École normale supérieure **2019-2020**
Département d'informatique

Introduction à la cryptologie
TD n° 12 : Correction.

Exercice 1 (Ring signature)**.**

1. (a) Le signataire connaît une clef privée sk_s pour un certain s, correspondant à la clef publique pk_s. Pour signer un message m avec les clef publiques $pk_1, \ldots, pk_n$, le signataire tire des valeurs x_i pour $i \in [\![1, n]\!] \setminus \{s\}$ uniformément aléatoirement. Il calcule $h = H(m)$ et $y_i = E_{pk_i}(x_i)$ pour $i \neq s$. Il résout ensuite l'équation $\mathsf{Eq}(h, y_1, \ldots, y_n)$ en y_s. La résolution est immédiate :
$$y_s = h \oplus \bigoplus_{i \neq s} y_i.$$
En utilisant sa connaissance de sk_s, le signataire calcule $x_s = D_{sk_s}(y_s)$. La signature est $(pk_1, \ldots, pk_n, x_1, \ldots, x_n)$.
Remarque. On triche ici un peu sur le fait que l'algorithme de signature n'est pas forcément surjectif sur l'ensemble sur l'ensemble des valeurs de y_s possibles. Autrement dit, la valeur y_s que le signataire calcule pourrait ne pas être un chiffré bien formé, auquel cas la fonction de déchiffrement ne fonctionnerait pas. En réalité il faut modifier un peu RSA de base pour avoir la propriété voulue, mais nous n'allons pas nous préoccuper de cela ici.

   (b) Soit m un message quelconque et $h = H(m)$. Soit b le nombre de bits des chiffrés, et soit $n > b$. On choisit n clefs publiques $pk_1, \ldots, pk_n$ arbitrairement, et on tire uniformément n paires de messages clairs $(x_0^1, x_1^1), \ldots, (x_0^n, x_1^n)$. Pour $1 \leq i \leq n$ et $v \in \{0, 1\}$, on calcule $y_v^i = E_{pk_i}(x_v^i)$. Notre but est de trouver $v_1, \ldots, v_n \in \{0, 1\}$ tels que $\mathsf{Eq}(h, y_{v_1}^1, \ldots, y_{v_n}^n)$. Le point clef est que cela revient à un système linéaire, à savoir le système suivant :
$$y_0^1 \oplus v_1(y_1^1 \oplus y_0^1) \oplus \cdots \oplus y_n^1 \oplus v_n(y_1^n \oplus y_0^n) = h.$$
Les deux quantités ci-dessus sont égales ssi tous leurs b bits sont égaux. On obtient b équations linéaires sur les $v_1, \ldots, v_n$. Comme $n > b$, avec forte probabilité il existe une solution (sinon on peut recommencer en tirant d'autres x_v^i). On obtient ainsi une signature valide $(pk_1, \ldots, pk_n, x_{v_1}^1, \ldots, x_{v_n}^n)$, pour un message quelconque et pour un ensemble quelconque de signataires.

2. On choisit $n > 9$ clefs publiques quelconques. On choisit un message m arbitraire ; soit $h = H(m)$. On tire arbitrairement $x_{10}, \ldots, x_n$. Soit $h' = h - E_{pk_{10}}(x_{10}) - \cdots - E_{pk_n}(x_n) \bmod \mathbb{Z}_{2^b}$. Avec une bonne probabilité sur le choix de ces x_i, $h' < 2^{b-1}$ (on identifie les éléments de Z_{2^b} avec $\{0, \ldots, 2^b - 1\}$). Si ce n'est pas le cas, on recommence le tirage jusqu'à avoir cette propriété. Par le théorème non-trivial de l'indication, il existe $x_1, \ldots, x_9 \in \mathbb{N}$ tels que $\sum x_i^3 = h'$. Comme chacun des x_i^3 pour $i < 10$ est nécessairement inférieur à $h' < 2^{b-1}$, les x_i^3 sont inférieurs aux modules RSA, donc pour $i < 10$, $E_{pk_i}(x_i) = x_i^3$ est vrai non seulement modulo le module RSA correspondant, mais directement dans les entiers. On déduit que $\mathsf{Eq}(h, y_1, \ldots, y_n)$ est vérifiée pour $y_i = E_{pk_i}(x_i) = x_i^3$. La signature $(pk_1, \ldots, pk_n, x_1, \ldots, x_n)$ est donc valide.

3. (a) Le processus est exactement le même que dans la première question : le signataire en possession de sk_s tire des valeurs x_i pour $i \in [\![1, n]\!] \setminus \{s\}$ uniformément aléatoirement. Il calcule $h = H(m)$ et $y_i = E_{pk_i}(x_i)$ pour $i \neq s$. La différence par rapport à la première question est de savoir comment résoudre $\mathsf{Eq}(h, y_1, \ldots, y_n)$ en y_s. Mais cette équation se résout en fait simplement (elle est conçue pour) du fait que S_h est inversible :
$$\begin{aligned} & S_h(y_1 \oplus S_h(y_2 \oplus \ldots S_h(y_n))) = 0 \\ \Leftrightarrow \quad & y_s \oplus S_h(y_{s+1} \oplus \ldots S_h(y_n)) = S_h^{-1}(\ldots S_h^{-1}(S_h^{-1}(S_h^{-1}(0) \oplus y_1) \oplus y_2) \cdots \oplus y_{s-1}). \end{aligned}$$

On obtient

$$y_s = S_h(y_{s+1} \oplus \dots S_h(y_n)) \oplus S_h^{-1}(\dots S_h^{-1}(S_h^{-1}(S_h^{-1}(0) \oplus y_1) \oplus y_2) \cdots \oplus y_{s-1}).$$

Comme dans la première question, il reste à calculer $x_s = D_{sk_s}(y_s)$. La signature est $(pk_1, \dots, pk_n, x_1, \dots, x_n)$.

(b) La résolution de l'équation donnée dans la question précédente montre que la solution est unique.

(c) Fixons $h = H(m)$ et $s \in [\![1, n]\!]$. Soit $\mathcal{E}$ l'ensemble des $y_1, \dots, y_n$ qui sont solutions de $\mathsf{Eq}(h, y_1, \dots, y_n)$. Soit $\mathcal{S}$ l'ensemble des choix possibles de $(y_1, \dots, y_{s-1}, y_{s+1}, \dots, y_n)$ (*i.e.* $\mathcal{S} = (\{0,1\}^b)^{n-1}$ si les y_i vivent dans $\{0,1\}^b$). Par la question précédente, pour chaque choix de $(y_1, \dots, y_{s-1}, y_{s+1}, \dots, y_n)$ il existe un unique y_s tel que $(y_1, \dots, y_n) \in \mathcal{E}$. Soit $f_s : \mathcal{S} \to \mathcal{E}$ la bijection correspondante. La question de l'énoncé revient à montrer que l'image de la distribution uniforme sur $\mathcal{S}$ par f_s est indépendante de s. Mais puisque f_s est une bijection, cette distribution est en fait égale à la distribution uniforme sur $\mathcal{E}$, qui est clairement indépendante de s.

(d) Pour signer un message m avec $h = H(m)$, le signataire qui possède la clef secrète correspondant à la clef publique pk_s tire $(x_1, \dots, x_{s-1}, x_{s+1}, \dots, x_n)$ uniformément aléatoirement, et pose $y_i = E_{pk_i}(x_i)$. Le signataire calcule ensuite y_s tel que $\mathsf{Eq}(h, y_1, \dots, y_n)$ est satisfaite. Puis il calcule $x_s = D_{sk_s}(y_s)$ en utilisant sa clef secrète sk_s. La signature est $(pk_1, \dots, pk_n, x_1, \dots, x_n)$.

Pour montrer que la signature ne révèle rien sur qui a signé parmi le cercle $pk_1, \dots, pk_n$, il suffit de montrer que la distribution de la signature ne dépend pas de s. C'est bien le cas : en effet, comme on utilise RSA de base, E et D sont bijectifs, donc la distribution de $(y_1, \dots, y_{s-1}, y_{s+1}, \dots, y_n)$ est uniforme. Par la question précédente, la distribution de $(y_1, \dots, y_n)$ qui en résulte ne dépend pas de s (elle est même uniforme parmi les solutions de Eq). Puisque E et D sont des bijections, la distribution de $(x_1, \dots, x_n)$ ne dépend donc pas non plus de s (elle est même uniforme parmi les signatures valides, pour h et $pk_1, \dots, pk_n$ fixés).

Remarque. Comme plus haut, on triche un peu sur un point : les images de E_{pk_i} ne vivent pas toutes dans le même espace $\{0,1\}^b$ puisque les modules RSA sont différents ; en réalité, il faut modifier légèrement RSA pour que ce soit le cas. L'énoncé nous dit de faire semblant que les y_i sont dans le même espace justement pour ne pas avoir à nous préoccuper de ça.

(e) On voit que la taille de la signature croît linéairement avec le nombre de signataires. Si les pk_i sont inclus dans la signature, c'est inévitable. S'ils sont connus par ailleurs, il existe des solutions sous-linéaires.

Exercice 2 (Mineurs égoïstes)**.**

1. Soit (p_i) la distribution de probabilité au point fixe. On a :

$$p_0 = (1-\alpha)p_0 + (1-\alpha)p_1 \tag{1}$$
$$p_n = \alpha p_{n-1} + (1-\alpha)p_{n+1} \quad \forall n > 0. \tag{2}$$

En résolvant la première équation, on obtient $p_1 = p_0 \cdot \alpha/(1-\alpha)$. Ici, on peut conclure rapidement en devinant que la solution est une suite géométrique de raison $\alpha/(1-\alpha)$, comme le suggère la première équation : on aurait donc

$$p_i = \beta \left(\frac{\alpha}{1-\alpha}\right)^i$$

pour un certain β. Pour s'en assurer, il suffit de vérifier par une récurrence triviale que cette solution vérifie les deux équations précédentes, donc que c'est bien une solution. Enfin, il reste

à s'assurer que la somme des p_i soit égale à 1, puisqu'on cherche une distribution de probilité. On obtient

$$\beta \sum \left(\frac{\alpha}{1-\alpha}\right)^i = 1$$
$$\beta = 1 - \frac{\alpha}{1-\alpha}$$
$$= \frac{1-2\alpha}{1-\alpha}.$$

Finalement, la réponse est

$$p_i = \frac{1-2\alpha}{1-\alpha}\left(\frac{\alpha}{1-\alpha}\right)^i.$$

C'est la manière rapide de répondre à la question.

La manière longue mais plus complète, c'est de remarquer que l'équation (2) définit une suite récurrente linéaire d'ordre 2. On peut donc déterminer l'ensemble des solutions de cette équation de manière générique (voir cours de prépa) ; il s'agit d'un sous-espace vectoriel de dimension 2, qui consiste en les combinaisons linéaires de deux suites géométriques dont les raisons sont les racines de l'équation du second degré correspondant à l'équation (2), c'est-à-dire : $x = \alpha + (1-\alpha)x^2$. En résolvant cette équation on trouve une racine $\alpha/(1-\alpha)$ qui correspond à la solution qu'on a devinée précédemment. L'autre racine est simplement égale à 1. Effectivement, si tous les p_i sont égaux, on est dans un état stable, mais leur somme est nulle ou infinie, donc cette solution ne peut pas correspondre à une distribution de probabilités.

2. Si on est dans un état $n > 0$, le prochain bloc incorporé dans la blockchain publique appartient nécessairement aux mineurs égoïstes. Le seul cas où le prochain bloc n'appartient *pas* à ce groupe, c'est si on est dans l'état 0, et le prochain bloc est trouvé hors du groupe (probabilité $1-\alpha$). La probabilité que cela arrive est

$$p_0(1-\alpha) = 1 - 2\alpha.$$

La probabilité que le prochain bloc soit dans le groupe est donc 2α.

3. La question du rendement est plus subtile qu'il n'y paraît. Une manière de définir le rendement du groupe E, c'est comme la proportion de blocs incorporés dans la blockchain publique qui proviennent du groupe. Cette définition a un sens, notamment si on part de 0 bitcoin au début : à terme, le groupe E détient la proportion correspondante de la quantité totale de bitcoins. Suivant cette définition, clairement le groupe E augmente son rendement en étant égoïste : avec un comportement honnête, le groupe E ne produirait qu'une proportion α des blocs, tandis qu'il produit maintenant une proportion 2α.

   Mais il y d'autres manières de définir le rendement. En particulier, à cause du comportement égoïste du groupe E, le nombre total de blocs produits par unité de temps a diminué, ce qu'il faudrait modéliser aussi. Si on définit le rendement comme le nombre de blocs incorporés dans la blockchain produits par le groupe E *par unité de temps*, alors le rendement n'a pas doublé. (Sauf à prendre en compte des mécanismes de réajustement de la difficulté dans Bitcoin.)

   Dans tous les cas, le rendement des mineurs hors du groupe E a diminué, ils perdent sur tous les tableaux. C'est logique : le groupe E forcent les mineurs hors du groupe à perdre du temps à chercher des blocs alors que le groupe E a déjà une réponse. Les mineurs hors du groupe E perdent ainsi une partie de leur temps en recherches inutiles, ce qui n'est pas le cas du groupe E. En principe, tout mineur aurait intérêt à se joindre au groupe E pour profiter de la chaîne privée (en théorie—en réalité bien sûr il y aurait bien d'autres considérations, par exemple si le groupe E faisait connaître son existence, il s'exposerait à des représailles). Cependant, si tout le monde rejoint le groupe égoïste, il n'y a plus de groupe égoïste.

Exercice 3 (Hachage pour preuve de travail)**.**

1. On choisit $m = d$ et on construit la fonction $H' : x \mapsto 0^d \parallel H(x)$. Cette fonction n'est clairement pas sûre pour les preuves de travail avec niveau de difficulté d (c'est le moins qu'on puisse dire). Par contre, étant donné une collision x, y telle que $H'(x) = H'(y)$, on déduit immédiatement une collision $H(x) = H(y)$ donc H' n'est pas moins résistante aux collisions que H.
2. Considérons une fonction H qui est sûre pour les preuves de travail avec niveau de difficulté d. On définit H' comme suit :

$$H' : \{0,1\}^* \to \{0,1\}^n$$
$$x \mapsto \begin{cases} 1^n & \text{si } x = 0^n \text{ ou } x = 1^n \\ H(x) & \text{sinon.} \end{cases}$$

   Clairement cette fonction ne résiste pas aux collisions, puisque $H(0^n) = H(1^n)$. Cependant elle n'est pas moins sûre pour les preuves de travail que H, puisque les seules valeurs modifiées arrivent sur 1^n.
3. Si H résiste aux préimages, on peut construire H' qui résiste aux préimages et qui n'est pas sûre pour les preuves de travail avec exactement la même construction que dans la première question. Réciproquement, supposons qu'il existe H qui est sûre pour les preuves de travail. On construit H' comme suit :

$$H' : \{0,1\}^* \to \{0,1\}^n$$
$$x \mapsto \begin{cases} x & \text{si } x \in \{0,1\}^n \\ H(x) & \text{sinon.} \end{cases}$$

   Clairement H' n'est pas du tout résistante aux préimages. Par contre, elle est tout aussi sûre pour les preuves de travail que H, puisque cette propriété ne porte que sur des entrées de la fonction de taille strictement supérieure à n.

Remarque. Il y a deux points généraux qui valent la peine d'être relevés. D'abord, dans toutes les constructions précédentes, on ne crée pas un fonction de toute pièce, on part toujours d'une fonction supposée sûre pour une certaine propriété, et on la modifie. Ce n'est pas un hasard : on ne peut pas prouver qu'il existe de fonction qui soit sûre pour les propriétés précédentes. Par exemple, si P = NP, on peut facilement voir qu'il n'y a pas de fonction de hachage sûre pour les preuves de travail (en généralisant un peu pour que ça devienne une notion asymptotique), si la fonction de hachage se calcule en temps polynomial. Deuxième point, les raisonnements ci-dessus sont informels, parce qu'il est en fait impossible de définir formellement qu'une fonction de hachage fixe résiste aux collisions. En effet, étant donné une fonction $H : \{0,1\}^* \to \{0,1\}^n$, il existe nécessairement x, y de longueur $n+1$ tels que $H(x) = H(y)$, et il existe un algorithme qui trouve une collision en temps $O(1)$: c'est l'algorithme d'une ligne qui renvoie la paire (x, y). On ne sait pas construire cet algorithme efficacement, mais ce n'est pas quelque chose qu'on sait exprimer formellement (sans se mordre la queue)...

Exercice 4 (Wifi Protected Setup, ou comment ne pas utiliser des commitments, exercice de détente)**.**

1. Deux essais de connexion suffisent. Le premier permet de récupérer m_1 à l'étape (b), en publiant des commitments sur des valeurs arbitraires à la première étape (nécessairement indinstinguables de commitments corrects, voir la propriété (b) de l'exercice **??**). Le second permet de récupérer le commitment d'Alice sur m_2 à l'étape (b), et l'aléa correspondant à l'étape (d), en publiant un commitment sur m_1 correct cette fois à la première étape. Il reste à tester toutes les valeurs de m_2 possibles (seulement 1000 à cause de la contrainte sur les chiffres du PIN) en regardant laquelle est compatible avec le commitment et l'aléa publiés par Alice, ce qui prend un temps négligeable.

2. Dans ce cas, 11000 essais de connexion sont nécessaires. En effet il faut (moins de) 10000 essais pour trouver m_1, en essayant chaque valeur possible jusqu'à ce que la box accepte la valeur à l'étape (c). Une fois m_1 connu, on recommence pour m_2. Il ne faut que 1000 essais à cause de la contrainte sur les chiffres du PIN. On note que 11000 essais est un nombre relativement élevé, mais en pratique c'est l'affaire de quelques heures si on les exécute d'un bloc. D'autre part comme le PIN de l'appareil ne change jamais, il est possible de procéder en plusieurs sessions, quand on veut, jusqu'à avoir fait tous les essais. C'est donc une faiblesse énorme. Ceci dit, le problème est intrinsèque à une identification par PIN où les deux parties peuvent être contrôlées par un adversaire.