Padding optimal pour le chiffrement et la signature

Sommaire

La sécurité forte va souvent de pair avec d'importantes contraintes concernant la construction des schémas cryptographiques : la sécurité sémantique implique que le chiffrement soit probabiliste et la sécurité contre les falsifications existentielles implique que les schémas de signature aient des redondances.

Il y a quelques années, Coron *et. al* [36] ont suggéré une construction commune pour le chiffrement et la signature : le « *padding universel* ». Cependant, comme le padding universel doit contenir à la fois un aléa et une redondance, le chiffrement et la signature qui en résulte ne sont pas optimaux.

Dans ce chapitre, nous raffinons cette notion de padding universel en permettant à une des parties d'être une chaîne aléatoire ou une chaîne de zéros, *i.e.* une certaine redondance. Ceci nous aide à construire, à partir d'un padding unique, un chiffrement et une signature efficaces : dans un premier temps, dans le modèle de la permutation aléatoire, et dans un deuxième temps, dans celui de l'oracle aléatoire. Dans les deux cas, nous étudions la taille effective des paramètres pour un niveau de sécurité spécifique et nous montrerons par la suite que le premier schéma est optimal en taille.

6.1 Introduction

Pour le chiffrement à clef publique, la notion de base reste la sécurité sémantique selon des attaques à chiffrés choisis [104]. De la même façon, pour la signature, l'exigence est désormais la sécurité contre les falsifications existentielles face aux attaques à messages choisis [61]. Cependant, la sécurité forte ne suffit pas, elle doit, de surcroît, être obtenue de manière efficace selon plusieurs critères : le temps de calcul, la taille du chiffré/de la signature, et la taille du code.

Les deux premiers critères cités sont courants. En effet, il existe dans la littérature des paddings rapides pour le chiffrement [10, 94] et pour la signature [11]. Concernant la bande passante, dans le chapitre précédent, nous avons proposé un padding optimal qui évite la redondance dans le chiffrement. De nombreux schémas de signature avec « reconstitution de message » (*message recovery* en anglais) [92, 86, 11] peuvent améliorer la bande de passante sans pour autant atteindre les solutions optimales en raison de la présence d'aléa et de redondance. Une exception à remarquer, fruit d'une idée récente de Katz et Wang, atteint une bonne réduction de sécurité en utilisant la construction FDH (« Full-Domain Hash » [11]) avec un seul bit additionnel dépendant du texte clair [71].

Pour le troisième critère, *i.e.* la taille du code, Coron *et. al* [36] ont introduit la notion de padding universel : la taille du code est réduite grâce à l'utilisation d'un padding commun pour le chiffrement et la signature. Ils ont proposé une variante de PSS, appelée PSS-ES pour la construction du padding universel. D'autres solutions, dont celle de Komano et Ohta [76], ont ensuite été proposées. Cependant, dans toutes ces constructions, le chiffrement contient de la redondance et la signature est probabiliste.

6.1.1 Redondance et aléa

Comme présenté dans le chapitre précédent, afin d'atteindre la sécurité sémantique, un chiffrement doit être probabiliste. De plus, les redondances sont souvent ajoutées aux chiffrés pour une preuve de connaissance du texte clair ou « plaintext awareness » [10, 7, 37]. Nous avons également prouvé que la redondance pouvait être évitée, au moins dans le modèle de l'oracle aléatoire et dans celui du chiffrement idéal.

De la même façon, afin d'obtenir la sécurité contre les falsifications, une certaine

redondance doit être introduite dans le couple (message, signature) ou dans une chaîne unique en cas de reconstitution de message : sans la clef de signature, il est difficile de créer un couple (message, signature) qui satisfasse la redondance. La plupart des schémas de signature sont, de plus, probabilistes [111, 89, 11, 38] bien que ceci ne soit pas une condition nécessaire (la signature FDH est déterministe, cependant, la réduction n'est pas efficace).

6.1.2 Padding universel

Il convient de noter que le padding universel est applicable en même temps au chiffrement et à la signature, avec les mêmes clefs d'utilisateur : la clef publique est utilisée pour le chiffrement et pour la vérification ; la clef privée, pour le déchiffrement et pour la signature. Même si ceci n'est pas obligatoire, dans la suite, nous considérons ce cas qui est le pire.

En effet, dans le modèle de sécurité, nous considérons les attaquants (contre la sécurité sémantique ou la falsification existentielle) ayant accès au déchiffrement et à la signature. L'oracle de déchiffrement pourra donc les aider à forger des signatures, puisque la même clef est utilisée, et réciproquement au sujet de la sécurité du chiffrement.

6.2 Modèle de sécurité pour la signature

Les schémas de signature sont les versions électroniques des signatures manuscrites pour les documents numériques : la signature d'un utilisateur sur un message m est une chaîne qui dépend de m et de ses données publiques et secrètes. N'importe qui peut vérifier la validité de la signature en n'utilisant que des données publiques.

Nous rappelons brièvement les principales notions de sécurité pour la signature [61].

Un schéma de signature $\mathsf{S} = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ est défini par trois algorithmes :
- L'*algorithme de génération de clef* $\mathcal{K}$: sur l'entrée 1^k, où k est le paramètre de sécurité, l'algorithme probabiliste $\mathcal{K}$ génère un couple clef publique, clef secrète $(\mathsf{pk}, \mathsf{sk})$. On appelle k le paramètre de sécurité. Les tailles des clefs ainsi que les autres paramètres dans le schéma dépendent de ce paramètre.
- L'*algorithme de signature* $\mathcal{S}$: étant donné un message m (dans l'espace des textes clairs $\mathcal{M}$) et un couple de clef publique-clef secrète $(\mathsf{pk}, \mathsf{sk})$, $\mathcal{S}$ produit une signature σ. L'algorithme de signature peut être probabiliste.
- L'*algorithme de vérification* $\mathcal{V}$: étant donné une signature σ, un (ou une partie, éventuellement vide du) message m, et une clef publique pk, $\mathcal{V}$ vérifie si σ est une signature valide et finalement, extrait le message m. En règle générale, l'algorithme de vérification n'a pas à être probabiliste.

6.2.1 Infalsifiabilité

L'objectif de l'attaquant est de produire un nouveau couple (message, signature) valide. Cette production s'appelle la *falsifiabilité existentielle*. La sécurité contre ce type d'attaque est l'*infalsifiabilité existentielle* (dénotée EUF). En ce qui concerne le moyen d'attaque, on considère habituellement les *attaques adaptatives à messages choisis* (dénotée CMA). Dans ce modèle, l'attaquant peut demander au signataire de signer n'importe quel message et adapter ses questions aux réponses obtenues. A la fin, l'attaquant retourne un couple (m, σ), avec m n'ayant jamais été soumis à l'oracle de signature. L'attaquant gagne si σ est une signature valide du message m. L'objectif est de résister aux falsifications existentielles selon des attaques adaptatives à messages choisis (EUF/CMA), *i.e.* la probabilité de succès de tout attaquant $\mathcal{A}$ en un temps raisonnable est négligeable. Cette probabilité de succès est définie comme suit :

$$\mathsf{Succ}_{\mathsf{S}}^{\mathsf{euf/cma}}(\mathcal{A}) = \Pr\left[\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{K}(1^k), (m, \sigma) \leftarrow \mathcal{A}^{\mathcal{S}_{\mathsf{sk}}}(\mathsf{pk}) : \mathcal{V}(\mathsf{pk}, m, \sigma) = 1\ \right].$$

6.2.2 Signature et chiffrement

Nous cherchons à construire un padding unifié qui peut être utilisé en même temps et avec une même primitive pour le chiffrement et pour la signature. L'attaquant a pour objectif soit de construire une falsification existentielle (*i.e.* casser EUF) contre le schéma de signature, soit de casser la sécurité sémantique du schéma de chiffrement (*i.e.* casser IND). Pour y parvenir, il dispose de l'accès adaptatif aux oracles de signature et de déchiffrement. Il s'agit d'une combinaison des attaques contre le chiffrement et contre la signature, d'où la notation de l'attaque CMA + CCA.

6.2.3 Permutations sans griffe

Dans [71], Katz et Wang ont prouvé que, en utilisant des permutations à trappe induites par des permutations sans griffe (*claw-free permutations* en anglais), on peut obtenir une variante de FDH (en y ajoutant un bit supplémentaire) avec une réduction fine. Nous utilisons également cette technique pour notre construction. L'hypothèse de l'existence des permutations sans griffe semble être raisonnable. En effet, toute permutation à sens-unique aléatoirement auto-réductible peut être considérée comme une permutations sans griffe [42] et la quasi-totalité des exemples connus de permutations à sens-unique à trappe sont aléatoirement auto-réductibles.

Définition 46 (Permutations sans griffe) *Une famille de permutations sans griffe est un ensemble d'algorithmes* $\{\mathsf{Gen}; f_i; g_i | i \in I\}$*, tels que :*

- Gen *retourne un indice aléatoire* i *et une trappe* td.
- f_i, g_i *sont des permutations sur le même domaine* D_i.
- *il existe un algorithme d'échantillonnage efficace qui, pour chaque indice* i*, retourne une valeur aléatoire* $x \in D_i$.

– *étant donné la trappe* td, f_i^{-1} *(l'inverse de* f_i*) et* g_i^{-1} *(l'inverse de* g_i*) sont efficacement calculables.*

Une griffe est un couple (x_0, x_1) *tel que* $f(x_0) = g(x_1)$*. On dit que l'algorithme probabiliste* $\mathcal{A}$ (t, ϵ)*-casse une famille de permutations sans griffe si* $\mathcal{A}$*, exécuté en temps* t*, peut retourner une griffe avec une probabilité supérieure à* ϵ *:*

$$\Pr\left[\ (i, \mathsf{td}) \leftarrow \mathsf{Gen}(1^k), (x_0, x_1) \leftarrow \mathcal{A}(i)\ : f_i(x_0) = g_i(x_1)\ \right] \geq \epsilon.$$

Une famille de permutations sans griffe est dite (t, ϵ)*-sûre s'il n'existe pas d'algorithme qui puisse* (t, ϵ)*-casser cette famille.*

6.3 Padding optimal fondé sur des permutations aléatoires

Dans ce qui suit, nous proposons un padding universel dans le modèle de la permutation aléatoire, fondé sur notre construction présentée au chapitre précédent. Il est optimal pour signer et pour chiffrer, *i.e.* il utilise 82 bits de redondance pour signer et seulement 82 bits aléatoires pour chiffrer. Dans la section suivante, nous proposons une autre construction, fondée sur OAEP 3-tours [96], qui a été prouvée sûre dans le modèle de l'oracle aléatoire, et « presque » optimale (161 bits d'aléa au lieu de 82).

Le chiffrement et la signature utilisent une permutation $\mathcal{P}$, supposée ressembler à une permutation véritablement aléatoire. Notons k le paramètre de sécurité et $\varphi_{\mathsf{pk}} : \{0,1\}^n \rightarrow \{0,1\}^n$ une permutation à sens-unique à trappe (dont l'inverse est notée ψ_{sk}). Les messages à signer ou à chiffrer seront de taille $\ell = n-k-1$. Le symbole « $\|$ » dénote la concaténation de chaînes de bits. De plus, on identifie $\{0,1\}^k \times \{0,1\}^\ell \times \{0,1\}$ à $\{0,1\}^n$. Finalement, dans ce qui suit, $\mathsf{prf}()$ désigne une fonction pseudo-aléatoire.

6.3.1 Padding

Le padding est relativement simple, il prend comme entrées un bit γ, le message m et une donnée additionnelle r, et retourne $\mathsf{OPbP}(\gamma, m, r) = \mathcal{P}(\gamma \| m \| r) = t \| u$. L'inverse de cette opération est naturellement : $\mathsf{OPbP}^{-1}(t, u) = \mathcal{P}^{-1}(t \| u) = \gamma \| m \| r$.

Algorithme de chiffrement

L'espace des textes clairs est $\mathcal{M} = \{0,1\}^\ell$. Le chiffrement utilise une chaîne aléatoire $r \in \mathcal{R} = \{0,1\}^k$ et un bit aléatoire γ. Il retourne un texte chiffré c de $\{0,1\}^n$:

$$t \| u = \mathsf{OPbP}(\gamma, m, r) \qquad c = \varphi_{\mathsf{pk}}(t \| u).$$

Algorithme de déchiffrement

À partir du texte chiffré c, on calcule d'abord $t\|u = \psi_{\mathsf{sk}}(c)$, où $t \in \{0,1\}^k$ et $u \in \{0,1\}^{\ell+1}$, puis on retrouve le texte clair m contenu dans : $\gamma\|m\|r = \mathsf{OPbP}^{-1}(t,u)$.

Algorithme de signature

L'espace des textes clairs est $\mathcal{M} = \{0,1\}^\ell$, l'algorithme de signature retourne une signature σ de $\{0,1\}^n$: à partir du message $m \in \mathcal{M}$, on calcule $\gamma = \mathsf{prf}(m)$, puis $t\|u = \mathsf{OPbP}(\gamma, m, 0^k)$ et $\sigma = \psi_{\mathsf{sk}}(t\|u)$.

Algorithme de vérification

À partir de la signature σ, on calcule d'abord $t\|u = \varphi_{\mathsf{pk}}(\sigma)$, où $t \in \{0,1\}^k$ et $u \in \{0,1\}^{\ell+1}$, puis $\gamma\|m\|r = \mathsf{OPbP}^{-1}(t,u)$. Si $r = 0^k$, la vérification retourne « Correcte » et retrouve m, sinon elle retourne « Incorrecte ».

6.3.2 Analyse de sécurité

Dans le chapitre précédent, ce padding, sans le bit additionnel de Katz et de Wang, conduit à un schéma de chiffrement $\mathsf{IND/CCA}$ sûr dans le modèle de la permutation aléatoire. Cependant, ce bit additionnel ne rend que plus aléatoire la phase de chiffrement. Dans cette section, on étend ce résultat au scénario d'attaque $\mathsf{CMA} + \mathsf{CCA}$ ainsi qu'à la signature ($\mathsf{EUF/CMA} + \mathsf{CCA}$).

Théorème 47 *Considérons les attaquants $\mathcal{A}$ et $\mathcal{B}$ contre respectivement le chiffrement et la signature, selon une attaque à la fois à chiffrés choisis (avec accès à l'oracle de déchiffrement) et à messages choisis (avec accès à l'oracle de signature). Supposons qu'en temps τ et qu'après q_p, q_s, q_d requêtes respectivement aux oracles de permutation, de signature et de déchiffrement, $\mathcal{A}$ peut avoir un avantage ε_E pour casser la sécurité sémantique du schéma, ou $\mathcal{B}$ peut avoir un succès ε_S pour produire une falsification existentielle, alors la permutation φ_{pk} peut être inversée avec probabilité ε' en temps t' où :*

$$\varepsilon' \geq \varepsilon_E - \frac{(q_p + q_d + q_s + 1)^2}{2^{k+\ell+1}} - \frac{(q_d+1)^2}{2^\ell} - \frac{2q_p + q_d + q_s + 2}{2^k}$$

ou

$$\varepsilon' \geq \frac{1}{q_p + q_s + 1} \times \left(\varepsilon_S - \frac{(q_p + q_d + q_s + 1)^2}{2^{k+\ell+1}} - \frac{(q_d+1)^2}{2^\ell} - \frac{2q_p + q_d + q_s + 2}{2^k} \right).$$

En particulier, si la fonction φ_{pk} est induite par une famille de permutations sans griffe (t', ε')-sûre, la deuxième relation peut être ré-écrite par :

$$\varepsilon' \geq \frac{1}{2}\left(\varepsilon_S - \frac{(q_p + q_d + q_s + 1)^2}{2^{k+\ell+1}} - \frac{(q_d+1)^2}{2^\ell} - \frac{2q_p + q_d + q_s + 2}{2^k}\right).$$

où $t' \leq t + (q_p + q_d + q_s + 1)T_f$, et T_f est le temps pour une évaluation de φ_{pk}.

Preuve. On utilise la technique des jeux successifs pour simuler les oracles de permutation aléatoire, l'oracle de déchiffrement, le challengeur et l'oracle de signature.

JEU $\mathbf{J}_0$: Il s'agit du jeu réel d'attaque dans le modèle de la permutation aléatoire. L'attaquant a accès aux oracles de permutation aléatoire $\mathcal{P}$ et $\mathcal{P}^{-1}$, à l'oracle de déchiffrement $\mathcal{D}_{\mathsf{sk}}$ et à l'oracle de signature $\mathcal{S}_{\mathsf{sk}}$.

Concernant la sécurité du chiffrement, on considère un attaquant $A = (A_1, A_2)$ contre ce schéma selon une attaque IND-CMA + CCA. Après avoir pris connaissance de la clef publique (la description de la fonction φ_{pk}), A_1 retourne deux textes clairs (m_0, m_1). Puis, le challengeur fabrique un challenge : un bit aléatoire b est choisi et le challenge $c^\star$ de $m^\star = m_b$ est donné par : $c^\star = \mathcal{E}(\gamma^\star, m_b, r^\star) = \varphi_{\mathsf{pk}}(\mathcal{P}(\gamma^\star, m_b, r^\star))$, où $r^\star \xleftarrow{R} \{0,1\}^k$, et $\gamma^\star$ est un bit aléatoire. Après avoir reçu le challenge $c^\star$, A_2 retourne un bit b'. On note Dist_0 l´événement $b' = b$ et Dist_n dans les jeux $\mathbf{J}_n$ ci-dessous.

En ce qui concerne la sécurité de la signature, on considère l'attaquant $\mathcal{B}$ contre ce schéma selon une attaque EUF-CMA + CCA. $\mathcal{B}$ retourne une falsification et on vérifie si elle est valide ou pas. On note l'événement où cette falsification est valide Forge_0 et Forge_n dans les jeux $\mathbf{J}_n$ ci-dessous.

Remarquons que l'attaquant a accès aux oracles de déchiffrement $\mathcal{D}_{\mathsf{sk}}$ et de signature $\mathcal{S}_{\mathsf{sk}}$ à toutes les étapes de l'attaque. Remarquons aussi que si l'attaquant soumet q_d, q_s et q_p requêtes respectivement aux oracles de déchiffrement, de signature et de permutation aléatoire, le nombre maximal de requêtes soumises à l'oracle de permutation aléatoire est $q_d + q_s + q_p + 2$. En effet, chaque requête de déchiffrement ou de signature peut faire une requête de permutation aléatoire, de même que l'étape de vérification ou l'étape de génération du challenge.

Par définition :

$$\begin{aligned} \varepsilon_E &= \mathsf{Adv}^{\mathsf{ind/cma+cca}}_{\mathsf{OPbP}}(\mathcal{A}) = 2\Pr[\mathsf{Dist}_0] - 1 \\ \varepsilon_S &= \mathsf{Succ}^{\mathsf{euf/cma+cca}}_{\mathsf{OPbP}}(\mathcal{B}) = \Pr[\mathsf{Forge}_0]. \end{aligned}$$

JEU $\mathbf{J}_1$: Une simulation parfaite du jeu réel est décrite dans la Figure 6.1. En effet, la règle $\mathsf{Chal}^{(1)}$ simule parfaitement la génération du challenge $c^\star$. Dans ce qui suit, on simule les permutations aléatoires $\mathcal{P}$ et $\mathcal{P}^{-1}$ grâce à la liste $\mathcal{P}$-List, à une permutation aléatoire P et à son inverse P^{-1}.

Oracle $\mathcal{P}$	Requête $\mathcal{P}(\gamma, m, r)$: la réponse est p, où ►**Règle** $\mathsf{EvalP}^{(0)}$ $\quad p = P(\gamma, m, r)$. En outre, si (γ, m, r) est une requête directe de l'attaquant à $\mathcal{P}$, ajouter $(\gamma, m, r, p, \perp, \varphi_{\mathsf{pk}}(p))$ à $\mathcal{P}$-List.
Oracle $\mathcal{P}^{-1}$	Requête $\mathcal{P}^{-1}(p)$: la réponse est (γ, m, r), où ►**Règle** $\mathsf{InvP}^{(0)}$ $\quad (\gamma, m, r) = P^{-1}(p)$. En outre, si p est une requête directe de l'attaquant à $\mathcal{P}^{-1}$, ajouter $(\gamma, m, r, p, \perp, \varphi_{\mathsf{pk}}(p))$ à $\mathcal{P}$-List.
Oracle $\mathcal{D}$	Requête $\mathcal{D}_{\mathsf{sk}}(c)$: la réponse est m : ►**Règle** $\mathsf{Decrypt}^{(0)}$ $\quad p = \psi_{\mathsf{sk}}(c)$, et $(\gamma, m, r) = \mathcal{P}^{-1}(p)$. Ajouter $(\gamma, m, r, \perp, \perp, c)$ à $\mathcal{P}$-List.
Oracle $\mathcal{S}$	Requête $\mathcal{S}_{\mathsf{sk}}(m)$: la réponse est σ : calculer $\gamma = \mathsf{prf}(m)$, puis soumettre $(\gamma, m, 0^k)$ à l'oracle de permutation EvalP : $p = \mathcal{P}(\gamma, m, 0^k)$. ►**Règle** $\mathcal{S}^{(0)}$ $\quad$ Calculer $\sigma = \psi_{\mathsf{sk}}(p)$. Ajouter $(\gamma, m, 0^k, p, \sigma, \varphi_{\mathsf{pk}}(p))$ à $\mathcal{P}$-List.
Challengeur	Pour deux messages (m_0, m_1) : choisir aléatoirement un bit b et définir $m^\star = m_b$; choisir aléatoirement $r^\star$ et répondre $c^\star$ où ►**Règle** $\mathsf{Chal}^{(0)}$ $\quad p^\star = \mathcal{P}(\gamma^\star, m^\star, r^\star)$; $c^\star = \varphi_{\mathsf{pk}}(p^\star)$. ►**Règle** $\mathsf{ChalAdd}^{(0)}$ $\quad$ Ajouter $(\gamma^\star, m^\star, r^\star, \perp, \perp, c^\star)$ à $\mathcal{P}$-List.
Oracle $\mathcal{V}$	Vérification de la falsification (σ) de l'attaquant. D'abord, calculer $t\|u = \varphi_{\mathsf{pk}}(\sigma)$. Ensuite, faire une requête à l'oracle de permutation $(\gamma, m, r) = \mathcal{P}^{-1}(t\|u)$. Finalement, vérifier si $r = 0^k$, auquel cas, la falsification est une signature valide de m.

FIG. 6.1 – Simulation dans le jeu $\mathbf{J}_1$

Il convient de noter que $\mathcal{P}$-List contient des éléments de la forme (γ, m, r, p, s, c) : p représente la valeur de $\mathcal{P}(\gamma, m, r)$; c , la valeur de $\varphi_{\mathsf{pk}}(p)$; et s , la valeur de $\psi_{\mathsf{sk}}(p)$, *i.e.* $p = \varphi_{\mathsf{pk}}(s)$. Dans l'organisation de $\mathcal{P}$-List, lors que l'on ajoute (γ, m, r, p, s, c) à $\mathcal{P}$-List, s'il y a déjà un élément de la forme $(\gamma, m, r, \alpha, \beta, y)$ dans $\mathcal{P}$-List, avec $\alpha = \perp$ ou $\beta = \perp$, on remplace cet élément par (m, r, p, s, c).

Maintenant, on note Δ_n la distance statistique entre les distributions de la vue de l'attaquant dans les jeux $\mathbf{J}_n$ et $\mathbf{J}_{n-1}$. On voit facilement que : $\Delta_1 = 0$.

JEU $\mathbf{J}_2$: Dans ce jeu, on modifie la simulation des oracles $\mathcal{P}$ et $\mathcal{P}^{-1}$ de telle sorte que chaque requête ne soit soumise au maximum qu'une fois à la permutation aléatoire P ou à son inverse P^{-1} :

►**Règle** $\mathsf{EvalP}^{(2)}$

- Si $(\gamma, m, r, \alpha, \beta, c)$ se trouve dans $\mathcal{P}$-List :
  - si $\alpha \neq \perp$, $p = \alpha$;
  - sinon, Terminer.
- Sinon, $p = P(\gamma, m, r)$.

►**Règle** $\mathsf{InvP}^{(2)}$

Calculer $c = \varphi_{\mathsf{pk}}(p)$ et vérifier si $(\gamma, m, r, \alpha, \beta, c)$ est dans $\mathcal{P}$-List :
- si oui, (γ, m, r) est déjà défini ;
- sinon $(\gamma, m, r) = P^{-1}(p)$.

On est amené à donner une réponse incorrecte lors de la simulation de l'oracle $\mathcal{P}$ quand il existe un élément $(\gamma, m, r, \alpha, \beta, c)$ dans $\mathcal{P}$-List et $\alpha = \perp$. Dans ce cas, on met fin au jeu mais la réponse correcte devrait être $\psi_{\mathsf{sk}}(c)$. On note cet événement BadP_2 et BadP_n dans les jeux $\mathbf{J}_n$ ci-dessous. À l'exception de cet événement, les jeux $\mathbf{J}_2$ et $\mathbf{J}_1$ sont parfaitement indistinguables :

$$\Delta_2 \leq \Pr[\mathsf{BadP}_2].$$

JEU $\mathbf{J}_3$: On modifie la simulation des oracles $\mathcal{P}$ et $\mathcal{P}^{-1}$. On n'utilise plus la permutation aléatoire P et son inverse P^{-1} mais retourne simplement une valeur aléatoire pour chaque nouvelle requête : à une nouvelle requête (γ, m, r) à l'oracle $\mathcal{P}$, on répond un aléa p, et à une nouvelle requête p à l'oracle $\mathcal{P}^{-1}(p)$, on répond aléatoirement (γ, m, r).

►**Règle** $\mathsf{EvalP}^{(3)}$

- Si $(\gamma, m, r, \alpha, \beta, c)$ se trouve dans $\mathcal{P}$-List :
  - si $\alpha \neq \perp$, $p = \alpha$;
  - sinon, Terminer.
- Sinon, $p \xleftarrow{R} \{0,1\}^n$.

►**Règle** $\mathsf{InvP}^{(3)}$

Calculer $c = \varphi_{\mathsf{pk}}(p)$ et vérifier si $(\gamma, m, r, \alpha, \beta, c)$ est dans $\mathcal{P}$-List :
- si oui, (γ, m, r) est défini ;
- sinon on choisit $(\gamma, m, r) \xleftarrow{R} \{0,1\}^n$.

Les jeux $\mathbf{J}_3$ et $\mathbf{J}_2$ sont parfaitement indistinguables sauf s'il y a une collision sur (γ, m, r) ou p dans $\mathcal{P}$-List. On note CollP_3 cet événement. On a :

$$\Delta_3 \leq \Pr[\mathsf{CollP}_3].$$

Comme il y a au plus $q_p + q_d + q_s + 1$ éléments dans $\mathcal{P}$-List, la probabilité d'occurrence d'une collision est $(q_p + q_d + q_s + 1)^2/2^n$:

$$\Pr[\mathsf{CollP}_3] \leq \frac{(q_p + q_d + +q_s + 1)^2}{2^{k+\ell+1}}.$$

Jeu $\mathbf{J}_4$: On peut maintenant simuler l'oracle de déchiffrement $\mathcal{D}_{\mathsf{sk}}$ sans utiliser la fonction ψ_{sk} :

►**Règle** $\mathsf{Decrypt}^{(4)}$

Vérifier s'il y a $(\gamma, m, r, \alpha, \beta, c)$ dans $\mathcal{P}$-List :

1. si oui, (γ, m, r) est défini,
2. sinon, on choisit $(\gamma, m, r) \stackrel{R}{\leftarrow} \{0,1\}^n$.

Les jeux $\mathbf{J}_4$ et $\mathbf{J}_3$ sont parfaitement indistinguables car dans le deuxième cas, on fait exactement comme si $\mathcal{P}^{-1}$ était simulé par la règle $\mathsf{InvP}^{(3)}$ dans le jeu précédent, y compris l'ajout à la $\mathcal{P}$-List :

$$\Delta_4 = 0.$$

Jeu $\mathbf{J}_5$: On considère les éléments de la forme $(\gamma, m, r, \perp, \beta, c)$ dans $\mathcal{P}$-List (il y en a au plus q_d+1). S'il y a une collision sur m, on met fin au jeu. On note CollM_5 cet événement :

$$\Delta_5 \leq \Pr[\mathsf{CollM}_5].$$

On remarque que m est aléatoirement choisi pour tout $(\gamma, m, r, \perp, \beta, c)$ sauf si $m = m^\star$. Par conséquent, la probabilité d'occurrence d'une collision sur m est majorée par $(q_d+1)^2/2^\ell$:

$$\Pr[\mathsf{CollM}_5] \leq \frac{(q_d + 1)^2}{2^\ell}.$$

Dans le cas où il n'y a pas de collision, pour chaque m, il existe au plus une valeur de r (et une valeur de c) telle(s) que $(\gamma, m, r, \perp, \beta, c) \in \mathcal{P}$-List. Donc, on peut facilement voir que :

$$\Pr[\mathsf{BadP}_5] \leq \frac{q_p + q_s + 1}{2^k}.$$

Jeu $\mathbf{J}_6$: On simule l'oracle de permutation pour ne plus avoir à utiliser la fonction ψ_{sk} dans la simulation de l'oracle de signature. Pour ce faire, on fait appel à la fonction $\varphi_{\mathsf{pk}}(s)$ lors de la simulation de l'oracle de permutation :

►**Règle** $\mathsf{EvalP}^{(6)}$

- Si $(\gamma, m, r, \alpha, \beta, c)$ se trouve dans $\mathcal{P}$-List :
  - si $\alpha \neq \perp$, $p = \alpha$;
  - sinon, Terminer.
- Sinon, choisir $s \xleftarrow{R} \{0,1\}^n$ et calculer $p = \varphi_{\mathsf{pk}}(s)$. Ajouter $(\gamma, m, r, p, s, \varphi_{\mathsf{pk}}(p))$ à $\mathcal{P}$-List.

Comme φ_{pk} est une permutation, p est un aléa et par conséquent, les jeux $\mathbf{J}_6$ et $\mathbf{J}_5$ sont parfaitement indistinguables :

$$\Delta_6 = 0.$$

JEU $\mathbf{J}_7$: Dans ce jeu, on continue à simuler l'oracle $\mathcal{P}^{-1}$ afin qu'il ne retourne pas d'élément de la forme $(\gamma, m, 0^k)$. En effet, si l'attaquant soumet $a = \mathcal{P}^{-1}(p)$, il sait que $\mathcal{P}(a) = p$. Par conséquent, si a est de la forme $\gamma \| m_0 \| 0^k$, on ne peut pas simuler la signature de m_0 sans utiliser ψ_{sk} car on ne connaît pas la valeur de s telle que $\mathcal{P}(a) = \varphi_{\mathsf{pk}}(s)$.

►**Règle** $\mathsf{InvP}^{(7)}$

Calculer $c = \varphi_{\mathsf{pk}}(p)$ et vérifier si $(\gamma, m, r, \alpha, \beta, c)$ est dans $\mathcal{P}$-List :
- si oui, (γ, m, r) est défini ;
- sinon on choisit $(\gamma, m, r) \xleftarrow{R} \{0,1\}^n$. Si $r = 0^k$, Terminer.

Pour chaque requête à $\mathcal{P}^{-1}$, la probabilité que l'on termine le jeu est égale à $1/2^k$:

$$\Delta_7 \leq (q_p + q_d + 1)/2^k.$$

JEU $\mathbf{J}_8$: Toutes les valeurs $\mathcal{P}(\gamma, m, 0^k)$ dont l'attaquant dispose sont des sorties de l'oracle $\mathcal{P}$. Par conséquent, $\mathcal{P}(\gamma, m, 0^k) = \varphi_{\mathsf{pk}}(s)$ et on peut donc simuler l'oracle de signature sans utiliser ψ_{sk} :

►**Règle** $\mathcal{S}_{\mathsf{sk}}{}^{(8)}$

Consulter $(\gamma, m, 0^k, p, s, c)$ dans $\mathcal{P}$-List, et définir $\sigma = s$.

Les jeux $\mathbf{J}_8$ et $\mathbf{J}_7$ sont parfaitement indistinguables :

$$\Delta_8 = 0.$$

On a simulé toutes les requêtes que l'attaquant soumet aux oracles de permutation, de déchiffrement et de signature. Ces simulations, décrites dans les figures 6.2 et 6.3, sont indépendantes de l'objectif de l'attaquant. Dans la suite, en fonction de son objectif (contre le chiffrement ou contre la signature), on complétera la réduction au problème d'inverser la fonction φ_{sk} pour une instance y donnée.

Attaque contre le chiffrement

JEU $\mathbf{J}_{8.1}$: On enlève l'élément $(\gamma^\star, m^\star, r^\star, \perp, \perp, c^\star)$ de $\mathcal{P}$-List lors de la génération du challenge.

Challengeur	Pour deux messages (m_0, m_1) : choisir aléatoirement un bit b et définir $m^\star = m_b$; choisir aléatoirement $r^\star$ et répondre $c^\star$ où ►**Règle** $\mathsf{Chal}^{(8)}$ $\quad p^\star = \mathcal{P}(\gamma^\star, m^\star, r^\star)$; $c^\star = \varphi_{\mathsf{pk}}(p^\star)$. ►**Règle** $\mathsf{ChalAdd}^{(8)}$ $\quad$ ajouter $(\gamma^\star, m^\star, r^\star, \perp, \perp, c^\star)$ à $\mathcal{P}$-List.
Oracle $\mathcal{V}$	Vérification de la falsification (σ) de l'attaquant. D'abord, calculer $t\|\|u = \varphi_{\mathsf{pk}}(\sigma)$. Ensuite, faire une requête à l'oracle de permutation $(\gamma, m, r) = \mathcal{P}^{-1}(t\|\|u)$. Finalement, vérifier si $r = 0^k$, dans ce cas, la falsification est une signature valide de m.

FIG. 6.2 – Simulation dans le Jeu $\mathbf{J}_8$

►**Règle** $\mathsf{ChalAdd}^{(8.1)}$

| Ne rien faire.

Les jeux $\mathbf{J}_{8.1}$ et $\mathbf{J}_8$ sont parfaitement indistinguables sauf si $(\gamma^\star, m^\star, r^\star)$ a été soumise à $\mathcal{P}$ ou $p^\star = \psi_{\mathsf{sk}}(c^\star)$ a été soumise à $\mathcal{P}^{-1}$. Le premier événement est compris dans l'événement $\mathsf{BadP}_{8.1}$, qui est déjà exclu. On note $\mathsf{AskInvP}_{8.1}$ le deuxième événement :

$$\Delta_{8.1} \leq \Pr[\mathsf{AskInvP}_{8.1}].$$

Dans ce jeu, $(\gamma^\star, m^\star, r^\star, \perp, \perp, c^\star)$ n'est plus dans $\mathcal{P}$-List, l'attaquant n'a aucun avantage pour deviner b : $\Pr[\mathsf{Dist}_{8.1}] = \frac{1}{2}$.

JEU $\mathbf{J}_{8.2}$: Au lieu de choisir $c^\star = \varphi_{\mathsf{pk}}(p^\star)$, on choisit $c^\star = y$, uniformément aléatoire.

►**Règle** $\mathsf{Chal}^{(8.2)}$

| $c^\star = y$.

Alors, on définit implicitement $p^\star = \psi_{\mathsf{sk}}(y)$. Comme $(\gamma^\star, m^\star, r^\star, \perp, \perp, c^\star)$ n'est plus utilisé dans la simulation, les jeux $\mathbf{J}_{8.2}$ et $\mathbf{J}_{8.1}$ sont parfaitement indistinguables : $\Delta_{8.2} = 0$.

Finalement, on peut facilement calculer $\psi_{\mathsf{sk}}(y)$ lorsque l'événement $\mathsf{AskInvP}_{8.2}$ se produit : en consultant dans $\mathcal{P}$-List (qui contient au plus $q_p + q_d + q_s + 1$ éléments), on peut extraire p tel que $y = \varphi_{\mathsf{pk}}(p)$. Alors :

$$\Pr[\mathsf{AskInvP}_{8.2}] \leq \mathsf{Succ}^{\mathsf{ow}}_{\varphi}(t'),$$

où T_φ est le temps nécessaire pour une évaluation d'une fonction φ_{pk}, et $t' \leq t + (q_p + q_d + q_s + 1) \times T_\varphi$, le temps de la simulation de ce jeu.

Attaque contre la signature (cas général)

JEU $\mathbf{J}_{8.1}$: Dans ce jeu, on énumère les requêtes de la forme $(\gamma, \star, 0^k)$ à l'oracle de permutation. Il s'agit des requêtes qui seront utilisées pour la signature. Pour ce faire, on

Oracle $\mathcal{P}$

Requête $\mathcal{P}(\gamma, m, r)$: la réponse est p, où :

►**Règle** $\mathsf{EvalP}^{(8)}$

- Si $(\gamma, m, r, \alpha, \beta, c)$ se trouve dans $\mathcal{P}$-List :
  - si $\alpha \neq \perp$, $p = \alpha$;
  - sinon, Terminer.
- Sinon, choisir $s \xleftarrow{R} \{0,1\}^n$, calculer $p = \varphi_{\mathsf{pk}}(s)$ et ajouter $(\gamma, m, r, p, s, \varphi_{\mathsf{pk}}(p))$ à $\mathcal{P}$-List.

En outre, si (γ, m, r) est une requête directe de l'attaquant à $\mathcal{P}$, ajouter $(\gamma, m, r, p, \perp, \varphi_{\mathsf{pk}}(p))$ à $\mathcal{P}$-List.

Oracle $\mathcal{P}^{-1}$

Requête $\mathcal{P}^{-1}(p)$: la réponse est (γ, m, r) :

►**Règle** $\mathsf{InvP}^{(8)}$

Calculer $c = \varphi_{\mathsf{pk}}(p)$ et vérifier si $(\gamma, m, r, \alpha, \beta, c)$ se trouve dans $\mathcal{P}$-List :
- si oui, (γ, m, r) est défini ;
- sinon, choisir $(\gamma, m, r) \xleftarrow{R} \{0,1\}^n$. Si $r = 0^k$, Terminer.

En outre, si p est une requête directe de l'attaquant à $\mathcal{P}^{-1}$, ajouter $(\gamma, m, r, p, \perp, \varphi_{\mathsf{pk}}(p))$ à $\mathcal{P}$-List.

Oracle $\mathcal{D}$

Requête $\mathcal{D}_{\mathsf{sk}}(c)$: la réponse est m :

►**Règle** $\mathsf{Decrypt}^{(8)}$

Vérifier si $(\gamma, m, r, \alpha, \beta, c)$ se trouve dans $\mathcal{P}$-List :
- si oui, (γ, m, r) est défini,
- sinon, choisir $(\gamma, m, r) \xleftarrow{R} \{0,1\}^n$.

Ajouter $(\gamma, m, r, \perp, \perp, c)$ à $\mathcal{P}$-List.

Oracle $\mathcal{S}$

Requête $\mathcal{S}_{\mathsf{sk}}(m)$: la réponse est σ : calculer $\gamma = \mathsf{prf}(m)$, puis soumettre $(\gamma, m, 0^k)$ à l'oracle de permutation EvalP : $p = \mathcal{P}(\gamma, m, 0^k)$.

►**Règle** $\mathcal{S}^{(8)}$

Consulter l'élément $(\gamma, m, 0^k, p, s, c)$ dans $\mathcal{P}$-List, et définir $\sigma = s$.

FIG. 6.3 – Simulation dans le Jeu $\mathbf{J}_8$

définit un compteur ν, initialisé à 0.

►**Règle** $\mathsf{EvalP}^{(8.1)}$

- Si $(\gamma, m, r, \alpha, \beta, c)$ se trouve dans $\mathcal{P}$-List :
  - si $\alpha \neq \perp$, $p = \alpha$;
  - sinon, Terminer.
- Sinon,
  - si $r = 0^k$, incrémenter ν ;
  - choisir $s \xleftarrow{R} \{0,1\}^n$ et calculer $p = \varphi_{\mathsf{pk}}(s)$. Ajouter $(\gamma, m, r, p, s, \varphi_{\mathsf{pk}}(p))$ à $\mathcal{P}$-List.

Évidemment, ce jeu est indistinguable du jeu $\mathbf{J}_8$: $\Delta_{8.1} = 0$.

Jeu $\mathbf{J}_{8.2}$: Comme le processus de vérification est compris dans le jeu d'attaque, la falsification est nécessairement soumise à l'oracle de permutation EvalP. On devine l'indice ν_0 de la première occurrence de cette requête. On considère seulement le cas où on le devine correctement :

$$\Pr[\mathsf{Forge}_{8.2}] \;\geq\; \Pr[\mathsf{Forge}_{8.1}]/(q_p + q_s + 1).$$

Jeu $\mathbf{J}_{8.3}$: On peut maintenant intégrer le challenge y dans la simulation de l'oracle de permutation. Grâce à cette méthode, on va extraire la pré-image x. L'idée est de répondre y à la ν_0-ième requête de l'oracle de permutation :

►**Règle** $\mathsf{EvalP}^{(8.3)}$

- Si $(\gamma, m, r, \alpha, \beta, c)$ se trouve dans $\mathcal{P}$-List :
  - si $\alpha \neq \perp$, $p = \alpha$;
  - sinon, Terminer.
- Sinon,
  - si $r = 0^k$, incrémenter ν ;
  - si $\nu \neq \nu_0$ ou si $r \neq 0^k$, choisir $s \xleftarrow{R} \{0,1\}^n$ et calculer $p = \varphi_{\mathsf{pk}}(s)$;
  - si $\nu = \nu_0$ et $r = 0^k$, définir $p = y$ et $s \xleftarrow{R} \{0,1\}^n$.
  - Ajouter $(\gamma, m, r, p, s, \varphi_{\mathsf{pk}}(p))$ à $\mathcal{P}$-List.

Comme l'instance y est aléatoirement choisie, les jeux $\mathbf{J}_{8.3}$ et $\mathbf{J}_{8.2}$ sont parfaitement indistinguables :

$$\Delta_{8.3} = 0.$$

Dans ce jeu, une falsification valide de signature conduit à calculer la pré-image de y :

$$\Pr[\mathsf{Forge}_{8.3}] = \mathsf{Succ}^{\mathsf{ow}}_{\varphi}(t + (q_p + q_d + q_s + 1)T_\varphi).$$

Cette égalité conclut la deuxième partie de la preuve.

Attaque contre la signature (avec une permutation sans griffe (t, ε')-sûre)

Dans le cas général, le bit γ peut en effet être supprimé car il est inutile. Dans le cas de permutation sans griffe $(\varphi_{\mathsf{pk}}, \lambda_{\mathsf{pk}})$ (t, ε')-sûres, l'idée est d'utiliser φ_{pk} à la sortie d'OPbP

pour une valeur du bit γ, et d'utiliser λ_{pk} pour l'autre valeur de γ. Comme la valeur de γ est imprévisible pour l'attaquant, la falsification conduit, avec une probabilité de $\frac{1}{2}$, à une griffe.

JEU $\mathbf{J}_{8.1}$: On exploite maintenant le bit γ pour la simulation de l'oracle de permutation, de la même manière que celle proposée par Katz et Wang [71].

▶**Règle** $\mathsf{EvalP}^{(8.1)}$
- Si $(\gamma, m, r, \alpha, \beta, c)$ se trouve dans $\mathcal{P}$-List :
  - si $\alpha \neq \perp$, $p = \alpha$;
  - sinon, Terminer.
- Sinon,
  - si $r \neq 0^k$ ou $\gamma = \mathsf{prf}(m)$, choisir $s \xleftarrow{R} \{0,1\}^n$ et calculer $p = \varphi_{\mathsf{pk}}(s)$.
  - si $r = 0^k$ ou $\gamma \neq \mathsf{prf}(m)$, choisir $s \xleftarrow{R} \{0,1\}^n$et calculer $p = \lambda_{\mathsf{pk}}(s)$.
  - Ajouter $(\gamma, m, r, p, s, \varphi_{\mathsf{pk}}(p))$ à $\mathcal{P}$-List.

Comme s est aléatoirement choisie, $\lambda_{\mathsf{pk}}(s)$ l'est aussi. Par conséquent, les jeux $\mathbf{J}_{8.1}$ et $\mathbf{J}_8$ sont parfaitement indistinguables :

$$\Delta_{8.1} = 0.$$

En utilisant les mêmes arguments que ceux présentés dans [71], on peut montrer qu'une falsification correcte de signature conduit à une griffe avec probabilité $\frac{1}{2}$. En effet, supposons que l'attaquant peut forger une signature $(\tilde{m}, \tilde{\sigma})$, où $(\tilde{m}, 0^k)$ a été soumise à l'oracle de permutation $\mathcal{P}$ lors d'une requête de permutation ou de l'étape de vérification. Le bit $b_{\tilde{m}} = \mathsf{prf}(\tilde{m})$ est un bit aléatoire et imprévisible pour l'attaquant. Alors, avec probabilité $\frac{1}{2}$, il existe un élément $(\tilde{m}, \tilde{r}, \tilde{p} = \lambda_{\mathsf{pk}}(\tilde{s}), \tilde{s}, \varphi_{\mathsf{pk}}(\tilde{p}))$ dans $\mathcal{P}$-List. Dans ce cas, on peut produire une griffe $\varphi_{\mathsf{pk}}(\tilde{\sigma}) = \lambda_{\mathsf{pk}}(\tilde{s})$. □

6.3.3 Proposition de taille pour les paramètres

Un schéma atteint le niveau de sécurité 2^κ si le rapport entre le temps t de fonctionnement de l'attaquant et sa probabilité de succès ε est au moins 2^κ. Il s'agit d'une approximation du temp pour obtenir un succès. Ainsi, on voudrait avoir $\varepsilon/t \leq 2^{-\kappa}$, avec la borne usuelle de sécurité $\kappa = 80$.

Tout d'abord, on cherche à simplifier le résultat ci-dessus. En effet, la taille ℓ du message est normalement significativement plus large que la taille k de l'aléa/la redondance : la quantité $Q/2^\ell$, voire $Q^2/2^\ell$, peut être ignorée en comparaison de $Q/2^k$ (où Q est le nombre total de requêtes, et donc Q est borné par 2^{80}). Par conséquent, le coût de la réduction dans le théorème 47 se simplifie :

$$\begin{aligned} & \frac{\varepsilon_E}{t} \leq \frac{\varepsilon'}{t} + \frac{2}{2^k} \\ \text{et} \quad & \frac{\varepsilon_S}{t} \leq \frac{Q\varepsilon'}{t} + \frac{2}{2^k} \text{ dans le cas général} \\ & \leq \frac{2\varepsilon'}{t} + \frac{2}{2^k} \text{ si la fonction } \varphi_{\mathsf{pk}} \text{ est induite par une permutation sans griffe.} \end{aligned}$$

Dans le cas particulier où la fonction φ_{pk} est induite par une permutation sans griffe (cas le plus courant où on utilise le RSA), il faut que la taille du message (*i.e.* module RSA) soit suffisamment grande pour que ε'/t soit inférieur à 2^{-82}. Grâce à l'estimation de Lenstra-Verheul [77], on peut utiliser un module RSA de 1024 bits. Avec seulement 82 bits de redondance, nous obtenons le même niveau de sécurité que RSA-PSS [10], et une meilleure bande passante. Concernant la sécurité du chiffrement, le résultat du chapitre précédent reste valable : 82 bits d'aléa sont convenables pour la sécurité sémantique, même selon des attaques $\mathsf{CMA} + \mathsf{CCA}$.

Dans le cas général, il nous faut un paramètre de sécurité et un message de taille relativement grande pour que ε'/t soit inférieur à 2^{-161}. Alors, si le niveau de sécurité de la fonction φ est 2^{161}, on peut choisir l'« overhead » $k = 82$ qui jouera le rôle de la redondance pour la signature et celui d'aléa pour le chiffrement.

6.4 OAEP 3-tours pour le chiffrement et la signature

6.4.1 Description

Afin de construire un schéma dans le modèle de l'oracle aléatoire [9], nous considérons la construction d'OAEP 3-tours présentée dans le chapitre précédent.

Le chiffrement et la signature utilisent trois fonctions de hachage : $\mathcal{F}$, $\mathcal{G}$ et $\mathcal{H}$, supposées ressembler à des oracles aléatoires dans les analyses de sécurité. Les paramètres de sécurité satisfont $n = k + \ell + 1$:

$$\mathcal{F} : \{0,1\}^k \to \{0,1\}^{\ell+1} \qquad \mathcal{G} : \{0,1\}^{\ell+1} \to \{0,1\}^k \qquad \mathcal{H} : \{0,1\}^k \to \{0,1\}^{\ell+1}.$$

Le chiffrement et la signature utilisent aussi une famille de permutations à sens-unique à trappe (φ_{pk}), sur l'espace $\{0,1\}^n$, dont les inverses sont les ψ_{sk}, où sk est la clef secrète (*i.e.* la trappe) associée à la clef publique pk. De plus, on identifie $\{0,1\} \times \{0,1\}^k \times \{0,1\}^\ell$ à $\{0,1\}^n$. Finalement, dans ce qui suit, $\mathsf{prf}()$ désigne une fonction pseudo-aléatoire.

Padding $\mathsf{OAEP3r}$ et unpadding $\mathsf{OAEP3r}^{-1}$

– $\mathsf{OAEP3r}(\gamma, m, r)$:

$$s = (\gamma \| m) \oplus \mathcal{F}(r) \qquad t = r \oplus \mathcal{G}(s) \qquad u = s \oplus \mathcal{H}(t)$$

$$\mathsf{OAEP3r}(\gamma, m, r) = t\|u.$$

– $\mathsf{OAEP3r}^{-1}(t, u)$:

$$s = u \oplus \mathcal{H}(t) \qquad r = t \oplus \mathcal{G}(s) \qquad \gamma\|m = s \oplus \mathcal{F}(r)$$

$$\mathsf{OAEP3r}^{-1}(t, u) = \gamma\|m\|r.$$

Algorithme de chiffrement

L'espace des textes clairs est $\mathcal{M} = \{0,1\}^\ell$. Le chiffrement utilise une chaîne aléatoire $r \in \mathcal{R} = \{0,1\}^k$ et un bit aléatoire γ. À partir du texte clair $m \in \mathcal{M}$, il retourne un texte chiffré c de $\{0,1\}^n$:

$$t\|u = \mathsf{OAEP3r}(\gamma, m, r) \qquad c = \varphi_{\mathsf{pk}}(t\|u).$$

Algorithme de déchiffrement

À partir du texte chiffré c, on calcule d'abord $t\|u = \psi_{\mathsf{sk}}(c)$, où $t \in \{0,1\}^k$ et $u \in \{0,1\}^{\ell+1}$, puis on retourne le texte clair m en calculant : $\gamma\|m\|r = \mathsf{OAEP3r}^{-1}(t, u)$.

Algorithme de signature

L'espace des textes clairs est $\mathcal{M} = \{0,1\}^\ell$, l'algorithme de signature retourne une signature σ de $\{0,1\}^n$: à partir du message $m \in \mathcal{M}$, on calcule $\gamma = \mathsf{prf}(m)$, puis $t\|u = \mathsf{OAEP3r}(\gamma, m, 0^k)$ et $\sigma = \psi_{\mathsf{sk}}(t\|u)$

Algorithme de vérification

À partir de la signature σ, on calcule d'abord $t\|u = \varphi_{\mathsf{pk}}(\sigma)$, où $t \in \{0,1\}^k$ et $u \in \{0,1\}^{\ell+1}$, puis $\gamma\|m\|r = \mathsf{OAEP3r}^{-1}(t, u)$. Si $r = 0^k$, la vérification retourne « Correcte » et retrouve m, sinon elle retourne « Incorrecte ».

6.4.2 Résultat de sécurité

Théorème 48 *Considérons les attaquants $\mathcal{A}$ et $\mathcal{B}$ contre respectivement le chiffrement et la signature selon une attaque à la fois à chiffrés choisis et à messages choisis. Supposons qu'en temps τ et qu'après q_f, q_g, q_h, q_s, q_d requêtes respectivement aux oracles $\mathcal{F}$, $\mathcal{G}$, $\mathcal{H}$, de signature et de déchiffrement, $\mathcal{A}$ peut avoir un avantage ε_E pour casser la sécurité sémantique du schéma, ou $\mathcal{B}$ peut avoir un succès ε_S pour produire une falsification existentielle, alors la permutation φ_{pk} peut être inversée avec probabilité ε' en temps τ' où :*

$$\begin{aligned}\varepsilon' &\geq \varepsilon_E - \left(q_d^2 \times \left(\frac{1}{2^{\ell+1}} + \frac{6}{2^k}\right) + \frac{4q_dq_g + q_g}{2^{\ell+1}} + \frac{5q_dq_f + q_gq_h + q_f + q_d}{2^k}\right)\\ \textit{ou}\\ \varepsilon' &\geq \frac{1}{q_g + q_s + 1} \times \\ &\quad \left(\varepsilon_S - \left(q_d^2 \times \left(\frac{1}{2^{\ell+1}} + \frac{6}{2^k}\right) + \frac{4q_dq_g + q_g}{2^{\ell+1}} + \frac{5q_dq_f + q_gq_h + q_f + q_d}{2^k}\right)\right).\end{aligned}$$

En particulier, si la fonction φ_{pk} est induite par une famille de permutations sans griffe et (τ, ε')-sûre, la deuxième relation peut être ré-écrite par :

$$\varepsilon' \geq \frac{1}{2} \times \left(\varepsilon_S - \left(q_d^2 \times \left(\frac{1}{2^{\ell+1}} + \frac{6}{2^k}\right) + \frac{4q_dq_g}{2^{\ell+1}} + \frac{5q_dq_f}{2^k} + \frac{q_gq_h}{2^k} + \frac{q_gq_s}{2^k} + \frac{q_g}{2^{\ell+1}} + \frac{q_f}{2^k} + \frac{q_d}{2^k}\right)\right),$$

avec $\tau' \leq \tau + (q_f + q_g + q_h + q_d)T_{lu} + q_d^2 T_{lu} + (q_d + 1)q_gq_h(T_\varphi + T_{lu})$, où T_φ est le temps pour une évaluation d'une fonction φ_{pk} et T_{lu}, le temps de recherche d'un élément dans une liste.

La preuve utilise une technique similaire à celle utilisée pour la sécurité sémantique d'OAEP 3-tours (voir la preuve du théorème 44). Elle considère les deux types d'attaque contre la signature et contre le chiffrement comme dans la preuve du théorème 47 dans le modèle de permutation aléatoire. Les détails sont dans [33].

6.4.3 Proposition de taille pour les paramètres

En utilisant les arguments similaires à ceux utilisées dans la section 6.3.3, on peut simplifier les contraintes sur les paramètres de sécurité :

– Pour le chiffrement, on a :

$$\frac{\varepsilon_E}{t} \leq \frac{\varepsilon'}{t} + \frac{Q}{2^k}.$$

Alors, $k = 161$ est convenable si les paramètres sont suffisamment grands, *i.e.* $\varepsilon'/t < 2^{-81}$).

– Pour la signature dans le cas général :

$$\frac{\varepsilon_S}{t} \leq \frac{Q\varepsilon'}{t} + \frac{Q}{2^k}.$$

Alors, $k = 161$ est aussi suffisant, à condition que $\varepsilon'/t < 2^{-161}$.

– Pour la signature dans le cas où la fonction φ_{pk} est induite par une permutation sans griffe :

$$\frac{\varepsilon_S}{t} \leq \frac{2\varepsilon'}{t} + \frac{Q}{2^k}.$$

On obtient une expression semblable à celle obtenue dans la construction dans la modèle de la permutation aléatoire pour le chiffrement : le terme ε'/t est remplacé par $2\varepsilon'/t$, ce qui permet des paramètres de sécurité plus courts. Par conséquent, $k = 161$ est suffisant, à condition que $\varepsilon'/t < 2^{-82}$.

En conclusion, pour le cas le plus intéressant de RSA, on peut choisir $k = 161$ à condition que le niveau de sécurité de la fonction φ soit de l'ordre de 2^{82}, *i.e.* un module de 1024 bits.

Diffusion de données chiffrées

7

Introduction

7.1 Traçage des traîtres

Le problème de sécurité dans la distribution de données est un des problèmes les plus importants de l'ère numérique. S'il n'y a pas assez de sécurité, on peut écrire des livres, composer de la musique mais on ne peut être payé qu'une seule fois car dès que le premier exemplaire est sorti, n'importe qui peut faire des copies et les distribuer. Les chaînes de télévision privées sont découragées car n'importe quel non - abonné peut capter ses programmes sans payer... Dans un tel monde, tous les fruits du travail sont des biens publics et les auteurs de ces travaux perdent tous leurs droits. Pour encourager l'investissement, il est indispensable que les problèmes de droit d'auteur, de sécurité, etc. occupent une place importante.

On peut distinguer deux façons de distribuer les données. La différence tient dans la nature du problème et l'objectif du pirate :

- Distribution « statique » de données. L'exemple type est la protection du contenu des CD-ROMs.
- Distribution « dynamique » de données. L'exemple type est la diffusion des programmes télévisés aux abonnés.

Concernant la distribution statique, un client, qui a acheté un CD-ROM par exemple peut toujours redistribuer le contenu de ce CD-ROM (on l'appellera alors « traîtres »). On ne peut pas l'en empêcher mais ce que l'on peut faire est de l'en décourager. La méthode utilisée est le « marquage » du produit, qui consiste à marquer quelques positions du contenu de telle manière que ces marques, bien que sans influence sur le contenu perçu, caractérisent chaque exemplaire. En un mot, chaque exemplaire est différent des autres, et si un client distribue le contenu de son exemplaire sans aucune modification, on peut le détecter. Le but ultime d'un traître ou d'une coalition de traîtres est de fabriquer un nouvel exemplaire et de le redistribuer sans être détectés. En revanche, l'objectif de l'autorité est de retrouver au moins un des traîtres de la coalition, source de l'exemplaire illégal.

Pour la distribution dynamique, où le contenu est transmis en temps réel sur un canal de diffusion, la technique de marquage est évidemment irréalisable car le centre ne peut émettre vers chaque abonné un programme différent. Le principe de la diffusion est d'envoyer un même signal à tous les abonnés. Pour y pallier, le contenu est chiffré avant d'être diffusé. Chaque abonné dispose donc d'un décodeur pour déchiffrer les signaux. Le but du pirate est de redistribuer le contenu. Une première méthode de fraude serait de redistribuer le contenu une fois déchiffré, mais cela suppose un système de diffusion. De plus, cette technique est facile à détecter et conduit souvent à des peines sévères. L'objectif réel du pirate est de construire lui-même un nouveau décodeur lui permettant de déchiffrer le signal émis par le centre. Pour ce faire, un traître, ou pire, une coalition de traîtres peut essayer d'ouvrir leur propre décodeur afin d'en extraire les clés de déchiffrement et en fabriquer un nouveau. Cependant, ce que ces traîtres veulent éviter (et que le centre veut atteindre) est qu'à partir d'un décodeur pirate, il soit possible de retrouver la source du piratage, *i.e.* un des traîtres.

Dans ce travail, nous nous concentrons sur le deuxième mode de distribution : le traçage de traîtres dans la distribution des données dynamiques. Il existe, dans la littérature, deux principales catégories de tels schémas :

1. Les schémas à taux de transmission *constant* [73]. Ces schémas conviennent particulièrement au chiffrement de messages volumineux. De plus, ils permettent une traçabilité efficace en *boîte noire*, *i.e.* la procédure de traçage n'a pas besoin d'ouvrir le décodeur pirate mais simplement d'interagir avec lui.
2. Les schémas à taille de textes clairs admissibles relativement petite [25, 22, 74]. Cependant, dans ce cas, le taux de transmission est *non constant* et est au moins linéaire en la taille de la coalition. De plus, pour ces schémas, il n'y pas de procédure de traçage en boîte noire efficace. Il est possible de faire du traçage en boîte noire [22] mais l'algorithme de traçage dans ce cas est irréaliste en raison de sa complexité plus grande que le coefficient binomial de n et c, où n est le nombre d'usagers et c est la taille maximale de la coalition.

Nous étudions ici la première catégorie en mettant l'accent sur un aspect important : l'optimisation du taux de transmission. Comme déjà mentionné, dans un système de transmission directe de données du centre aux abonnés, un point important est le rapport entre la taille du texte chiffré (le vrai signal envoyé par le centre) et celle du texte clair (le contenu du programme transmis aux abonnés). Dans ce cadre, les paramètres de transmission sont considérés : *le taux du texte chiffré*–le rapport entre la taille du texte chiffré et celle du texte clair ; *le taux de la clef de chiffrement*–le rapport entre la taille de la clef de chiffrement et celle du texte en clair dans un bloc primitif de données (car une clef est normalement utilisée plusieurs fois pour chiffrer plusieurs blocs de données) ; *le taux de la clef d'usager*–le rapport entre la taille de la clef d'usager et celle du texte clair dans un bloc primitif de données.

Plus intéressant, les systèmes de la première catégorie intègrent des codes résistants aux coalitions- la technique la plus connue pour faire du marquage proposée par Boneh-Shaw [25]. Cette méthode, proposée par Kiayias et Yung [73], va être utilisée dans notre construction. De plus, nous utilisons les propriétés particulières des couplages pour amé-

liorer l'efficacité et pour obtenir une nouvelle fonctionnalité : la traçabilité publique.

Avant de présenter notre construction, nous rappelons brièvement les applications des couplages en cryptographie. Puis, nous présentons notre attaque sur un schéma de traçage de traîtres fondé sur des couplages.

7.2 Couplages en cryptographie

La recherche sur les courbes elliptiques est une longue histoire, depuis les études des nombres congrues et des équations diophantiennes. Cependant, l'approche arithmétique pour étudier ces courbes est relativement nouvelle, et n'est vraiment utilisée que depuis le siècle dernier (avec les travaux de Mordell [85]) et la notion de groupes de points. Initialement, le champ de recherche dans cette direction était essentiellement limité aux courbes elliptiques sur des corps infinis avec le théorème fondamental selon lequel le groupe des points rationnels d'une courbe elliptique est engendré par un nombre fini de points. C'est dans les années 80 que les courbes elliptiques sur des corps finis ont trouvé, pour la première fois, leur application en cryptographie grâce aux travaux de Koblitz [75] et Miller [83]. Ces auteurs ont indépendamment trouvé que les schémas cryptographiques fondés sur la difficulté du problème du logarithme discret pourraient atteindre un meilleur niveau de sécurité avec un groupe de points sur une courbe elliptique qu'avec le groupe multiplicatif conventionnel d'un corps fini. En effet, à un même niveau de sécurité, l'utilisation des courbes elliptiques se contente d'une clef plus courte. À titre d'exemple, dans des applications cryptographiques, l'utilisation d'un groupe elliptique de taille environ 2^{160} garantit une sécurité équivalente à celle offerte par l'utilisation d'un groupe classique de taille 2^{1024}.

La raison de cet écart est que le problème du logarithme discret est en générale beaucoup plus difficile sur les courbes elliptiques. Ce n'est, cependant, pas toujours le cas. En 1993, Menezes, Okamoto et Vanstone [80] ont décrit la première attaque non triviale contre la difficulté du problème du logarithme discret sur certaines classes de courbes elliptiques. Ils ont prouvé que ce problème sur une courbe elliptique se réduit à ce même problème sur une extension du corps de base. Or, dans le cas de courbes elliptiques super singulières, le degré d'extension est très petit. Cette attaque, appelée la réduction MOV, utilise la propriété de bilinéarité du couplage de Weil. Un an après, Frey et Ruck [50] ont utilisé le couplage de Tate pour décrire une attaque similaire, appelée la réduction FR. Les couplages apportent des propriétés intéressantes : ce sont des applications bilinéaires qui sont efficacement calculables et non-dégénérées.

Ainsi les couplages furent-ils initialement considérés comme outil d'attaque et, pendant longtemps, ces applications cryptanalytiques étaient considérées comme les seules applications des couplages. C'est en 2000 qu'ils ont été utilisés pour la première fois comme un outil de construction de schémas cryptographiques : Joux [67] a construit un schéma d'échange de clef à trois parties qui généralise le protocole Diffie-Hellman de base ; Sakai *et. al* [110] ont ensuite proposé des schémas de signature et d'échange de clef fondés sur

l'identité. Ces travaux ouvrent un nouveau courant d'études : l'utilisation des couplages dans la construction des schémas cryptographiques. L'application la plus importante des couplages se trouve sans doute dans la construction de schémas fondés sur l'identité. En 1984, Shamir [112] a proposé une variante de la cryptographie à clef publique où l'on utilise l'identité comme la clef publique et la clef secrète correspondante est générée par un tiers de confiance. Le fait que la clef de chiffrement soit liée à l'identité de l'usager garantit implicitement son authenticité et exclut la nécessité d'avoir une infrastructure des clefs publiques pour la certification. La mise en œuvre d'un schéma de chiffrement efficace fondé sur l'identité restait cependant un problème ouvert jusqu'au travail de Boneh et Franklin [23]. Depuis ce résultat, plusieurs améliorations, aussi bien en termes d'efficacité qu'en termes de sécurité, ont été apportées [20, 21, 119]. Boneh *et. al* ont également utilisé les couplages pour construire des schémas de signature de petite taille [24, 26]. Ils ont èa cet effet utilisé des groupes où le problème CDH est difficile mais où le problème DDH est facile (pour la vérification). L'étude de tels groupes a été entreprise par Joux et Nguyen [69].

Aujourd'hui, cinq ans après la première utilisation des couplages, cet outil est largement utilisé dans presque toutes les branches de la cryptographie asymétrique. Le « Pairing-Based Crypto Lounge » de Barreto [3] regroupe environ deux cents articles. En ce qui concerne le problème de traçage de traîtres, le premier schéma fondé sur les couplages a été proposé par Mitsunari, Sakai et Kasahara [84] en 2002 sans fournir de preuve de sécurité. To, Safavi-Naini et Zhang [117] ont trouvé, en 2003, une attaque sur ce schéma et ont proposé de nouveaux schémas avec un nouvel algorithme de traçage de traîtres. Dans ce chapitre, nous montrons une attaque sur le schéma de To *et. al* et expliquons quelle était l'erreur dans leur analyse de sécurité. Dans le chapitre suivant, nous proposons une nouvelle construction qui devient la seule construction sûre fondée sur les couplages.

On rappelle brièvement la notion d'*application bilinéaire admissible*. On utilise les notations suivantes :

Notation 49

1. *$\mathcal{G}_1$ et $\mathcal{G}_2$ sont deux groupes cycliques d'ordre q, un grand nombre premier. $\mathcal{G}_1$ est un groupe additif et $\mathcal{G}_2$ est un groupe multiplicatif.*
2. *P est un générateur de $\mathcal{G}_1$.*
3. *e est une application $e : \mathcal{G}_1 \times \mathcal{G}_1 \to \mathcal{G}_2$.*
4. *$g = e(P, P)$.*

Définition 50 (Applications bilinéaires) *Étant donné deux groupes $\mathcal{G}_1$ et $\mathcal{G}_2$ (définis ci-dessus). Une application $e : \mathcal{G}_1 \times \mathcal{G}_1 \to \mathcal{G}_2$ est dite* application bilinéaire admissible *si elle satisfait les trois propriétés suivantes :*

Bilinéarité : $e(aU, bV) = e(U, V)^{ab}$ *quels que soient $U, V \in \mathcal{G}_1$ et $a, b \in \mathbb{Z}_q$;*

Non-dégénération : $e(P,P) \neq 1$;

Efficacement calculable : *Il existe un algorithme efficace pour calculer* $e(U,V)$ *pour tous* $U, V \in \mathcal{G}_1$.

Remarquons que, comme $\mathcal{G}_1, \mathcal{G}_2$ sont des groupes d'ordre premier et e est non-dégénérée, si P est un générateur de $\mathcal{G}_1$ alors $g = e(P,P)$ est un générateur de $\mathcal{G}_2$.

7.2.1 Exemples

On peut utiliser les couplages de Weil et de Tate pour construire des applications bilinéaires admissibles [118, 68].

7.2.2 Description du schéma de TSZ

Le schéma de To, Safavi-Naini et Zhang (appelé TSZ) est fondé sur une application bilinéaire $e : \mathcal{G}_1 \times \mathcal{G}_1 \to \mathcal{G}_2$, où $\mathcal{G}_1, \mathcal{G}_2$ sont des groupes d'ordre premier q.

Initialisation : choisir aléatoirement deux générateurs $P, Q \in \mathcal{G}_1$ et un polynôme unitaire de degré $2k-1$ à coefficients dans $\mathbb{Z}_q$:

$$f(x) = a_0 + a_1 x + \ldots + a_{2k-2} x^{2k-2} + x^{2k-1}.$$

Posons $Q_0 = a_0 Q$, $Q_1 = a_1 Q$, $\ldots$, $Q_{2k-2} = a_{2k-2} Q$ et $g = e(P,Q) \in \mathcal{G}_2$.

Clef secrète du centre : le générateur P et le polynôme f

Clef de chiffrement : le multiplet $(g, Q, Q_0, Q_1, \ldots, Q_{2k-2})$

Clef d'usager u **:** $K_u = f(u)^{-1} P$

Algorithme de chiffrement : l'algorithme de chiffrement génère un aléa $r \in \mathbb{Z}_q$. Ensuite, une clef de session $s \in \mathcal{G}_2$ est chiffrée par : $c = (sg^r, rQ, rQ_0, \ldots, rQ_{2k-2})$.

Algorithme de déchiffrement : grâce à la clef K_u, l'usager u calcule g^r pour retrouver s :

$$g^r = e(K_u, rQ_0) \cdot \ldots \cdot e(u^{2k-2} K_u, rQ_{2k-2}) \cdot e(u^{2k-1} K_u, rQ).$$

7.2.3 Attaque du schéma TSZ

Dans [117], les auteurs ont montré qu'une coalition d'au moins k usagers ne peut produire un décodeur anonyme. Ici, nous expliquons comment *un seul* usager peut construire un décodeur anonyme. Considérons un usager u, il choisit aléatoirement des éléments $z_0, z_1, \ldots, z_{2k-2} \in^R \mathbb{Z}_q$ et construit le décodeur suivant

– $X_i = u^i K_u + z_i Q$, pour i allant de 0 à $2k-2$.

– X_{2k-1} est déterminé par la relation :

$$X_{2k-1} = u^{2k-1}K_u - (z_0Q_0 + z_1Q_1 + \ldots + z_{2k-2}Q_{2k-2}). \tag{7.1}$$

L'usager u peut donc publier $X_0, X_1, \ldots, X_{2k-1}$, ce qui permet à n'importe qui de calculer g^r pour retrouver s :

$$g^r = e(X_0, rQ_0) \cdot \ldots \cdot e(X_{2k-2}, rQ_{2k-2}) \cdot e(X_{2k-1}, rQ).$$

Nous montrons que cette construction de décodeur est intraçable, *i.e.* à partir de $X_0, \ldots, X_{2k-2}, X_{2k-1}$, il est impossible de retrouver l'usager u. En effet, $X_0, \ldots, X_{2k-2}$, X_{2k-1} satisfont la relation suivante :

$$\begin{aligned} \sum_0^{2k-1} a_i X_i &= \left(\sum_0^{2k-2} a_i u^i\right) K_u + \left(\sum_0^{2k-2} a_i z_i\right) Q + u^{2k-1}K_u - \left(\sum_0^{2k-2} z_i a_i\right) Q \\ &= f(u)K_u = P. \end{aligned} \tag{7.2}$$

Remarquons aussi que, pour chaque usager et pour tout i ($i = 0, \ldots, 2k-2$), l'application de $\mathbb{Z}_q$ vers $\mathcal{G}_1$ qui envoie z_i sur X_i est une bijection. Alors, au lieu de choisir $z_0, z_1, \ldots, z_{2k-2}$, on peut choisir aléatoirement $X_0, \ldots, X_{2k-2}$ dans $\mathcal{G}_1$, qui détermine $(z_0, z_1, \ldots, z_{2k-2})$. La valeur X_{2k-1} est donc déterminée de façon unique par la relation (7.1). Elle satisfait également la relation (7.2). Comme elle détermine une valeur unique, on peut l'utiliser pour caractériser X_{2k-1} : elle est donc indépendante de l'usager u.

On peut facilement voir que tout usager peut produire un même ensemble de décodeurs pirates avec les paramètres $(X_0, X_1, \ldots, X_{2k-2}, X_{2k-1})$ où $X_0, X_1, \ldots, X_{2k-2}$ sont aléatoirement choisis dans $\mathcal{G}_1^{2k-1}$ et X_{2k-1} est défini par la relation (7.2).

7.2.4 Faille dans l'analyse de sécurité

Notons que cette attaque est bien différente de celle contre le schéma de Mitsunari *et. al* dans [117]. Notre construction de décodeur pirate combine en effet l'information de la clef privée d'un usager et l'*information publique* du système. Elle montre une faille dans l'algorithme de traçage dans [117].

La procédure de traçage dans [117] fonctionne comme suit : pour un ensemble d'usagers suspects $A = \{u_1, \ldots, u_t\}$ ($t \leq k$), on construit un autre polynôme $f'(x) = f(x) + \alpha(x - u_1)\ldots(x - u_t)$. Pour chaque usager dans A, la clef de u est la même qu'elle soit dans le schéma utilisant f (appelé $\mathsf{S}(f)$) ou dans le schéma utilisant f' (appelé $\mathsf{S}(f')$). De ce fait, To *et. al* déclarent que si tous les traîtres sont dans A, alors tout décodeur pirate produit par ces traîtres dans $\mathsf{S}(f)$ est aussi un décodeur pirate dans $\mathsf{S}(f')$. Par conséquent, ce décodeur pirate déchiffre de la même façon un texte chiffré selon $\mathsf{S}(f')$ et un texte chiffré selon $\mathsf{S}(f)$. Grâce à cet argument et à l'envoi d'un texte chiffré au décodeur pirate, le centre peut facilement détecter si les traîtres sont inclus dans A.

Malheureusement, ce raisonnement est fausse. En effet, il suppose que la construction du décodeur pirate ne dépende que des clefs privées des usagers et pas des informations

publiques. Lorsque la construction du décodeur dépend aussi des informations publiques (qui sont évidemment disponibles aux traîtres), l'algorithme de traçage du schéma échoue, comme le montre le contre-exemple ci-dessus.