Université Paris 8 J. Lavauzelle

Master 1 Mathématiques et applications — année 2023–24

Cryptographie à clé publique – Feuille de TD 5

26/02/2024

Le corrigé de certains exercices sera disponible à l'adresse suivante :

`https://lvzl.fr/teaching/2023-24/cp.html`

$(\star)$ exercice fondamental $(\star\star)$ pour s'entraîner $(\star\star\star)$ pour aller plus loin 💻 sur machine

Exercice 1. $(\star)$ Signature RSA : falsification sélective à message choisi.

On s'intéresse au schéma de signature RSA sous sa forme "brute", c'est-à-dire sans l'utilisation de fonction de hachage. Dans le cours, nous en avons vu une falsification existentielle à clef seule. Le but de cet exercice est de monter une falsification sélective à message choisi.

Une falsification **sélective** signifie que l'attaquant fixe le message dont il veut falsifier la signature **avant** de monter son attaque (et donc, avant de demander au signataire d'autres signatures valides). C'est donc une attaque moins forte que la falsification universelle, mais plus forte que la falsification existentielle.

Dans l'exercice, on note $\mathtt{pk} = (n, e)$ et $\mathtt{sk} = d$ les clefs publique et privée du schéma de signature RSA brut.

Question 1.– Soient $m_1, m_2 \in \mathbb{Z}/n\mathbb{Z}$ deux messages, et s_1, s_2 leurs signatures correspondantes. Que vaut la signature s du message $m = m_1 m_2 \mod n$, en fonction de s_1 et s_2 ?

Question 2.– Soit $m \in \mathbb{Z}/n\mathbb{Z}$ quelconque. Déduire de la questions précédente une falsification de la signature de m, après avoir demandé à Alice la signature de deux messages m_1 et m_2 (différents de m) judicieusement choisis.

Exercice 2. $(\star)$ Signature ElGamal : réutilisation de l'aléa.

On considère le schéma de signature ElGamal dans lequel le message est haché avant d'être signé. On choisit comme groupe $\mathbb{F}_p^\times$, et on a donc à disposition une fonction de hachage $H : \{0,1\} \to \mathbb{F}_p^\times$. La génération de clés est identique au cadre du cours, mais l'algorithme de signature est légèrement modifier pour donnr l'Algorithme 1. Pour la vérification de la signature, on effectue également un haché du message avant de tester les égalité requise.

Algorithme 1 : Algorithme de signature d'ElGamal avec fonction de hachage

Entrée : un message $\boldsymbol{m} \in \{0,1\}^*$, la clé privée $a \in \{1, \ldots, p-2\}$
Sortie : une signature $s \in \mathbb{F}_p^\times \times \{0, \ldots, p-2\}$

1 Choisir $k \in \mathbb{Z}/(p-1)\mathbb{Z}$ inversible.
2 Calculer $b = g^k \mod p$.
3 Calculer $h = H(\boldsymbol{m})$.
4 Calculer $c = (h - ab)k^{-1} \mod (p-1)$.
5 Retourner la signature $s = (b, c)$.

On suppose dans cet exercice qu'**Alice réutilise le même aléa k pour plusieurs de ses signatures**, et l'on souhaite en déduire une faille de sécurité.

Question 1.– Soient $s = (b, c)$ et $s' = (b', c')$ les signatures de deux messages distincts m et m' effectuées avec le même aléa k. Comparer b et b', puis déterminer une égalité liant $h = H(\boldsymbol{m})$, $h' = H(\boldsymbol{m}')$, a, b, c et c'.

Question 2.– En déduire une attaque à message connu sur la clé privée d'Alice, qui réussit avec très bonne probabilité.

Exercice 3. $(\star\star)$ Signature de Lamport.

Dans cet exercice, on s'intéresse au schéma de signature de Lamport, qui ne présuppose que l'utilisation d'une fonction à sens unique. Pour cela, soient E et E' deux ensembles finis et $f : E \to E'$ une fonction à sens unique. Le schéma de signature est défini pour un certain paramètre entier $k \geq 1$.

Algorithme 2 : Génération des clefs

Entrée : une taille $k \geq 1$
Sortie : une paire de clés publique/privée

1 Tirer uniformément $2k$ éléments distincts de E, et les stocker dans une matrice de taille $(2 \times k)$:

$$\boldsymbol{A} = \begin{pmatrix} a_{0,1} & a_{0,1} & \dots & \dots & a_{0,k} \\ a_{1,1} & a_{1,2} & \dots & \dots & a_{1,k} \end{pmatrix} \in E^{2\times k}$$

2 Calculer la matrice $\boldsymbol{B}$ de taille $(2 \times k)$ sur E', constituée des $b_{i,j} = f(a_{i,j})$:

$$\boldsymbol{B} = \begin{pmatrix} f(a_{0,1}) & f(a_{0,2}) & \dots & \dots & f(a_{0,k}) \\ f(a_{1,1}) & f(a_{1,2}) & \dots & \dots & f(a_{1,k}) \end{pmatrix} \in (E')^{2\times k}$$

3 La clé publique est $\boldsymbol{B}$, la clé privée est $\boldsymbol{A}$.

Algorithme 3 : Signature

Entrée : la clé privée $\boldsymbol{A}$, le message à signer $\boldsymbol{m} \in \{0,1\}^k$
Sortie : la signature $\boldsymbol{s}$

1 Pour tout $i \in \{1, \dots, k\}$, définir $s_i := a_{m_i,i}$.
2 Retourner la signature $\boldsymbol{s} = (s_1, \dots, s_k) \in E^k$

Algorithme 4 : Vérification

Entrée : la clé publique $\boldsymbol{B}$, la signature $\boldsymbol{s}$, le message à signer $\boldsymbol{m} \in \{0,1\}^k$
Sortie : un booléen `true/false`

1 Vérifier que $f(s_i) = B_{m_i,i}$ pour tout $i \in \{1, \dots, k\}$.

Question 1.– Expliquer pourquoi, si la fonction f n'est pas à sens unique, alors la signature n'est pas sûre.

Question 2.– Expliquer pourquoi la même paire de clés ne peut pas être utilisée pour signer deux messages.

Question 3.– A priori, le schéma semble construit de sorte que la longueur des messages et le nombre de colonnes de la clé publique (et de la clé privée) sont égaux.

1. En quoi cela pose-t-il problème d'un point de vue pratique ?
2. Proposer une modification de la signature afin qu'elle puisse rester pratique pour n'importe quel message à signer.
3. En déduire (approximativement) la taille des clés de cette signature. Justifier.

Exercice 4. $(\star\star)$ Une proposition de schéma de signature.

Dans cet exercice, on considère un nombre premier p pour lequel le problème du logarithme discret dans $\mathbb{F}_p^\times$ est supposé difficile. On note g un générateur du groupe cyclique $\mathbb{F}_p^\times$. Enfin, on considère une fonction de hachage $H : \{0,1\}^* \to \mathbb{Z}/(p-1)\mathbb{Z}$.

Un schéma de signature est décrit par les trois algorithmes suivants.

Algorithme 5 : Génération de clefs

Entrée : les paramètres du système
Sortie : une paire de clefs publique/privée

1 Tirer x aléatoirement dans $\mathbb{Z}/(p-1)\mathbb{Z}$.
2 Tirer y aléatoirement dans $\mathbb{Z}/(p-1)\mathbb{Z}$.
3 Calculer $X = g^x \mod p$ et $Y = g^y \mod p$.
4 La clef publique est $\mathtt{pk} = (X,Y)$, la clef privée est $\mathtt{sk} = (x,y)$.

Algorithme 6 : Signature

Entrée : un message $\boldsymbol{m} \in \{0,1\}^*$, la clé privée $\mathtt{sk} = (x,y)$
Sortie : une signature $s \in \mathbb{Z}/(p-1)\mathbb{Z}$

1 Calculer $h = H(\boldsymbol{m}) \in \mathbb{Z}/(p-1)\mathbb{Z}$
2 Calculer et retourner l'élément $s = xh + y \in \mathbb{Z}/(p-1)\mathbb{Z}$.

Algorithme 7 : Vérification

Entrée : une signature $s \in \mathbb{Z}/(p-1)\mathbb{Z}$, un message $\boldsymbol{m} \in \{0,1\}^*$, la clé publique $\mathtt{pk} = (X,Y)$
Sortie : vrai ou faux

1 Calculer $h = H(\boldsymbol{m}) \in \mathbb{Z}/(p-1)\mathbb{Z}$.
2 Calculer $a = g^s \mod p$ et $b = X^h Y \mod p$.
3 Faire le test $a \equiv b \mod p$ et retourner le booléen associé.

Question 1.– Vérifier que le schéma de signature est valide.

Question 2.– Proposer une attaque sur la clé privée $\mathtt{sk} = (x,y)$. On précisera le moyen d'attaque utilisé.

Exercice 5. (⋆⋆⋆) 💻 Implantation de RSA-FDH.

Dans cet exercice, on considère le schéma de signature RSA-FDH (*full-domain-hash*), implanté avec la fonction de hachage SHA-3, de sortie 224 bits.

On note $\boldsymbol{m}$ le message à signer, qui est vu comme une chaîne de caractères. Afin d'obtenir des hachés à valeur dans $\mathbb{Z}/n\mathbb{Z}$ (où n est le module public), on procède ainsi :

1. On note t le nombre de bits de p et q, les facteurs de n.
2. On calcule d'abord une séquence de hachés $\boldsymbol{h}$ suivant le schéma de remplissage :

$$\boldsymbol{h} = \texttt{SHA3\_224}(\boldsymbol{m} \,\|\, 0) \,\|\, \texttt{SHA3\_224}(\boldsymbol{m} \,\|\, 1) \,\|\, \cdots\cdots \,\|\, \texttt{SHA3\_224}(\boldsymbol{m} \,\|\, r)$$

   où « $\boldsymbol{m} \,\|\, i$ » est la chaîne de caractères associée à la concaténation de $\boldsymbol{m}$ et de l'entier i, et où l'entier r est fixé de sorte tel que $\boldsymbol{h}$ soit de longueur $\ell > 2t$ bits. Autrement dit, on prend $r = \lfloor 2t/224 \rfloor$.
3. Enfin, on convertit $\boldsymbol{h} \in \{0,1\}^\ell$ en un entier (les bits de poids fort étant devant), et on le réduit modulo n.

Aide (en python).

— On pourra utiliser la bibliothèque `pycryptodome` pour accéder à la fonction de hachage `SHA3_224`. Cette fonction prend en entrée une séquence d'octets [1].

— On peut également convertir une chaine de caractères $\boldsymbol{m}$ (encodée en `utf-8`) en une chaîne d'octets par la fonction `bytes(m, 'utf-8')`. Ainsi, pour transformer une chaîne `m` en haché `h` sous forme hexadécimale, grâce à la fonction de hachage `SHA3_224`, on écrit les instructions python suivantes :

```
from Cryptodome.Hash import SHA3_224
h = SHA3_224.new(bytes(m, "utf-8")).hexdigest()
```

— Il est également possible d'utiliser la bibliothèque `hashlib`.

— Pour lire le contenu d'un fichier, il existe les fonction `open()` et `read()`. Exemple d'utilisation :

```
# pour ouvrir l'objet fichier
f = open("nom_du_fichier.txt", 'r')
# pour lire le contenu  et le stocker dans une chaine de caracteres
s = f.read()
# pour fermer le fichier (important)
f.close()
```

— Pour transformer un nombre `a` sous forme hexadécimale en entier, on peut utiliser `int(a, 16)`.

Question 1.– Implanter le schéma de remplissage pour la fonction de hachage `SHA3_224`. Vérifier que pour le message $\boldsymbol{m}$ = « test » (ne pas considérer les guillemets) et la valeur $t = 128$, on obtient le haché

$$\boldsymbol{h} = 3489e8bf24b660896180884567a6c1bb971fe104137e713de8a7bc98$$
$$d6e981ecb4de6889ecd6d33a5022bffee9af2aa272c2060fb86f098a$$

en écriture hexadécimale [2].

1. voir la documentation ici : `https://pycryptodome.readthedocs.io/en/latest/src/hash/sha3_224.html`
2. on pourra s'aider par exemple de `https://emn178.github.io/online-tools/sha3_224.html` pour vérifier la valeur des hachés.

Question 2.– Implanter les algorithmes de signature et de vérification du schéma de signature RSA-FDH, en accord avec le schéma de remplissage présenté ci-dessus. On appellera ces fonctions `sign` et `verif`.

Question 3.– Exemple-jouet[3]**.** On prend $t = 16$ pour simplifier.

1. Vérifier que pour $n = 3089616301$ et $d = 1118240705$, la signature de $\boldsymbol{m} =$ « exemple » est $s = 2870643504$.
2. Vérifier que pour $n = 2509359689$ et $e = 65537$, la valeur $s = 520203729$ est une signature correcte de $\boldsymbol{m} =$ « jouet ».

On fixe maintenant $t = 1024$ afin d'avoir au moins 80 bits de sécurité. On donne également un ensemble de fichiers accessibles à l'adresse suivante :

`https://lvzl.fr/teaching/2023-24/docs/cp/td/rsafdh.zip`

Décompresser l'archive `.zip`. Vous y trouverez :

1. un fichier `lebateauivre.txt`,
2. un fichier `pk.txt` dans lequel se trouve le module public n,
3. trois fichiers `s1.txt`, `s2.txt`, `s3.txt` dans lequels se trouvent trois signatures potentielles du texte dans `lebateauivre.txt`.

On appelle $\boldsymbol{m}$ la chaîne de caractères correspondant au fichier `lebateauivre.txt`. Vérifiez que la chaîne de caractères commence par :

```
Le Bateau Ivre.

Comme je descendais des Fleuves impassibles,
Je ne me sentis plus guidé par les haleurs:
Des Peaux-Rouges criards les avaient pris pour cibles,
Les ayant cloués nus aux poteaux de couleurs.
```

Faites notamment attention à l'encodage des accents, et à la présence de retours à la ligne.

Question 4.– Vérifier que le haché de $\boldsymbol{m}$, selon le schéma de remplissage décrit ci-dessus, a pour écriture hexadécimale :

```
d79227398fb4bf19d182410afcb19bc72f8c0a8390b7f4fe071e52caf15d73cfb347ad3d1456e60
667a75ef47ab32dc6f41bf8f5542e6ef55d16abc0eda0992feff93e51fb5756341fa28144e6d2ad
72b113d97b6dddb90b12f701bf835d633d78d6dd29ba78f355772316d53e6f77c2063a61f95187e
e5cb3a9ef760f2da35a145522f1d02de2f9eed9f67f4fcc7a72a3a615b7dd5c60dd4f50d3bbbb4a
61310f51d1d21b08d2a657cd9803f28dd3d6dc280f6ebc98c9833d58b4d6a16fe5e7d0ca55331b6
92c8163dc0c9c9b2284ca014c30590596948e22240e307e695574ef9fee6de8df46065719c2f403
bb3ba780e428d94369016d6029415e94474cbf50fbbeef652911fe01995f593d4560a7d8e580e85
957867b
```

Question 5.– Alice engendre une clef publique $\mathtt{pk} = (n, e)$ où $e = 65537$ et n est l'entier écrit dans le fichier `pk.txt`.

Parmi les entiers donnés dans les fichiers `s1.txt`, `s2.txt` et `s3.txt`, lequel correspond à une signature de $\boldsymbol{m}$ avec la clé publique d'Alice ?

3. c'est-à-dire, avec des valeurs « petites »