Introduction

Dans la mise en œuvre des schémas de chiffrement, l'efficacité est un des facteurs les plus importants. Depuis longtemps, on essaie de construire des schémas de chiffrement à la fois efficaces et sûrs. Malheureusement, la sécurité forte va rarement de pair avec la mise en œuvre pratique. Un compromis doit être fait et en général, on fait quelques hypothèses sur l'attaque. À titre d'exemple, elle pourrait être générique et indépendante de l'exécution réelle de certains objets tels que les fonctions de hachage (dans le modèle de l'oracle aléatoire [49, 9]), le chiffrement symétrique par bloc (dans le modèle du « chiffrement idéal ») ou les groupes algébriques (dans le modèle générique [27]). Parmi ces hypothèses idéalistes, celle sur les fonctions de hachage — le modèle de l'oracle aléatoire — est la plus étudiée. En effet, nombre de schémas cryptographiques utilisent des fonctions de hachage telles que MD5 [105], les normes américaines SHA-1 [90], SHA-256, SHA-384 et SHA-512 [91].

Dans le « modèle de l'oracle aléatoire », introduit par Bellare et Rogaway [9], la fonction de hachage est formalisée comme un oracle qui retourne une valeur parfaitement aléatoire, sous réserve qu'elle produise toujours le même résultat pour les entrées identiques. Ce modèle est une passerelle entre la cryptographie théorique et la cryptographie pratique : le schéma est d'abord prouvé sûr dans le modèle de l'oracle aléatoire, les oracles sont remplacés par des fonctions réelles de hachage convenablement choisies. Bien qu'elle ne fournisse que des preuves de sécurité idéalisées, cette méthode a permis la mise au point de schémas particulièrement efficaces comme OAEP (*Optimal Asymmetric Encryption Padding*) [10] pour le chiffrement ou PSS (*Probabilistic Signature Scheme*) [11, 35] pour la signature.

Dans la littérature, plusieurs séparations entre le modèle standard et celui de l'oracle aléatoire ont déjà été montrées [29, 88, 5, 62, 30] (quelques unes ont également été montrées par rapport au modèle générique [116]). Cependant, les contre-exemples sont artificiels et contre-intuitifs ; ils ne sont pas applicables aux scénarios réels. De plus, une preuve de sécurité dans le modèle de l'oracle aléatoire sous une hypothèse faible peut apporter plus d'intérêt pratique qu'une preuve de sécurité dans le modèle standard sous une hypothèse forte. Le modèle de l'oracle aléatoire est donc encore largement accepté.

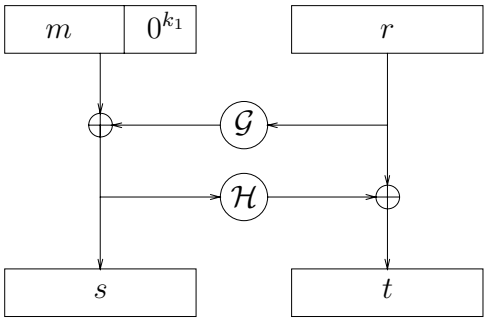

FIG. 4.1 – Optimal Asymmetric Encryption Padding

Dans cette partie, nous mettons l'accent sur l'efficacité des schémas de chiffrement et considérons la construction la plus célèbre du chiffrement à clef publique : la construction OAEP, introduite par Bellare et Rogaway [10].

4.1 OAEP

Avec l'introduction de la notion formelle du modèle de l'oracle aléatoire en 1993, Bellare et Rogaway ont proposé un schéma efficace de chiffrement [10] pour optimiser l'efficacité. Cet objectif entraîne deux problèmes de minimisation : l'un concerne l'écart entre la taille du texte chiffré et celle du texte clair (*i.e.* la bande passante) et l'autre, la taille du chiffré par rapport à celle de la primitive utilisée. Leur construction, appelée OAEP, résout le deuxième problème d'optimisation : avec une permutation à sens-unique à trappe de n bits vers n bits, la longueur du texte chiffré peut être optimale, *i.e.* n bits. Cette construction, illustrée dans la figure 4.1, permet de convertir une permutation à sens-unique à trappe en un schéma de chiffrement [10] décrit comme suit :

4.1.1 Description

Notations et paramètres communs

Le chiffrement et le déchiffrement utilisent deux fonctions de hachage : $\mathcal{G}$, $\mathcal{H}$, modélisées par des oracles aléatoires dans les analyses de sécurité. Les paramètres de sécurité satisfont $n = k + \ell$:

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

Le chiffrement utilise une famille de permutations à sens-unique (φ_{pk}) (pour simplifier les notations, la deuxième indice pk signifie l'espace des clef publiques) dont les inverses sont respectivement les ψ_{sk}, où sk est la clef secrète (*i.e.* la trappe) associée à la clef publique

pk. Le symbole « ‖ » dénote la concaténation des chaînes de bits. De plus, on identifie $\{0,1\}^k \times \{0,1\}^\ell$ à $\{0,1\}^n$.

Algorithme de chiffrement

L'espace des textes clairs est $\mathcal{M} = \{0,1\}^{\ell-k_1}$. Le chiffrement utilise un aléa $r \in \mathcal{R} = \{0,1\}^k$ et retourne un texte chiffré c dans $\mathrm{F} = \{0,1\}^n$: à partir du texte clair $m \in \mathcal{M}$, on calcule :

$$s = (m\|0^{k_1}) \oplus \mathcal{G}(r) \qquad t = r \oplus \mathcal{H}(s).$$

Le texte chiffré correspondant est alors $c = \varphi_{\mathsf{pk}}(s,t)$.

Algorithme de déchiffrement

A partir du texte chiffré c, on calcule d'abord $s\|t = \psi_{\mathsf{sk}}(c)$, où $s \in \{0,1\}^k$ et $t \in \{0,1\}^\ell$, puis

$$r = t \oplus \mathcal{H}(s) \quad M = s \oplus \mathcal{G}(r).$$

Si les k derniers bits de M sont tous 0, le déchiffrement retourne le message m qui est M sans les k derniers bits. Dans le cas contraire, le déchiffrement retourne « chiffré non valable », *i.e.* le texte chiffré ne correspond à aucun texte clair.

L'instantiation du schéma avec la fonction RSA [107], appelée RSA-OAEP, devient la norme pour le chiffrement RSA (PKCS#1 version 2, IEEE P1363). Ce schéma est très efficace en termes de temps de calcul et de l'expansion de message, il est aussi compatible avec des applications plus traditionnelles du chiffrement RSA. Outre son efficacité, la raison déterminante de la popularité de f-OAEP fut la confiance en sa sécurité : sémantiquement sûr contre des attaques à chiffrés choisis adaptatives, sous réserve que la permutation utilisée soit à sens-unique à trappe—malgré aucune preuve complète.

Cependant, en 2001, Shoup [114] a montré que pour ce schéma, la sécurité ne peut reposer sur l'hypothèse générale de l'existence de permutations à sens-unique à trappe. En effet, il a introduit une nouvelle technique, appelée la technique des « jeux successifs », pour prouver la sécurité. Appliquée au schéma OAEP, Shoup s'est heurté à un obstacle insurmontable. Il en a alors déduit un contre-exemple pour mettre en évidence le problème de sécurité d'OAEP. Cette technique sera étudiée en détail dans la section 4.2

4.1.2 Attaque de Shoup

Tout d'abord, on rappelle les étapes pour casser un schéma au sens de la sécurité sémantique : l'attaquant choisit deux messages m_0 et m_1 ; reçoit un challenge c-le chiffré d'un de ces deux messages ; et finalement, devine à quel message c correspond. Dans le jeu d'attaque CCA2, on peut accéder à l'oracle de déchiffrement à n'importe quel moment. L'idée de Shoup est décrite comme suit :

Quand on reçoit le challenge c, on vise à produire un autre texte chiffré c' tel que les valeurs r et r' dans la construction de ces deux chiffrés soient égales. Dans le cas où $r = r'$, on obtient une relation entre les deux texte clairs : $m \oplus m' = s \oplus s'$. Par conséquent, si on connaît s, s', grâce à une seule requête de déchiffrement sur le texte chiffré c', on obtient m', puis le m correspondant au challenge c. Le problème restant est de comprendre la façon avec laquelle un tel texte chiffré c' est généré. Pour cela, Shoup a introduit la notion de permutation à sens-unique à trappe « XOR-malléable » : pour une telle permutation f_0, on peut, avec probabilité non-négligeable, calculer $f_0(t \oplus a)$ à partir de $f_0(t)$ et a.

Supposons maintenant que l'on dispose d'une permutation f_0 à sens-unique à trappe « XOR-malléable ». Définissons f par $f(s||t) = s||f_0(t)$. Il est facile de montrer que f est également une permutation à sens-unique à trappe. Alors, du challenge $c = f(s||t) = s||f_0(t)$, on extrait directement s. On choisit aléatoirement s' et calcule $a = \mathcal{H}(s) \oplus \mathcal{H}(s')$. À partir de a, $f_0(t)$, grâce à la propriété de permutation XOR-malléable, l'attaquant peut calculer $f_0(t' = t \oplus \mathcal{H}(s) \oplus \mathcal{H}(s'))$. Considérons la valeur r' du texte chiffré $c' = s'||f_0(t') : r' = t' \oplus \mathcal{H}(s') = t \oplus \mathcal{H}(s) \oplus \mathcal{H}(s') \oplus \mathcal{H}(s') = t \oplus \mathcal{H}(s) = r$. Par conséquent, par le même raisonnement, une fois que c' est demandé à l'oracle de déchiffrement, on peut calculer le texte clair m du m' retourné par l'oracle de déchiffrement.

Dans son article, Shoup a construit un modèle de calcul non standard où il existe une permutation à sens-unique à trappe XOR-malléable. Par conséquent, il est impossible de formuler une preuve de la propriété $\mathsf{IND-CCA2}$ sûr de f-OAEP en utilisant la fonction f comme une boîte noire. f-OAEP ne pourrait être $\mathsf{IND-CCA2}$ sûr qu'avec des propriétés spécifiques de la fonction f.

4.1.3 Sécurité de RSA-OAEP

Heureusement, OAEP est presque immédiatement sauvé par Fujisaki *et al.* [54]. Ces auteurs ont montré que : premièrement, l'utilisation d'une permutation à « sens-unique partielle » [3] à trappe dans OAEP garantit son niveau fort de sécurité ; deuxièmement, la fonction RSA est une permutation à sens-unique partielle à trappe sous l'hypothèse bien connue qu'elle soit simplement une permutation à sens-unique à trappe. La sécurité de RSA-OAEP est définitivement prouvée. Cependant, la réduction dans leur preuve n'est pas aussi bonne qu'espérée. En effet, le coût de la réduction du problème d'inverser partiellement la fonction RSA au problème de sécurité de RSA-OAEP est quadratique en temps. De plus, le coût de la réduction du problème d'inverser complètement la fonction RSA au problème d'inverser partiellement la fonction RSA est aussi quadratique en temps.

[3] Une fonction f est k-partiellement à sens-unique si, à partir de $f(x)$, il est calculatoirement difficile de trouver les k premiers bits de x.

4.2 Technique des jeux successifs

Au cours du temps, l'exigence du niveau de sécurité grandit. Les notions de sécurité deviennent ainsi de plus en plus complexes. Par conséquent, il est de plus en plus difficile de prouver la sécurité d'un schéma et de la vérifier. À titre d'exemple, la construction OAEP a été utilisée pendant une longue période sans être ni prouvée sûre ni attaquée. Dans ce contexte, Shoup a introduit la technique des « jeux successifs » pour une construction progressive des preuves de sécurité. Chaque étape est donc simple et facile à vérifier.

4.2.1 Description

La sécurité d'un schéma peut être évaluée au travers d'un « jeu » d'attaque entre un « attaquant » et un « challengeur ». L'attaquant et le challengeur sont des algorithmes probabilistes interactifs. Ils génèrent ainsi un espace probabiliste. Normalement, la définition de sécurité est liée à un « événement » particulier Ev : l'attaquant casse le schéma. La sécurité exige que, pour tout attaquant, la probabilité d'occurrence de cet événement soit « très proche » d'une valeur « cible », typiquement 0 ou $\frac{1}{2}$. La technique de jeux successifs consiste à changer petit à petit, pour chaque jeu, le comportement du challengeur. Le changement est effectué de telle manière que, entre deux jeux successifs, la différence de probabilité d'occurrence de l'événement Ev est « négligeable ». L'objectif est d'obtenir, dans le jeu final, une estimation de la probabilité d'occurrence de l'événement Ev. L'un des avantages de cette technique est que l'on peut facilement vérifier la preuve grâce à la simplicité des transitions des jeux successifs.

Les transitions entre deux jeux successifs les plus utilisées sont au nombre de trois :

- **Transition 1 : Transition basée sur l'indistinguabilité**. Un attaquant qui détecte le changement peut être transformé en un algorithme efficace, capable de distinguer deux distributions indistinguables (statistiquement ou calculatoirement).
- **Transition 2 : Transition basée sur un événements d'échec**. Les i-ème et $(i+1)$-ème jeux procèdent de façon identique sauf si un événement d'échec Ech se produit. Autrement dit :

  $$\Pr[\mathsf{Ev}_i \wedge \neg\mathsf{Ech}] = \Pr[\mathsf{Ev}_{i+1} \wedge \neg\mathsf{Ech}].$$

  On constate que l'écart entre les probabilités d'occurrence des événements Ev_i et Ev_{i+1} dans les deux jeux successifs est, elle-même, « négligeable » si la probabilité d'occurrence de l'événement d'échec Ech est « négligeable ». Ceci résulte du lemme suivant (dont la preuve est évidente) :

  Lemme 39 *Soient* E, F *et* G *des événements dans un espace de probabilités, alors*

  $$\Pr[\mathsf{E} \wedge \neg\mathsf{G}] = \Pr[\mathsf{F} \wedge \neg\mathsf{G}] \implies |\Pr[\mathsf{E}] - \Pr[\mathsf{F}]| \leq \Pr[\mathsf{G}].$$

- **Transition 3 : Transition de ré-écriture**. Il s'agit d'une étape intermédiaire, qui décrit les façons équivalentes de calculer certaines quantités dans les jeux successifs. Cette transition semble inutile mais elle permet de suivre la preuve plus facilement.

L'état de l'art de cette technique est dans [115]

4.2.2 Exemple

Dans [9], outre la formalisation de la notion de l'oracle aléatoire, Bellare et Rogaway ont proposé une construction générique de chiffrement $\mathsf{IND}-\mathsf{CCA2}$ à partir de n'importe quelle permutation à sens-unique à trappe.

Cette construction fait appel à deux oracles aléatoires $\mathcal{G}$ et $\mathcal{H}$, à valeurs respectivement dans $\{0,1\}^k$ et $\{0,1\}^\ell$. L'algorithme de génération des clés définit une permutation φ_{pk} dans l'espace E de taille n bits, calculable grâce à la clé publique pk. Son inverse ψ_{sk} ne peut être calculée que grâce à la clef privée sk (la trappe). Pour chiffrer un message $m \in \{0,1\}^k$, on choisit $r \stackrel{R}{\leftarrow} E$, puis calcule :

$$\mathcal{E}(m;r) = \varphi_{\mathsf{pk}}(r) \,\|\, m \oplus \mathcal{G}(r) \,\|\, \mathcal{H}(m,r).$$

Le déchiffrement d'un chiffré $C = a \,\|\, b \,\|\, c$ se fait en deux étapes : tout d'abord, on retrouve $r = \psi_{\mathsf{sk}}(a)$, grâce à la trappe sk, puis $m = b \oplus \mathcal{G}(r)$; ensuite, avant de retourner le message m, on vérifie la consistance du chiffré, *i.e.* si $c = \mathcal{H}(m,r)$.

Ce schéma est prouvé $\mathsf{IND}-\mathsf{CCA2}$ sûr sous l'hypothèse que la famille des permutation (φ_{pk}) φ_{pk} est à sens-unique.

Théorème 40 *[102] Considérons un attaquant $\mathcal{A}$ à chiffrés choisis adaptatif. Supposons qu'après q_D requêtes à l'oracle de déchiffrement et q_G, q_H requêtes aux oracles $\mathcal{G}$ et $\mathcal{H}$, $\mathcal{A}$ a un avantage ε en temps t, alors on peut inverser φ avec succès $\varepsilon/2 - q_D/2^\ell$, en temps $t + (q_G + q_H)T_\varphi$, où T_φ désigne le temps pour une évaluation d'une fonction φ_{pk}.*

Une preuve utilisant la technique des jeux successifs est présentée dans [102]. On va l'étudier plus en détails pour apprécier l'efficacité de cette technique.

Preuve. Considérons l'attaquant $\mathcal{A} = (A_1, A_2)$ contre ce schéma. Dans les deux étapes, A_1 et A_2 ont accès à l'oracle de déchiffrement.

JEU $\mathbf{J}_0$: Il s'agit d'une simulation parfaite du comportement du challengeur. On exécute l'algorithme de génération de clés qui retourne une permutation φ_{pk} et son inverse ψ_{sk}. On génère également $x \stackrel{R}{\leftarrow} E$ et $y = \varphi_{\mathsf{pk}}(x)$. Après avoir vu la clé publique (la description de la fonction φ_{pk}), A_1 retourne deux messages m_0 et m_1. Après avoir reçu le chiffré $C^\star = a^\star \,\|\, b^\star \,\|\, c^\star$ du message m_δ, A_2 retourne un bit δ'. On dénote $r^\star$ l'unique élément tel que $C^\star = \mathcal{E}(m_\delta, r^\star)$. Avec probabilité $(\varepsilon+1)/2$, $\delta' = \delta$. On note cet événement S_0, ainsi que S_i dans les jeux Jeu_i ci-dessous : $\Pr[S_0] = (1+\varepsilon)/2$.

Pour la suite, on pourra supposer que toute question $\mathcal{H}(\star, \rho)$ est précédée de la question $\mathcal{G}(\rho)$, ainsi $q'_G = q_H + q_G$, où q'_G est le nombre total de requêtes à l'oracle $\mathcal{G}$.

JEU $\mathbf{J}_1$: dans un premier temps, on remplace les oracles $\mathcal{G}$ et $\mathcal{H}$ par des simulations classiques : pour toute nouvelle question à l'un de ces oracles, on répond par une chaîne

aléatoire dans l'espace correspondant, puis on stocke les questions-réponses respectivement dans les listes $\mathcal{G}$-List et $\mathcal{H}$-List. Il s'agit de simulations parfaites, $\Pr[S_1] = \Pr[S_0]$.

Commentaire. *Il s'agit d'une transition de type 1 : la simulation des oracles aléatoires est parfaite, les deux distributions sont parfaitement identiques.*

JEU $\mathbf{J}_2$: dans ce jeu, on simule l'oracle de déchiffrement. À la question $C = a \,\|\, b \,\|\, c$, pour $a = f(r)$, si r n'est pas dans $\mathcal{G}$-List, on rejette le chiffré ; si de même $(b \oplus G(r), r)$ n'est à son tour pas dans $\mathcal{H}$-List, on rejette le chiffré ; dans les autres cas, on continue à utiliser l'oracle de déchiffrement.

Avec l'hypothèse ci-dessus que toute question $\mathcal{H}(\star, \rho)$ est précédée de la question $\mathcal{G}(\rho)$, on ne peut refuser un chiffré valide que si $(b \oplus \mathcal{G}(r), r)$ n'a pas été demandé à $\mathcal{H}$. Mais alors, $\mathcal{H}(b \oplus \mathcal{G}(r), r)$ retourne un élément parfaitement aléatoire, qui est égal à c avec probabilité $1/2^{k_1}$. Ainsi, $|\Pr[S_2] - \Pr[S_1]| \leq q_D/2^{k_1}$.

Commentaire. *Il s'agit d'une transition de type 2 : l'événement d'échec* Ech *est « r n'est pas dans $\mathcal{G}$-List ou $(b \oplus \mathcal{G}(r), r)$ n'est pas dans $\mathcal{H}$-List ». À l'exception de cet événement, les deux jeux sont identiques.*

JEU $\mathbf{J}_3$: on poursuit la simulation de l'oracle de déchiffrement, sur $C = a \,\|\, b \,\|\, c$, pour $a = f(r)$. On sait que $r \in \mathcal{G}$-List et $(b \oplus \mathcal{G}(r), r) \in \mathcal{H}$-List. On peut alors trouver ce r (en testant si $\varphi_{\mathsf{pk}}(r) = a$ sur toutes les questions à $\mathcal{G}$, grâce à la propriété de permutation de φ_{pk}), puis déchiffrer correctement : $\Pr[S_3] = \Pr[S_2]$.

Commentaire. *Il s'agit d'une transition de type 3 : le jeu $\mathbf{J}_3$ est identique au jeu 2 mais m est calculé de manière différente. En effet, dans le jeu 2 , $r = f^{-1}(a)$, alors que dans le jeu $\mathbf{J}_3$, r sera extrait de $\mathcal{G}$-List (le cas où r n'est pas dans la liste est déjà exclu).*

JEU $\mathbf{J}_4$: dans ce jeu, on définit $a^\star = y = f(x)$, $b^\star = m_\delta \oplus g^+$ et $c = h^+$, où x, g^+ et h^+ sont aléatoires. De plus, à la question $\mathcal{G}(x)$, on répond g^+, et à la question $\mathcal{H}(m_\delta, x)$ on répond h^+. Il s'agit simplement de spécifier certaines valeurs de $\mathcal{G}$ et $\mathcal{H}$, par des valeurs aléatoires, on ne modifie donc pas les distributions : $\Pr[S_4] = \Pr[S_3]$.

Commentaire. *Il s'agit évidemment d'une transition de type 3. Cette transition est importante dans la mesure où l'instance x est insérée dans la génération du challenge.*

JEU $\mathbf{J}_5$: maintenant, on supprime les modifications locales de $\mathcal{G}$ et $\mathcal{H}$. Les réponses aux question $\mathcal{G}(x)$ et $\mathcal{H}(m_\delta, x)$ sont indépendantes de x et m_δ. La seule différence apparaît si l'événement « x a été demandé », nommé AskG (impliqué aussi par la requête à $\mathcal{H}$), a lieu : $|\Pr[S_5] - \Pr[S_4]| \leq \Pr[\mathsf{AskG}]$. Cependant, dans ce dernier jeu, δ est indépendant de la vue de l'attaquant, ainsi $\Pr[S_5] = 1/2$.

L'inégalité triangulaire nous donne :

$$\frac{\varepsilon}{2} = \frac{1+\varepsilon}{2} - \frac{1}{2} = |\Pr[S_0] - \Pr[S_5]| \leq \frac{q_D}{2^{k_1}} + \Pr[\mathsf{AskG}].$$

Commentaire. *Il s'agit d'une transition de type 2. L'événement d'échec est « x a été demandé »,* i.e. AskG. *Il nous reste à montrer que la probabilité d'occurrence de cet événement est « négligeable ». Cependant, l'événement* AskG *permet d'inverser* $\varphi_{\mathsf{pk}}(x)$ *en testant toutes les questions posées à* $\mathcal{G}$*, d'où le résultat.*

□

Cette construction est très efficace par rapport aux autres schémas dans le modèle standard. Cependant, elle n'est pas optimale car la taille du texte chiffré est encore longue, *i.e.* $n + k + \ell$ bits, par rapport à k bits du texte clair. Cette remarque est la motivation de OAEP.

5

Sécurité à chiffrés choisis sans redondance

Sommaire

En étudiant les limites des schémas précédents, nous proposons deux nouveaux schémas de chiffrement : le premier, dont l'efficacité est « optimale », est dans le modèle de la permutation aléatoire et le deuxième, dans le modèle de l'oracle aléatoire. Ce dernier, appelé f-OAEP 3 tours, dispose des propriétés importantes suivantes :

Sécurité : sous l'hypothèse d'une permutation f à sens-unique à trappe, ce schéma est sémantiquement sûr face aux attaques à chiffrés choisis adaptatives.

Sans redondance : tout chiffré correspond au chiffrement d'un texte clair. Il s'agit du premier schéma IND−CCA2 sans redondance. Cette propriété améliore l'efficacité du schéma en optimisant la bande passante.

Flexibilité : f n'est pas nécessairement une permutation mais peut être une fonction satisfaisant quelques conditions spécifiques. Quelques exemples de fonctions admissibles sont le chiffrement ElGamal ou celui de Paillier, d'où les nouvelles variantes d'OAEP 3 tours : ElGamal-OAEP 3 tours et Paillier-OAEP 3 tours. Rappelons que le chiffrement ElGamal ou celui de Paillier ne sont pas applicables aux précédentes variantes d'OAEP.

5.1 Chiffrement sans redondance

Dans le schéma OAEP ainsi que dans les variantes dont la primitive de base est une permutation à sens-unique à trappe (SAEP, SAEP+ [19] et OAEP+ [114]), le texte clair est toujours complété avec de la redondance avant d'être chiffré.

D'autres paddings, applicables à des familles de fonctions plus générales, ont été proposés par Fujisaki et Okamoto [53, 52], Pointcheval [100] et Okamoto et Pointcheval [93]. La sécurité y est également garantie par de la redondance, mais dans ce cas, la redondance concerne le texte chiffré : avant de retourner le texte chiffré, on lui ajoute de la redondance.

L'introduction de redondance dans le texte clair ou dans le texte chiffré a toujours le même objectif : un texte chiffré qui n'est pas proprement généré à partir d'un texte clair n'est valide qu'avec une probabilité négligeable. Cette propriété est formellement définie par la notion de *plaintext-awareness* dans [10, 7]. Elle entraîne le fait que l'oracle de déchiffrement ne donne aucune information aux attaquants.

Ainsi, la redondance renforce la sécurité. Cependant, dans certains cas, elle est source d'attaques. En effet, le mécanisme de déchiffrement pour les textes chiffrés valides pourrait être différent de celui pour les textes chiffrés invalides : après un test de validité, l'algorithme de déchiffrement continue la phase de calcul si le texte chiffré vérifie la forme de redondance, sinon il retourne immédiatement la réponse que le texte chiffré est invalide. Par conséquent, des attaques peuvent être menées [13, 79].

A ce stade, le problème qui se pose est donc d'éviter ces redondances dans le chiffrement. Ceci éviterait d'éventuelles attaques et, surtout, améliorerait l'efficacité du schéma.

5.2 FDP : permutation sur un domaine complet

Dans le même esprit que Full-Domain Hash signature [11, 34], nous proposons un chiffrement dans un nouveau modèle appelé permutation sur un domaine complet (Full-Domain Permutation). Dans ce modèle, nous appliquons d'abord une permutation aléatoire sur le couple texte clair - chaîne aléatoire, le résultat est ensuite chiffré avec une permutation à sens-unique à trappe. Cette construction conduit au premier schéma de

chiffrement $\mathsf{IND}-\mathsf{CCA2}$ sûr, à bande passante optimale et sans redondance, *i.e.* tout chiffré est valide.

5.2.1 Description

Le chiffrement FDP est doté d'une grande efficacité grâce à l'utilisation d'une permutation aléatoire $\mathcal{P}$ (qui est un oracle aléatoire bijectif ou un chiffrement idéal avec une clef particulière, 0 par exemple, voir aussi [108]). L'algorithme de génération de clef sélectionne une permutation à sens-unique à trappe φ_{pk} (son inverse ψ_{sk}, calculable à l'aide de la trappe sk) sur $\{0,1\}^{\ell+k}$, et une permutation aléatoire $\mathcal{P}$ sur le même espace $\{0,1\}^{\ell}\times\{0,1\}^{k}$ qui est identifié à $\{0,1\}^{\ell+k}$. La clef publique définit la permutation φ_{pk}, alors que la clef secrète sk définit l'inverse ψ_{sk} de φ_{pk}. Ainsi,

$$\mathcal{E}_{\mathsf{pk}}(m;r)=\varphi_{\mathsf{pk}}(\mathcal{P}(m,r)) \qquad \mathcal{D}_{\mathsf{sk}}(c)=m,\ \text{où}\ (m,r)=\mathcal{P}^{-1}(\psi_{\mathsf{sk}}(c)).$$

L'espace des textes clairs est $\{0,1\}^{\ell}$, alors que celui des aléas r est $\{0,1\}^{k}$. Remarquons que $\mathcal{P}$ et $\mathcal{P}^{-1}$ sont des permutations publiques.

5.2.2 Résultat de sécurité

Les avantages de ce schéma sont au nombre de trois :
- Le premier, comme déjà mentionné, est la validité de tout texte chiffré : n'importe quel élément de l'espace d'arrivée de l'algorithme de chiffrement peut être atteint par cet algorithme.
- Le deuxième découle du résultat de sécurité dans le théorème 41 ci-dessous : il atteint la sécurité $\mathsf{IND}-\mathsf{CCA2}$ sous la seule hypothèse de la difficulté d'inverser φ.
- Le troisième vient de l'optimalité de la bande passante : pour un niveau de sécurité 2^k, on n'a besoin que d'une chaîne aléatoire de k bits. Évidemment, cette remarque est applicable uniquement au cas général où $\ell \geq k$ (*e.g.*, $k=80$ et $k+\ell=1024$).

Théorème 41 *Considérons un attaquant $\mathcal{A}$ contre φ-FDP, selon une attaque à chiffrés choisis adaptative. Supposons qu'après q_p et q_d requêtes respectivement aux oracles de permutation et de déchiffrement, $\mathcal{A}$ peut avoir un avantage ϵ en temps τ, alors on peut inverser φ avec succès $\mathsf{Succ}^{\mathsf{ow}}_{\varphi}(\tau+2q_p\times T_\varphi)$ en temps $\tau+2q_p\times T_\varphi$, où T_φ est le temps nécessaire pour une évaluation d'une fonction de φ, où :*

$$\epsilon \leq 2\times \mathsf{Succ}^{\mathsf{ow}}_{\varphi}(\tau+2q_p\times T_\varphi)+2\times\left(\frac{(q_p+q_d+1)^2}{2^{k+\ell}}+\frac{q_p}{2^k}+\frac{(q_d+1)^2}{2^{\ell}}\right).$$

Rappelons brièvement que pour tout algorithme $\mathcal{A}$:

$$\mathsf{Succ}^{\mathsf{ow}}_{\varphi}(\mathcal{A})=\Pr_{\substack{\mathsf{pk},\\ x\in\{0,1\}^{k+\ell}}}[\mathcal{A}(\mathsf{pk},\varphi_{\mathsf{pk}}(x))=x],\ \text{et}\ \mathsf{Succ}^{\mathsf{ow}}_{\varphi}(\tau)=\max_{|\mathcal{A}|\leq\tau}\left\{\mathsf{Succ}^{\mathsf{ow}}_{\varphi}(\mathcal{A})\right\}.$$

Oracle $\mathcal{P}$

Une requête $\mathcal{P}(m, r)$ aura une réponse p, où

►**Règle** $\mathsf{EvalP}^{(5)}$

| $p = P(m, r)$.

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

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

Une requête $\mathcal{P}^{-1}(p)$ aura une réponse (m, r), où

►**Règle** $\mathsf{InvP}^{(5)}$

| $(m, r) = P^{-1}(p)$.

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

Oracle $\mathcal{D}$

Une requête $\mathcal{D}_{\mathsf{sk}}(c)$ aura une réponse m, où

►**Règle** $\mathsf{Decrypt}^{(5)}$

| $p = \psi_{\mathsf{sk}}(c)$, et $(m, r) = \mathcal{P}^{-1}(p)$.

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

Challengeur

Des deux messages (m_0, m_1), choisir un bit b et définir $m^\star = m_b$, puis choisir aléatoirement $r^\star$.

►**Règle** $\mathsf{Chal}^{(5)}$

| $p^\star = \mathcal{P}(m^\star, r^\star)$; $c^\star = \varphi_{\mathsf{pk}}(p^\star)$.

►**Règle** $\mathsf{ChalAdd}^{(5)}$

| Ajouter $(m^\star, r^\star, p^\star, c^\star)$ à $\mathcal{P}$-List.

Répondre $c^\star$

FIG. 5.1 – Simulation formelle dans le jeu $\mathsf{IND-CCA}$ contre φ-FDP

5.2.3 Idée de la preuve

L'objectif est de simuler les oracles $\mathcal{P}$, $\mathcal{P}^{-1}$ et $\mathcal{D}_{\mathsf{sk}}$ de telle manière que l'attaquant ne puisse distinguer les simulations des oracles réels. Dans cette simulation, la réponse de l'algorithme de déchiffrement à un texte chiffré, qui n'avait pas été obtenu, se veut être une nouvelle valeur aléatoire (indépendante des autres). Afin d'obtenir cette propriété, nous devons rendre la simulation de la permutation aléatoire cohérente. D'autre part, le challenge sera rendu indépendant des textes clairs m_0 et m_1 : l'attaquant n'aura aucun avantage.

La preuve est constituée des modifications successives des règles appliquées dans la simulation (parfaite), où les oracles $\mathcal{P}$ et $\mathcal{P}^{-1}$ sont initialement simulés par l'utilisation d'une permutation parfaitement aléatoire P et son inverse P^{-1}. Le dernier jeu fournira

une simulation de $\mathcal{D}_{\mathsf{sk}}$ sans inverser φ_{pk}.

La simulation est parfaite à moins que l'attaquant ne fasse une requête sur la préimage du challenge à la permutation aléatoire $\mathcal{P}^{-1}$. Mais dans ce cas, l'attaquant aide aussi à inverser la fonction φ.

La preuve est en fait un cas particulier de celle du théorème 47 qui va être présentée en détails dans le chapitre suivant.

5.3 OAEP 3 tours : un padding générique et efficace

5.3.1 Relâchement de la sécurité CCA

À Eurocrypt '02, An *et al* [2] ont proposé la notion de sécurité « CCA généralisée », où l'attaquant n'a pas le droit de demander des textes chiffrés ayant une certaine *relation* avec le challenge à l'oracle de déchiffrement. Cette relation doit être une relation publique et efficacement calculable. De plus, elle doit respecter l'équivalence de déchiffrement (si deux textes chiffrés sont en relation, ils chiffrent nécessairement le même texte clair). Ce relâchement a pour but de montrer que des bits supplémentaires, qui peuvent être facilement ajoutés ou supprimés, ne rendent pas le schéma théoriquement fragile. En effet, d'un point de vue pratique, ils ne jouent aucun rôle.

Un autre relâchement a été récemment proposé par Canetti *et al* [31]. Il s'agit d'une extension de la relation ci-dessus. Lorsqu'une requête c est soumise à l'oracle de déchiffrement, elle est, dans un premier temps, déchiffrée en m : si m est identique à un des deux textes clairs retournés par l'attaquant à la fin de la première étape (m_0 ou m_1), l'oracle refuse de répondre (retourne $\perp$ ou test), sinon il retourne m. Les auteurs appellent cette variante la sécurité « CCA rejouable ». D'après eux, ce niveau de sécurité, bien que plus faible que le niveau usuel CCA, est suffisant dans presque toutes les applications pratiques.

Dans notre travail, nous pouvions utiliser le deuxième relâchement, le « CCA rejouable ». Cependant, afin d'obtenir une preuve de sécurité plus simple et un résultat plus précis (avec un corollaire intéressant pour les cas particuliers tels que RSA), nous proposons une notion de sécurité plus forte que « CCA rejouable » : « CCA relâchée », dénotée RCCA. Un schéma sûr dans ce scénario l'est dans celui de « CCA rejouable », mais ne l'est pas nécessairement dans celui de « CCA généralisée » ou « CCA classique ». Les relations entre ces scénarios dépendent de la manière dont la chaîne aléatoire est utilisée. Dans la notation formelle de l'algorithme de chiffrement, on découpe en effet la chaîne aléatoire en deux parties r et ρ : $c = \mathcal{E}_{\mathsf{pk}}(m; r, \rho)$. L'algorithme de chiffrement est donc une fonction de $\mathcal{M} \times \mathcal{R} \times \mathrm{R}$ dans l'ensemble des textes chiffrés. Pour garantir la consistance du schéma de chiffrement, elle doit être une injection par rapport à $\mathcal{M}$ (plusieurs éléments de $\mathcal{M} \times \mathcal{R} \times \mathrm{R}$ peuvent correspondre à un même texte chiffré, mais leur projection sur $\mathcal{M}$ doit obligatoirement donner un texte clair unique). Dans le cadre de notre relâchement, la chaîne aléatoire $\mathcal{R} \times \mathrm{R}$ est découpée de telle manière que le chiffrement soit une injection

par rapport à $\mathcal{M} \times \mathcal{R}$.

Soit le challenge $c^\star = \mathcal{E}_{\mathsf{pk}}(m^\star; r^\star, \rho^\star)$. Considérons le texte chiffré $c = \mathcal{E}_{\mathsf{pk}}(m; r, \rho)$, d'après les commentaires précédents, $(m^\star, r^\star)$ et (m, r) sont uniques et définis respectivement à partir de $c^\star$ et c, tandis que $\rho^\star$ et ρ pourraient ne pas être uniques. A la réception de c, un *oracle de déchiffrement relâché* est défini pour vérifier si $(m^\star, r^\star) = (m, r)$ auquel cas il retourne test (*i.e.* il refuse de répondre), sinon, il retourne m.

Définition 42 (CCA relâché) *Dans le scénario « CCA relâché », un attaquant a accès à l'oracle de déchiffrement relâché mais pas à l'oracle de déchiffrement standard.*

Propriété 43 *La sécurité dans le scénario « CCA relâché » implique la sécurité dans le scénario « CCA rejouable ».*

Preuve. Il s'agit d'ici d'une relation triviale car l'oracle à « CCA rejouable » peut être facilement simulé par l'oracle « relâché » : s'il retourne test, cette valeur sera transmise, sinon, la réponse m sera comparée avec les deux textes clairs $\{m_0, m_1\}$, retournés par l'attaquant à la fin de la première étape. A l'issue de cette comparaison, test sera retournée si $m \in \{m_0, m_1\}$, sinon m sera la réponse. □

Cette propriété montre que notre cadre de travail reste acceptable d'un point de vue pratique. De plus, si R est un ensemble vide et si f est une bijection, RCCA sera identique à CCA.

5.3.2 Primitive de base

Notre objectif est de proposer une variante d'OAEP qui peut être utilisée avec une grande classe de fonctions à sens-unique. Plus précisément, on n'a besoin que d'une famille de fonctions injectives, *probabilistes* à sens-unique à trappe. Une telle famille est dénotée (φ_{pk}) de l'ensemble E_{pk} vers l'ensemble F_{pk}, pour tout indice pk. Nous verrons que plusieurs primitives de chiffrement, où les espaces de textes clairs et de textes chiffrés sont respectivement représentés par E_{pk} et F_{pk}, sont convenables : pour chaque pk (la clef publique), il existe une fonction ψ_{sk} (où sk est la clef secrète) qui retourne la pré-image dans E_{pk}. Une famille de fonctions injectives, *probabilistes* à sens-unique à trappe de E vers F contient des fonctions $f : \mathrm{E} \times \mathrm{R} \to \mathrm{F}$, qui prend un couple (x, ρ) et retourne $y \in \mathrm{F}$. L´élément x, dans E, est considéré comme l'entrée et ρ est une chaîne aléatoire dans R qui rend la fonction probabiliste. L'injectivité implique que pour tout y, il existe au plus un x (mais peut-être plusieurs ρ) tel que $y = f(x, \rho)$. La fonction g, sur l'entrée y, retourne x. Cette fonction est l'inverse de la fonction probabiliste f. Clairement, on a besoin que la fonction f soit efficacement calculable et que son inverse soit difficilement calculable sans la trappe (la propriété à sens-unique à trappe). Cependant, pour prouver la sécurité de notre construction, nous avons besoin de deux propriétés additionnelles :

– la fonction $f : \mathrm{E} \times \mathrm{R} \to \mathrm{F}$ est une bijection ;

– sans la connaissance de la trappe, il est difficile d'inverser f, même si on a accès à un oracle décisionnel $\mathsf{Same}_f(y, y')$ qui vérifie si $g(y) = g(y')$.

La deuxième propriété est en effet la notion du problème de « séparation » (« gap problem » en anglais), qui est défini par la probabilité de succès $\mathsf{Succ}_f^{\mathsf{gap}}(t, q)$: pour tout attaquant $\mathcal{A}$ dont le temps de calcul est borné par t, et le nombre de requêtes à l'oracle décisionnel Same_f est borné par q :

$$\mathsf{Succ}_f^{\mathsf{gap}}(t, q) = \max_{\mathcal{A}}\{x \stackrel{R}{\leftarrow} \mathrm{E}, \rho \stackrel{R}{\leftarrow} \mathrm{R}, y = f(x, \rho) : \mathcal{A}^{\mathsf{Same}_f}(y) = x\}.$$

Pour une famille de fonctions (φ_{pk}), on note $\mathsf{Succ}_{\varphi}^{\mathsf{gap}}(t, q)$ la probabilité de succès $\mathsf{Succ}_{\varphi_{\mathsf{pk}}}^{\mathsf{gap}}(t, q)$ où pk est choisie aléatoirement dans l'espace des clefs publiques.

Considérons ces deux propriétés d'un point de vue pratique :

– Le premier exemple à considérer est évidemment la permutation RSA [107]. Pour une clef publique $\mathsf{pk} = (n, e)$ donnée, les ensembles $\mathrm{E}, \mathrm{F}, \mathrm{R}$ sont définis par : $\mathrm{E} = \mathrm{F} = \mathbb{Z}_n^\star$ et R est l'ensemble vide. Il s'agit clairement d'une fonction injective (et déterministe), qui est, en outre, une bijection. Grâce au déterminisme, l'oracle décisionnel $\mathsf{Same}(y, y')$ vérifie simplement si $y = y'$: le problème de séparation RSA est ainsi le problème classique de RSA.
– L'objectif de notre extension d'OAEP est de l'appliquer au chiffrement bien connu d'ElGamal [45] dans un groupe cyclique $\mathbb{G}$ d'ordre q, généré par g. Étant donnée une clef publique $\mathsf{pk} = y \in \mathbb{G}$, les ensembles $\mathrm{E}, \mathrm{R}, \mathrm{F}$ sont définis par : $\mathrm{E} = \mathbb{G}$, $\mathrm{R} = \mathbb{Z}_q$ et $\mathrm{F} = \mathbb{G} \times \mathbb{G}$. La fonction de chiffrement $\varphi_y(x, \rho) = (g^\rho, x \times y^\rho)$ est à la fois une injection probabiliste de E vers F et une bijection de $\mathrm{E} \times \mathrm{R}$ vers F. À propos de l'oracle décisionnel, il devra vérifier, sur les entrées $(a = g^\rho, b = x \times y^\rho)$ et $(a' = g^{\rho'}, b' = x' \times y^{\rho'})$, si $x = x'$, ce qui est équivalent à décider si $(g, y, a'/a = g^{\rho'-\rho}, b'/b = (x'/x) \times y^{\rho'-\rho})$ est un quadruplet Diffie-Hellman : le problème de séparation est ainsi le problème connu Gap Diffie-Hellman [93, 94].
– Avec des arguments similaires, on peut montrer que le chiffrement de Paillier [95] peut également être utilisé avec notre construction.

5.3.3 Description d'OAEP 3 tours

Notations et paramètres communs

Pour une présentation plus simple, et aussi des analyses de sécurité plus claires, nous nous concentrons sur le cas où $\mathrm{E} = \{0, 1\}^n$ (un ensemble binaire). Une analyse du cas plus général est considérée dans [96]. On remarque que la plupart des fonctions peut être, à faible coût, transformées afin de satisfaire cette contrainte [4]. Cette construction est illustrée sur la figure 5.2 Le chiffrement et le déchiffrement utilisent trois fonctions de hachage : $\mathcal{F}, \mathcal{G}, \mathcal{H}$, modélisées ultérieurement dans les analyses de sécurité par des oracles aléatoires. Les paramètres de sécurité satisfont $n = k + \ell$:

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

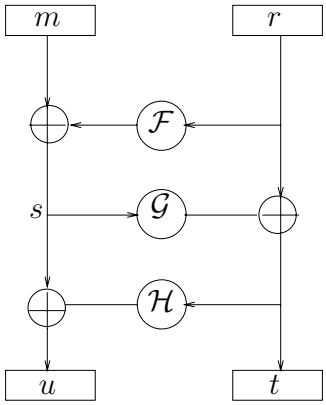

FIG. 5.2 – OAEP 3 tours

Le chiffrement utilise une famille de permutations à sens-unique (φ_{pk}) dont les inverses sont respectivement les ψ_{sk}, où sk est la clef secrète (*i.e.* la trappe) associée à la clef publique pk. Le symbole « $\|$ » dénote la concaténation des chaînes de bits. De plus, on identifie $\{0,1\}^k \times \{0,1\}^\ell$ à $\{0,1\}^n$.

Algorithme de chiffrement

L'espace des textes clairs est $\mathcal{M} = \{0,1\}^\ell$. Le chiffrement utilise deux chaînes aléatoires $r \in \mathcal{R} = \{0,1\}^k$ et $\rho \in \mathrm{R}$. A partir du texte clair $m \in \mathcal{M}$, il retourne un texte chiffré c de F :

$$s = m \oplus \mathcal{F}(r) \qquad t = r \oplus \mathcal{G}(s) \qquad u = s \oplus \mathcal{H}(t) \qquad c = \varphi_{\mathsf{pk}}(u\|t, \rho).$$

Algorithme de déchiffrement

A partir du texte chiffré c, on calcule d'abord $u\|t = \psi_{\mathsf{sk}}(c)$, où $u \in \{0,1\}^\ell$ et $t \in \{0,1\}^k$, puis retrouve le texte clair m par :

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

5.4 Résultat de sécurité

Dans cette section, nous présentons le résultat de l'analyse de sécurité du schéma OAEP 3 tours et la preuve de ce résultat.

Théorème 44 *Considérons un attaquant $\mathcal{A}$ contre la construction d'OAEP 3 tours, selon une attaque* IND−RCCA. *Supposons qu'après q_f, q_g, q_h et q_d requêtes respectivement aux*

oracles $\mathcal{F}$, $\mathcal{G}$, $\mathcal{H}$, et à l'oracle de déchiffrement, $\mathcal{A}$ peut avoir un avantage $\mathsf{Adv}^{\mathsf{ind-rcca}}_{\mathsf{oaep\text{-}3}}(\tau)$ *en temps τ, alors* $\mathsf{Succ}^{\mathsf{gap}}_{\varphi}(\tau', q_d(q_g q_h + q_d))$ *est borné par :*

$$\frac{1}{2} \times \mathsf{Adv}^{\mathsf{ind-rcca}}_{\mathsf{oaep\text{-}3}}(\tau') - q_d^2 \times \left(\frac{1}{2^\ell} + \frac{6}{2^k}\right) - (4q_d + 1) \times \left(\frac{q_g}{2^\ell} + \frac{q_f}{2^k}\right) - q_d \times \frac{q_f + 1}{2^k},$$

avec $\tau' \leq \tau + (q_f + q_g + q_h + q_d)T_{lu} + q_d^2 T_{\mathsf{Same}} + (q_d + 1)q_g q_h (T_\varphi + T_{\mathsf{Same}})$, où T_φ est le temps pour une évaluation d'une fonction φ_{pk}, T_{Same} est le temps pour une décision de l'oracle $\mathsf{Same}_{\varphi_{\mathsf{pk}}}$, et T_{lu} est le temps pour la consultation d'un élément dans une liste de taille inférieure au nombre total des requêtes.

5.4.1 Permutations à sens-unique

Avant de prouver ce résultat général, considérons le cas particulier où (φ_{pk}) est une famille de permutations. Le résultat général engendre en effet quelques inconvénients :

- le coût de la réduction présente une facteur cubique $q_d q_g q_h$ qui nécessite l'utilisation de grandes clefs afin d'obtenir un niveau de sécurité convenable.
- le résultat de sécurité repose sur le problème de séparation, qui est une hypothèse forte dans certains cas.
- l'impossibilité d'obtenir le niveau usuel de sécurité $\mathsf{IND{-}CCA}$.

Ces inconvénients sont acceptables car ils sont le prix de la généralisation qui s'applique aux chiffrements d'ElGamal et de Paillier. Cependant, pour les permutations à sens-unique à trappe telles que RSA, plusieurs autres variantes d'OAEP offrent une meilleure efficacité. Mais on devrait interpréter notre résultat dans ce cas particulier des permutations à sens-unique à trappe :

- d'abord, le problème de séparation devient le problème RSA classique à sens-unique grâce au déterminisme de la permutation ;
- le scénario de RCCA devient le scénario CCA classique ;
- finalement, grâce au déterminisme de la permutation, on peut éviter le facteur cubique dans la réduction, et obtenir ainsi le facteur quadratique usuel $q_g q_h$, comme dans toutes les constructions OAEP+, SAEP et SAEP+.

On peut alors obtenir un résultat de sécurité bien meilleur :

Théorème 45 *Considérons un attaquant $\mathcal{A}$ contre la construction d'OAEP 3 tours utilisant une famille de permutations à sens-unique à trappe (φ_{pk}) et selon une attaque* $\mathsf{IND{-}CCA}$. *Supposons qu'après q_D requêtes à l'oracle de déchiffrement et q_f, q_g, q_h requêtes respectivement aux oracles $\mathcal{F}$, $\mathcal{G}$ et $\mathcal{H}$, $\mathcal{A}$ peut avoir un avantage* $\mathsf{Adv}^{\mathsf{ind-cca}}_{\mathsf{oaep\text{-}3}}(\tau)$ *en temps τ, alors on peut inverser φ avec succès* $\mathsf{Succ}^{\mathsf{ow}}_{\varphi}(\tau')$, *en temps τ', borné par :*

$$\frac{1}{2} \times \mathsf{Adv}^{\mathsf{ind-cca}}_{\mathsf{oaep\text{-}3}}(\tau') - q_d^2 \times \left(\frac{1}{2^\ell} + \frac{6}{2^k}\right) - (4q_d + 1) \times \left(\frac{q_g}{2^\ell} + \frac{q_f}{2^k}\right) - q_d \times \frac{q_f + 1}{2^k},$$

avec $\tau' \leq \tau + (q_f + q_g + q_h + q_d)T_{lu} + q_g q_h T_\varphi$, où T_φ est le temps pour une évaluation d'une fonction φ_{pk}, et T_{lu} est le temps pour la consultation d'un élément dans une liste de taille inférieure au nombre total des requêtes.

5.4.2 Idée de la preuve

L'objectif de la preuve est de simuler les oracles de telle manière que l'attaquant ne puisse distinguer les simulations des oracles réels.

La simulation des oracles aléatoires utilise des listes pour stocker les questions-réponses. L'oracle de déchiffrement est simulé comme suit : quand une requête y est faite, si les valeurs s et t correspondantes ont toutes été soumises à $\mathcal{G}$ et à $\mathcal{H}$, on pourra extraire m, sinon on retournera un texte clair aléatoire. Cependant, cette simulation entraîne plusieurs relations implicites entre les oracles aléatoires $\mathcal{F}$, $\mathcal{G}$ et $\mathcal{H}$. Nous prouvons que, si la fonction φ_{pk} est difficile à inverser, les éventuelles contradictions engendrées par ces relations ne peuvent être détectées par l'attaquant.

Il est à noter que nous étudions ici une classe plus générale que les seules permutations : les fonctions probabilistes injectives. Ceci peut provoquer quelques problèmes. En effet, pour une permutation, une pré-image correspond à une image unique. Au contraire, dans le cas d'une fonction f, une pré-image x peut correspondre à plusieurs images $y = f(x, \rho)$, avec ρ une chaîne aléatoire. L'attaquant peut donc construire d'autres c' dont la pré-image est identique à celle du challenge c : $g(c) = g(c')$. Une telle requête c' peut être soumise à l'oracle de déchiffrement dans le scénario CCA, auquel cas on ne peut la détecter et ne peut ainsi donner une réponse cohérente. Pour résoudre ce problème, on a introduit la version relâchée de la sécurité à chiffrés choisis, puis on utilise l'oracle décisionnel Same_f. Cet oracle permet de détecter les textes chiffrés dont la pré-image est identique à celle du challenge et peut donc répondre test, *i.e.* refuser de répondre. Il peut également détecter si une requête à l'oracle de déchiffrement y a la même pré-image qu'une requête précédente de déchiffrement y' : si oui, on retournera le même texte clair ; sinon, y est un nouveau texte chiffré. Dans le cas d'un nouveau texte chiffré, grâce à Same_f, on vérifie si s et t ont été soumises respectivement à $\mathcal{G}$ et à $\mathcal{H}$: si oui, le texte clair m est facilement extrait ; sinon, on retourne un texte clair aléatoire.

5.4.3 Preuve

Nous présentons les principales étapes de la preuve. Elle utilise la technique des jeux successifs pour prouver que le simulateur de déchiffrement, décrit ci-dessus, est calculatoirement indistinguable de l'algorithme de déchiffrement réel pour tout attaquant. Puis, dans ce nouvel environnement, un attaquant $\mathsf{IND{-}RCCA}$ peut être facilement utilisé pour inverser la fonction à sens-unique.

JEU $\mathbf{J}_0$: On considère l'attaquant $A = (A_1, A_2)$ contre ce schéma selon une attaque $\mathsf{IND{-}RCCA}$. Après avoir vu la clef publique (la description de la fonction φ_{pk}), A_1 retourne deux textes clairs (m_0, m_1). Après avoir reçu le chiffré $C^\star = a^\star \,\|\, b^\star \,\|\, c^\star$ du message m_b, A_2 retourne un bit b'. On note $r^\star, t^\star, u^\star$ les uniques éléments tels que $c^\star = \mathcal{E}_{\mathsf{pk}}(m_b, r^\star, \rho^\star) = \varphi_{\mathsf{pk}}(u^\star \| t^\star, \rho^\star)$. Remarquons que l'attaquant a accès à l'oracle de déchiffrement $\mathcal{D}_{\mathsf{sk}}$ et aux oracles aléatoires $\mathcal{F}$, $\mathcal{G}$ et $\mathcal{H}$ à toute étape de l'attaque.

Oracles $\mathcal{F}$, $\mathcal{G}$ et $\mathcal{H}$	Requête $\mathcal{F}(r)$: si r est dans la liste, *i.e.* un couple (r, f) se trouve dans la liste $\mathcal{F}$-List, la réponse sera f. Sinon, la réponse f est aléatoirement choisie dans $\{0,1\}^\ell$ et le couple (r, f) sera ajouté à la $\mathcal{F}$-List. Requête $\mathcal{G}(s)$: si s est dans la liste, *i.e.* un couple (s, g) se trouve dans la liste $\mathcal{G}$-List, la réponse sera g. Sinon, la réponse g est aléatoirement choisie dans $\{0,1\}^k$ et le couple (s, g) sera ajouté à la $\mathcal{G}$-List. ►**Règle** $\mathsf{EvalGAdd}^{(0)}$ Ne rien faire `% Sera défini après` Requête $\mathcal{H}(t)$: si t est dans la liste, *i.e.* un couple (t, h) se trouve dans la liste $\mathcal{H}$-List, la réponse sera h. Sinon, la réponse h est aléatoirement choisie dans $\{0,1\}^\ell$ et le couple (t, h) sera ajouté à la $\mathcal{H}$-List.

FIG. 5.3 – Simulation formelle dans le jeu IND−RCCA : les oracles aléatoires

On note S_0 l'événement $b' = b$ ainsi que S_n dans les jeux $\mathbf{J}_n$ ci-dessous.

$$\Pr[\mathsf{S}_0] = \frac{1}{2} \times \left(\mathsf{Adv}^{\mathsf{ind-rcca}}_{\mathsf{oaep\text{-}3}}(\tau) + 1\right).$$

Avantage nul

L'objectif des deux premiers jeux est de modifier le jeu réel en un jeu dans lequel le bit b est parfaitement indistinguable.

JEU $\mathbf{J}_1$: Les simulations des oracles aléatoires $\mathcal{F}$, $\mathcal{G}$ et $\mathcal{H}$, ainsi que celle de l'oracle de déchiffrement $\mathcal{D}_{\mathsf{sk}}$, sont présentées dans les figures 5.3 et 5.4, et la simulation du challengeur est présentée dans la figure 5.5. Ces simulations sont effectuées grâce à l'utilisation des listes $\mathcal{F}$-List, $\mathcal{G}$-List, $\mathcal{H}$-List et $\mathcal{D}$-List. Leur mission est de répondre de façon cohérente aux requêtes soumises aux oracles aléatoires et à l'oracle de déchiffrement. Il s'agit de simulations parfaites.

JEU $\mathbf{J}_2$: Pour que l'avantage de tout attaquant soit nul, nous définissons le masque $f^\star$ de telle sorte qu'il soit totalement indépendant de la vue de l'attaquant :

►**Règle** $\mathsf{Chal}^{(2)}$

Les deux valeurs $r^+ \xleftarrow{R} \{0,1\}^k$ et $f^+ \xleftarrow{R} \{0,1\}^\ell$ sont choisies de façon aléatoire, puis on définit $r^\star = r^+, f^\star = f^+$ et $s^\star = m^\star \oplus f^+, g^\star = \mathcal{G}(s^\star), t^\star = r^+ \oplus g^\star, h^\star = \mathcal{H}(t^\star), u^\star = s^\star \oplus h^\star$.

Oracle $\mathcal{D}$

Requête $\mathcal{D}_{\mathsf{sk}}(c)$: Cas particulier : pour une requête dans la deuxième étape, une requête $(c, c^\star)$ est faite à l'oracle décisionnel $\mathsf{Same}_{\varphi_{\mathsf{pk}}}$. Si la décision est positive, la réponse sera test et la simulation est terminée. Sinon, on se trouve dans le cas général.
Cas général : pour chaque (m', c') dans $\mathcal{D}$-List, soumettre (c, c') à l'oracle décisionnel $\mathsf{Same}_{\varphi_{\mathsf{pk}}}$. Dans le cas d'une décision positive, la réponse sera m'.
Dans le cas contraire, la réponse est m et est définie par les règles suivantes :

►**Règle** Decrypt−Init$^{(0)}$

Calculer $u \| t = \psi_{\mathsf{sk}}(c)$;

– Si t est dans la liste, *i.e.* un couple (t, h) se trouve dans $\mathcal{H}$-List, calculer $s = u \oplus h$.

– si s est dans la liste, *i.e.* un couple (s, g) se trouve dans $\mathcal{G}$-List, calculer $r = t \oplus g$.

– si r est dans la liste, *i.e.* un couple (r, f) se trouve dans $\mathcal{F}$-List

►**Règle** Decrypt−TSR$^{(0)}$

$h = \mathcal{H}(t)$,
$s = u \oplus h, \quad g = \mathcal{G}(s)$,
$r = t \oplus g, \quad f = \mathcal{F}(r)$,
$m = s \oplus f$.

– sinon

►**Règle** Decrypt−TSnoR$^{(0)}$

même règle que Decrypt−TSR$^{(0)}$.

– sinon

►**Règle** Decrypt−TnoS$^{(0)}$

même règle que Decrypt−TSR$^{(0)}$.

– sinon

►**Règle** Decrypt−noT$^{(0)}$

même règle que Decrypt−TSR$^{(0)}$.

Répondre m et ajouter (m, c) à la $\mathcal{D}$-List.

FIG. 5.4 – Simulation formelle dans le jeu IND−RCCA : l'oracle de déchiffrement

Challengeur

Pour deux messages (m_0, m_1), lancer une pièce à pile ou face pour b et poser $m^\star = m_b$, choisir aléatoirement $r^\star$ puis répondre $c^\star$ où :

▶**Règle** $\mathsf{Chal}^{(0)}$

$f^\star = \mathcal{F}(r^\star), \quad s^\star = m^\star \oplus f^\star,$
$g^\star = \mathcal{G}(s^\star), \quad t^\star = r^\star \oplus g^\star,$
$h^\star = \mathcal{H}(t^\star), \quad u^\star = s^\star \oplus h^\star.$

▶**Règle** $\mathsf{ChalC}^{(0)}$

et $c^\star = \varphi_{\mathsf{pk}}(u^\star \| t^\star, \rho^\star)$, pour une chaîne aléatoire $\rho^\star$.

FIG. 5.5 – Simulation formelle dans le jeu IND−RCCA : le générateur du challenge

Les deux jeux $\mathbf{J}_2$ et $\mathbf{J}_1$ sont parfaitement indistinguables sauf si $r^\star$ a été soumise à $\mathcal{F}$ (par l'attaquant ou par l'oracle de déchiffrement). On note AskF_2 cet événement ainsi que AskF_n dans les jeux $\mathbf{J}_n$ ci-dessous.

$$|\Pr[\mathsf{S}_2] - \Pr[\mathsf{S}_1]| \leq \Pr[\mathsf{AskF}_2].$$

Comme souhaité, f^+ est utilisée dans la génération du challenge pour masquer le texte clair, mais n'apparaît pas ailleurs puisque $\mathcal{F}(r^+)$ n'est plus défini comme étant f^+. Ainsi, la réponse de $\mathcal{A}_2$ suit une distribution indépendante de b : $\Pr[\mathsf{S}_2] = 1/2$. On obtient la première conclusion :

$$\mathsf{Adv}^{\mathsf{ind-rcca}}_{\mathsf{oaep\text{-}3}}(t) \leq 2 \times \Pr[\mathsf{AskF}_2]. \tag{5.1}$$

L'objectif est désormais d'étudier l'événement AskF.

Exclusion des requêtes aux oracles $\mathcal{G}$ et $\mathcal{H}$ dans la simulation du déchiffrement

JEU $\mathbf{J}_3$: On commence la phase de simulation de l'oracle de déchiffrement. Tout d'abord, les règles Decrypt−noT, Decrypt−TnoS et Decrypt−TSnoR sont modifiées de telle manière qu'elles retournent des messages aléatoires : les réponses des oracles $\mathcal{F}$, $\mathcal{G}$ et $\mathcal{H}$ sont remplacées par des valeurs aléatoires.

▶**Règle** $\mathsf{Decrypt-noT}^{(3)}$

Choisir $m \xleftarrow{R} \{0,1\}^\ell$, $h \xleftarrow{R} \{0,1\}^\ell$ et $g \xleftarrow{R} \{0,1\}^k$
Définir $s = u \oplus h$, $r = t \oplus g$ et calculer $f = m \oplus s$.
Ajouter (r, f) à la $\mathcal{F}$-List, (s, g) à la $\mathcal{G}$-List, (t, h) à la $\mathcal{H}$-List.

▶**Règle** $\mathsf{Decrypt-TnoS}^{(3)}$

Choisir $m \xleftarrow{R} \{0,1\}^\ell$ et $g \xleftarrow{R} \{0,1\}^k$
Définir $r = t \oplus g$ et calculer $f = m \oplus s$.
Ajouter (r, f) à la $\mathcal{F}$-List et (s, g) à la $\mathcal{G}$-List.

►**Règle** Decrypt−TSnoR$^{(3)}$

Choisir $m \xleftarrow{R} \{0,1\}^\ell$.
Calculer $f = m \oplus s$.
Ajouter (r, f) à la $\mathcal{F}$-List.

Ces règles sont en général similaires à celles utilisées dans les jeux précédents. La relation suivante peut être facilement démontrée (les détails sont dans l'article [98]) :

$$|\Pr[\mathsf{AskF}_3] - \Pr[\mathsf{AskF}_2]| \leq q_d \times \left(\frac{q_g + q_d}{2^\ell} + 2 \times \frac{q_f + q_d}{2^k}\right). \tag{5.2}$$

JEU $\mathbf{J}_4$: Dans les jeux précédents, les simulations des oracles étaient quasi-parfaites grâce à l'ajout de nouvelles relations aux listes correspondantes. Dans ce jeu, on effectue des modifications plus techniques : $\mathcal{G}$-List ne stocke plus les nouvelles relations (s, g). Par conséquent, g n'est plus explicitement définie, la valeur de r correspondante ne peut plus être calculée, et (r, f) ne sera plus dans la $\mathcal{F}$-List. Cependant, dès que $\mathcal{G}(s)$ est connue, on peut définir $\mathcal{F}(r)$ de façon cohérente et mettre à jour les listes :

►**Règle** Decrypt−TnoS$^{(4)}$

Choisir $m \xleftarrow{R} \{0,1\}^\ell$.

►**Règle** Decrypt−noT$^{(4)}$

Choisir $h \xleftarrow{R} \{0,1\}^\ell$ et $m \xleftarrow{R} \{0,1\}^\ell$.
Ajouter (t, h) à la $\mathcal{H}$-List.

►**Règle** EvalGAdd$^{(4)}$

Pour chaque $(t, h) \in \mathcal{H}$-List et chaque $(m, c) \in \mathcal{D}$-List, choisir une chaîne aléatoire $\rho \in \mathrm{R}$ et demander $(c, c' = \varphi_{\mathsf{pk}}(h \oplus s \| t, \rho))$ à l'oracle décisionnel $\mathsf{Same}_{\varphi_{\mathsf{pk}}}$. Si $\mathsf{Same}_{\varphi_{\mathsf{pk}}}$ retourne une réponse positive, on calcule $r = t \oplus g$ et $f = m \oplus s$ et on ajoute (r, f) à la $\mathcal{F}$-List.

Avec ces nouvelles règles, les réponses données dans $\mathbf{J}_3$ et $\mathbf{J}_4$ sont parfaitement indistinguables, à l'exception du cas où r est soumise à $\mathcal{F}$ avant la soumission de s à $\mathcal{G}$. On note cet événement AskRbS_4 ainsi que AskRbS_n dans les jeux $\mathbf{J}_n$ ci-dessous. Quand r est soumise après s, la simulation est parfaite : au moment où s est soumise, grâce à la simulation de $\mathcal{G}$ (et la règle supplémentaire EvalGAdd) on trouve (t, h) et ainsi (r, f) est calculé d'une manière cohérente, exactement comme dans $\mathbf{J}_3$, et ajouté à $\mathcal{F}$-List.

Remarquons que pour chaque texte chiffré c, t est unique, par conséquent, h et s le sont aussi. Alors, le cas d'échec apparaît au plus une fois pour chaque texte chiffré soumis à l'oracle de déchiffrement.

Cependant, tant que s n'est pas soumise, g est une variable aléatoire uniformément distribuée. Par conséquent, r l'est aussi. La probabilité d'échec (*i.e.* r a déjà été soumise à $\mathcal{F}$) est ainsi $q_f/2^k$:

$$\Pr[\mathsf{AskRbS}_4] \leq q_d \times \frac{q_f + q_d}{2^k}.$$

Un problème survient en raison de la suppression de quelques éléments (s, g) de $\mathcal{G}$-List, et (r, f) de $\mathcal{F}$-List. Cette suppression peut éventuellement avoir un impact sur la simulation des requêtes de déchiffrement et sur l'événement AskF :

- si le couple (r, f) où $r = r^\star$ est supprimé, l'événement AskF ne peut se produire que dans $\mathbf{J}_3$ mais pas dans $\mathbf{J}_4$. Heureusement, puisque $r = t \oplus g$, où g est aléatoirement choisie, la probabilité d'occurrence de cet événement est au plus $1/2^k$.
- quand on simule une requête de déchiffrement ultérieure $c' = \varphi_{\mathsf{pk}}(u' \| t', \rho')$, l'élément $s' = s$ peut être dans $\mathcal{G}$-List de $\mathbf{J}_3$ mais pas dans celle de $\mathbf{J}_4$. Par conséquent, la règle Decrypt−TnoS est utilisée au lieu de Decrypt−TSR ou Decrypt−TSnoR. g est donc définie lors du premier déchiffrement dans $\mathbf{J}_3$ mais ne serait jamais révélée. La probabilité que $r' = t' \oplus g' = t' \oplus g$ ne soit pas dans la liste $\mathcal{F}$-List est donc $(q_f + q_d)/2^k$ (modification de Decrypt−TSR à Decrypt−TnoS). Dans le cas où r' n'est pas dans $\mathcal{F}$-List, m' est aléatoire à la fois dans $\mathbf{J}_3$ et dans $\mathbf{J}_4$: un passage de Decrypt−TSnoR à Decrypt−TnoS.

$$|\Pr[\mathsf{AskF}_4] - \Pr[\mathsf{AskF}_3]| \leq \Pr[\mathsf{AskRbS}_4] + q_d \times \frac{q_f + q_d}{2^k} + q_d \times \frac{1}{2^k} \leq 2q_d \times \frac{q_f + q_d}{2^k} + q_d \times \frac{1}{2^k}.$$

Jeu $\mathbf{J}_5$: On continue à simplifier la simulation de l'oracle de déchiffrement : les nouvelles relations (t, h) ne sont plus stockées dans $\mathcal{H}$-List.

▶**Règle** Decrypt−noT$^{(5)}$

Choisir $m \xleftarrow{R} \{0,1\}^\ell$.

Par rapport à $\mathbf{J}_4$, les réponses des simulations de l'oracle de déchiffrement dans $\mathbf{J}_5$ sont identiques (valeurs aléatoires). Néanmoins, $\mathcal{H}$-List a changé et ce changement peut avoir quelques impacts :

- $\mathcal{F}$ peut répondre différemment. On note AskSbT_5, ainsi que AskSbT_n dans les jeux $\mathbf{J}_n$ ci-dessous, l'événement où s est soumise à $\mathcal{G}$ avant que t ne soit soumise à $\mathcal{H}$. Quand AskSbT_5 se produit, la règle EvalGAdd n'est plus applicable. Heureusement, tant que t n'est pas soumise à $\mathcal{H}$, h est une variable aléatoire uniformément distribuée, et donc $s = u \oplus h$ l'est également. Par conséquent, la probabilité d'occurrence de AskSbT_5 est égale à $q_g/2^\ell$ (puisqu'aucune nouvelle relation de $\mathcal{G}$ n'est ajoutée par la simulation de déchiffrement) :

$$\Pr[\mathsf{AskSbT}_5] \leq q_d \times \frac{q_g}{2^\ell}.$$

- la suppression de l'élément (t, h) de la $\mathcal{H}$-List peut éventuellement avoir un impact sur la simulation d'une requête de déchiffrement ultérieure $c' = \varphi_{\mathsf{pk}}(u' \| t', \rho')$:
  - si s' est dans $\mathcal{G}$-List et $t' = t$ a été trouvée dans le jeu $\mathbf{J}_4$. Comme t' n'est plus dans $\mathcal{H}$-List, la règle Decrypt−noT est utilisée au lieu de Decrypt−TSR ou Decrypt−TSnoR. Dans ce cas, $h' = h$ est définie lors du premier déchiffrement

dans $\mathbf{J}_4$ mais elle n'est plus révélée. La probabilité que $s' = t' \oplus h$ soit dans $\mathcal{G}$-List est au plus $q_g/2^\ell$.

– si s' n'est pas dans $\mathcal{G}$-List mais $t' = t$ a été trouvée dans le $\mathbf{J}_4$. Comme t' n'est plus dans $\mathcal{H}$-List, la règle Decrypt−noT est utilisée au lieu de Decrypt−TnoS. Dans ce cas, le déchiffrement reste le même car comme dans le jeu $\mathbf{J}_4$, il retourne un texte clair aléatoire et n'ajoute rien aux listes.

On obtient finalement :

$$|\Pr[\mathsf{AskF}_5] - \Pr[\mathsf{AskF}_4]| \leq \Pr[\mathsf{AskSbT}_5] + q_d \times \frac{q_g}{2^\ell} \leq 2q_d \times \frac{q_g}{2^\ell}.$$

Remarquons que les listes $\mathcal{G}$-List et $\mathcal{H}$-List ne contiennent maintenant que les requêtes posées par l'attaquant et par la génération du challenge. La simulation de déchiffrement ne fait plus appel aux oracles $\mathcal{G}$ ou $\mathcal{H}$, mais seulement à $\mathcal{F}$.

On note AskGA_5 (resp. AskHA_5) l'événement où $s^\star$ (resp. $t^\star$) (valeur obtenue lors de la génération du challenge), est soumise par l'attaquant. On désigne aussi AskGHA_5 l'événement où les deux événements AskGA_5 et AskHA_5 se produisent.

Extracteur du texte clair

On complète maintenant la simulation de l'oracle de déchiffrement en rendant son comportement identique à celui d'un extracteur classique de textes clairs, brièvement décrit dans l'« idée de la preuve ».

Jeu $\mathbf{J}_6$: Avant de faire des modifications, on introduit la règle Abort pour supprimer quelques exécutions. Si une des conditions d'application de cette règle est vérifiée, le jeu s'arrête et on retourne une réponse aléatoire b'.

►**Règle** $\mathsf{Abort}^{(6)}$

Si $\mathsf{AskGA}_6 \wedge \neg\mathsf{AskHA}_6$.

Si $\neg\mathsf{AskHA}_6$, $\mathcal{H}(t^\star) = u^\star \oplus m_b \oplus f^+$ ne sera jamais révélée. Puisque f^+ est une valeur aléatoire indépendante de la vue de l'attaquant, $\mathcal{H}(t^\star)$ est donc une variable aléatoire uniformément distribuée et $s^\star = u^\star \oplus H(t^\star)$ l'est également. Ceci implique que $s^\star$ ne peut être soumise qu'avec une probabilité $q_g/2^\ell$.

$$|\Pr[\mathsf{AskF}_6] - \Pr[\mathsf{AskF}_5]| \leq \frac{q_g}{2^\ell}.$$

De plus, $\Pr[\mathsf{AskF}_6]$ peut être facilement bornée par la relation suivante dont la preuve se trouve dans l'article [98] :

$$\Pr[\mathsf{AskF}_6] \leq \frac{q_f}{2^k} + \Pr[\mathsf{AskGHA}_6]. \tag{5.3}$$

Conclusion intermédiaire :

$$\Pr[\mathsf{AskF}_2] \leq q_d^2 \times \left(\frac{4}{2^k} + \frac{1}{2^\ell}\right) + (3q_d + 1) \times \left(\frac{q_f}{2^k} + \frac{q_g}{2^\ell}\right) + q_d \times \frac{q_f + 1}{2^k} + \Pr[\mathsf{AskGHA}_6]. \tag{5.4}$$

On s'intéresse désormais à l'événement AskGHA_6.

JEU $\mathbf{J}_7$: Quelques autres exécutions supplémentaires seront également supprimées si une des conditions suivantes est vérifiée :

▶**Règle** $\mathsf{Abort}^{(7)}$

- Si $\mathsf{AskGA}_7 \wedge \neg\mathsf{AskHA}_7$.
- Si la règle $\mathsf{Decrypt{-}TSR/Decrypt{-}TSnoR}$ est appliquée où $t = t^\star$, et $\mathcal{H}(t^\star)$ n'a pas encore été soumise par l'attaquant.
- Si la règle $\mathsf{Decrypt{-}TSR}$ est appliquée où $s = s^\star$, et que $\mathcal{G}(s^\star)$ n'a pas encore été soumise par l'attaquant.

La différence suivante entre ces deux jeux est expliquée dans l'article [98] :

$$|\Pr[\mathsf{AskGHA}_7] - \Pr[\mathsf{AskGHA}_6]| \leq q_d \times \left(\frac{q_f + q_d}{2^k} + \frac{q_g}{2^\ell}\right). \tag{5.5}$$

JEU $\mathbf{J}_8$: On complète la simulation de l'oracle de déchiffrement en la rendant indépendante des requêtes faites par la génération du challenge. Si l'attaquant ne demande pas de requête sur $s^\star$, l'oracle de déchiffrement n'utilise plus $(s^\star, g^\star)$.

▶**Règle** $\mathsf{Decrypt{-}TSnoR}^{(8)}$

Si $s = s^\star$ et que $s^\star$ n'a pas été directement soumise par l'attaquant : $m \xleftarrow{R} \{0,1\}^\ell$.
Sinon, on choisit $m \xleftarrow{R} \{0,1\}^\ell$ et calcule $f = m \oplus s$ et on ajoute (r, f) à la $\mathcal{F}$-List.

Si $s = s^\star$ mais $s^\star$ n'a pas été directement soumise par l'attaquant, en application de la règle $\mathsf{Decrypt{-}TSnoR}$ dans $\mathbf{J}_7$, $f = \mathcal{F}(r)$ est une variable aléatoire uniformément distribuée. On peut donc retourner une réponse aléatoire m. Cependant, dans ce cas, (r, f) ne sera plus stocké et ceci peut causer quelques problèmes si la valeur r' d'une requête de déchiffrement ultérieure c' est identique à r. Comme cela signifie que $g^\star \oplus t = g' \oplus t'$:

- si $s' = s$, alors $t' = t$, ceci implique que la réponse m' est identique à m. Cette éventualité peut cependant être détectée dès le début de la simulation grâce à l'utilisation de l'oracle décisionnel $\mathsf{Same}_{\varphi_{\mathsf{pk}}}$ sur (c, c').
- si $s' \neq s$, on obtient que $g = \mathcal{G}(s) = g^\star$ est indépendante de g'. De plus, $s^\star$ n'est pas soumise, donc $r = g^\star \oplus t$ est une variable aléatoire uniformément distribuée : la probabilité d'occurrence d'une telle requête de déchiffrement c' est $1/2^k$.

On a :

$$|\Pr[\mathsf{AskGHA}_8] - \Pr[\mathsf{AskGHA}_7]| \leq \frac{q_d^2}{2^k}.$$

JEU $\mathbf{J}_9$: Dans $\mathbf{J}_8$, la simulation de l'oracle de déchiffrement n'utilise plus les requêtes faites aux oracles $\mathcal{G}$ et $\mathcal{H}$ par le générateur du challenge :

- $\mathsf{Decrypt{-}TSR}$
  - $r = r^\star$: impossible si la requête $r^\star$ n'a pas encore été soumise directement par l'attaquant, car elle n'a pas été soumise au cours de la génération du challenge ;

  - $s = s^\star$: exclu dans le jeu $\mathbf{J}_7$;
  - $t = t^\star$: exclu dans le jeu $\mathbf{J}_7$;
- Decrypt−TSnoR
  - $s = s^\star$: similaire à Decrypt−TnoS du jeu $\mathbf{J}_8$;
  - $t = t^\star$: exclu dans le jeu $\mathbf{J}_7$;
- Decrypt−TnoS
  - $t = t^\star$: similaire à Decrypt−noT du jeu $\mathbf{J}_5$;

On peut donc modifier la simulation du challenge sans demander de requêtes à $\mathcal{G}$ ou à $\mathcal{H}$:

►**Règle** Chal$^{(9)}$

> Les deux valeurs $r^+ \xleftarrow{R} \{0,1\}^k$ et $f^+ \xleftarrow{R} \{0,1\}^\ell$ sont données, ainsi que $g^+ \xleftarrow{R} \{0,1\}^k$ et $h^+ \xleftarrow{R} \{0,1\}^\ell$ et puis on définit $r^\star = r^+$, $f^\star = f^+$, $s^\star = m^\star \oplus f^+$, $g^\star = g^+$, $t^\star = r^+ \oplus g^\star$, $h^\star = h^+$ et $u^\star = s^\star \oplus h^\star$.

Cette règle n'a aucun impact ni sur la simulation de déchiffrement ni sur la règle EvalGAdd car la modification a déjà été prise en compte dans $\mathbf{J}_5$. Ainsi, les distributions de probabilité ne sont pas modifiées.

Dans ce jeu, la simulation de l'oracle déchiffrement sur c est en effet l'extracteur simple de textes clairs [10, 54, 96]. En utilisant l'oracle décisionnel $\mathsf{Same}_{\varphi_{\mathsf{pk}}}$, elle obtient les valeurs (s, g) et (t, h) correspondant à $c = \varphi_{\mathsf{pk}}(s \oplus h \| t, \rho)$ grâce à une consultation de $\mathcal{G}$-List et $\mathcal{H}$-List (qui ne contiennent maintenant que des requêtes directement faites par l'attaquant). Il est à noter qu'elle n'a pas besoin de ψ_{sk} pour cela :

►**Règle** Decrypt−Init$^{(9)}$

> Pour chaque $(s, g) \in \mathcal{G}$-List et $(t, h) \in \mathcal{H}$-List : on choisit arbitrairement $\rho \in \mathrm{R}$, calcule $c' = \varphi_{\mathsf{pk}}(s \oplus h \| t, \rho)$ et on demande (c, c') à l'oracle décisionnel $\mathsf{Same}_{\varphi_{\mathsf{pk}}}$.
> - s'il existe (s, g) et (t, h) tels que $\mathsf{Same}_{\varphi_{\mathsf{pk}}}$ retourne une réponse affirmative, on définit $u = s \oplus h$.
> - sinon, on pose $t = \perp$ et $u = \perp$.

Le fait de définir $t = \perp$ et $u = \perp$ a pour but de rendre la réponse m aléatoire. Le temps de calcul est donc borné par $q_g q_h \times (T_\varphi + T_{\mathsf{Same}})$ plus le temps de consultation initiale de $\mathcal{D}$-List : $T_{lu} + q_d T_{\mathsf{Same}}$, où T_φ est le temps pour évaluer une fonction dans la famille φ, et T_{Same} est le temps d'exécution de l'oracle décisionnel. Alors, le temps total d'exécution (y compris toutes les consultation dans les listes) est borné par :

$$\tau' \leq \tau + q_d q_g q_h \times (T_\varphi + T_{\mathsf{Same}}) + q_d^2 \times T_{\mathsf{Same}} + (q_f + q_g + q_h + q_d) \times T_{lu}.$$

5.4.4 Cas particulier

Dans le cas où (φ_{pk}) est une famille de permutations, ce résultat peut être amélioré grâce à l'utilisation d'une liste supplémentaire de taille $q_g q_h$, qui stocke des multiplets $(s, g = \mathcal{G}(s), t, h = \mathcal{H}(t), c' = \varphi_{\mathsf{pk}}(s \oplus h \| t))$. Le temps de calcul est réduit à $\tau + q_g q_h \times T_\varphi + (q_f + q_g + q_h + q_d) \times T_{lu}$. De plus, le scénario RCCA devient le scénario classique CCA. On obtient donc un meilleur résultat en terme de sécurité (voir le théorème 45).

5.5 Conclusion

Toute variante d'OAEP [114, 19] appliquée à RSA avec des exposants généraux (*i.e.*, ni Rabin ni $e = 3$) admet, dans le cas optimal, une réduction quadratique en temps au problème RSA [101]. OAEP est même moins efficace à cause de sa réduction au problème d'inverser une permutation à sens-unique partiel. De plus, pour un niveau de sécurité de 2^{-k}, une chaîne aléatoire de taille $2k$ bits ainsi qu'une redondance de taille k bits sont exigées.

Dans ce chapitre, nous avons montré qu'OAEP 3 tours admet une réduction aussi efficace que celle de la meilleure des variantes d'OAEP. En particulier, notre construction n'exige pas de redondance : on gagne donc k bits. Cependant, ceci n'est pas l'avantage principal.

Selon tous les critères, OAEP 3 tours est au moins aussi efficace que les autres variantes d'OAEP. De plus, en pratique :

- grâce à l'absence de redondance, la mise en œuvre est facilitée, en particulier, pour le processus de déchiffrement [79] ;
- des familles de fonctions probabilistes à sens-unique peuvent être utilisées. Cela rend ce schéma plus flexible : il peut être utilisé avec différentes primitives comme les chiffrements d'El-Gamal et de Paillier.

En guise de conclusion, OAEP 3 tours est le padding le plus générique et le plus simple à pouvoir être utilisé avec plusieurs primitives de chiffrement asymétrique.