Exercices de sécurité par les jeux (EUF–CMA, réductions)

(inspiré Katz & Lindell)

Conventions

On travaille dans le cadre standard des jeux de sécurité : pour un schéma Π, un adversaire PPT A et un jeu Game_Π, on note

$$\mathsf{Adv}_{A,\Pi} := \Pr[\mathsf{Game}_\Pi(A) = 1].$$

Un schéma de *signature* $\Pi = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ est (t, ε)-*EUF–CMA sûr* si pour tout A probabiliste s'exécutant en temps $\leq t$ on a $\mathsf{Adv}_{A,\Pi} \leq \varepsilon$ dans le jeu INFORG04 (l'adversaire dispose d'un oracle de signature et doit forger une signature valide nouvelle).

Exercice 1 (Concaténation de deux signatures)**.** *Soient deux schémas de signatures* Π_1 *et* Π_2 *respectivement* (t_1, ε_1)- *et* (t_2, ε_2)-*EUF–CMA sûrs. On définit* Π_{concat} *par :*

$$\begin{aligned}
\mathsf{KeyGen}: &\ (sk_1, vk_1) \leftarrow \Pi_1.\mathsf{KeyGen},\ (sk_2, vk_2) \leftarrow \Pi_2.\mathsf{KeyGen},\ \textit{retourner}\ ((sk_1, sk_2), (vk_1,\\
\mathsf{Sign}((sk_1, sk_2), m): &\ (\sigma_1, \sigma_2) \leftarrow (\Pi_1.\mathsf{Sign}(sk_1, m),\ \Pi_2.\mathsf{Sign}(sk_2, m));\\
\mathsf{Verify}((vk_1, vk_2), m, (\sigma_1, \sigma_2)): &\ \textit{retourner}\ \Pi_1.\mathsf{Verify}(vk_1, m, \sigma_1) \wedge \Pi_2.\mathsf{Verify}(vk_2, m, \sigma_2).
\end{aligned}$$

Montrer qu'il existe une constante K *telle que* Π_{concat} *est*

$$(\max(\varepsilon_1, \varepsilon_2),\ \min(t_1, t_2) - K)\text{-}\textit{EUF–CMA sûr.}$$

Indications / preuve esquissée. *Soit* A^* *un adversaire contre* Π_{concat}*. Construire* B_1 *qui attaque* Π_1 *:* B_1 *reçoit* vk_1 *et un oracle* $\mathsf{Sign}_1(\cdot)$ *; il génère localement* (sk_2, vk_2) *; fournit* (vk_1, vk_2) *à* A^* *; lorsqu'on lui demande* $\mathsf{Sign}(m)$ *pour le système concaténé,* B_1 *renvoie* $(\mathsf{Sign}_1(m),\ \Pi_2.\mathsf{Sign}(sk_2, m))$*. Si* A^* *sort* $(m^*, (\sigma_1^*, \sigma_2^*))$ *valide avec* m^* *nouveau, alors* B_1 *rend* (m^*, σ_1^*) *et gagne contre* Π_1*. Simulation parfaite* $\Rightarrow \mathsf{Adv}_{B_1,\Pi_1} = \mathsf{Adv}_{A^*,\Pi_{concat}}$ *et* $\mathrm{Temps}(B_1) = \mathrm{Temps}(A^*) + K$*. Par sécurité de* Π_1*,* $\mathsf{Adv}_{A^*,\Pi_{concat}} \leq \varepsilon_1$ *pour tout* A^* *de temps* $\leq t_1 - K$*. De manière symétrique, on obtient* $\mathsf{Adv}_{A^*,\Pi_{concat}} \leq \varepsilon_2$ *si* $\mathrm{Temps}(A^*) \leq t_2 - K$*. En combinant, pour tout* A^* *de temps* $\leq \min(t_1, t_2) - K$ *:*

$$\mathsf{Adv}_{A^*,\Pi_{concat}} \leq \max(\varepsilon_1, \varepsilon_2).$$

Exercice 2 (Signer puis chiffrer)**.** *Soient un schéma de signature* Π_S *(EUF–CMA sûr) et un schéma de chiffrement à clé publique* Π_E *(IND–CPA sûr). Définir* Π_{SE} *:*

$$\begin{aligned}
\mathsf{KeyGen}: &\ (sk_S, vk_S) \leftarrow \Pi_S.\mathsf{KeyGen},\ (sk_E, vk_E) \leftarrow \Pi_E.\mathsf{KeyGen};\\
\mathit{SignEnc}((sk_S, sk_E), m): &\ \sigma \leftarrow \Pi_S.\mathsf{Sign}(sk_S, m);\ c \leftarrow \Pi_E.\mathsf{Enc}(vk_E, (m, \sigma));\\
\mathit{VerifyDec}((vk_S, vk_E), c): &\ (m, \sigma) \leftarrow \Pi_E.\mathsf{Dec}(sk_E, c);\ \textit{return}\ \Pi_S.\mathsf{Verify}(vk_S, m, \sigma).
\end{aligned}$$

Montrer que Π_{SE} *est EUF–CMA sûr si* Π_S *est EUF–CMA et* Π_E *IND–CPA.*

Indications / preuve esquissée. *Adversaire A^* contre EUF de Π_{SE}. Construire B contre EUF de Π_S : B reçoit vk_S et un oracle Sign_S. Il gère localement (sk_E, vk_E) et publie (vk_S, vk_E) à A^*. Pour simuler $\mathsf{SignEnc}(m)$, B obtient $\sigma \leftarrow \mathsf{Sign}_S(m)$ puis chiffre localement $c \leftarrow \mathsf{Enc}(vk_E, (m, \sigma))$ et rend c. Quand A^* rend une forge c^* menant à (m^*, σ^*) tel que $\mathsf{Verify}_S(vk_S, m^*, \sigma^*) = 1$ et m^* nouveau, B rend (m^*, σ^*). La vue d'A^* est identique (chiffrement public). Donc $\mathsf{Adv}_{B,\Pi_S} = \mathsf{Adv}_{A^*,\Pi_{SE}}$ et la sécurité EUF de Π_S conclut. L'hypothèse IND–CPA assure que le chiffrement n'aide pas à détecter d'enseignements supplémentaires (mais ici B chiffre lui-même, donc simulation parfaite).*

Exercice 3 (Hash-then-Sign)**.** *Soit Π un schéma de signature EUF–CMA sûr et H une fonction de hachage résistante aux collisions. On définit*

$$\Pi_H : \quad \mathsf{Sign}_H(sk, m) = \Pi.\mathsf{Sign}(sk, H(m)), \qquad \mathsf{Verify}_H(vk, m, \sigma) = \Pi.\mathsf{Verify}(vk, H(m), \sigma).$$

Montrer que Π_H est EUF–CMA sûr si Π est EUF–CMA et H collision-résistante.

Indications / preuve esquissée. *Soit A^* un adversaire qui gagne contre Π_H. Deux cas mutuellement exclusifs selon le message forgé m^* :*

- *(i) $H(m^*)$ a déjà été signé pour un autre message $m \neq m^*$ demandé à l'oracle. Alors $H(m) = H(m^*)$ avec $m \neq m^*$ est une collision : on construit un réducteur B_{coll} qui relaie les requêtes de A^*, collecte (m, m^*) et sort la collision. On obtient $\Pr[\textit{collision}] \geq \Pr[\textit{cas (i)}]$.*
- *(ii) $H(m^*)$ est* nouveau*. Alors $(H(m^*), \sigma^*)$ est une forge pour Π. Construire B_Π qui utilise A^* et remplace l'oracle $\mathsf{Sign}_H(m)$ par $\mathsf{Sign}(H(m))$ de Π. Simulation parfaite $\Rightarrow$ $\Pr[\textit{forge } \Pi] \geq \Pr[\textit{cas (ii)}]$.*

Par lemme de partition, $\Pr[A^* \textit{ gagne } \Pi_H] \leq \Pr[\textit{collision}] + \Pr[\textit{forge } \Pi] \leq \varepsilon_H + \varepsilon_\Pi$.

Exercice 4 (Répétition r fois (majoration par union bound))**.** *Soit Π un schéma (t, ε)-EUF–CMA sûr. On définit Π_{rep} qui signe r fois le même message avec des clés indépendantes :*

$$\mathsf{KeyGen} : \ (sk_i, vk_i) \leftarrow \Pi.\mathsf{KeyGen} \textit{ pour } i = 1..r; \qquad \mathsf{Sign} : \ \sigma_i \leftarrow \Pi.\mathsf{Sign}(sk_i, m); \mathsf{Verify} : \ \bigwedge_{i=1}^{r} \Pi.\mathsf{Verify}(vk_i, m, \sigma_i)$$

Montrer que Π_{rep} est $(t - K, r\varepsilon)$-EUF–CMA sûr.

Indications / preuve esquissée. *Si A^* forge sur Π_{rep} alors il existe un i pour lequel (m^*, σ_i^*) est une forge contre la i-ème instance de Π. Construire B_i qui fixe l'instance i comme challenge et génère localement les autres clés. Par union bound,*

$$\Pr[A^* \textit{ gagne } \Pi_{rep}] \leq \sum_{i=1}^{r} \Pr[B_i \textit{ gagne } \Pi] \leq r\varepsilon,$$

et le surcoût de simulation est une constante K (générations et signatures locales).

Remarque pratique. Dans toutes les preuves ci-dessus, la constante K regroupe les coûts fixes de simulation (une ou plusieurs KeyGen locales, quelques appels à Sign locaux, tenue d'ensembles Q). Les jeux utilisés sont des simulations parfaites (pas de perte sauf K en temps), d'où des réductions *serrées*.