G \wedge \mathsf{OG}(G, 1)$, on choisit un exposant aléatoire $\beta \in \mathbb{Z}_q$
  et on calcule $\overline{Y} = (Bg^\beta)$;
- sinon, on choisit un exposant aléatoire $y \in \mathbb{Z}_q$, on calcule $\overline{Y} = h^y$ et $K = \overline{X}^y$.

on simule $\Pi_2 = \mathtt{NIZKPDL}(X^\star; h, \overline{Y})$, sans utiliser y.

Si l'événement AskHF a lieu, $\mathcal{A}$ peut extraire le Diffie-Hellman des éléments A et B dans $\Lambda_{\mathcal{H}}$, ce qui signifie que $\mathcal{A}$ peut casser le problème $\mathsf{CDH}_{\mathbb{G}}$. Nous avons en effet :

$$K = \mathsf{CDH}_{\mathbb{G}}(Ag^\alpha, Bg^\beta) = \mathsf{CDH}_{\mathbb{G}}(Ag^\alpha, B)\mathsf{CDH}_{\mathbb{G}}(Ag^\alpha, g^\beta) = \mathsf{CDH}_{\mathbb{G}}(A, B)B^\alpha A^\beta g^{\alpha\beta}.$$

Nous obtenons alors $\mathsf{CDH}_{\mathbb{G}}(A, B) = K/B^\alpha A^\beta g^{\alpha\beta}$. On a donc :

$$\Delta_6 \leqslant \Pr[\mathsf{AskHF}_{6.1}] \leqslant q_{\mathcal{H}} \times \mathsf{Succ}^{\mathsf{cdh}}(t') = \mathsf{negl}(),$$

où $t' = t + (2q_{\mathsf{send}} + 3)\tau$ avec τ le temps de calcul d'une exponentiation dans le groupe, et q_{send} le nombre de requêtes Send.

Game $\mathbf{G}_7$: nous considérons désormais les sessions pour lesquelles la passerelle est corrompue : l'adversaire peut se faire passer pour le serveur auprès de la passerelle (puisque nous avons supposé une authentification symétrique entre les passerelles et le serveur). Intuitivement, l'adversaire ne connaît pas les mots de passe, le client rejettera alors les requêtes issues de telles sessions. C'est en fait comme si l'adversaire jouait le rôle de la passerelle sans que le serveur ne soit impliqué ; on notera $\mathsf{OGA}(S)$ pour caractériser les requêtes posées à l'oracle S et dont l'entrée a été générée par un oracle ;

►**Rule C2**$^{(7)}$

– si $\neg\mathsf{corrupt}_C \wedge \neg\mathsf{corrupt}_G \wedge \neg\mathsf{OG}(C,4)$, on rejette.
– if $\neg\mathsf{corrupt}_C \wedge \neg\mathsf{corrupt}_G \wedge \mathsf{OG}(G,1)$,
  on calcule $\mathsf{AuthG}' = \mathcal{H}'_2(C,G,X^\star,\overline{Y})$;
  si Π_1 et Π_2 sont valides et si $\mathsf{AuthG}' = \mathsf{AuthG}$,
  on calcule $\mathsf{sk} = \mathcal{H}'_1(C,G,X^\star,\overline{Y})$, sinon on rejette.
– si $\neg\mathsf{corrupt}_C \wedge (\neg\mathsf{OGA}(S) \vee \neg\mathsf{OG}(G,3))$, on rejette ;
– on identifie dans $\Lambda_{\mathcal{H}}$ l'élément K tel que : $\mathsf{AuthG}' = \mathcal{H}_2(C,G,X^\star,\overline{Y},K)$;
  – s'il n'y en a pas, rejeter,
  – si Π_1 et Π_2 sont valides et si $K = \overline{Y}^x$,
    on calcule alors $\mathsf{sk} = \mathcal{H}_1(C,G,X^\star,\overline{Y},K)$, sinon on rejette.

Comme pour l'analyse du jeu $\mathbf{G}_3$, $\mathbf{G}_7$ est identique au jeu $\mathbf{G}_6$ sauf si l'adversaire essaie de jouer le rôle du serveur, et devine le bon mot de passe. Dans ce cas, nous rejetterons à tort un authentifiant valide ; nous notons $\mathsf{BadReject\text{-}C\text{-}Uncorrupted}_7$ un tel événement. Cet événement se produit avec probabilité négligeable puisque l'adversaire ne connaît pas le bon mot de passe (ici, l'analyse est restreinte aux sessions pour lesquelles le client n'est pas corrompu). Pour tout événement Ev, on a :

$$|\Pr[\mathsf{Ev}_7] - \Pr[\mathsf{Ev}_6]| \leqslant \Delta_7 \leqslant \Pr[\mathsf{BadReject\text{-}C\text{-}Uncorrupted}_7].$$

Notons qu'un tel événement ne se produit que pour un mot de passe pour chaque session, puisque l'adversaire ne peut retourner un quadruplet $\mathsf{CDH}_{\mathbb{G}}$, $(g, X^\star/\mathtt{PW}, h, \overline{X})$ que pour un seul mot de passe $\mathtt{PW}$. Puisque notre simulation utilise toujours le mot de passe, nous ne pouvons affirmer que cette probabilité est bornée par q_{send}/N, où q_{send} est le nombre de requêtes Send posés au client. Les prochains jeux nous permettront d'aboutir à cette conclusion.

Game $\mathbf{G}_8$: désormais, nous considérons les attaques pour lesquelles $\mathcal{A}$ interagit directement avec le serveur (la passerelle peut éventuellement être corrompue). Soulignons que lors de la simulation d'une requête INIT destinée au client, nous ne pouvons savoir si la valeur X^* générée sera transférée à S ou non. En effet, l'adversaire peut très bien envoyer une valeur différente. Et le client rejettera l'authentifiant avec grande probabilité mais comme la simulation n'est pas encore totalement indépendante du mot de passe (le serveur l'utilise encore), l'adversaire peut très bien obtenir de l'information sur le mot de passe du client. Nous montrons justement que l'attaquant n'obtient aucune information significative sur le mot de passe ; commençons par simuler le serveur sans l'utiliser.

►**Rule C1**$^{(8)}$
– si $\neg\mathsf{corrupt}_C$, on choisit un exposant aléatoire $x^\star \in \mathbb{Z}_q$, on calcule $X^\star = g^{x^\star}$;
– sinon, on choisit un exposant aléatoire $x \in \mathbb{Z}_q$, on calcule $X = g^x$ et $X^\star = X \times \mathtt{PW}$.

►**Rule S**$^{(8)}$
on choisit un exposant aléatoire $s \in \mathbb{Z}_q$ et on calcule $h = g^s$.
Puis,
– si $\neg\mathsf{corrupt}_C$, on choisit un exposant $t \xleftarrow{R} \mathbb{Z}_q$ et on calcule $\overline{X} = g^t$.
– sinon on calcule $\overline{X} = (X^\star/\mathtt{PW}_i)^s$, où $\mathtt{PW}_i$ est le mot de passe de C_i.
on simule $\Pi_1 = \mathtt{NIZKPDL}(X^\star; g, h)$.

►**Rule C2**$^{(8)}$

– si $\neg\mathsf{corrupt}_C \wedge \neg\mathsf{corrupt}_G \wedge \neg\mathsf{OG}(C,4)$, on rejette.
– si $\neg\mathsf{corrupt}_C \wedge \neg\mathsf{corrupt}_G \wedge \mathsf{OG}(G,1)$,
  on calcule $\mathsf{AuthG}' = \mathcal{H}'_2(C,G,X^\star,\overline{Y})$;
  si Π_1 et Π_2 sont valides, et si $\mathsf{AuthG}' = \mathsf{AuthG}$,
  on calcule alors $\mathsf{sk} = \mathcal{H}'_1(C,G,X^\star,\overline{Y})$, sinon on rejette ;
– si $\neg\mathsf{corrupt}_C \wedge (\neg\mathsf{OGA}(S) \vee \neg\mathsf{OG}(G,3))$, on rejette ;
– on identifie dans $\Lambda_{\mathcal{H}}$ l'élément K tel que $\mathsf{AuthG} = \mathcal{H}_2(C,G,X^\star,\overline{Y},K)$;
  – s'il n'y en a pas, on rejette l'authentifiant ;
  – si $\neg\mathsf{corrupt}_C$, si Π_1 et Π_2 sont valides et si $\overline{Y}^t = K^s$,
    on calcule alors $\mathsf{sk} = \mathcal{H}_1(C,G,X^\star,\overline{Y},K)$, sinon on rejette ;
  – sinon, si Π_1 et Π_2 sont valides et si $K = \overline{Y}^x$,
    on calcule alors $\mathsf{sk} = \mathcal{H}_1(C,G,X^\star,\overline{Y},K)$, sinon on rejette.

Les jeux $\mathbf{G}_8$ et $\mathbf{G}_7$ sont identiques sauf si le client n'est pas corrompu :
– $X^\star$ est généré sans utiliser le mot de passe (ce qui ne modifie pas la vue de l'attaquant),
– et le serveur simule h et $\overline{X}$ de manière indépendante, sans utiliser le mot de passe.
La simulation du client est modifiée en conséquence. Notre de but est de prouver que quelque soit le participant générant la valeur $X^\star$ impliquée dans une requête Send, l'adversaire ne peut distinguer cette nouvelle simulation (du serveur) de la précédente qu'avec un avantage négligeable.
Afin de prouver ce résultat, nous utilisons un argument hybride pour montrer que si l'avantage précédent est significatif alors nous pouvons casser le problème $\mathsf{PCDDH}_{\mathbb{G}}$. Nous construisons une séquence de $q_{\mathsf{send}}+1$ jeux, noté Hyb_k, pour $0 \leqslant k \leqslant q_{\mathsf{send}}$, où q_{send} est le nombre de requêtes Send. Dans chaque jeu Hyb_k, la i-ème (on ordonne selon le nombre de requêtes Send) session est définie de manière suivante :
– si $(i \leqslant k)$, les requêtes Send sont simulées comme au jeu $\mathbf{G}_8$.
– sinon, les requêtes Send sont simulées comme au jeu $\mathbf{G}_7$.
Définissons un jeu intermédiaire G_k, similaire au jeu Hyb_k :

►**Rule S**$^{(8,k)}$
– si $i \leqslant k$, on procède comme au jeu $\mathbf{G}_8$;
– si $i = k+1$, on envoie $(\mathcal{D}, X^\star)$ au challenger qui possède alors une entrée dans l'expérience $\mathsf{PCDDH}_{\mathbb{G}}$. Ce dernier répond par $(\overline{X}, h, \mathtt{PW})$;
  on retourne $\overline{X}$ et h comme défini précédemment par la sortie du challengeur PCDDH ;
  on simule $\Pi_1 = \mathtt{NIZKPDL}(X^\star; g, h)$;
– sinon, on procède exactement comme dans le jeu $\mathbf{G}_7$, en utilisant $\mathtt{PW}$.

►**Rule C2**$^{(8,k)}$

- si $i \leqslant k$, on procède comme au jeu $\mathbf{G}_8$;
- si $i = k+1$,
  - si $\neg\mathsf{corrupt}_C \wedge \neg\mathsf{corrupt}_G \wedge \neg\mathsf{OG}(C,4)$, on rejette l'authentifiant ;
  - si $\neg\mathsf{corrupt}_C \wedge \neg\mathsf{corrupt}_G \wedge \mathsf{OG}(G,1)$,
    on calcule $\mathsf{AuthG}' = \mathcal{H}'_2(C,G,X^\star,\overline{Y})$.
    Si Π_1 et Π_2 sont valides et si $\mathsf{AuthG}' = \mathsf{AuthG}$,
    on calcule alors $\mathsf{sk} = \mathcal{H}'_1(C,G,X^\star,\overline{Y})$; sinon on rejette ;
  - si $\neg\mathsf{corrupt}_C \wedge (\neg\mathsf{OGA}(S) \vee \neg\mathsf{OG}(G,3))$, on rejette ;
  - on identifie dans $\Lambda_{\mathcal{H}}$ l'élément K tel que $\mathsf{AuthG} = \mathcal{H}_2(C,G,X^\star,\overline{Y},K)$;
    - s'il n'y en a pas, on rejette ;
    - si $\neg\mathsf{corrupt}_C$;
      - si Π_1 ou Π_2 ne sont pas valides, on rejette ;
      - on extrait y de Π_2 (avec un *rewind*, ou on renvoie "echec") ;
      - si $K = \overline{X}^y$,
        on calcule $\mathsf{sk} = \mathcal{H}_1(C,G,X^\star,\overline{Y},K)$, sinon on rejette ;
    - sinon, si Π_1 et Π_2 sont valides et si $K = \overline{Y}^x$, on calcule $\mathsf{sk} = \mathcal{H}_1(C,G,X^\star,\overline{Y},K)$, sinon on rejette l'authentifiant ;
- sinon on procède comme au jeu $\mathbf{G}_7$.

$\mathcal{A}'$ retourne sa réponse $d' = (b = b')$ pour d, où b' est la réponse de $\mathcal{A}$ pour le bit b ; rappelons que nous considérons l'événement S. Mais l'événement $d' = 1$ se produira si et seulement si on peut détecter un événement Ev quelconque.
En effet, ce jeu est soit identique au jeu défini par l'expérience Hyb_k si $d = 0$, ou à l'expérience Hyb_{k+1} sinon, grâce à l'extraction de y :
- si $d = 0$, $(g, X^\star/\mathtt{PW}, h, \overline{X})$ est un quadruplet $\mathsf{CDH}_{\mathbb{G}}$, le serveur est simulé comme défini par le jeu $\mathbf{G}_7$. Alors, du point de vue du client, tester $K = \overline{X}^y$ est équivalent à tester $K = \overline{Y}^x$;
- si $d = 0$, il est facile de voir que le serveur est simulé comme au jeu $\mathbf{G}_8$, puisque $(g, X^\star/\mathtt{PW}, h, \overline{X})$ est un quadruplet aléatoire. Pour le client, nous vérifions seulement que l'adversaire calcule $\overline{Y}$ et K, avec un exposant y commun.

Les jeux Hyb_k et Hyb_{k+1} sont indistinguables sauf si l'adversaire a un avantage non négligeable à deviner le bit b (ou à distinguer un événement quelconque Ev dans les deux jeux). Si c'est la cas, l'attaquant $\mathcal{A}'$ devine le bit d avec un avantage non négligeable :

$$\begin{aligned}\mathsf{Adv}^{\mathsf{pcddh}}_{\mathbb{G}}(t) \geqslant \mathsf{Adv}^{\mathsf{pcddh}}_{\mathbb{G}}(\mathcal{A}') &= \Pr[d'=1|d=1] - \Pr[d'=1|d=0] \\ &= \Pr_{k+1}[\mathsf{Ev}] - \Pr_{k}[\mathsf{Ev}].\end{aligned}$$

Par conséquent, sous l'hypothèse $\mathsf{PCDDH}_{\mathbb{G}}$ (voir définition 7.2.3), on a :

$$\Delta_8 \leqslant \frac{q_{\mathsf{send}}}{N} + \mathsf{negl}().$$

Remarquons que l'ordre est déterminé par les requêtes Send. Il est légitime de se demander si la preuve ne s'applique que pour des exécutions non concurrentes. Avant que cette requête ne soit posée, la simulation des messages est la même dans tous les jeux hybrides, puisque nous nous restreignons aux corruptions non adaptatives. Par conséquent, notre analyse tient également compte des exécutions concurrentes.

Game $\mathbf{G}_9$: dans ce jeu, nous examinons les sessions pour lesquelles le client est corrompu ; ce qui signifie que l'attaquant (mais aussi le simulateur) connaît le mot de passe du client.

Pour ces sessions, on remplace les oracles publiques $\mathcal{H}_1$ et $\mathcal{H}_2$ par les oracles secrets $\mathcal{H}'_1$ et $\mathcal{H}'_2$ dans les sessions susceptibles d'être fraîches ; dans ce cas :
– même si le client est corrompu, l'adversaire peut jouer le rôle du client. Si c'est la cas et si $\mathsf{OG}(G, 1)$, il se peut que la session reste *fraîche* ;
– cependant, si $\neg\mathsf{OG}(G, 1)$, nous savons que la session ne sera pas fraîche. Mais le protocole doit tout de même garantir que le client est capable de vérifier la validité de l'authentifiant, quelque soit l'émetteur.

Nous avons déjà considéré les cas triviaux au jeu $\mathbf{G}_3$, ceux pour lesquels l'adversaire construit un authentifiant sans poser de requêtes de hachage puis, après réception d'un message $(\overline{Y}, \mathsf{AuthG})$, le client extrait l'entrée $(C, G, X^\star, \overline{Y}, K)$ telle que $\mathsf{AuthG} = \mathcal{H}_2(C, G, X^\star, \overline{Y}, K)$. Si $\neg\mathsf{OG}(C, 4)$, nous devons nous assurer (dans le cas particulier on effectue les calculs par appels aux oracles publics) que l'oracle simulant le client est capable de rejeter les *transcripts* pour lesquels $(g, X^\star/\mathtt{PW}, \overline{Y}, K)$ n'est pas quadruplet $\mathsf{CDH}_{\mathbb{G}}$:

▶**Rule G1**$^{(9)}$
on vérifie la validité de Π_1 ;
– si $\neg\mathsf{corrupt}_C \wedge \neg\mathsf{corrupt}_G$ ou si $\mathsf{OG}(S, 2) \wedge \mathsf{OG}(G, 3)$;
on choisit un exposant aléatoire $y \in \mathbb{Z}_q$ et on calcule $\overline{Y} = h^y$;
– sinon on choisit un exposant aléatoire $y \in \mathbb{Z}_q$ puis on calcule $\overline{Y} = h^y$ et $K = \overline{X}^y$;
puis on simule $\Pi_2 = \mathtt{NIZKPDL}(X^\star; h, \overline{Y})$.

▶**Rule G2**$^{(9)}$
– si $\neg\mathsf{corrupt}_C \wedge \neg\mathsf{corrupt}_G$ ou $\mathsf{OG}(S, 2) \wedge \mathsf{OG}(G, 3)$,
on calcule $\mathsf{AuthG} = \mathcal{H}'_2(C, G, X^\star, \overline{Y})$ et $\mathsf{sk} = \mathcal{H}'_1(C, G, X^\star, \overline{Y})$;
– sinon, on calcule $\mathsf{AuthG} = \mathcal{H}_2(C, G, X^\star, \overline{Y}, K)$;
et $\mathsf{sk} = \mathcal{H}_1(C, G, X^\star, \overline{Y}, K)$.

▶**Rule C2**$^{(9)}$
– si $\neg\mathsf{corrupt}_C \wedge \neg\mathsf{corrupt}_G \wedge \neg\mathsf{OG}(C, 4)$, on rejette ;
– si $\neg\mathsf{corrupt}_C \wedge (\neg\mathsf{OGA}(S) \vee \neg\mathsf{OG}(G, 3))$, on rejette ;
– si $\mathsf{OG}(G, 3) \wedge \mathsf{OG}(C, 4)$, on calcule $\mathsf{AuthG}' = \mathcal{H}'_2(C, G, X^\star, \overline{Y})$;
si Π_1 et Π_2 sont valides, et si $\mathsf{AuthG}' = \mathsf{AuthG}$,
on calcule alors $\mathsf{sk} = \mathcal{H}'_1(C, G, X^\star, \overline{Y})$, sinon on rejette ;
– on identifie dans $\Lambda_{\mathcal{H}}$ l'élément K tel que $\mathsf{AuthG} = \mathcal{H}_2(C, G, X^\star, \overline{Y}, K)$.
– s'il n'y en a pas, on rejette ;
– si $\neg\mathsf{corrupt}_C$,
si Π_1 et Π_2 sont valides et si $\overline{Y}^t \neq K^s$,
on calcule alors $\mathsf{sk} = \mathcal{H}_1(C, G, X^\star, \overline{Y}, K)$, sinon on rejette ;
– sinon, si Π_1 et Π_2 sont valides et si $K = \overline{Y}^x$,
on calcule alors $\mathsf{sk} = \mathcal{H}_1(C, G, X^\star, \overline{Y}, K)$, sinon on rejette.

La différence entre les deux jeux $\mathbf{G}_9$ et $\mathbf{G}_8$ est que, dans le premier cas, nous savons les oracles $\mathcal{H}_1$ et $\mathcal{H}_2$ et pour le second cas, nous avons utilisé les oracles $\mathcal{H}'_1$ et $\mathcal{H}'_2$ dans toutes les sessions dans lesquelles la clé de session n'est pas compromise (c.à.d. choisie par des oracles). Notre but est de montrer que la différence de probabilité de tout événement dans les deux jeux est négligeable. Nous définissons une séquence de jeux hybrides Hyb_k pour $0 \leqslant k \leqslant q_{\mathsf{send}}$. Ici, l'ordre est déterminé par les requêtes Send. Pour la i-ième session, nous avons :
– si $i \leqslant k$, on simule les requêtes Send comme nous l'avons fait dans le jeu $\mathbf{G}_9$;
– sinon, on simule les requêtes Send comme au jeu $\mathbf{G}_8$.

Nous voulons montrer que la différence de la probabilité de succès de $\mathcal{A}$ à casser la sécurité sémantique de la clé de session dans les jeux Hyb_k et Hyb_{k+1} est négligeable : l'adversaire pose la requête $\mathcal{H}_1$ et/ou $\mathcal{H}_2$ sur certains *transcripts* $C\|G\|X^\star\|\overline{Y}\|K$ qui ont été simulés en utilisant $\mathcal{H}'_1$ et/ou $\mathcal{H}'_2$ respectivement. Dans ce cas, l'adversaire trouve la clé Diffie-Hellman, $K = \mathsf{CDH}(X, \overline{Y}) = \overline{Y}^x$. On appelle cet événement AskHA_9, car désormais, certains participants peuvent être corrompus. Pour tout événement Ev, nous avons :

$$\left|\Pr_{k+1}[\mathsf{Ev}] - \Pr_{k}[\mathsf{Ev}]\right| \leqslant \Pr[\mathsf{AskHA}_9].$$

Afin de borner la probabilité de l'événement AskHA_k, nous réduisons le challenge de l'adversaire relatif aux jeux Hyb_k et Hyb_{k+1} au challenge d'un adversaire, $\mathcal{A}'$ défini par l'expérience CDH. Soit (A, B) une instance $\mathsf{CDH}_{\mathbb{G}}$.

►**Rule C1**$^{(9,k)}$
- si $(i \neq k)$, on simule les requêtes $\mathsf{Send}(C, \mathsf{INIT})$ comme dans le jeu $\mathbf{G}_8$=$\mathbf{G}_9$;
- sinon, si $i = k$
  - si $\neg\mathsf{corrupt}_C$, on choisit un exposant aléatoire $x^\star \in \mathbb{Z}_q$, et on calcule $X^\star = g^{x^\star}$;
  - sinon si $\mathsf{corrupt}_C \wedge \neg\mathsf{corrupt}_G$, on pose $X = A$, et on calcule $X^\star = X \times \mathtt{PW}$;
  - sinon on choisit un exposant aléatoire $x \in \mathbb{Z}_q$ et on calcule $X = g^x$, puis $X^\star = X \times \mathtt{PW}$.

►**Rule G1**$^{(9,k)}$
- si $i < k$, on procède comme dans le jeu $\mathbf{G}_9$;
- sinon, si $i = k$,
  - on vérifie l validité de Π_1 ;
  - puis,
    - si $\neg\mathsf{corrupt}_C \wedge \neg\mathsf{corrupt}_G$ ou $\mathsf{OG}(S, 2) \wedge \mathsf{OG}(G, 3)$, poser $\overline{Y} = B$;
    - sinon, on choisit un exposant $y \xleftarrow{R} \mathbb{Z}_q$ et on calcule $\overline{Y} = h^y$ et $K = \overline{X}^y$;
  - on construit $\Pi_2 = \mathtt{NIZKPDL}(X^\star; h, \overline{Y})$;
- sinon on procède comme au jeu $\mathbf{G}_8$

Si l'événement AskHA_k a lieu, nous pouvons alors extraire le valeur CDH des éléments A et B dans $\Lambda_{\mathcal{H}}$: $\Pr[\mathsf{AskHA}_k] \leqslant \mathsf{Succ}^{\mathsf{cdh}}(t)$. Nous avons donc,

$$\Delta_9 \leqslant q_{\mathsf{send}} \times \mathsf{Succ}^{\mathsf{cdh}}(t) = \mathsf{negl}().$$

Dans ce dernier jeu, nous remarquons deux choses : la première est que les clés sont calculées en utilisant une fonction secrète pour les sessions fraîches. L'attaquant ne peut donc pas distinguer ces clés de clés aléatoires.

$$\Pr[\mathsf{S}_9] = \frac{1}{2}.$$

De plus, le mot de passe n'est plus utilisé, sauf si le client est corrompu. Dans ce cas, nous pouvons tout simplement choisir le mot de passe lorsque la simulation en a réellement besoin, c.à.d. :
- lorsque le client est corrompu ;
- lorsque l'adversaire envoie sa réponse b' pour le bit b,

et on vérifie alors si les événements $\mathsf{BadReject\text{-}Uncorrupted}_9$ et $\mathsf{BadReject\text{-}C\text{-}Uncorrupted}_9$ ont lieu ou non. Par ailleurs, comme déjà précisé, ces événements ne peuvent engagés au plus qu'un seul mot de passe. Nous avons donc :

$$\Pr[\mathsf{BadReject\text{-}Uncorrupted}_9] \leqslant \frac{q_{\mathsf{send}}}{N} \qquad \Pr[\mathsf{BadReject\text{-}C\text{-}Uncorrupted}_9] \leqslant \frac{q_{\mathsf{send}}}{N}.$$

$$\Delta_2 = \mathsf{negl}() \quad \Delta_3 = \Pr[\mathsf{BadReject\text{-}Uncorrupted}_3] + \mathsf{negl}() \quad \Delta_4 \leqslant \Delta_5 = \mathsf{negl}()$$

$$\Delta_6 = \mathsf{negl}() \quad \Delta_7 = \Pr[\mathsf{BadReject\text{-}C\text{-}Uncorrupted}_7] \quad \Delta_8 \leqslant \frac{q_{\mathsf{send}}}{N} + \mathsf{negl}() \quad \Delta_9 = \mathsf{negl}()$$

Par conséquent,

$$\Delta_7 = \Pr[\mathsf{BadReject\text{-}C\text{-}Uncorrupted}_7] \leqslant \frac{2q_{\mathsf{send}}}{N} + \mathsf{negl}()$$

$$\Delta_3 = \Pr[\mathsf{BadReject\text{-}Uncorrupted}_3] + \mathsf{negl}() \leqslant \frac{q_{\mathsf{send}}}{N} + \frac{3q_{\mathsf{send}}}{N} + \mathsf{negl}()$$

$$\mathsf{S}_0 \leqslant \frac{1}{2} + \frac{q_{\mathsf{send}}}{N} + \frac{5q_{\mathsf{send}}}{N} + \mathsf{negl}() \leqslant \frac{1}{2} + \frac{6q_{\mathsf{send}}}{N} + \mathsf{negl}(),$$

où q_{send} est le nombre de sessions où l'adversaire est actif (c.à.d. interagit avec le protocole *via* des requêtes Send).

Remarque 15 *Dans la formule ci-dessus,* q_{send} *est le nombre de requêtes* Send *total : ces requêtes comprennent à la fois les requêtes où l'attaquant modifie le message en entrée et celles où il transfert le ou les message(s).*

□

Dans cette partie, nous avons présenté un protocole d'échange de clés par mot de passe à trois parties : un client se connecte à un serveur sans savoir s'il s'agit d'un serveur d'application ou d'authentification. Ce scénario est tout à fait pertinent puisqu'en pratique, la gestion d'un service d'authentification est en générale déléguée à plusieurs entités. Si le client 'a aucune information sur l'implémentation de l'infrasctucture d'authentifiaction, il est tout à fait légitime qu'il veuille conserver son anonymat : la notion de transparence ne s'en trouve que plus consolidée. De manière paradoxale, en vue de considérer l'anonymat du client (voir chapitre 8 section 8.1) vis à vis du serveur d'authentification, nous avons ajouté des preuves de connaisssance, affaiblissant la propriété de transparence. Par ailleurs, pour avoir une garantie d'anonymat, il est évident que les serveurs doivent être coopératif : nous avons étendu le modèle de Bellare *et al.* [17, 14], en considérant certaines sessions fraîches, en cas de corruptions (dans le cas où les participants jouent le protocole de manière honnête). Il reste encore à explorer les corruptions adaptatives dans le scénario à trois parties.

Chapitre 8

Introduction de l'Anonymat pour l'authentification : IB-PAKE

Dans cette partie, nous montrons comment combiner la notion d'*anonymat* avec les protocoles d'échange de clés à deux parties et trois parties de manière à garantir l'*anonymat* du client vis à vis du serveur. Nous considérerons que cette garantie d'anonymat s'établit par deux moyens : l'utilisation des $\mathcal{PIR}$ comme interface au protocole GPAKE [2] du chapitre précédent, et l'utilisation d'un schéma $\mathcal{IB}$-$\mathcal{KEM}$ possédant la propriété d'anonymat au sens KwrtA présentée à la partie précédente.

- Par le premier moyen, l'utilisateur a un accès confidentiel à des données liées à son identité ou mot de passe. Nous pouvons utiliser un protocole de $\mathcal{PIR}$ quelconque comme interface au protocole GPAKE pour garantir l'anonymat du client.
- Par le second moyen, l'utilisateur s'authentifie en utilisant son mot de passe comme identité. Nous montrons comment préserver la confidentialité de l'identité du client dans un protocole 2-PAKE : nous décrivons un protocole générique IB-PAKE qui utilise comme brique de base un schéma $\mathcal{IB}$-$\mathcal{KEM}$ possédant outre les propriétés de sécurité sémantique et d'anonymat traditionnel, le nouvelle propriété au sens KwrtA, que nous avons introduite au chapitre 4 partie II. Rappelons que le schéma présenté paragraphe 4.3.3 (notre candidat) dans ce même chapitre satisfait toutes ces conditions. Afin de garantir la sécurité du protocole IB-PAKE, nous avons besoin d'introduire une nouvelle notion de sécurité traduisant l'absence d'une certaine forme de malléabilité. Nous verrons que notre schéma vérifie également cette propriété.

8.1 Anonymat dans GPAKE

Pour certaines applications, il peut être primordial de préserver l'anonymat d'un acteur. Dans le cas contraire, pour les protocoles d'échange de clés, toutes les connexions peuvent être combinées de telle sorte à analyser le profil des utilisateurs. Pour des raisons de confidentialité, un client peut vouloir rendre ses connexions anonymes et non reliées. Une première solution serait d'utiliser des passerelles différentes ; mais le serveur d'authentification reste malgré tout unique, et on ne peut pas utiliser un serveur différent pour chaque connexion. En revanche, puisque le serveur est le seul participant à autoriser la connexion, nous pouvons très bien masquer l'identité du client vis à vis du serveur.

Le serveur peut être vu comme une base de données dynamique virtuelle : pour chaque requête d'autorisation d'accès, le serveur construit les réponses pour tous les clients possibles, tandis que la passerelle se charge d'être l'interface du client : la passerelle et le

serveur s'engagent dans un protocole de $\mathcal{PIR}$ [58], sans que le serveur n'apprenne une quelconque information sur l'identité du client. Remarquons qu'en utilisant un schéma $\mathcal{SPIR}$, nous assurons en plus de la confidentialité vis-à-vis du client, celle du serveur, puisque la passerelle n'apprend aucune information sur les autres mots de passe stockés dans la base de données.

Rappelons qu'un protocole de $\mathcal{PIR}$ est une primitive où l'efficacité est mesurée en terme de complexité de communication. Elle permet à un utilisateur de retrouver une chaîne de caractères dans une base de données de taille n sans laisser fuir d'information sur l'indice de la valeur demandée.

Une des propriétés intéressantes de notre protocole GPAKE est son implémentation efficace avec un $\mathcal{SPIR}$ quelconque : nous n'avons donc pas besoin de choisir ou de décrire un schéma spécifique ; nous pouvons tout simplement utiliser une requête comme un système de boîte noire.

Nous verrons qu'il est possible de diminuer considérablement la taille de la base de données virtuelle ; ce qui améliore l'implémentation de notre schéma en pratique et ce, quel que soit le schéma de $\mathcal{PIR}$ que nous utilisons : avec notre construction, chaque client possède un mot de passe indexé par i, assimilé au secret de la passerelle. Le serveur possède une base de données de taille n (où n est le nombre de clients) qui contient tous les mots de passe des clients.

Une manière triviale d'introduire l'anonymat du client vis-à-vis du serveur dans notre protocole est d'utiliser un $\mathcal{SPIR}$ en générant de manière dynamique une base de données pour chaque session : dés la réception d'une requête Send, avec $X^\star$ en entrée , le serveur calcule les réponses correspondant à chacun des messages $(C_i, G, X^\star)$, ces réponses sont donc calculées pour tous les clients possibles, puisque le serveur ne sait pas lequel d'entre eux interagit avec la passerelle. La base de données dynamique est constituée de tous les blocs $B_i = (g^{s_i}, (X^\star/pw_i)^{s_i}, \Pi_i)$. La passerelle fait appel à un protocole de $\mathcal{SPIR}$ afin d'obtenir le B_i correspondant à sa requête, tout en préservant l'anonymat du client. Cette transformation est générique. Cependant, un inconvénient majeur est sa complexité calculatoire ; celles-ci est très importante du fait des multiples preuves Π_i. Dans notre cas, nous pouvons très facilement améliorer l'efficacité, en effectuant les calculs en une seule étape (en effectuant même des pré-calculs), puis en envoyant $h = g^s$ et Π_1. Alors, la base de données est constituée de n entrées $B_i = (X^\star/pw_i)^s$; ce qui améliore de manière considérable le coût calculatoire, mais diminue également les contraintes d'espace.

8.2 Notion de non-malléabilité d'identités

Nous considérons une nouvelle notion pour les schémas $\mathcal{IB}$-$\mathcal{KEM}$: la non-malléabilité d'identités. L'idée est la suivante : étant donnés un chiffré c et l'identité de son destinataire ID, il doit être difficile de savoir si un autre destinataire ID′ pourrait déchiffrer ce même c.

Ceci se traduit par le fait que lorsqu'on *encapsule* une clé, on produit une clé de session éphémère pour un unique destinataire et non plusieurs clés correspondant à différentes identités pour un même chiffré.

8.2.1 Définition

Pour la notion de sécurité précédente, le jeu est défini de la manière suivante : étant donnés un paramètre de sécurité λ et un schéma $\mathcal{IB}$-$\mathcal{KEM}$ $\stackrel{\text{déf}}{=} \langle \mathsf{Setup}_{\mathsf{IBK}}, \mathsf{Extract}_{\mathsf{IBK}},$

$\mathsf{Encaps}_{\mathsf{IBK}}, \mathsf{Decaps}_{\mathsf{IBK}}\rangle$, nous générons les clés maîtres publique mpk et secrète msk du schéma $\mathcal{IB}$-$\mathcal{KEM}$. L'attaquant reçoit la clé mpk. Par ailleurs, nous supposons que l'attaquant a accès à un oracle d'extraction de clés et de déchiffrement. Il gagne s'il parvient à produire deux paires clé/identité et un chiffré associé à chacune de ces paires. Plus précisément, nous définissons le jeu de sécurité pour la notion de non-malléabilité d'identités de la manière suivante :

$$
\begin{array}{l}
\text{Expérience } \mathsf{Exp}^{\mathsf{id\text{-}nm}}_{\mathcal{IB}\text{-}\mathcal{KEM},\mathcal{A}}(\lambda) \\
\mathsf{IDSet} \leftarrow \emptyset; \mathsf{CSet} \leftarrow \emptyset; \\
(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}_{\mathsf{IBK}}(\lambda)\,; \\
((K_0, \mathsf{ID}_0), (K_1, \mathsf{ID}_1), c^*) \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{Extract}(),\mathcal{O}\mathsf{Decaps}()}(\mathtt{FIND}, \mathsf{mpk}) \\
\text{tels que } \mathsf{ID}_0, \mathsf{ID}_1 \notin \mathsf{IDSet}, c^* \notin \mathsf{CSet}
\end{array}
$$

Les deux accès aux oracles $\mathcal{O}\mathsf{Extract}$ et $\mathcal{O}\mathsf{Decaps}$ sont déterminés ci-dessous :

$$
\begin{array}{ll}
\mathcal{O}\mathsf{Extract}(\mathsf{ID}) & \mathcal{O}\mathsf{Decaps}(\mathsf{ID}, C) \\
\quad \mathsf{IDSet} \leftarrow \mathsf{IDSet} \cup \{\mathsf{ID}\}\,; & \quad \mathsf{CSet} \leftarrow \mathsf{CSet} \cup \{C\}\,; \\
\quad \mathsf{usk}_{\mathsf{ID}} \leftarrow \mathsf{Extract}_{\mathsf{IBK}}(\mathsf{msk}, \mathsf{ID})\,; & \quad m \leftarrow \mathsf{Decaps}(\mathsf{usk}_{\mathsf{ID}}, C)\,; \\
\quad \mathtt{retourner}\ \mathsf{usk}_{\mathsf{ID}} & \quad \mathtt{retourner}\ \mathsf{usk}_{\mathsf{ID}}
\end{array}
$$

L'attaquant gagne le jeu précédent si $c^* \notin \mathsf{CSet}, \mathsf{ID}_0, \mathsf{ID}_1 \notin \mathsf{IDSet}$ et si les deux égalités ci-dessous sont vérifiées :

$$K_0 = \mathsf{Decaps}_{\mathsf{IBK}}(\mathsf{ID}_0, c^*) \text{ et } K_1 = \mathsf{Decaps}_{\mathsf{IBK}}(\mathsf{ID}_1, c^*).$$

Nous définissons la probabilité de succès de $\mathcal{A}$ à gagner le jeu de non-malléabilité d'identités d'un schéma $\mathcal{IB}$-$\mathcal{KEM}$ par :

$$\mathsf{Succ}^{\mathsf{id\text{-}nm}}_{\mathcal{IB}\text{-}\mathcal{KEM},\mathcal{A}}(\lambda) = \Pr\left[\begin{array}{l}(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}_{\mathsf{IBK}}(\lambda); \\ (c, (K_0, \mathsf{ID}_0), (K_1, \mathsf{ID}_1)) \leftarrow \mathcal{A}(\mathsf{mpk}) : \\ K_0 = \mathsf{Decaps}_{\mathsf{IBK}}(\mathsf{ID}_0, c) \wedge K_1 = \mathsf{Decaps}_{\mathsf{IBK}}(\mathsf{ID}_1, c)\end{array}\right].$$

8.2.2 Analyse de constructions

Dans cette partie, nous faisons un petit détour sur les schémas présentés à la section 4.2.2. Notre but est de trouver un schéma vérifiant les propriétés d'anonymat classique, celles au sens KwrtA mais également à identités non-malléables. Un certain nombre de ces candidats sont déjà écartés, puisqu'il ne sont pas anonymes au sens KwrtA. Dans cette section, nous étudions malgré tout, pour tous les schémas, la propriété de non-malléabilité d'identités ; ce qui nous donnera un ordre d'idée de la difficulté de cette condition.

Le schéma de Boneh-Franklin [31]

Nous avons décrit ce schéma figure 4.1. Nous avons déjà un candidat potentiel dans le modèle de l'oracle aléatoire : cette construction vérifie les deux propriétés d'anonymat auxquelles nous nous intéressons (notion IND-ANON et anonymat au sens KwrtA). Rappelons seulement qu'un chiffré de la forme $c = g^r \in \mathbb{G}$ correspond à la clé $K = e(F_{\mathsf{mpk}}(\mathsf{ID}), \mathsf{mpk})^r = \mathsf{BDH}_g(\mathsf{mpk}, c, F(\mathsf{ID})) = e(\mathsf{usk}_{\mathsf{ID}}, c)$, où $\mathsf{usk}_{\mathsf{ID}} = F_{\mathsf{mpk}}(\mathsf{ID})^s = \mathsf{co\text{-}CDH}_g F_{\mathsf{mpk}}(\mathsf{ID})(\mathsf{mpk}) \in \mathbb{G}$ (la fonction co-CDH a été définie section 3.3.3).

Nous remarquons que le chiffré ne dépend pas de l'identité ; il est donc facile de casser la notion de non-malléabilité d'identités. Connaissant r et $c = g^r$, nous pouvons facilement calculer $K = \mathsf{BDH}_g(\mathsf{mpk}, c, F_{\mathsf{mpk}}(\mathsf{ID})) = e(F(\mathsf{ID}), \mathsf{mpk})^r$, pour toute identité choisie.

Le schéma de Boneh-Boyen [25]

Nous avons décrit le schéma $\mathcal{IBE}$ figure 3.4, rappelons que ce schéma est anonyme sous l'hypothèse $\mathsf{DLIN}_{\mathbb{G}}$. Un chiffré de la forme $c = (g^s, F_{\mathsf{mpk}}(\mathsf{ID})^s)$ correspond à la clé :

$$K = e(g_1, g_2)^s = e(c_1, \mathsf{usk}_2)/e(\mathsf{usk}_1, c_2),$$

si $\mathsf{usk}_{\mathsf{ID}}$ est définie par $(g^r, \mathsf{msk} \cdot F_{\mathsf{mpk}}(\mathsf{ID})^r)$, pour un certain $r \stackrel{R}{\leftarrow} \mathbb{Z}_p$.

Nous remarquons que le chiffré est spécifique à un utilisateur unique. Le problème est de savoir si une autre identité ID' peut déchiffrer ce chiffré. Compte tenu de l'aléa r utilisé lors de l'extraction de la clé, pour tout utilisateur malhonnête, possédant une clé secrète $\mathsf{usk}' = (g^{r'}, g_2^{\alpha} F_{\mathsf{mpk}}(\mathsf{ID}')^{r'})$, et un chiffré $c = (g^s, F_{\mathsf{mpk}}(\mathsf{ID}')^{s'})$ ($s' \neq s$ puisque ID' n'est pas le destinataire du chiffré c), $K' = K \times H^{r'}$, pour $H \neq 1$, et r' est complètement aléatoire. Par conséquent, ce schéma est à identités non-malléables au sens de la théorie de l'information.

Le schéma de Gentry [71]

Nous avons décrit ce schéma figure 4.3. Rappelons que ce schéma est anonyme sous l'hypothèse q-$\mathsf{ABDHE}_{\mathbb{G}}$, et pourtant il n'est pas anonyme au sens KwrtA. Puisque le chiffré est spécifique au destinataire, $\mathcal{A}$ ne sait pas quelle autre identité ID' permet de déchiffrer le chiffré $c = (c_1 = F_{\mathsf{mpk}}(\mathsf{ID}')^{s'}, c_2 = e(g,g)^s)$, puisque

$$\begin{aligned} K' &= e(c_1, \mathsf{usk}'_2) \cdot {c_2}^{\mathsf{usk}'_1} = e(F_{\mathsf{mpk}}(\mathsf{ID}')^{s'}, hg^{-\mathsf{usk}'_1})^{1/(\alpha - \mathsf{ID}')} \cdot e(g,g)^{s \cdot \mathsf{usk}'_1} \\ &= e(g,h)^{s'} \cdot e(g,g)^{(s-s') \cdot \mathsf{usk}'_1}. = e(g,h)^{s'} \cdot e(g,g)^{(s-s') \cdot \mathsf{usk}'_1} \\ &= K \cdot e(g,h)^{s'-s} \cdot e(g,g)^{(s-s') \cdot \mathsf{usk}'_1} = K \cdot (e(g,g)^{\mathsf{usk}'_1} / e(g,h))^{s-s'}, \end{aligned}$$

est un élément aléatoire dans $\mathbb{G}_T$. Ce schéma est donc à identités non-malléables au sens de la théorie de l'information.

Non-malléabilité d'identités de notre candidat

Revenons au candidat que nous avons proposé paragraphe 4.3.3 partie II. Cette construction vérifie les propriétés d'anonymat recherchées. Considérons le chiffré c, et son déchiffrement pour une identité ID_i pour $i \in \{0,1\}$. Rappelons que c et la clé associée sont de la forme : $c = F(\mathsf{ID}_i)^{r_i}$, et $K_i = e(g,h)^{r_i}$, ce qui définit formellement r_i. Par conséquent, la propriété de non-malléabilité d'identités repose sur la difficulté de trouver $c, \{\mathsf{ID}_i, K_i\}$, avec $\mathsf{ID}_0 \neq \mathsf{ID}_1$ tels que $r_i = \log_{e(g,h)}(K_i) = \log_{F(\mathsf{ID}_i)}(c)$; ce qui permet de trouver une solution au problème co-CDH commun.

Théorème 13 *La non-malléabilité d'identités (ou* Identity-Based Non-Malleability *en anglais) de notre schéma (décrit paragraphe 4.3.3) repose sur le problème du* co-CDH *commun (défini paragraphe 3.3.3 partie I) dans les groupes $\mathbb{G}$ et $\mathbb{G}_T$.*

Démonstration: Supposons qu'il existe un attaquant $\mathcal{A}$ qui parvienne à casser la non-malléabilité d'identités de notre schéma avec un avantage ϵ non négligeable. Il est alors possible de résoudre le problème $\mathsf{Cco\text{-}CDH}_{\mathbb{G},\mathbb{G}_T}$ avec un avantage ϵ. Soit $g, g_1 \stackrel{R}{\leftarrow} \mathbb{G}$ une instance du problème Cco-CDH, avec $\mathbb{G}$ d'ordre p ; nous choisissons $h \in \mathbb{Z}_p$ tel que $V = e(g,h) \in \mathbb{G}_T$. Nous définissons la clé $\mathsf{mpk} \stackrel{\text{déf}}{=} (g, g_1, h)$, avec $g, g_1 \stackrel{R}{\leftarrow} \mathbb{Z}_p$. Nous renvoyons cette clé à l'attaquant $\mathcal{A}$. ce dernier renvoie les éléments $r_i, (\mathsf{ID}_i, K_i)$ avec $r_i, \mathsf{ID}_i \in \mathbb{Z}_p$ et $K_i \in \mathbb{G}_T$ vérifiant :

$$c = F_{\mathsf{mpk}}(\mathsf{ID}_i)^{r_i} = g \cdot {g_1}^{\mathsf{ID}} \qquad K_i = e(g,h)^{r_i} \quad \text{et} \quad \mathsf{ID}_0 \neq \mathsf{ID}_1.$$

Nous obtenons donc une solution au problème $\mathsf{Cco\text{-}CDH}_{\mathbb{G},\mathbb{G}_T}$: $(c, \mathsf{ID}_0, \mathsf{ID}_1, K_0, K_1)$. □

8.3 Protocole IB-PAKE

8.3.1 Protocole 2-PAKE générique

Revenons sur une classe particulière de protocole 2-PAKE dérivée de EKE [22], où le premier message (la clé publique) envoyé n'est pas chiffré : il s'agit de la famille de protocole OKE, proposé par Lucks en 1997 [93]. Rappelons que pour les anciennes variantes, le chiffrement de la clé publique restreignait le choix de schémas. Cette nouvelle structure permet d'étendre la classe de crytposytèmes pouvant être utilisé, en particulier les schémas RSA. Cependant, lors d'une exécution, rien empêche l'attaquant d'envoyer une clé publique non valide ; dans le cas de RSA, on ne peut plus garantir que la fonction de chiffrement est une permutation et les attaques *par partition* deviennent possibles. Dans l'article [48], Catalano *et al.* proposent une nouvelle primitive d'isomorphisme possédant toutes les conditions nécessaires et suffisantes à la conception d'un protocole OKE sûr, IPAKE. Ils généralisent par la même occasion la classe de schémas pouvant être utilisés, incluant le chiffrement RSA et Rabin. Dans le même esprit, dans l'article [85], nous avons proposé la primitive d'$\mathcal{IB}$-$\mathcal{KEM}$ vérifiant de nouvelles notions (deux formes d'anonymat dont celui au sens KwrtA et non-malléabilité d'identités) nécessaires et suffisantes à la généralisation de protocole 2-PAKE : dans la prochaine section, nous présentons le protocole générique IB-PAKE qui en découle.

8.3.2 Description du schéma IB-PAKE

Nous décrivons le protocole générique obtenu à partir d'un $\mathcal{IB}$-$\mathcal{KEM}$ anonyme au sens usuel et au sens KwrtA et à identités non-malléables figure 8.1. Ce protocole permet à un client de partager une clé de session avec un serveur en utilisant un schéma $\mathcal{IB}$-$\mathcal{KEM}$ où l'identité est le mot de passe commun. Les propriétés suivantes sont vérifiées :

- un participant et son partenaire peuvent l'un ou l'autre retrouver la clé de session facilement ;
- la sécurité sémantique de la clé de session et la propriété de *forward-secrecy* sont toutes deux garanties. Ces deux propriétés sont impliquées par les propriétés d'anonymat du schéma $\mathcal{IB}$-$\mathcal{KEM}$ sous-jacent. Nous précisons ce résultat juste à la suite.

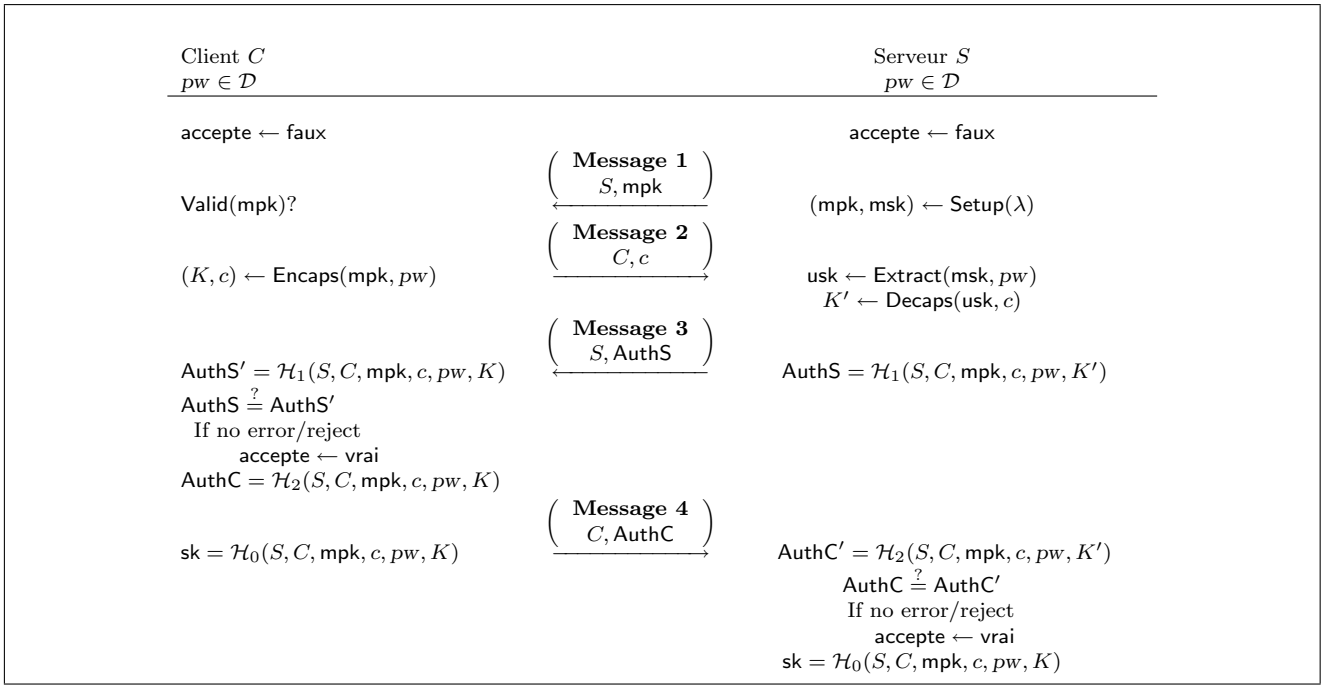


FIG. 8.1 – IB-PAKE : un protocole (générique) authentifié d'échange de clés à base de mot de passe.

8.3.3 Analyse de sécurité

Résultat de sécurité

Nous rappelons que nous considérons qu'une attaque est active si elle est générée (éventuellement pour des instances parallèles) *via* une succession de requêtes Send, dont l'une n'a pas été transférée mais initiée par l'attaquant.

Théorème 14 *Considérons un schéma d'*encapsulation de clés *à base d'identités* $\mathcal{IB}$-$\mathcal{KEM} \stackrel{déf}{=} \langle \mathsf{Setup}_{\mathsf{IBK}}, \mathsf{Extract}_{\mathsf{IBK}}, \mathsf{Encaps}_{\mathsf{IBK}}, \mathsf{Decaps}_{\mathsf{IBK}} \rangle$*, sémantiquement sûr (résistant aux attaques à messages clairs choisis, à identités non orientées et sans requêtes d'extraction), à la fois anonyme et anonyme au sens KwrtA, à identités non malléables. Alors notre protocole* IB-PAKE *garantit la confidentialité de clé de session et vérifie la propriété de* Perfect Forward-Secrecy *:*

$$\mathsf{Adv}^{\mathsf{ake}}_{\mathsf{IB\text{-}PAKE},\mathcal{A}}(\lambda) \leqslant 4 \times \frac{q_{\mathsf{active}}}{N} + \mathsf{negl}(\lambda),$$

où $q_{\mathsf{active}} = q_{\mathsf{activeC}} + q_{\mathsf{activeS}}$ *est le nombre d'attaques actives (générées* via *au moins une requête* Send*) et* N *est la taille du dictionnaire.*

Démonstration: La preuve est définie par une séquence de jeux modélisant une succession d'attaques passives et actives, où le premier jeu $\mathbf{G}_0$ représente l'attaque dans le jeu réel Dans tous ces jeux, on s'intéresse aux deux événements suivants :

– S (pour la *sécurité sémantique*) : cet événement a lieu si l'adversaire devine le bit b en jeu dans les requêtes Test ;

– A (pour l'*authentification mutuelle*) : cet événement a lieu si une instance accepte une session sans avoir de partenaire ou en étant partenaire avec l'adversaire.

Notre but est d'estimer la probabilité de l'événement S dans le jeu réel. Pour atteindre cet objectif, nous allons modifier pas à pas la simulation, de manière à obtenir un jeu final où l'adversaire ne gagne qu'avec probabilité négligeable. Nous notons Δ_n la différence de probabilité de tout événement Ev dans le jeu $\mathbf{G}_n$ et le jeu qui lui précède.
Remarquons que l'événement Ev n'est détectable que lorsque cette différence est calculatoirement bornée ; cela dit lorsque cette distance est statistiquement bornée cet événement peut être quelconque.

Game $\mathbf{G}_0$: il s'agit du jeu réel et par définition, on a :

$$\mathsf{Adv}^{\mathsf{ake}}_{\mathsf{IB\text{-}PAKE},\mathcal{A}}(\lambda) = 2\Pr[\mathsf{S}_0] - 1.$$

Game $\mathbf{G}_1$: dans ce jeu, nous simulons les requêtes Send comme précisées par la description du simulateur ci-dessous. Nous répondons aux requêtes Test en fonction de la valeur du bit b : si b vaut 1 (cas réel), nous retournons la valeur sk (provenant du calcul) ; si $b = 0$ (cas aléatoire), nous retournons $\mathcal{H}'_0(S, C, \mathsf{mpk}, c)$, où $\mathcal{H}'_0$ est un oracle secret : il s'agit d'une valeur complètement aléatoire. Remarquons simplement que l'entrée de $\mathcal{H}'_0$ est l'identifiant de la session et la propriété suivante est donc vérifiée : les requêtes Test posées à deux instances partenaires renvoient toujours la même valeur aléatoire, dans le cas où b vaut 0. Le but des jeux suivants est de rendre la simulation de sk identique dans le cas réel et dans le cas aléatoire.
Nous décrivons le simulateur ci-dessous :

Description du simulateur IB-PAKE

– **Simulation des requêtes** Send **à** C **:** les requêtes Send posées aux instances C sont simulées de la manière suivante :
  – on répond à une requête $\mathsf{SendC}(C; S, \mathsf{mpk})$ en appliquant la règle suivante :

    ▶**Rule C1**$^{(1)}$

    | on calcule $(K, c) \leftarrow \mathsf{Encaps}_{\mathsf{IBK}}(\mathsf{mpk}, pw)$;

    puis on répond par c.
  – on répond à une requête $\mathsf{SendC}(C; S, \mathsf{mpk})$ en appliquant la règle suivante :

    ▶**Rule C2**$^{(1)}$

    | on calcule $\mathsf{AuthS}' = \mathcal{H}_2(S, C, \mathsf{mpk}, c, pw, K)$ et $\mathsf{sk} = \mathcal{H}_0(S, C, \mathsf{mpk}, c, pw, K)$;

    on vérifie si AuthS' est égal à AuthS ; si l'égalité est vérifiée,
    on accepte avec la clé de session sk, et répondre par AuthC ; sinon on rejette ;
  – tous les autres cas sont ignorés.

– **Simulation des requêtes** Send **posées à** S : les requêtes Send posées à une instance S sont simulées de la manière suivante :
  – on répond à une requête $\mathsf{SendS}(S; \mathsf{INIT})$ en appliquant la règle suivante :

    ▶**Rule S1**$^{(1)}$

    | on génère générer $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}_{\mathsf{IBK}}(\lambda)$;

    on répond à cette requête par mpk.

– on répond à une requête $\mathsf{SendS}(S; C, c)$ en appliquant les règles suivantes :

▶**Rule S2**$^{(1)}$

on calcule $\mathsf{usk} \leftarrow \mathsf{Extract}_{\mathsf{IBK}}(\mathsf{msk}, pw)$;
on décapsule la clé, $K' \leftarrow \mathsf{Decaps}_{\mathsf{IBK}}(\mathsf{usk}, c)$;
puis on calcule $\mathsf{AuthS} = \mathcal{H}_1(S, C, \mathsf{mpk}, c, pw, K')$;

répondre par AuthS ;

– on répond aux requêtes $\mathsf{SendS}(S; C, \mathsf{AuthC})$ en appliquant les règles suivantes :

▶**Rule S3**$^{(1)}$

on calcule $\mathsf{AuthC}' = \mathcal{H}_2(S, C, \mathsf{mpk}, c, pw, K')$ et $\mathsf{sk} = \mathcal{H}_0(S, C, \mathsf{mpk}, c, pw, K')$.

on vérifie si AuthC' est égal à AuthC. Si l'égalité est vérifiée,
on accepte avec la clé de session sk, et on termine ; sinon on rejette ;

– tous les autres cas sont ignorés.

– **Simulation des orales $\mathcal{H}$ et $\mathcal{H}'$:** pour une requête de hachage $\mathcal{H}_n(q)$ (resp. $\mathcal{H}'_n(q)$), telle qu'un triplet (n, q, r) apparaît dans la liste $\Lambda_{\mathcal{H}}$ (resp. $\Lambda_{\mathcal{H}'}$), la réponse est r ; sinon on choisit un élément aléatoire r dans l'ensemble approprié qu'on renvoie ; on ajoute le triplet (n, q, r) à la liste $\Lambda_{\mathcal{H}}$ (resp. $\Lambda_{\mathcal{H}'}$).

Game $\mathbf{G}_2$: tout d'abord, on élimine les jeux où des collisions apparaissent. Celles-ci sont liées aux identifiants de session ; les collisions peuvent être problématiques si elles conduisent à des clés de session identiques qui correspondent chacune à des sessions différentes. Cette procédure permet juste de simplifier l'analyse de la preuve :
– collisions sur les *transcripts* partiels (S, C, mpk, c).
– collisions sur les sorties $\mathcal{H}_1, \mathcal{H}_2$ et $\mathcal{H}_0$.

$$\Delta_2 \leqslant \frac{q_{\mathcal{H}}^2}{2^\ell} + \gamma q_{\mathsf{session}}^2,$$

où $q_{\mathcal{H}}$ est le nombre de requêtes de hachage, et q_{session} le nombre de sessions, où γ est une borne supérieure de la probabilité de deviner mpk et c, et où ℓ est la taille des sorties des fonctions de hachage.

Game $\mathbf{G}_3$: dans ce jeu, nous considérons les sessions pour lesquelles l'adversaire reste passive au cours les deux premiers échanges. On note q_{passive} le nombre de requêtes $\mathsf{Execute}$, modélisant de tels échanges. Par la suite, $\mathsf{OG}(C, n)$ (resp. $\mathsf{OG}(S, n)$) signifiera que le $n^{\text{ième}}$ message reçu par le client (resp. par le serveur) a été généré par la simulation : nous connaissons donc l'aléa utilisé contrairement à l'adversaire.
Nous nous intéressons au cas où l'adversaire peut participer au protocole au troisième ou quatrième échange, et essaie de construire un authentifiant valide soit au nom de serveur (avec AuthS), soit au nom du client (avec AuthC). Intuitivement, l'adversaire n'a aucune information sur les valeurs secrètes, msk et K. La probabilité de construire un authentifiant valide est donc négligeable, même si $\mathcal{A}$ connaît le mot de passe.
Nous modifions uniquement ces sessions (qui peuvent tout à fait être déterminées avant) : l'authentifiant et la clé de session sont désormais calculés non plus par les oracles publiques $\mathcal{H}_i$ (pour $i = 0, 1, 2$) mais par les oracles secrets $\mathcal{H}'_i$ (pour $i = 0, 1, 2$), avec en entrée, non plus $(S, C, \mathsf{mpk}, c, pw, K)$ mais l'identifiant de session (ou session ID) (S, C, mpk, c), qui lui est public.
Une telle modification ne peut être détectée par l'attaquant qu'avec probabilité négligeable. Pour prouver ce résultat, nous allons montrer que si cette modification est détectée, l'attaquant casse la sécurité sémantique du schéma $\mathcal{IB}$-$\mathcal{KEM}$ sous-jacent.
Tout d'abord, précisons la manière dont nous modifions la simulation :

►**Rule S2**$^{(3)}$
- si $\neg$Corrupt$\wedge$OG$(C,1)\wedge$OG$(S,2)$, on calcule $\mathsf{AuthS} = \mathcal{H}'_1(S, C, \mathsf{mpk}, c)$;
- sinon,
  on extrait la clé $\mathsf{usk} \leftarrow \mathsf{Extract}_{\mathsf{IBK}}(\mathsf{msk}, pw)$;
  on déchiffre le chiffré, $K' \leftarrow \mathsf{Decaps}_{\mathsf{IBK}}(\mathsf{usk}, c)$;
  puis on calcule $\mathsf{AuthS} = \mathcal{H}_1(S, C, \mathsf{mpk}, c, pw, K')$.

►**Rule C2**$^{(3)}$
- si $\neg$Corrupt$\wedge$OG$(C,1)\wedge$OG$(S,2)$, on calcule $\mathsf{AuthS}' = \mathcal{H}'_1(S, C, \mathsf{mpk}, c)$,
  $\mathsf{sk} = \mathcal{H}'_0(S, C, \mathsf{mpk}, c)$, et $\mathsf{AuthC} = \mathcal{H}'_2(S, C, \mathsf{mpk}, c)$;
- sinon, on calcule $\mathsf{AuthS}' = \mathcal{H}_1(S, C, \mathsf{mpk}, c, pw, K)$,
  $\mathsf{sk} = \mathcal{H}_0(S, C, \mathsf{PK}, c, pw, K)$, $\mathsf{AuthC} = \mathcal{H}_2(S, C, \mathsf{mpk}, c, pw, K)$

►**Rule S3**$^{(3)}$
- si $\neg$Corrupt $\wedge$ OG$(C,1) \wedge$ OG$(S,2) \wedge$ OG$(C,3)$,
  on calcule $\mathsf{AuthC} = \mathcal{H}'_2(S, C, \mathsf{mpk}, c)$ et $\mathsf{sk} = \mathcal{H}'_0(S, C, \mathsf{mpk}, c)$;
- sinon, on calcule $\mathsf{AuthC}' = \mathcal{H}_2(S, C, \mathsf{mpk}, c, pw, K')$
  et $\mathsf{sk} = \mathcal{H}_0(S, C, \mathsf{mpk}, c, pw, K')$.

Les jeux $\mathbf{G}_3$ et $\mathbf{G}_2$ sont identiques sauf si l'adversaire remarque l'utilisation des oracles secrets ; ce qui n'est possible que si $\mathcal{A}$ pose des requêtes $\mathcal{H}_i$ sur des *transcripts* (S, C, mpk, c), pour une paire (pw, K) appropriée. Nous appelons cet événement AskHF (pour *Passive*, puisque l'adversaire a été passive durant les deux premiers échanges). Pour estimer la probabilité de l'événement AskHF, on définit un jeu auxiliaire qui permet de capturer la différence de probabilité de succès entre les jeux $\mathbf{G}_2$ et $\mathbf{G}_3$. On aura alors : si $\mathcal{A}$ est capable de distinguer les deux exécutions, on peut construire un attaquant contre la sécurité sémantique *au sens faible* (c.à.d. sans requêtes aux oracles, résistant aux attaques à messages choisis) du schéma $\mathcal{IB}$-$\mathcal{KEM}$ sous-jacent.
On suppose par contradiction qu'il existe un adversaire $\mathcal{A}'$ qui est capable de distinguer la simulation des jeux $\mathbf{G}_3$ et $\mathbf{G}_2$, avec un avantage non négligeable. On construit alors un attaquant contre la sécurité sémantique du schéma $\mathcal{IB}$-$\mathcal{KEM}$.
Game $\mathbf{G}_{2.1}$: nous modifions uniquement la $i^{\text{ième}}$ session, pour un i choisi entre 1 et q_{session}, avant corruption d'un des participants :

►**Rule S1**$^{(2.1)}$
- S'il s'agit de la $i^{\text{ième}}$ session
  on fait appel à l'algorithme $\mathsf{Setup}(\lambda)$ pour obtenir mpk avec pour identité challenge pw ;
- sinon,
  on génère $(\mathsf{mpk}, \mathsf{msk})$ par appel à $\mathsf{Setup}(\lambda)$.

►**Rule C1**$^{(2.1)}$
- s'il s'agit de la $i^{\text{ième}}$ session et si $\mathsf{OG}(C,1)$,
  on demande pw au challenger, qui répond par une paire (K, c) (si $d = 1$, on pose $K = K_1$ qui correspond réellement au chiffré c, et si $d = 0$, on pose $K = K_0$ qui est indépendante du chiffré c).
- sinon,
  on obtient $(K, c) \leftarrow \mathsf{Encaps}_{\mathsf{IBK}}(\mathsf{mpk}, pw)$.

►**Rule S2**$^{(2.1)}$

– s'il s'agit de la $i^{\text{ième}}$ session et si $\mathsf{OG}(C,1) \wedge \mathsf{OG}(S,2)$,
  on calcule $\mathsf{AuthS} = \mathcal{H}_1(S, C, \mathsf{mpk}, c, pw, K)$;
– sinon,
  on extrait la clé $\mathsf{usk} \leftarrow \mathsf{Extract}_{\mathsf{IBK}}(\mathsf{msk}, pw)$;
  on déchiffre le chiffré c, $K' \leftarrow \mathsf{Decaps}_{\mathsf{IBK}}(\mathsf{usk}, c)$;
  on calcule $\mathsf{AuthS} = \mathcal{H}_1(S, C, \mathsf{mpk}, c, pw, K')$.

Nous pouvons remarquer que dans le cas réel ($d = 1$), la simulation est identique à celle du jeu $\mathbf{G}_2$. Dans le cas aléatoire, la probabilité de demander $\mathcal{H}_i$ sur une entrée appropriée K et spécifique à la session est négligeable (bornée par $\gamma q_{\mathcal{H}}$, où γ est une borne supérieure sur la probabilité de deviner la clé K correcte) puisqu'aucune information sur la clé K ne fuit.
Et donc,

$$\Delta_{2.1} \leqslant \Pr[\mathsf{AskHF}_2] \leqslant q_{\mathsf{session}} \times \mathsf{Adv}^{\mathsf{ind-cpa}}_{\mathsf{IBK}}(t) + \gamma q_{\mathcal{H}} \leqslant \mathsf{negl}().$$

Game $\mathbf{G}_4$: dans ce jeu, nous ne modifions que formellement la simulation, sans modifier la vue de l'adversaire : on prend un chiffré aléatoire c avec la distribution appropriée, qui peut éventuellement dépendre du mot de passe. Notre but est de rendre la simulation indépendante du mot de passe, lorsqu'il n'y a pas de corruption (le flag Corrupt indique si l'adversaire connaît le mot de passe en ayant poser une requête Corrupt). On définit les deux distributions $\mathcal{C}_{\mathsf{mpk}}$ et $\mathcal{C}_{\mathsf{mpk},pw}$:

$$\mathcal{C}_{\mathsf{mpk}} = \{\mathsf{Encaps}_{\mathsf{IBK}}(\mathsf{mpk}, pw, m) \mid m \in \mathcal{M}\} \text{ et}$$

$$\mathcal{C}_{\mathsf{mpk},pw} = \{\mathsf{Encaps}_{\mathsf{IBK}}(\mathsf{mpk}, pw, m) \mid m \in \mathcal{M}, pw \in \mathcal{D}\}.$$

▶**Rule C1**$^{(4)}$
– si $\neg\mathsf{Corrupt} \wedge \mathsf{OG}(C,1)$, on calcule $c \leftarrow \mathcal{C}_{\mathsf{mpk},pw}$;
– sinon, $(K, c) \leftarrow \mathsf{Encaps}_{\mathsf{IBK}}(\mathsf{mpk}, pw)$.

On remarque que si $\mathsf{OG}(C,1)$ et $\mathsf{OG}(S,2)$, la clé K n'est plus utilisée par la suite. Cette simulation n'apporte aucune modification : $\Delta_4 = 0$.

Game $\mathbf{G}_5$: À présent, nous simulons les instances du client lorsque ce dernier reçoit une clé publique mpk générée par un oracle, et indépendamment du mot de passe. Notre nouveau jeu sera indistinguable du précédent, sous l'hypothèse d'anonymat du schéma $\mathcal{IB}$-$\mathcal{KEM}$ sous-jacent.

▶**Rule C1**$^{(5)}$
– si $\neg\mathsf{Corrupt} \wedge \mathsf{OG}(C,1)$, on choisit un chiffré $c \leftarrow \mathcal{C}_{\mathsf{mpk}}$;
– sinon on a $(K, c) \leftarrow \mathsf{Encaps}_{\mathsf{IBK}}(\mathsf{mpk}, pw)$.

La différence de probabilité de succès entre les jeux $\mathbf{G}_5$ et $\mathbf{G}_4$ est négligeable sauf si l'adversaire $\mathcal{A}$ est capable de distinguer les deux distributions $\mathcal{C}_{\mathsf{mpk}}$ et $\mathcal{C}_{\mathsf{mpk},pw}$, où pw peut être connu de l'adversaire. Or mpk est générée honnêtement (par l'autorité). En utilisant une séquence de jeux hybrides, nous pouvons montrer que

$$\Delta_5 \leqslant q_{\mathsf{passive}} \times \mathsf{Adv}^{\mathsf{anon}}_{\mathsf{IBK}}(t) \leqslant \mathsf{negl}(),$$

où t le temps de calcul de $\mathcal{A}$ nécessaire à casser l'anonymat du schéma IBKEM sous-jacent.

Remarque 16 *Nous pouvons remarquer que dans la séquence de jeux $\mathbf{G}_{2.1}$, nous avons commencé la simulation (spécifique au $i^{ième}$ jeu hybride) sans pouvoir prédire si la session allait être passive ou non ; ce qui explique la présence du facteur q_{session}. Dans cette séquence en revanche, la modification n'a lieu que si la session est passive, d'où le facteur q_{passive}.*

Game $\mathbf{G}_6$: dans ce jeu, nous considérons les attaques pour lesquelles l'adversaire initie une session active à partir du second échange : l'adversaire choisit le chiffré c. Nous allons montré qu'il a une faible chance d'envoyer un authentifiant AuthC valide, sauf s'il devine le bon mot de passe, celui utilisé pour le calcul de c : Nous voulons garantir que dans ce cas, au plus un mot de passe est testé.
Nous poursuivons les modifications en remplaçant les oracles $\mathcal{H}_i$ par les oracles secrets dans le cas où mpk est générée par un oracle.

▶**Rule S2**$^{(6)}$
- si $\neg\mathsf{Corrupt} \wedge \mathsf{OG}(C,1)$, on calcule $\mathsf{AuthS} = \mathcal{H}'_1(S,C,\mathsf{mpk},c)$.
- sinon,
  on extrait $\mathsf{usk} \leftarrow \mathsf{Extract}_{\mathsf{IBK}}(\mathsf{msk},pw)$;
  on déchiffre le chiffré , $K' \leftarrow \mathsf{Decaps}_{\mathsf{IBK}}(\mathsf{usk},c)$;
  puis on calcule $\mathsf{AuthS} = \mathcal{H}_1(S,C,\mathsf{mpk},c,pw,K')$.

▶**Rule C2**$^{(6)}$
- si $\neg\mathsf{Corrupt} \wedge \mathsf{OG}(C,1)$, on calcule $\mathsf{AuthS}' = \mathcal{H}'_1(S,C,\mathsf{mpk},c)$,
  $\mathsf{sk} = \mathcal{H}'_0(S,C,\mathsf{mpk},c)$, $\mathsf{AuthC} = \mathcal{H}'_2(S,C,\mathsf{mpk},c)$
- sinon, on calcule $\mathsf{AuthS}' = \mathcal{H}_1(S,C,\mathsf{mpk},c,pw,K)$,
  $\mathsf{sk} = \mathcal{H}_0(S,C,\mathsf{mpk},c,pw,K)$, $\mathsf{AuthC} = \mathcal{H}_2(S,C,\mathsf{mpk},c,pw,K)$

Les jeux $\mathbf{G}_6$ et $\mathbf{G}_5$ sont identiques sauf si l'adversaire se rend compte que le calcul a été effectué par des oracles secrets $\mathcal{H}'_i$ $(i \in \{0,1\})$ dans les sessions pour lesquelles il a choisi le chiffré c sans avoir choisi la clé publique mpk. Un tel événement a lieu si et seulement si l'adversaire pose une des requêtes de hachage $\mathcal{H}_i$ $(i \in \{0,1\})$ sur un *transcript* (S,C,mpk,c) associée à une paire (pw,K) appropriée, c.à.d. telle que $K = \mathsf{Decaps}_{\mathsf{IBK}}(pw,c)$ pour le mot de passe en jeu. Dans ce cas, il existe au moins un mot de passe pw tel que $(S,C,\mathsf{mpk},c,pw,K) \in \Lambda_{\mathcal{H}}$ et $K = \mathsf{Decaps}_{\mathsf{IBK}}(pw,c)$. Nous appelons cet événement AskHS_6. De manière à clarifier notre analyse, nous distinguons les deux cas suivant :
- pour toutes les sessions (S,C,mpk,c), où c est choisi par l'adversaire, il y a au plus un mot de passe π tel que $(S,C,\mathsf{mpk},c,\pi,K = \mathsf{Decaps}_{\mathsf{IBK}}(\pi,c)) \in \Lambda_{\mathcal{H}}$. Nous notons cet événement AskHSU_6. S'il a lieu, alors au plus un mot de passe est testé par attaque active contre le serveur. Notre but est de montrer que :

$$\Pr[\mathsf{AskHSU}_6] \leqslant \frac{q_{\mathsf{activeS}}}{N}.$$

- pour une session (S,C,mpk,c), il existe deux mots de passe π_1,π_2 tels que $(S,C,\mathsf{mpk},c,\pi_i,K_i = \mathsf{Decaps}(\pi_i,c)) \in \Lambda_{\mathcal{H}}$. Nous notons cet événement AskHSM_6. S'il a lieu, $\mathcal{A}$ peut alors construire $\{c,(\pi_1,K_1),(\pi_2,K_2)\}$ tels que $K_1 = \mathsf{Decaps}_{\mathsf{IBK}}(\pi_1,c)$ et $K_2 = \mathsf{Decaps}_{\mathsf{IBK}}(\pi_2,c)$; ce qui contredit l'hypothèse de non-malléabilité d'identités du schéma $\mathcal{IB}$-$\mathcal{KEM}$ sous-jacent :

$$\Pr[\mathsf{AskHSM}_6] \leqslant \mathsf{Adv}^{\mathsf{id\text{-}nm}}_{\mathsf{IBK}}(t).$$

Et donc,

$$\Delta_6 \leqslant \Pr[\mathsf{AskHS}_6] \leqslant \mathsf{Adv}^{\mathsf{id\text{-}nm}}_{\mathsf{IBK}}(t) + \Pr[\mathsf{AskHSU}_6],$$

où t est le temps maximum pour casser la non-malléabilité d'identités du schéma $\mathcal{IB}$-$\mathcal{KEM}$.

Game $\mathbf{G}_7$: dans ce jeu, nous considérons les attaques actives contre le client qui commence tout au début de la session : l'adversaire envoie le premier message et contrôle

mpk. Il se peut très bien qu'il connaisse la clé maître msk. Comme dans le jeu $\mathbf{G}_5$, on veut rendre la génération du chiffré indépendante du mot de passe, y compris dans ce cas. On remplace alors les oracles de hachage par des oracles secrets dans toutes les sessions où personne n'est corrompue :

►**Rule S2**$^{(7)}$
- si ¬Corrupt, on calcule $\mathsf{AuthS} = \mathcal{H}'_1(S, C, \mathsf{mpk}, c)$;
- sinon,
  on extrait $\mathsf{usk} \leftarrow \mathsf{Extract}_{\mathsf{IBK}}(\mathsf{msk}, pw)$;
  on déchiffre le chiffré, $K' \leftarrow \mathsf{Decaps}_{\mathsf{IBK}}(\mathsf{usk}, c)$;
  on calcule $\mathsf{AuthS} = \mathcal{H}_1(S, C, \mathsf{mpk}, c, pw, K')$.

►**Rule C2**$^{(7)}$
- si ¬Corrupt, on calcule $\mathsf{AuthS}' = \mathcal{H}'_1(S, C, \mathsf{mpk}, c)$,
  $\mathsf{sk} = \mathcal{H}'_0(S, C, \mathsf{mpk}, c)$, $\mathsf{AuthC} = \mathcal{H}'_2(S, C, \mathsf{mpk}, c)$;
- sinon, on calcule $\mathsf{AuthS}' = \mathcal{H}_1(S, C, \mathsf{mpk}, c, pw, K)$,
  $\mathsf{sk} = \mathcal{H}_0(S, C, \mathsf{mpk}, c, pw, K)$, $\mathsf{AuthC} = \mathcal{H}_2(S, C, \mathsf{mpk}, c, pw, K)$

►**Rule S3**$^{(7)}$
- si ¬Corrupt, on calcule $\mathsf{AuthC} = \mathcal{H}'_2(S, C, \mathsf{mpk}, c)$
  et $\mathsf{sk} = \mathcal{H}'_0(S, C, \mathsf{mpk}, c)$;
- sinon, on calcule $\mathsf{AuthC}' = \mathcal{H}_2(S, C, \mathsf{mpk}, c, pw, K')$
  et $\mathsf{sk} = \mathcal{H}_0(S, C, \mathsf{mpk}, c, pw, K')$.

Les jeux $\mathbf{G}_7$ et $\mathbf{G}_6$ sont identiques sauf si mpk est générée par un oracle : tous les oracles publics sont remplacés par des oracles secrets. $\mathcal{A}$ se rend compte de cette différence seulement s'il pose $\mathcal{H}_1$ sur un *transcript* (S, C, mpk, c) pour une paire appropriée (pw, K), c.à.d. telle que $K = \mathsf{Decaps}_{\mathsf{IBK}}(pw, c)$ et $K \in \Lambda_{\mathcal{H}}$ pour le mot de passe en jeu. Cette requête est nécessaire pour le calcul d'un authentifiant valide, AuthS.
Puisque nous avons supprimé les collisions (jeu $\mathbf{G}_2$), il existe au plus un t-uplet $(S, C, \mathsf{mpk}, c, pw, K)$ tel que $\mathsf{AuthS} = \mathcal{H}_1(S, C, \mathsf{mpk}, c, pw, K)$ pour chaque authentifiant. Nous notons AskHC un tel événement.

$$\Delta_7 \leqslant \Pr[\mathsf{AskHC}_7].$$

Game $\mathbf{G}_8$: puisque nous utilisons encore le mot de passe dans certaines sessions (pour le calcul de c), nous ne pouvons pas encore nous prononcer sur l'évaluation de la probabilité des événements AskHC etAskHSU. Donc, comme dans le jeu $\mathbf{G}_5$, on rend la génération du chiffré c indépendante du mot de passe, y compris dans le cas où mpk est générée par l'adversaire. Puisque l'adversaire choisit mpk, la différence est donc bornée par la probabilité de succès défini par le jeu pour l'anonymat au sens *KwrtA* du schéma $\mathcal{IB}$-$\mathcal{KEM}$ sous-jacent :

►**Rule C1**$^{(8)}$
- si ¬Corrupt, on choisit $c \leftarrow \mathcal{C}_{\mathsf{mpk}}$;
- sinon, on calcule $(K, c) \leftarrow \mathsf{Encaps}_{\mathsf{IBK}}(\mathsf{mpk}, pw)$.

La différence de probabilité de succès entre les jeux $\mathbf{G}_8$ et $\mathbf{G}_7$ est négligeable sauf si $\mathcal{A}$ est capable de distinguer les distributions $\mathcal{C}_{\mathsf{mpk}}$ et $\mathcal{C}_{\mathsf{mpk},pw}$, où pw peut éventuellement être connu de l'adversaire et mpk est choisi par l'adversaire. En utilisant une séquence de jeu hybride comme pour le jeu $\mathbf{G}_5$, sauf que $\mathcal{A}$ peut très bien connaître msk, on montre facilement que :

$$\Delta_8 \leqslant q_{\mathsf{activeC}} \times \mathsf{Adv}^{\mathsf{kwrta-anon}}_{\mathsf{IBK}}(t) \leqslant \mathsf{negl}().$$

Game $\mathbf{G}_9$: Remarquons que s'il n'y a pas de corruption, les authentifiants et clés de session sont calculés en utilisant des oracles secrets, sans le mot de passe en entrée ; le mot de passe n'est plus utilisé dans la simulation sauf si une corruption a lieu, plus exactement, dans les cas suivant :
– le serveur génère les paramètres publics,
– le client génère le chiffré selon la distribution $\mathcal{C}_{\mathsf{mpk}}$.
On peut alors choisir le mot de passe à la fin de la simulation, ou lorsqu'une corruption a lieu : $\Delta_9 = 0$. Nous pouvons alors évaluer la probabilité des événements AskHC et AskHSU :
– l'événement AskHSU a lieu si pour chaque session (S, C, mpk, c), où c est choisi par l'adversaire (attaque active contre le serveur), il y a au plus une paire appropriée (pw, K). Dans ce cas, seulement un mot de passe peut donner lieu à cet événement pour chaque session où l'adversaire est actif contre le serveur :

$$\Pr[\mathsf{AskHSU}_9] \leqslant \frac{q_{\mathsf{activeS}}}{N}$$

– l'événement AskHC a lieu si $\mathcal{A}$ envoie un authentifiant valide (il s'agit alors d'une attaque active contre le client), calculé avec le bon mot de passe. Mais une seule valeur du mot de passe peut donner lieu à cet événement pour chaque session où l'adversaire est actif contre le client, puisque nous avons supprimé les collisions sur $\mathcal{H}_1$:

$$\Pr[\mathsf{AskHC}_9] \leqslant \frac{q_{\mathsf{activeC}}}{N}.$$

Comme toutes les clés de session sont calculées à partir d'un oracle secret dans le dernier jeu, il n'y a pas de différence entre le cas réel et le cas aléatoire pour la requête Test :

$$\Pr[S_9] = \frac{1}{2},$$

ce qui permet de conclure la preuve. □

Nous avons présenté une application des notions d'anonymat pour le chiffrement $\mathcal{IB}$-$\mathcal{KEM}$ présentées au chapitre 4 : l'aboutissement est un protocole 2-PAKE générique, IB-PAKE permettant à un client et un serveur d'établir une clé de session commune, tout en garantissant l'anonymat du client. Notre protocole est prouvé sûr dans une extension du modèle de Bellare *et al.* [17, 14], présentée au chapitre 6. Par ailleurs, la preuve que nous obtenons est très intuitive : toutes les propriétés de l'$\mathcal{IB}$-$\mathcal{KEM}$ sous-jacent induisent la sécurité sémantique de la clé de session échangée, la propriété de *perfect-forward-secrecy* et enfin une garantie d'anonymat du client. D'un point de vue théorique, il est intéressant de remarquer que la primitive de chiffrement à base d'identité est à la source d'une nouvelle application pour laquelle elle n'est *a priori* pas destinée ; il serait intéressant d'exploiter ces multiples fonctionnalités dans d'autres contexte que celui de l'échange de clés.

Conclusion

Nous avons confronté le concept d'anonymat à des situations réelles dans les protocoles de chiffrement et de mise en accord de clé : à chaque étape, nous avons proposé un modèle, en justifiant nos choix, parfois au détriment d'un niveau de sécurité plus fort. L'anonymat a un coût réel mais des solutions à son emploi sont possibles : nous avons tenté, dans chaque situation, d'établir le meilleur compromis.

Un premier état de l'art des mécanismes conçus pour le chiffrement anonyme nous a permis d'exposer les techniques de constructions de schémas anonymes. Nous avons proposé une nouvelle notion de sécurité renforçant ainsi les outils caractérisant les primitives que nous connaissons. Ensuite, s'est posée la question du chiffrement anonyme dans un contexte multi-acteurs : nous avons complété les fonctionnalités de la primitive de chiffrement de groupe initialement proposée, en exhibant un modèle et un schéma efficace, sans faire usage de preuves interactives. Nous avons prouvé la sécurité de notre schéma dans le modèle standard.

Pour les protocoles d'échange de clé, nous avons recensé les notions de sécurité pour le scénario à deux et trois parties. Nous avons étendu le modèle existant aboutissant à une mécanisation (partielle) des preuves de sécurité dans ce contexte : l'idée est essentiellement basée sur une anticipation des attaques dans le cas où les corruptions sont statiques. Ensuite, nous avons proposé deux protocoles d'échanges de clés en considérant l'anonymat du client, d'abord en utilisant un schéma d'interrogation confidentielle de bases de données comme interface à une variante d'un protocole à trois parties, proposés en 2005 dans l'article [2] ; puis en utilisant une primitive de chiffrement anonyme. Dans ce dernier cas, il s'agit précisément d'une application de la notion de sécurité que nous avons introduite.

Bibliographie

[1] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi. Searchable encryption revisited : Consistency properties, relation to anonymous IBE, and extensions. In V. Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 205–222. Springer, Aug. 2005.

[2] M. Abdalla, O. Chevassut, P.-A. Fouque, and D. Pointcheval. A simple threshold authenticated key exchange from short secrets. In B. K. Roy, editor, *ASIACRYPT 2005*, volume 3788 of *LNCS*, pages 566–584. Springer, Dec. 2005.

[3] M. Abdalla, P.-A. Fouque, and D. Pointcheval. Password-based authenticated key exchange in the three-party setting. In S. Vaudenay, editor, *PKC 2005*, volume 3386 of *LNCS*, pages 65–84. Springer, Jan. 2005.

[4] M. Abdalla, M. Izabachène, and D. Pointcheval. Anonymous and transparent gateway-based password-authenticated key exchange. In *CANS 08*, LNCS, pages 133–148. Springer, 2008.

[5] M. Abdalla and D. Pointcheval. Interactive Diffie-Hellman assumptions with applications to password-based authentication. In A. Patrick and M. Yung, editors, *FC 2005*, volume 3570 of *LNCS*, pages 341–356. Springer, Feb. / Mar. 2005.

[6] A. Ambainis. Upper bound on communication complexity of private information retrieval. In P. Degano, R. Gorrieri, and A. Marchetti-Spaccamela, editors, *ICALP 97*, volume 1256 of *LNCS*. Springer, July 1997.

[7] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In M. Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 255–270. Springer, Aug. 2000.

[8] J. Baek, D. Galindo, W. Susilo, and J. Zhou. Constructing strong kem from weak kem (or how to revive the kem/dem framework). In *SCN 06*, LNCS, pages 358–374. Springer, Sept. 2008.

[9] A. Baimel and Y. Ishai. Information retrieval private information retrieval : A unified construction. unpublished manuscript.

[10] J. Bajard and N. ElMrabet. Pairing in cryptography : an arithmetic point of view. In *Advanced Signal Processing Algorithms, Architectures, and Implementations (SPI)*, volume 6697, pages 480–494, 2007.

[11] M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval. Key-privacy in public-key encryption. In C. Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 566–582. Springer, Dec. 2001.

[12] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *38th FOCS*, pages 394–403. IEEE Computer Society Press, Oct. 1997.

[13] M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures : Formal definitions, simplified requirements, and a construction based on general assumptions. In E. Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 614–629. Springer, May 2003.

[14] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In B. Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 139–155. Springer, May 2000.

[15] M. Bellare and T. Ristenpart. Simulation without the artificial abort : Simplified proof and improved concrete security for waters' ibe scheme. In A. Joux, editor, *EUROCRYPT 2009*, LNCS. Springer, Apr. 2009.

[16] M. Bellare and P. Rogaway. Random oracles are practical : A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93*, pages 62–73. ACM Press, Nov. 1993.

[17] M. Bellare and P. Rogaway. Entity authentication and key distribution. In D. R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 232–249. Springer, Aug. 1994.

[18] M. Bellare and P. Rogaway. The exact security of digital signatures : How to sign with RSA and Rabin. In U. M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 399–416. Springer, May 1996.

[19] M. Bellare and P. Rogaway. Provably secure session key distribution — the three party case. In *28th ACM STOC*, pages 57–66. ACM Press, May 1996.

[20] M. Bellare and P. Rogaway. The AuthA protocol for password-based authenticated key exchange. Contributions to IEEE P1363, Mar. 2000.

[21] M. Bellare, H. Shi, and C. Zhang. Foundations of group signatures : The case of dynamic groups. In A. Menezes, editor, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 136–153. Springer, Feb. 2005.

[22] S. M. Bellovin and M. Merritt. Encrypted key exchange : Password-based protocols secure against dictionary attacks. In *1992 IEEE Symposium on Security and Privacy*, pages 72–84. IEEE Computer Society Press, May 1992.

[23] K. Bentaha, P. Farshim, J. Malone-Lee, and N. P. Smart. Generic constructions of identity-based and certificateless KEMs. Cryptology ePrint Archive, Report 2005/058, 2005. `http://eprint.iacr.org/`.

[24] I. Blake, G. Seroussi, and N. Smart. *Elliptic Curves in Cryptography*. London Mathematical Society Lecture Note Series. Cambridge University Press, 1999.

[25] D. Boneh and X. Boyen. Efficient selective-ID secure identity based encryption without random oracles. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238. Springer, May 2004.

[26] D. Boneh and X. Boyen. Secure identity based encryption without random oracles. In M. Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 443–459. Springer, Aug. 2004.

[27] D. Boneh and X. Boyen. Short signatures without random oracles. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer, May 2004.

[28] D. Boneh and X. Boyen. On the impossibility of efficiently combining collision resistant hash functions. In C. Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 570–583. Springer, Aug. 2006.

[29] D. Boneh, X. Boyen, and E.-J. Goh. Hierarchical identity based encryption with constant size ciphertext. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*. Springer, May 2005.

[30] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In M. Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, Aug. 2004.

[31] D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Aug. 2001.

[32] D. Boneh, E.-J. Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. In J. Kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 325–341. Springer, Feb. 2005.

[33] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In C. Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Dec. 2001.

[34] X. Boyen. The bb1 identity-based crytposystem : A standard for encryption and key encapsulation. In *2006 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2006. available at http ://grouper.ieee.org/groups/1363/.

[35] X. Boyen. A tapestry of identity-based encryption : Practical frameworks compared. *International Journal of Applied Cryptography*, 1(1) :3–21, 2008.

[36] X. Boyen and B. Waters. Anonymous hierarchical identity-based encryption (without random oracles). In C. Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 290–307. Springer, Aug. 2006.

[37] X. Boyen and B. Waters. Compact group signatures without random oracles. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 427–444. Springer, May / June 2006.

[38] V. Boyko, P. D. MacKenzie, and S. Patel. Provably secure password-authenticated key exchange using Diffie-Hellman. In B. Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 156–171. Springer, May 2000.

[39] E. Bresson, O. Chevassut, and D. Pointcheval. New security results on encrypted key exchange. In F. Bao, R. Deng, and J. Zhou, editors, *PKC 2004*, volume 2947 of *LNCS*, pages 145–158. Springer, Mar. 2004.

[40] J. Camenisch. Efficient and generalized group signatures. In W. Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 465–479. Springer, May 1997.

[41] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In B. Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer, May 2001.

[42] J. Camenisch and M. Michels. A group signature scheme with improved efficiency. In K. Ohta and D. Pei, editors, *ASIACRYPT'98*, volume 1514 of *LNCS*, pages 160–174. Springer, Oct. 1998.

[43] J. Camenisch, G. Neven, and A. Shelat. Simulatable adaptive oblivious transfer. In M. Naor, editor, *EUROCRYPT 2007*, LNCS. Springer, May 2007.

[44] J. Camenisch and M. Stadler. Efficient group signature schemes for large groups (extended abstract). In B. S. K. Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 410–424. Springer, Aug. 1997.

[45] S. Canard and J. Traoré. On fair e-cash systems based on group signature schemes. published in acisp, 2003.

[46] R. Canetti, S. Halevi, and J. Katz. A forward-secure public-key encryption scheme. In E. Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 255–271. Springer, May 2003.

[47] R. Canetti, H. Krawczyk, and J. B. Nielsen. Relaxing chosen-ciphertext security. In D. Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 565–582. Springer, Aug. 2003.

[48] D. Catalano, D. Pointcheval, and T. Pornin. IPAKE : Isomorphisms for password-based authenticated key exchange. In M. Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 477–493. Springer, Aug. 2004.

[49] Y.-C. Chang. Single database private information retrieval with logarithmic communication. published in acisp, 2004.

[50] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the Association for Computing Machinery*, 24(2) :84–88, 1981.

[51] D. Chaum. Blind signature system. In D. Chaum, editor, *CRYPTO'83*, page 153. Plenum Press, New York, USA, 1984.

[52] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In S. Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 319–327. Springer, Aug. 1990.

[53] D. Chaum and T. P. Pedersen. Wallet databases with observers. In E. F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 89–105. Springer, Aug. 1993.

[54] D. Chaum and E. van Heyst. Group signatures. In D. W. Davies, editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 257–265. Springer, Apr. 1991.

[55] L. Chen and Z. Cheng. Security proof of sakai-kasahara's identity-based encryption scheme. Cryptology ePrint Archive, Report 2005/226, 2005. `http://eprint.iacr.org/`.

[56] L. Chen, Z. Cheng, J. Malone-Lee, and N. Smart. An efficient id-kem based on the sakai-kasahara key construction. Cryptology ePrint Archive, Report 2005/224, 2005. `http://eprint.iacr.org/`.

[57] L. Chen and T. P. Pedersen. New group signature schemes (extended abstract). In A. D. Santis, editor, *EUROCRYPT'94*, volume 950 of *LNCS*, pages 171–181. Springer, May 1994.

[58] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan. Private information retrieval. *Journal of the ACM*, 45(6) :965–981, 1998.

[59] C. Cocks. An identity based encryption scheme based on quadratic residues. In B. Honary, editor, *Cryptography and Coding, 8th IMA International Conference*, volume 2260 of *LNCS*, pages 360–363, Cirencester, UK, Dec. 17–19, 2001. Springer.

[60] J.-S. Coron. On the exact security of full domain hash. In M. Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 229–235. Springer, Aug. 2000.

[61] R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1) :167–226, 2003.

[62] C. Crépeau. Equivalence between two flavours of oblivious transfers. In C. Pomerance, editor, *CRYPTO'87*, volume 293 of *LNCS*, pages 350–354. Springer, Aug. 1988.

[63] I. Damgård and M. Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In K. Kim, editor, *PKC 2001*, volume 1992 of *LNCS*, pages 119–136. Springer, Feb. 2001.

[64] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. In *23rd ACM STOC*, pages 542–552. ACM Press, May 1991.

[65] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31 :469–472, 1985.

[66] A. Enge. *Elliptic Curves and Their Application to Cryptography : An Introduction*. Kluwer Academic Publishers, 1999.

[67] A. Fiat and A. Shamir. How to prove yourself : Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Aug. 1987.

[68] G. Frey and H. Ruck. A remark concerning divisibility and the discrete logarithm in the divisor class group of curves. *Mathematics of Computation*, 62 :865–874, 1994.

[69] S. Galbraith, K. Paterson, and N. Smart. Pairings for cryptographers. Cryptology ePrint Archive, Report 2006/165, 2006. `http://eprint.iacr.org/`.

[70] P. Gaudry. *Algorithmique des courbes hyperelliptiques et applications à la cryptologie*. PhD thesis, École polytechnique, France, 2000.

[71] C. Gentry. Practical identity-based encryption without random oracles. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 445–464. Springer, May / June 2006.

[72] O. Goldreich and Y. Lindell. Session-key generation using human passwords only. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 408–432. Springer, Aug. 2001. `http://eprint.iacr.org/2000/057`.

[73] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2) :270–299, 1984.

[74] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th ACM STOC*, pages 291–304. ACM Press, May 1985.

[75] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2) :281–308, Apr. 1988.

[76] P. Golle, M. Jakobsson, A. Juels, and P. F. Syverson. Universal re-encryption for mixnets. In T. Okamoto, editor, *CT-RSA 2004*, volume 2964 of *LNCS*, pages 163–178. Springer, Feb. 2004.

[77] J. Groth. Rerandomizable and replayable adaptive chosen ciphertext attack secure cryptosystems. In M. Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 152–170. Springer, Feb. 2004.

[78] J. Groth. Fully anonymous group signatures without random oracles. In K. Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 164–180. Springer, Dec. 2007.

[79] J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. Cryptology ePrint Archive, 2007. `http://eprint.iacr.org/2007/155`.

[80] S. Halevi and H. Krawczyk. Public-key cryptography and password protocols. *ACM Transactions on Information and System Security*, 2(3) :230–268, Aug. 1999.

[81] R. Hartshorne. *Algebraic geometry*. Springer, 1977.

[82] J. Herranz, D. Hofheinz, and E. Kiltz. Kem/dem : Necessary and sufficient conditions for secure hybrid encryption. Cryptology ePrint Archive, 2006. `http://eprint.iacr.org/`.

[83] J. Herranz, D. Hofheinz, and E. Kiltz. The kurosawa-desmedt key encapsulation is not chosen-ciphertext secure. Cryptology ePrint Archive, 2006. `http://eprint.iacr.org/`.

[84] Y. Ishai and E. Kushilevitz. Improved upper bounds on information-theoretic private information retrieval (extended abstract). In *31st ACM STOC*, pages 79–88. ACM Press, May 1999.

[85] M. Izabachène and D. Pointcheval. New anonymity notions for identity-based encryption. In *SCN 06*, LNCS, pages 375–391. Springer, Sept. 2008.

[86] M. Izabachene, D. Pointcheval, and D. Vergnaud. Traceable anonymous encryption. Technical report, Ecole Normale Superieure de Paris, 2009. Avalaible at http ://www.di.ens.fr/ izabache/docs/anon_pdf.

[87] D. P. Jablon. Extended password key exchange protocols immune to dictionary attacks. In *6th IEEE International Workshops on Enabling Technologies : Infrastructure for Collaborative Enterprises (WETICE 1997)*, pages 248–255, Cambridge, MA, USA, June 18–20, 1997. IEEE Computer Society.

[88] A. Kiayias, Y. Tsiounis, and M. Yung. Group encryption. In K. Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 181–199. Springer, Dec. 2007.

[89] N. Koblitz. *Algebraic Aspects of Cryptography*, volume 3. Springer, 1997.

[90] N. Koblitz and A. Menezes. Pairing-based cryptography at high security levels. In B. Honary, editor, *Cryptography and Coding, 10th IMA International Conference*, volume 3796 of *LNCS*, pages 13–36, Cirencester, UK, 2005. Springer.

[91] E. Kushilevitz and R. Ostrovsky. Replication is NOT needed : SINGLE database, computationally-private information retrieval. In *38th FOCS*, pages 364–373. IEEE Computer Society Press, Oct. 1997.

[92] E. Kushilevitz and R. Ostrovsky. One-way trapdoor permutations are sufficient for non-trivial single-server private information retrieval. In B. Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 104–121. Springer, May 2000.

[93] S. Lucks. Open key exchange : How to defeat dictionary attacks without encrypting public keys. In *Workshop on Security Protocols*, École Normale Supérieure, 1997.

[94] P. MacKenzie and R. Swaminathan. Secure authentication with a short secret. Contributions to IEEE P1363, Aug. 1999.

[95] A. Menezes. *Elliptic Curve Public Key Cryptosystems*. Kluwer Academic Publishers, 1993.

[96] A. Menezes, T. Okamoto, and S. Vanstone. Reducing elliptic curve to logarithms in a fnite field. *IEEE Transactions on Information Theory*, 39 :1639–1646, 1993.

[97] V. Miller. Short programs for functions on curves. available at http//crypto.standford.edu/miller/miller.ps, 1986.

[98] V. S. Miller. The Weil pairing, and its efficient calculation. *Journal of Cryptology*, 17(4) :235–261, Sept. 2004.

[99] S. Mitsunari, R. Saka, and M. Kasahara. A new traitor tracing. *IEICE Transactions*, E85-A(2) :481–484, Feb. 2002.

[100] A. Miyaji, M. Nakabayashi, and S. Takano. New explicit conditions of elliptic curve traces for fr-reduction. *IEICE Transactions*, E84-A(5) :1234–1243, Feb. 2001.

[101] M. Naor and B. Pinkas. Computationally secure oblivious transfer. *Journal of Cryptology*, 18(1) :1–35, Jan. 2005.

[102] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *21st ACM STOC*, pages 33–43. ACM Press, May 1989.

[103] K. Q. Nguyen and J. Traoré. An online public auction protocol protecting bidder privacy. published in acisp, 2000.

[104] T. Okamoto and S. Uchiyama. A new public-key cryptosystem as secure as factoring. In K. Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 308–318. Springer, May / June 1998.

[105] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238. Springer, May 1999.

[106] D. Pointcheval and J. Stern. Security proofs for signature schemes. In U. M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 387–398. Springer, May 1996.

[107] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3) :361–396, 2000.

[108] M. Prabhakaran and M. Rosulek. Rerandomizable rcca encryption. In A. Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 517–534. Springer, Aug. 2007.

[109] N. S. R. Granger, D. Page. High security pairing-based cryptography revisited. In *Seventh Algorithmic Number Theory Symposium (ANTS)*, volume 4096 of *LNCS*, pages 480–494. Springer, 1998.

[110] M. O. Rabin. How to Exchange Secrets with Oblivious Tranfer. Technical Report TR-81, Aiken Computation Lab, Harvard University, Jan. 1981. Available as Cryptology ePrint Archive Report 2005/187.

[111] R. Sakai and M. Kasahara. Id based cryptosystems with pairing on elliptic curve. Cryptology ePrint Archive, Report 2003/054, 2003. `http://eprint.iacr.org/`.

[112] K. Sako. An auction protocol which hides bids of losers. In H. Imai and Y. Zheng, editors, *PKC 2000*, volume 1751 of *LNCS*, pages 422–432. Springer, Jan. 2000.

[113] C.-P. Schnorr. Efficient identification and signatures for smart cards. In G. Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer, Aug. 1990.

[114] C.-P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3) :161–174, 1991.

[115] A. Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and D. Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 47–53. Springer, Aug. 1985.

[116] V. Shoup. Using hash functions as a hedge against chosen ciphertext attack. In B. Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 275–288. Springer, May 2000.

[117] V. Shoup. ISO 18033-2 : An emerging standard for public-key encryption. `http://shoup.net/iso/std6.pdf`, Dec. 2004. Final Committee Draft.

[118] J. Silverman. *The arithmetic of elliptic curves*. Springer, 1986.

[119] G. J. Simmons. The prisoners' problem and the subliminal channel. In D. Chaum, editor, *CRYPTO'83*, pages 51–67. Plenum Press, New York, USA, 1984.

[120] B. R. Waters. Efficient identity-based encryption without random oracles. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127. Springer, May 2005.