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

Introduction à la cryptologie
TD n° 10 : Correction.

Exercice 1 (Hachage et synchronisation)**.**

1. On calcule le haché de chaque fichier séparément, puis on construit un arbre de Merkle dont les feuilles sont les hachés de chaque fichier. La racine de l'arbre est le haché global. Le serveur stocke le haché de chaque nœud de l'arbre. Comme le nombre de nœuds est linéaire en le nombre de fichiers, et que la taille d'un haché est faible (*e.g.* 32 octets), la surcoût en stockage est faible. Quand un fichier est modifié, $\log n$ nœuds ont besoin d'être recalculés.

Exercice 2 (Signatures de Lamport)**.**

1. Soit $\mathcal{A}$ un adversaire qui étant donné une clef publique du schéma, parvient à produire une signature en temps polynomial. On souhaite montrer que H n'est pas à sens unique. On choisit donc x uniformément aléatoirement dans $\{0,1\}^\lambda$. On tire y indépendamment de la même manière, et c uniformément dans $\{0,1\}$. On donne à l'adversaire la clef publique $pk_0 = H(x)$, $pk_1 = H(y)$ si $c = 0$, et $pk_0 = H(y)$, $pk_1 = H(x)$ sinon. L'adversaire renvoie x_b tel que $H(x_b) = pk_b$. Si $b = c$, il a donc trouvé une préimage de $H(x)$, et l'évènement $b = c$ a probabilité 50%, puisque c est uniforme et que $H(x)$, $H(y)$ sont par ailleurs indistinguables (ils proviennent de la même distribution). En utilisant $\mathcal{A}$ de cette manière, on obtient un algorithme qui trouve une préimage de $H(x)$ pour x uniforme en temps polynomial avec probabilité de succès 50% (moyenne à la fois sur x, et sur le choix de tous les aléas consommés par l'algorithme), ce qui est une probabilité non-négligeable, donc H n'est pas à sens unique.
   Remarque. On traite « non-négligeable » de manière informelle. Une définition formelle est « bornée inférieurement par l'inverse d'un polynôme (en le paramètre de sécurité, ici λ) ».
2. D'abord, on crée λ instances indépendantes du schéma précédent, numérotées de 1 à λ. Chacune peut nous servir à signer une fois un bit. Pour signer un message de longueur quelconque, comme pour une signature classique, on commence par hacher le message. On obtient un haché de longueur fixe $H(m) \in \{0,1\}^\lambda$. On signe chaque bit de ce haché (dans l'ordre) avec l'instance correspondante du schéma de signature à 1 bit. La longueur d'une signature est λ^2 bits. La clef publique est de taille $2\lambda^2$.
3. Pour signer ℓ messages, on peut utiliser ℓ instances du schéma de la question précédente. Le problème (enfin, un des problèmes) est que la taille de la clef publique est de $2\ell\lambda^2$ bits. Pour compacter un peu, on peut organiser les ℓ clefs publiques comme les feuilles d'un arbre de Merkle de hauteur $\log \ell$ (on va supposer que ℓ est une puissance de 2). La clef publique devient alors simplement la racine de cet arbre, de longueur λ bits. Pour signer un message, on choisit la feuille correspondant au premier schéma de signature non encore utilisé, on signe le message normalement avec cette instance, puis on ajoute dans la signature la clef publique de l'instance, et le haché de tous les frères le long du chemin qui mène de la racine de l'arbre de Merkle jusqu'à la feuille utilisée (utilisation habituelle de l'arbre de Merkle). Cela permet de recalculer la racine de l'arbre à partir de la clef publique de l'instance utilisée, et donc de vérifier qu'elle correspond bien à la i-ième instance de la clef publique. La taille d'une signature est λ^2 pour l'instance de signature à usage unique utilisée, plus $2\lambda^2$ pour la clef publique de l'instance (en fait on peut omettre la moitié de la clef publique correspondant à aux valeurs révélées dans la signature, donc cela coûte seulement λ^2 bits supplémentaires), plus $\lambda \log \ell$ bits pour l'arbre de Merkle.

4. Avec un arbre de Merkle binaire la contribution de l'arbre dans la signature est $\lambda \log \ell$. Pour un arbre k-aire, la hauteur de l'arbre est $\log_k \ell$ (on ne se préoccupe pas des arrondis), et pour chaque nœud il faut révéler $(k-1)\lambda$ bits, donc la taille totale est $(k-1)\lambda \log_k \ell = \lambda \log \ell (k-1)/\log k$. La question est donc de minimiser $(k-1)/\log k$. On vérifie aisément que c'est une fonction croissante, donc le choix optimal est $k = 2$. ($k = 1$ n'est malheureusement pas valide.)

Exercice 3 (Signatures de Winternitz)**.**

1. Il suffit de vérifier $H^{a-1-v}(H^v(x_0)) = pk_0$ et $H^v(H^{a-1-v}(x_1)) = pk_1$.
2. Étant donné une signature modifée $(H^v(x_0), H^v(x_1))$ pour un message $v < a-1$, un attaquant peut signer le message $v+1$ en donnant $(H(H^v(x_0)), H(H^v(x_1)))$.
3. Supposons qu'on sait attaquer le schéma de l'énoncé. Alors étant donné $y = H(x)$ pour $x \in \{0,1\}^\lambda$ uniforme, montrons qu'on sait retrouver x efficacement. Soit $x' \in \{0,1\}^\lambda$ uniforme. Donnons à l'attaquant la paire de clefs publiques $pk_0 = y$, $pk_1 = H^{a-1}(x')$, et la signature du message $v = a-1 : (y, x')$. Du fait que H est une permutation, on observe que pk_0, pk_1 set une paire de valeurs uniformes indépendantes, qui est identique à la distribution d'une vraie paire de clefs publiques, et la signature est également distribuée identiquement à une signature réelle. Il s'ensuit que comme l'attaquant fonctionne pour n'importe quel message, il doit pouvoir trouver efficacement et avec probabilité non-négligeable une signature valide d'un message $w < a-1$: soit (s, s') la signature. Alors $H^{a-2-w}(s) = x$ est la préimage de y, ce qui a contredit l'assertion que H est à sens unique. (On peut adapter le raisonnement pour le cas où l'attaquant choisit le message qu'il signe, ce qui donne un résultat plus fort.)
4. Il suffit d'utiliser une graine aléatoire de λ bits, et de générer toutes les valeurs de la clef privée x_0, x_1 en utilisant un générateur pseudo-aléatoire. Par exemple, si H' est une fonction de hachage avec des entrées de taille quelconque, si s est la graine aléatoire, on peut définir $x_0 = H'(s||0)$, $x_1 = H'(s||1)$, où $||$ dénote la concaténation. Le point crucial pour ce type de compaction est que les éléments de la clef privée sont tous uniformes et indépendants, donc la clef privée dans son ensemble peut être vue comme une chaîne de bits uniformes indépendants.
5. Il suffit d'utiliser k instances du schéma de signature, et de compacter les clefs privées comme dans la question précédente. Mais on fait mieux à la question suivante.
6. Suivant l'indication, soit $x_1, \ldots, x_k, x'$ un ensemble de valeurs uniformes indépendantes constituant la clef privée (qu'on peut compacter en λ bits comme précédemment). La clef publique est $H^{a-1}(x_0), \ldots, H^{a-1}(x_k), H^{ka-1}(x')$. Pour signer un message $v = (v_1, \ldots, v_k) \in \{0, a-1\}^k$, on publie la signature $(H^{v_1}(x_1), \ldots, H^{v_k}(x_k), H^{ka-1-\sum v_i}(x'))$.

Exercice 4 (Huile et vinaigre)**.**

1. Tout d'abord, on observe que l'ensemble des matrices dont le quadrant supérieur droit est nul forme une algèbre. Pour le voir, il suffit de considérer les opérations par bloc, où chaque quadrant est un bloc. (Alternativement, on peut observer que ces matrices sont le noyau du morphisme $M \mapsto PMP'$, où P et P' désignent la projection sur les $n/2$ premières et $n/2$ dernières coordonnées respectivement.) Ensuite on observe que si $\mathcal{A}$ est une algèbre, son conjugué $S^{-1}\mathcal{A}'S$ aussi. Au passage, on peut d'ailleurs noter que lorsque l'inverse existe, il est également interne à l'algèbre.
2. C'est équivalent à montrer que $M_i^{-1}M_i$ a un quadrant supérieur droit nul. À nouveau, c'est immédiat en écrivant les opérations par bloc : M_i a un quadrant inférieur droit nul, son inverse M^{-1} doit donc avoir un quadrant supérieur gauche nul (on peut le voir par exemple en utilisant la relation de l'inverse avec la comatrice), et donc leur produit a un quadrant supérieur droit nul.
3. Cela découle des deux questions précédantes, en observant ue Z est un espace propre pour toute matrice dont la quadrant supérieur droit est nul.

4. L'adversaire calcule les $F_{i,j}$ pour un (i, j) arbitraire. Il calcule le polynôme caractéristique P, puis le factorise (ce qui peut se faire en temps polynomial, par exemple avec l'algorithme de Berlekamp). Avec une probabilité suffisamment élevée, comme dit l'énoncé, il y a une unique factorisation non-triviale $P = P_1P_2$, et l'espace H de la question précédente doit être égal à $P_1(M)$ ou $P_2(M)$. Si cela ne fonctionne pas, l'adversaire recommence avec une nouvelle valeur de (i, j). De cette manière, l'adversaire trouve un ou plusieurs espaces candidats pour H. Pour confirmer qu'il a trouvé H, il vérifie que c'est un espace propre pour tous les $F_{i,j}$. Une fois qu'il a trouvé H, il complète la base de H en une base de l'espace entier (en mettant les vecteurs générant H en seconde moitié de la base). La décomposition des F_i dans cette base lui donne une représentation de F de même nature que la clef privée. Il peut en particulier signer des messages arbitraires, comme le signataire légitime. Noter que cet attaquant n'a même pas besoin d'observer de signature, la connaissance de la clef publique lui suffit à casser le schéma.