PROJET D'INITIATION À LA RECHERCHE

SCHÉMA EFFICACE DE TRAÇAGE DU TRAÎTRE A CLÉ PUBLIQUE

Encadré par : Duong Hieu Phan
Réalisé par : BOSSELUT Cédric
BENALI Mohamed
Mai 2017

Table des matières

Introduction

Dans la cryptographie multi-utilisateurs, le traçage des traitres est une notion importante, en effet un système de cryptographie multi-utilisateur correspond à un message chiffré une fois, qui puisse être déchiffré par plusieurs clés. Plus usuellement on peut regarder le système du décodeur TV de télévision payante, en effet le message c'est à dire le flux de la chaine est chiffré une fois à l'émission, donc impossible à regarder si on capte le signal, mais chaque client de la chaine de télévision va recevoir une clé via un décodeur qui sera différente des autres clients, et pourtant tous les clients pourront déchiffrer le flux et donc regarder la chaine. Un "traitre" est alors une personne qui utilise des clés officielles pour créer une fausse clé qui déchiffrera le flux, la notion de traçage des traitres est donc de retrouver si une clé est officielle ou non, et de retrouver les clés originales qui ont permis de créer la fausse clé.

Nous allons donc présenter dans ce projet quelques moyens de traçage basés sur le modèle de Dan Boneh et Matthew Franklin.

Chapitre 1

Schéma de traçage

1.1 Définition :

1.1.1 modèle de traçage du traître

La clé publique du schéma de cryptage est un système de chiffrement à clé publique dans lequel il y a une unique clé de chiffrement et plusieurs clé pour décrypter.
le schéma est composé de quatre éléments :

Génération des clés

L'algorithme de génération de clés est le suivant :

Entrée :
- Un paramètre sécurisé s.
- Un nombre l de clés privées à générer.
- Un nombre k.

Sortie :
- Clé publique e (pour chiffrer).
- Une liste de clés de décryptage privée $d_1, ..., d_l$.

On peut utiliser n'importe quelle clé pour déchiffrer un texte chiffré crée en utilisant la clé de chiffrement.

Encodage

Entrée :
- Une clé de décryptage publique e.
- Un message M.

Sortie :
- Un texte chiffré C.

Décodage

Entrée :
- Texte chiffré C.
-Une clé de décryptage d_i.

Sortie :
- Message M.

Traçage

Entrée :
- Tous les bits aléatoires utilisés
pendant la génération de la clé
(ce qui peut être vu comme la clé de traçage)
- Un décodeur pirate D.

Sortie :
- $i \in \{1, ..., l\}$.

On modélise le pirate par un algorithme qui prend en entrée une clé publique et K clé de décryptage aléatoire $d_1, ..., d_k$ de l clé. En utilisant la clé publique et les k clé privée, le pirate crée une décodeur pirate (ou un système de décryptage) D. Le décodeur du pirate doit décrypter correctement tout mais pas forcement une fraction négligeable du texte chiffré validé généré par l'algorithme encodage.

Traçage de la boite noire

La première notion de cet algorithme est que le traceur utilise le décodeur du pirate comme u oracle de décryptage. Prendre un texte C, le décodeur renvoie soit "invalide", c'est à dire que C n'est pas un chiffré valide, soit "valide" ce qui veut dire que le texte clair déchiffré de C est vrai. Dans ce modèle le traceur voit la totalité de la sortie du décodeur du pirate qui est le décryptage de C.
Il existe deux types de Traçage de la boite noire

Traçage de la boite noire à accès total

Dans ce modèle, le traceur demande au décodeur pirate de lire C. Le décodeur retourne soit le texte clair M qui est normalement le clair de C, soit "invalide".

Traçage de la boite noire à accès minimal

Dans ce modèle le but du décodeur pirate est la paire $< C, M >$ où C est le texte chiffré et M un peu du texte clair. La question nous donne accès à une troisième partie que le décodeur exécute et qui prend en entée C. Si le décodeur retourne M, la troisième partie retourne "valide". Sinon elle retourne "invalide". En général, la troisième partie limite l'accès aux du décodeur pirate.

Représentation

Notre schéma de traçage du traître est relié au problème de représentation, lorsque $y = \prod_{i=1}^{2k} h_i^{\delta_i}$, on dit que $\delta_1, ..., \delta_{2k}$ est la représentation de y dans la base $h_1, ..., h_{2k}$.
Si $\overline{d_1}, ..., \overline{d_m}$ sont les représentations de y dans la même base, donc il y a une combinaison convexe des représentations : $\overline{d} = \sum_{i=1}^{m} \alpha_i \overline{d_i}$ où $\alpha_i, ..., \alpha_m$ sont des scalaires tq $\sum_{i=1}^{m} \alpha_i = 1$.

1.2 schéma d'encodage

Soit S un paramètre de sécurité et K le nombre maximal de complices, le schéma de traçage de l'encodage traître défend contre la complicité de k parties. On veut générer une clé publique et l clés privés correspondantes. On doit avoir $l \geq 2k + 2$ si ($l < 2k + 2$, on prend $l = 2k + 2$ est on génère l clé privé).

Notre schéma utilise un certain espace linéaire de traçage σ qui est une collection de l éléments dans $\mathbb{Z}^{2k}$. La construction de l'ensemble σ et les propriétés qu'il doit satisfaire sont décrit dans la prochaine section. Les l éléments dans σ sont des vecteurs de taille 2k. L'ensemble σ est fixé et est publique.
Soit G_q un groupe d'ordre q premier. La sécurité de notre schème est reliés à la difficulté du calcule du logarithme discret dans G_q. Plus précieusement, la sécurité est basé sur la difficulté de la Décision Diffie-Hellman (DDH) dans G_q.

Génération de clés

1. Soit $g \in G_q$
2. Pour $i = 1, .., 2k$, on prend aléatoirement un $r_i \in \mathbb{F}_q$ et on calcule $h_i = g^{r_i}$
3. La clé publique est : $< y, h_1, ..., h_{2k} >$, où $y = \prod_{i=1}^{2k} h_i^{\alpha_i}$ pour $\alpha_1, ..., \alpha_{2k} \in \mathbb{F}_q$ aléatoires
4. La clé privé est un élément $\theta_i \in \mathbb{F}_q$ tq $\theta_i * \gamma^{(i)}$ est une représentation de y en respectant la base $h_1, ..., h_{2k}$

La ième clé, θ_i, est dérivée du ième élément $\gamma^{(i)} = (\gamma_1, ..., \gamma_{2k}) \in \sigma$, avec $\theta_i = \frac{(\sum_{j=1}^{2k} r_j \alpha_j)}{(\sum_{j=1}^{2k} r_j \gamma_j)} (mod\ q)$.
Pour simplifier, on va souvent définir la clé privée comme la représentation $\overline{d_i} = \theta_i * \gamma^{(i)}$, cependant, les θ_i doivent restés secrets tant que le code γ est publique.

Encodage

Pour chiffrer un message M dans G_q, on prend a aléatoire $a \in \mathbb{F}_q$, le texte chiffré est $C =< M * y^a, h_1^a, ..., h_{2k}^a >$.

Décryptage

Pour déchiffrer le texte chiffré $C =< S, H_1, ..., H_{2k} >$ en utilisant la ième clé privé, θ_i, on calcule $m = \frac{M}{U^{\theta_i}}$ où $U = \prod_{j=1}^{2k} H_j^{\gamma_j}$.
Ici $\gamma^{(i)} = (\gamma_1, ..., \gamma_{2k}) \in \sigma$ d'où dérive les θ_i. Le coût du calcule de U est beaucoup plus petit que $2k + 1$. Notamment U peut être calculer sans la connaissance de la clé privée.
Par tout les clés privés θ_i, on peut déchiffrer correctement le texte chiffré.

En prenant un texte chiffré $C =< M * y^a, h_1^a, ..., h_{2k}^a >$, le rendement du décryptage est de $M * y^a / U^{\theta_i}$ où $U = \prod_{j=1}^{2k} (h_j^a)^{\sigma_j}$. Alors,

$$U^{\theta_i} = (\prod_{j=1}^{2k} g^{a r_j \gamma_j})^{\theta_i} = (g^{\sum_{2k}^{j=1} r_j \gamma_j})^{\theta_i} a = (g^{\sum_{j=1}^{2k} r_j \alpha_j})^a = (\prod_{j=1}^{2k} h_j^{\alpha_j})^a = y^a$$

Preuve de sécurité :
Maintenant, on va vous montrer que notre schème de cryptage est bien sécurisé contre des adversaires passives en prenant en considération la difficulté du problème (DDH) dans G_q. L'hypothèse suppose que dans G_q, il n'existe pas de test statistique en temps polynomial qui peut distingué ; avec des avantages non négligeables entre les deux distributions $D =< g_1, g_2, g_1^a, g_2^a >$ et $R =< g_1, g_2, g_1^a, g_2^b >$ où g_1, g_2 sont choisit aléatoirement dans G_q et $a, b \in \mathbb{F}_q$.
Théorème :
Le schème de chiffrement est sémantiquement sécurisé contre les adversaires passifs comprenant la difficulté des DDH dans G_q.
Preuve :
On suppose le contraire. Donc il existe un adversaire qui peut, à partir de la clé publique $< y, h_1, ..., h_{2k} >$, produire deux messages $M_0, M_1 \in G_q$. Donner le cryptage C de l'un des messages permet à l'adversaire avec des avantages ϵ non- négligeables lequel des deux messages a eu. On voit qu'un tel adversaire peut se permettre de décider DDH dans G_q. Sachant $< g_1, g_2, u_1, u_2 >$, on suit les étapes suivantes pour déterminer si $< g_1, g_2, u_1, u_2 >$ est pris de R ou de D.

ÉTAPE 1 : Prendre $r_2, ..., r_{2k} \in \mathbb{F}_q$ aléatoires. Poser $y = g_1$, $h_1 = g_2$ et $h_i = g^{r_i}$ pour $i = 2, ..., 2k$.

ÉTAPE 2 : Envoyer $< y, h_1, ..., h_{2k} >$ à l'adversaire. Ce dernier retourne $M_0, M_1 \in G_q$.

ÉTAPE 3 : Prendre un b aléatoire $\in \{0, 1\}$. et construire le texte chiffré

$$C =< M_b u_1, u_2, u_2^{r_2}, ..., u_2^{r_{2k}} >$$

ÉTAPE 4 : Envoyer le texte chiffré C à l'adversaire. Celui-ci retourne $b' \in \{0, 1\}$.

ÉTAPE 5 : Si $b = b'$ on retourne D et R sinon.

On remarque que si $< g_1, g_2, u_1, u_2 >$ est choisit dans D, alors le texte chiffré

C est un chiffrement de M_b. Si le quadruple est dans R, le texte chiffré est un chiffrement de $M_b g_1^{(a_1-a_2)}$ où $u_1 = g_1^{a_1}$ et $u_2 = g_2^{a_2}$. En d'autres termes, le texte chiffré est obtenue à partir d'un message aléatoire. Néanmoins, on a $b = b'$ avec une probabilité égale à $\frac{1}{2}$. Avec des arguments standards, l'adversaire à une probabilité de succès non-négligeable ce qui veut dire une probabilité de succès non négligeable pour décider DDH.

Construction de nouvelles représentations :
Pour décrypté, il suffit de savoir n'importe quelle représentation de y dans la base $h_1, ..., h_{2k}$. On a noté que si $\overline{d_1}, ..., \overline{d_m} \in \mathbb{F}_q^{2k}$ sont des représentations de y, alors toute combinaison convexe est une nouvelle représentation de y. Le lemme suivant montre que les uniques nouvelles représentations de y qu'on peut construire efficacement à partir de $\overline{d_1}, ..., \overline{d_m} \in \mathbb{F}_q^{2k}$.
Lemme :
Soit $< y, h_1, ..., h_{2k} >$ une clé publique. On suppose qu'un adversaire a la clé publique et m clé privé $\overline{d_1}, ..., \overline{d_m} \in \mathbb{F}_q^{2k}$ pour $m < 2k-1$. Si l'adversaire arrive à générer une nouvelle représentation $\overline{d}$ de y dans la base $h_1, ..., h_{2k}$ qui n'est pas une combinaison convexe de $\overline{d_1}, ..., \overline{d_m}$ alors l'adversaire peut calculer le logarithme discret dans G_q.
Preuve :
Soit g un générateur de G_q.
On suppose qu'on a $z = g^x$.
On va voir comment l'adversaire va calculer x.
On choisit aléatoirement $a, b, r_1, ..., r_m, s_1, ..., s_{2k} \in \mathbb{Z}_q$.
On construit l'ensemble $\{h_1, ..., h_{2k}\}$ où $h_i = z^{r_i} g^{s_i}$ pour $m+1 \leq i \leq 2k$. On calcul $y = z^a g^b$. On cherche m solutions linéaires indépendantes $\overline{\alpha_1}, ..., \overline{\alpha_m}$ de $\overline{\alpha}.\overline{s} = b \ mod \ q$. Tant qu'on conserve les m premières entrées de chaque $\overline{\alpha}_i$ inchangés. Ses m vecteurs sont des représentations de y dans la base $h_1, ..., h_{2k}$. On suppose que l'adversaire peut trouver une autre représentation $\overline{\beta}$ qui n'est pas une combinaison convexe de $\alpha_1, ..., \alpha_m$. Puisque ce n'est pas une combinaison convexe, on doit avoir $\beta_1 r_1 + ... + \beta_m r_m = a' \neq a \ mod \ q$, mais $a'x + \overline{\beta}.\overline{s} = ax \ mod \ q$ ce qui veut dire que $\log x = (\overline{\beta}.\overline{s})(a - a')^{-1}$.

Chapitre 2

Traçage des boites non-noires

Le terme boite non-noire signifie que nous pouvons "voir" l'intérieur du décodeur au point de trouver la fausse clé utilisée. L'algorithme de traçage va donc permettre à partir de la clé trouvée $\bar{d}$, de trouver une clé originale qui a servi à créer cette fausse clé. En effet on a vu que la seule façon de créer une fausse clé était de faire une combinaison convexe de vraies clés, l'algorithme de traçage va donc permettre de retrouver une clé parmi celles qui ont servi à faire la combinaison convexe, et donc de trouver un "traître" qui a permis l'existence du décodeur.

2.1 L'algorithme

La construction des clés privées est faite à partir d'un ensemble Γ de l mots de code et chaque utilisateur reçoit une clé privée $\bar{d}_i \in \mathbb{F}_q^{2k}$ qui est un multiple d'un mot du code de Γ. Soit $\bar{d}$ un point dans l'espace vectoriel engendré par k mots du code $\gamma^1, ..., \gamma^k \in \Gamma$, il y a alors au moins un de ces k mots de code qui doit être parmi ceux qui ont fabriqué $\bar{d}$, et donc on peut retrouver la clé qui a servi à créer cette fausse clé. L'algorithme va donc prendre en entrée $\bar{d}$, la fausse clé et sortir γ, ou les γ^i qui sont dans la combinaison linéaire de $\bar{d}$.

Pré-requis L'ensemble Γ contient l mots de code. Avec q un grand nombre premier tel que $q > max(l, 2k)$, on peut construire la matrice de taille $(l - 2k) \times l$ dont les lignes correspondent à un polynôme de degré maximum

$l-2k-1$ évalué aux point $1,...,l$:

$$A = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 2 & 3 & \dots & l \\ 1 & 2^2 & 3^2 & \dots & l^2 \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 1 & 2^{l-2k-1} & 3^{l-2k-1} & \dots & l^{l-2k-1} \end{pmatrix} mod\, q$$

Maintenant on prend $b_1, ..., b_{2k}$ une base de l'espace vectoriel qui satisfait $A\bar{x} = 0\, mod\, q$ et on place les vecteurs obtenues en colonne dans la matrice B, qui sera donc de taille $l \times 2k$:

$$B = \begin{pmatrix} b_1 & b_2 & \dots & b_{2k} \end{pmatrix}$$

On définit donc notre Γ comme l'ensemble des lignes de B, cet ensemble contient donc l éléments, des mots de code de longueur $2k$. On peut remarquer qu'il est possible de construire directement un mot de code sans construire tout l'ensemble avec l'interpolation de Lagrange, en l opérations arithmétiques modulo q.

Calculs Maintenant que nous avons Γ, soit $\bar{d}$ un vecteur formé d'une combinaison linéaire d'au plus k vecteurs de Γ. On va déterminer pour $\bar{d}$ un vecteur $\bar{w} \in \mathbb{F}_q^l$ de poids de Hamming au plus k tel que $\bar{w}B = \bar{d}$, ce vecteur contiendra les coefficients de la combinaison linéaire des lignes de B.

On commence par trouver un vecteur $\bar{v} \in \mathbb{F}_q^l$ tel que $\bar{v}.B = \bar{d}$, il en existe plusieurs mais celui qu'on va prendre n'a pas d'importance, cependant si B est sous forme systématique, on peut prendre directement $\bar{v} = (\bar{d}||0...0)$. On a alors $(\bar{v} - \bar{w}).B = 0$, et on sait que $\bar{v} - \bar{w}$ est dans l'espace vectoriel engendré par les lignes de A, donc il doit exister un polynôme $f \in \mathbb{F}_q[X]$ de degré maximum $l - 2k - 1$ tel que $\bar{v} - \bar{w}$ soit $(f(1), ..., f(l))$.

On sait que $\bar{w}$ a un poids de Hamming d'au plus k, et que $(f(1), ...f(l))$ est égal à $\bar{v}$ partout sauf pour k éléments. On utilise alors l'algorithme de Berlekamp (détaillé ci-dessous) pour trouver f, le polynôme trouvé nous donne le vecteur $\bar{v} - \bar{w}$, qui nous donne $\bar{w}$

L'algorithme de Berlekamp L'algorithme de Berlekamp prend en entrée le vecteur $\bar{v} \in \mathbb{F}_q^l$ et renvoie le polynôme f cherché.

— Soit g, un polynôme de degré maximum k, tel que $\forall i = 1, ..., l$, si $f(i) \neq v_i$ alors $g(i) = 0$. On a alors $f(i)g(i) = g(i)v_i$

— Le polynôme fg a un degré maximum de $l-k-1$, on a alors l équations, une pour chaque i, et l variables qui sont les coefficients des polynômes unitaires g et fg.
— Soit h et g des solutions, g polynôme non nul de degré au plus k et h polynôme de degré au plus $l-k-1$
— On sait que $f(i) = v_i$ en $l-k$ points, on a donc $h(i) = g(i)v_i = g(i)f(i)$, d'où $f = h/g$

2.2 Propriétés de l'algorithme

Aucune erreur Cet algorithme de traçage dans le cas où on a réussit à avoir la fausse clé n'est pas probabiliste, la clé qu'il retourne a donc servi à faire la fausse clé, il n'y a aucune chance d'erreur dans le résultat.

Limite pour l'algorithme S'il y a plus de k clés qui ont servies à créer la fausse clé, alors l'algorithme de Berlekamp ne pourra pas retrouver le polynôme f, et donc nous ne pourrons pas retrouver une clé qui a servie à la clé pirate. Au dessus de cette limite, on peut utiliser les résultats de Guruswami et Sudan qui améliore les résultats du code de Reed-Solomon pour trouver une liste de polynômes pour f, si il y a au maximum $2k-1$ clés qui ont été utilisées.

Temps d'exécution Cet algorithme demande de résoudre un système linéaire de dimension l, l'algorithme le plus rapide à l'heure actuelle est celui de Pan qui permet de calculer en $O(l \log l \log\log l)$ opérations.

Chapitre 3

Traçage des boites noires à accès minimal

3.1 Concept de confirmation de boite noire

Dans le chapitre précédent, on pouvait directement trouver la clé utilisée dans le décodeur, maintenant on va regarder le cas où on ne peut pas extraire la clé, le concept de la confirmation de boite noire est de savoir si une personne suspectée est bien à l'origine du décodeur pirate trouvé. L'algorithme de confirmation de boite noire reçoit la clé publique et des bits aléatoires utilisés pendant la génération des clés, un accès en boite noire au décodeur pirate trouvé ainsi qu'une liste de clés que l'on suspecte d'avoir participé à la création du décodeur. L'algorithme va donc regarder chaque clé suspecté et dire si cette clé a été utilisée ou non.

Plus formellement, on écrit $T_{suspect}$ les clés suspectées et T_D les clés utilisées dans la création du décodeur pirate D. Alors l'algorithme renverra pour chaque $\bar{d} \in T_{suspect}$:

— "*non coupable*" si $\bar{d}$ n'a pas été utilisée.
— "$\bar{d}$ *est coupable*" si $\bar{d} \in T_D$

La sortie de l'algorithme de confirmation de boite noire doit donc respecter deux conditions :

1. **La confirmation** : Si $T_D \subseteq T_{suspect}$ alors l'algorithme de confirmation doit désigner au moins une clé $\bar{d} \in T_{suspect}$ comme coupable
2. **La solidité** : Si l'algorithme de confirmation donne une clé $\bar{d}$ comme coupable alors $\bar{d} \in T_D$

On garde l'hypothèse que le nombre de clés pirates est inférieur à k la limite de clés dans la collusion. Avant d'utiliser l'algorithme on ne sait pas si $T_D \subseteq T_{suspect}$ donc par la condition de solidité (2), si $T_D \nsubseteq T_{suspect}$, le pirate ne peux pas accuser une utilisateur innocent en trompant l'algorithme.
Cet algorithme de confirmation permettra d'avoir un algorithme de traçage de boite noire avec la méthode suivante :

On exécute l'algorithme de confirmation sur toutes les $\binom{l}{k}$ coalitions possibles, où l est le nombre total d'utilisateurs du système. Avec la propriété de confirmation (1), si une coalition essayé contient une coalition coupable, les membres coupables de la coalition essayé seront repérés, on peut donc trouver un coupable en examinant une coalition où il est mais qui n'est pas la vraie coalition de pirates. Par la propriété de solidité (2), si une clé $\bar{d}$ est désignée coupable, alors on peut dire que cette clé est membre de la coalition coupable avec un très faible taux d'erreurs. On obtient donc un algorithme de traçage de boite noire de complexité $O(\binom{l}{k}k^2)$.
Ce n'est donc pas un algorithme très efficace mais il montre que le traçage est possible ce qui signifie qu'il est possible d'inculper un pirate sans connaitre la clé du décodeur pirate.

Comme on l'a dit, il est possible de créer un algorithme de confirmation de boîte noire et l'idée est la suivante. Soit $T_{suspect} = \{\bar{d}_1, \ldots, \bar{d}_k\}$ les k clés suspectées, g un générateur de G_q et $(y, h_1, \ldots, h_{2k})$ une clé publique. Pour confirmer nos suspicions sur les clés suspectées, on va demander au décodeur de déchiffrer un message chiffré aléatoire $\tilde{C} = (S, g^{z_1}, \ldots, g^{z_{2k}})$ où $\bar{z} = (z_1, \ldots, z_{2k})$ vérifie $\bar{z} \cdot \bar{d}_i = w, \forall i \in T_{suspect}$ avec w un élément aléatoire de $\mathbb{F}_q$, ce message chiffré n'est pas valide car la plupart du temps $(g^{z_1}, \ldots, g^{z_{2k}})$ n'est pas de la forme de $({h_1}^r, \ldots, {h_{2k}}^r)$ pour tout r. On montre que quand $T_D \nsubseteq T_{suspect}$ le décodeur pirate ne fait pas la distinction entre ce message et un message correct. Le décodeur répond alors avec $A = S/g^{\bar{z} \cdot \bar{d}}$ où $\bar{d}$ est une représentation de y dans l'enveloppe convexe de $\bar{d}_1, \ldots, \bar{d}_k$. Donc si $T_D \nsubseteq T_{suspect}$ le décodeur pirate répondra $A = S/g^w$. Et si les clés suspectés n'appartiennent pas aux clés utilisés dans le décodeur ne pourra pas interpréter le faux message. On peut donc dire si $T_{suspect}$ contient une des clés pirates, et en allant un peu plus loin il est possible de savoir quelle clé suspectée $\bar{d} \in T_{suspect}$ est possédé par le pirate.

3.2 Notations

Pour la suite, et notamment pour l'algorithme, nous allons avoir besoin de plusieurs nouvelles définitions, ainsi que de nouveaux outils :

$T_i = \{\bar{d_1}, \ldots, \bar{d_k}\} \subseteq T_{suspect}$ pour $i = 0, \ldots, k$ des partitions de l'ensemble des clés suspectées, où T_0 représente l'ensemble de clé vide.

$CT_i(W), i = 0, \ldots, k$ avec $W \in G_q$ est défini tel que : $CT_i = \{C = (S, H_1, \ldots, H_{2k})$ tel que $S \in G_q$ et $W = \prod_{i=0}^{2k} H_i^{\delta_i}, \forall \bar{d} = (\delta_1, \ldots, \delta_{2k}) \in T_i\}$.
On définit donc $CT_i = \cup_{W \in G_q} CT_i(W)$

Pour $i = 0, \ldots, k$, on définit la distribution CW_i de la paire (W, C) tel que : on prend un élément W aléatoire de G_q puis on prend un élément aléatoire $C \in CT_i(W)$ et ressort (W, C)

Pour $i = 0, \ldots, k$, on définit $P_{D,i}$ tel que : $P_{D,i} = Pr[D(C, S/W) = valide]$ pour $(W, C) \in CW_i$ où $C = (S, H_1, \ldots, H_{2k})$ et (W, C) est choisi dans la distribution CW_i et $D(C, S/W)$ détermine la réussite ou l'échec du décodeur pirate qui prend en entrée $C, S/W$.

Pour une clé publique donnée $(y, h_1, \ldots, h_{2k})$, on se réfère à l'ensemble $\{(y, {h_1}^r, \ldots, {h_{2k}}^r) : r \in \mathbb{F}_q\}$ comme l'ensemble des chiffrés valides. On se réfère à tous les autres chiffrés comme chiffrés invalides. On définit la distribution CW_{valide} des pairs (W, C) tel que :

1. On prend $r \in \mathbb{F}_q$ et $S \in G_q$ aléatoires
2. $W = y^r$ et $C = (S, {g_1}^r, \ldots, {g_{2k}}^r)$
3. Renvoie (W, C)

On remarque que C est un chiffré valide et que son clair est S/W

On remarque aussi que pour n'importe quel i on peu efficacement prendre un échantillon de la distribution CW_i tel que :

1. On prend un $w \in \mathbb{F}_q$ et on met $W = g^w$
2. On prend un vecteur aléatoire $\bar{z} = (z_1, \ldots, \bar{z}_{2k}) \in {\mathbb{F}^{2k}}_q$ tel que $\bar{z} \cdot \bar{d} = w \forall \bar{d} \in T_i$
3. On prend un $S \in G_q$ aléatoire
4. On met $C = (S, g^{z_1}, \ldots, g^{z_{2k}})$ et renvoie (W, C)

On affirme que $P_{D,0} \leq 1/q$ et on voit que quand (W, C) est choisi de CW_0 nous avons que W est indépendant de C. Il découle que $D(C, S/W)$ a une probabilité d'au plus $1/q$ d'être valide.

3.3 L'algorithme

On peut maintenant décrire un algorithme de traçage sur un décodeur D avec un accès minimal.

Pour chaque i de 0 à k :
  Pour j de 1 à λ :
    Prendre une paire (W_j, C_j) aléatoire et indépendante
     choisie suivant la distribution CW_i
  Exécuter le décodeur D sur les entrées $C_1, \ldots, C_\lambda$:
    On compte c_i le nombre de fois que $D(C_j, S_j/W_j) = valide$
  $\tilde{p}_i = c_i/\lambda$
Si $\tilde{p}_k \leq 3/4$: Retourner $non - coupable$
Sinon comme $P_{D,0} \leq 1/q$, on a $\tilde{p}_0 \leq 1/4$ avec une grande probabilité, donc :
$1/2 \leq \mid \tilde{p}_k - \tilde{p}_0 \mid = \mid \sum_{i=1}^{k} \tilde{p}_i - \tilde{p}_{i-1} \mid \leq \sum_{i=1}^{k} \mid \tilde{p}_i - \tilde{p}_{i-1} \mid$
Il existe donc pour un j entre 1 et k tel que : $\tilde{p}_i - \tilde{p}_{i-1} \leq \dfrac{1}{2k}$
Renvoyer $\bar{d}_j \in T_{suspecte}$ est *coupable*

L'algorithme demande $\lambda(k+1)$ requêtes au décodeur pirate D, cet algorithme n'a pas besoin de la clé publique ou des bits utilisés surant la génération des clés, seulement de k clés privées de la coalition suspectée.

Théorème L'algorithme présenté est un algorithme de confirmation de boite noire.
Pour le prouver, il faut montrer que cet algorithme vérifie les 2 propriétés énumérés dans la partie 3.1, c'est à dire **La confirmation** et **La solidité**, nous n'allons pas le prouver entièrement mais donner les idées qui permettent de comprendre cet algorithme.

3.4 L'algorithme vérifie les conditions

3.4.1 La confirmation

Cette propriété est celle qui permet de dire que l'algorithme ne se trompera pas si on teste les clés du pirates sont dans les clés testées. Il faut donc montrer que un pirate in-traçable n'existe pas.

On dit qu'une pirate P est $\epsilon - non - traable$ si avec la clé publique et un ensemble de clés $T_D = \{\bar{d}_1, \ldots, \bar{d}_k\}$, le pirate peut créer un décodeur pirate D avec les 2 propriétés suivantes :
(1)D peut décoder correctement tous les chiffrés valides.
(2)Quand $T_D \subseteq T_{suspect}$ l'algorithme retourne $non - coupable$ sur le décodeur D avec une probabilité d'au moins ϵ

L'idée est alors de prouver qu'avec $\epsilon > 0$ et $\lambda = 1$, si un pirate P est $\epsilon - non - traable$ alors on peut s'en servir pour distinguer des tuples de Diffie-Hellman aléatoires. C'est à dire dans un groupe cyclique G d'ordre g, étant donné $(g, g^a, g^b), a, b \in \{0, \ldots, q-1\}$ on peut calculer g^{ab} facilement.
En effet, on suppose que $T_D \subseteq T_{suspect}$, alors pour un décodeur D donné crée par P, si l'algorithme retourne$non - couapble$ avec une probabilité d'au moins ϵ alors $\tilde{p}_k \leq 3/4$ avec une probabilité d'au moins ϵ. Comme $\lambda = 1$ on a $P_{D,k} < 1 - \epsilon$. Ce qui signifie que le pirate est capable de construire un décodeur pirate qui décrypte tous les chiffrés valides, mais pour un $C \in CT_k(W)$ aléatoire, retourne *invalide* pour une entrée $C, S/W$ avec une probabilité d'au moins ϵ.

Pour montrer comment le pirate peut être utilisé pour résoudre l'hypothèse décisionnelle de Diffie-Hellman dans G_q. l'idée est de construire un algorithme qui différencie les tuples aléatoires des tuples de Diffie-Hellman.

3.4.2 La solidité

Pour montrer l'autre propriété à savoir la solidité, il est a nouveau question de ramener la possibilité que l'algorithme désigne une clé coupable alors qu'elle ne l'est pas au problème de Diffie-Hellman.
Il faut d'abord vérifier que si on prend D un décodeur pirate construit avec un ensemble T_D d'au plus k clés privées, et $T_{suspect}$ l'ensemble des clés suspectées d'au moins k clés. Soit $\epsilon > 0$ et $\lambda = 64klog \div k\epsilon$, alors quand l'algorithme sur D renvoie $\bar{d}coupable$, alors $\bar{d} \in T_D$ avec une probabilité de $1 - \epsilon$.

Et soit $(y, h_1, \ldots, h_{2k})$ une clé publique et soit $T_{suspect} = \{\bar{d}_1, \ldots, \bar{d}_k\}$ ensemble de k clés privées utilisées pour définir les ensembles CT_i avec $i = 0, \ldots, k$. On suppose le pirate en possession de cette clé publique et d'un ensemble aléatoire de k clés privées T_D, est capable de construire un décodeur D tel que : $\exists j_0 \in \{1, \ldots, k\}, \bar{d}_{j_0} \notin T_D et \mid P_{D,j_0} - P_{D,j_0-1}| \geq \epsilon$. Alors le pirate peut résoudre l'hypothèse décisionnelle de Diffie-Hellman.
Il est alors possible de définir un algorithme qui permet à un décodeur comme celui-ci de différencier un tuple aléatoire d'un tuple de Diffie-Hellman.

Conclusion

On a donc vu les concepts généraux du traçage du traitre et du modèle de cryptographie multi-utilisateurs utilisés ici, avec notamment la génération des clés et le concept de la représentation des clés. Nous avons vu ensuite 2 algorithmes de traçage, un dans le cas où on a un accès total au décodeur et un où on a un accès minimal.
Le thème de la cryptographie multi-utilisateur est vaste, et même en se concentrant sur les problèmes du traçage des traitres, il est difficile d'en faire le tour, mais ce sont des concepts intéressants et vraiment différents de ce qu'on à connu jusqu'à présent et pourtant qui utilisent les outils mathématiques que nous connaissons, c'est pour cette raison que ce sujet de Projet d'initiation à la Recherche était une bonne expérience.