Université Paris 8 — Année 2022–23 — J. Lavauzelle

Master 1 Mathématiques et applications

Cryptographie à clé publique – Solutions feuille de TD 5

24/02/2023

Retrouvez le sujet du TD et d'autres exercices à l'adresse :

`www.math.univ-paris13.fr/~lavauzelle/teaching/2022-23/clef-publique.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 brut. 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 $\texttt{pk} = (n, e)$ et $\texttt{sk} = d$ les clefs publique et privée du schéma de signature RSA brut.

Question 1.– Soient $m_1, m_2 \in (\mathbb{Z}/n\mathbb{Z})^\times$ 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.– En déduire la falsification de la signature d'un message $m \in (\mathbb{Z}/n\mathbb{Z})^\times$ quelconque, après avoir demandé à Alice la signature de deux messages m_1 et m_2 (différents de m) judicieusement choisis.

Solutions de l'Exercice 1.

Solution Q1. On a

$$\begin{cases} s_1 &= (m_1)^d \mod n \\ s_2 &= (m_2)^d \mod n \\ s &= (m_1 m_2)^d \mod n \end{cases}$$

Par conséquent,

$$s = s_1 s_2 \mod n$$

Solution Q2. Supposons que l'on veuille falsifier une signature pour un message cible $m \in \mathbb{Z}/n\mathbb{Z}$. L'idée de l'attaque est la suivante.

1. On choisit aléatoirement un message $m_1 \in (\mathbb{Z}/n\mathbb{Z})^\times$.
2. On calcule $m_2 = m \cdot m_1^{-1} \mod n$.
3. On demande à Alice de signer m_1 et m_2 (notons s_1 et s_2 les signatures associées).
4. On calcule $s = s_1 s_2 \mod n$

On obtient alors une signature valide s du message m.

Remarque. L'attaque est donc une falsification sélective à message choisi.

Exercice 2. ($\star$) Signatures et fonctions de hachage.

Soit H une fonction de hachage à valeurs dans $\{0,1\}^t$. On rappelle qu'une *collision* sur H est un couple de messages distincts $\boldsymbol{m} \neq \boldsymbol{m}'$ tels que $H(\boldsymbol{m}) = H(\boldsymbol{m}')$.

Question 1.– On peut obtenir une collision sur la fonction de hachage H par un compromis temps-mémoire, et exploiter ainsi le *paradoxe des anniversaires*. Décrire la méthode qui permet d'obtenir cette collision, et donner une approximation de sa complexité en fonction de t. Pour cela, on supposera que le coût d'évaluation de H, et le test d'appartenance d'un haché h à une liste de hachés L se font en temps constant.

On considère maintenant le schéma de signature DSA dans le groupe multiplicatif $\mathbb{F}_p^\times$, et on note g un générateur d'un sous-groupe d'ordre q de $\mathbb{F}_p^\times$, où q divise $(p-1)$. Pour simplifier, on suppose également que la fonction de hachage H est utilisée **sans schéma de remplissage** additionnel. Les algorithmes de signature et de vérification de DSA sont rappelés ci-dessous. On note $\mathcal{S} = (\mathbb{Z}/q\mathbb{Z})^\times \times (\mathbb{Z}/q\mathbb{Z})^\times$ l'espace des signatures.

Algorithme 1 : Signature DSA

Entrée : un message $\boldsymbol{m}$, la clé privée a
Sortie : une signature $s \in \mathcal{S}$

1. Calculer l'entier h associé à $H(\boldsymbol{m}) \in \{0,1\}^t$.
2. Choisir $k \in (\mathbb{Z}/q\mathbb{Z})^\times$ aléatoirement.
3. Calculer $b = (g^k \mod p) \mod q$.
4. Calculer $c = (h + ab)k^{-1} \mod q$.
5. Si b ou c n'est pas inversible $\mod q$, revenir à l'étape 2.
6. Sinon, retourner $s = (b,c)$.

Algorithme 2 : Vérification DSA

Entrée : une signature $s \in \mathcal{S}$, un message $\boldsymbol{m}$, la clé publique $\alpha = g^a$
Sortie : vrai ou faux

1. Calculer l'entier h associé à $H(\boldsymbol{m}) \in \{0,1\}^t$.
2. Calculer $x = g^{hc^{-1} \bmod q}\, \alpha^{bc^{-1} \bmod q}$.
3. Faire le test $x \equiv b \mod q$ et retourner le booléen associé.

Question 2.– Expliquer comment une collision sur H peut mener à une attaque sur le schéma de signature. On précisera la nature et les moyens de l'attaque.

Question 3.– En déduire la valeur de t minimale pour espérer obtenir une sécurité EUF-CMA (infalsifiabilité existentielle à message choisi) de 128 bits.

Solutions de l'Exercice 2.

Solution Q1. Le paradoxe des anniversaires stipule que, si l'on tire M fois avec remise un élément dans un ensemble à N éléments, alors la probabilité d'avoir tiré des éléments deux-à-deux distincts décroît asymptotiquement en

$$e^{-M^2/2N}.$$

Autrement dit, en tirant $M \gg \sqrt{N}$ éléments aléatoirement, on peut espérer avoir deux éléments identiques avec bonne probabilité.

La preuve de ce résultat est la suivante. Soit X_i la valeur du i-ème élément tiré, et $S_i = \{X_1, \ldots, X_i\}$ la liste des i premiers éléments tirés. On note également E_i l'évènement « les éléments tirés sont deux à deux distincts ». Alors, si l'on suppose les tirages de X_i indépendants, on a :

$$\begin{aligned}
\mathbb{P}(E_M) &= \mathbb{P}(X_2 \notin S_1 \text{ et } X_3 \notin S_2 \text{ et } \ldots \text{ et } X_M \notin S_{M-1}) \\
&= \mathbb{P}(X_2 \notin S_1)\mathbb{P}(X_3 \notin S_2 \mid \#S_2 = 2) \cdots \mathbb{P}(X_M \notin S_{M-1} \mid \#S_{M-1} = M-1) \\
&= (1-1/N)(1-2/N)\cdots(1-(M-1)/N) \simeq e^{-M^2/2N}
\end{aligned}$$

Étant donné ce résultat de probabilités, pour construire une collision sur H, voici une idée d'algorithme :

1. Initialiser une table $T = [\,]$
2. **Faire** :
   (a) tirer $\boldsymbol{m}$ aléatoirement et calculer $H(\boldsymbol{m})$
   (b) s'il existe $H(\boldsymbol{m}_i) \in T$ tel que $H(\boldsymbol{m}) = H(\boldsymbol{m}_i)$, alors **retourner** la paire $(H(\boldsymbol{m}), H(\boldsymbol{m}_i))$
   (c) sinon, ajouter $H(\boldsymbol{m})$ à T

D'après l'analyse asymptotique précédente, la complexité de l'algorithme est en moyenne $O(2^{t/2})$, si l'on considère que le calcul de $H(\cdot)$ et le test $h \in T$ ont un coût $O(1)$.

Solution Q2. Si un attaquant connaît deux messages $\boldsymbol{m} \neq \boldsymbol{m}'$ tels que $H(\boldsymbol{m}) = H(\boldsymbol{m}')$, alors il peut simplement :

- demander une signature s de $\boldsymbol{m}$,
- retourner la falsification $(\boldsymbol{m}', s)$.

On observe que s est bien une signature valide de $\boldsymbol{m}'$.

C'est une attaque existentielle (l'attaquant ne choisit pas le message attaqué, car il ne maîtrise pas la collision de $\boldsymbol{m}$ et $\boldsymbol{m}'$) à message choisi (il demande la signature d'un message spécifique).

Solution Q3. Il faut rendre la recherche de collision impossible, ce qui nécessite :

$$2^{t/2} \geq 2^{128} \implies t \geq 256\,.$$

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 3 : 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 4 : 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 $\boldsymbol{s} = (s_1, \dots, s_k) \in E^k$

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

Algorithme 5 : 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 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.

Solutions de l'Exercice 3.

Solution Q1. Si f n'est pas à sens unique, cela signifie qu'il est possible de retrouver certaines préimages $a_{j,}$ à partir de $b_{j,i} = f(a_{j,i})$ uniquement. Un attaquant, qui a accès à la clé publique $\boldsymbol{B}$, pourra donc reconstituer une partie de la clé secrète $\boldsymbol{A}$. Sur les indices i correspondant à cette partie de la clé secrète, l'attaquant pourra donc forger les éléments s_i de la signature.

Solution Q2. Supposons que deux messages $\boldsymbol{m}, \boldsymbol{m}' \in \{0,1\}^k$ soit signés avec une même paire de clefs $(\boldsymbol{A}, \boldsymbol{B})$, pour obtenir deux signatures $\boldsymbol{s}, \boldsymbol{s}'$. Alors, les bits communs $m_i = m'_i$ correspondent à deux éléments de signatures égaux : $s_i = s'_i$. Autrement dit, avec la connaissance de $\boldsymbol{m}$, un attaquant peut forger la partie de la signature de $\boldsymbol{m}'$ qui correspond aux bits identiques de $\boldsymbol{m}, \boldsymbol{m}'$: ce n'est clairement pas un comportement voulu.

Solution Q3.

1. Cela signifie que les clés sont au moins aussi longues que le message (en nombre de bits) : ce n'est pas un comportement voulu pour une signature.
2. On peut hacher le message avant de procéder à la signature.
3. D'un point de vue calculatoire, si l'on souhaite une sécurité de 128 bits, il faut que la taille des ensembles de départ et d'arrivée d'une fonction à sens unique soit $\geq 2^{256}$ (pour éviter les attaques de type paradoxe des anniversaires). Chaque élément de E et E' doit donc être codé sur 256 bits. Les fonctions de hachages cryptographiquement sures donnent des hachés de plusieurs centaines de bits (voir les fonctions SHA par exemple). On obtient donc des clés de taille $\simeq 200 \times 2 \times 256 \times \simeq 1000$ bits. C'est raisonnable.

Exercice 4. ($\star\star$) Vérification simultanée de signatures RSA.

Dans cete exercice, on s'intéresse au schéma de signature RSA « brut ». Soit $(n = pq, e)$ une clé publique RSA, et d la clé privée associée. On suppose que n est de taille t bits.

Question 1.– En fonction de t, quel est le coût algorithmique (en nombre de multiplications et carrés dans $\mathbb{Z}/n\mathbb{Z}$) d'une signature RSA ?

Bob reçoit une série de $\ell \geq 2$ messages signés par Alice : $(m_1, s_1), \dots, (m_\ell, s_\ell)$. Pour vérifier ces signatures RSA plus rapidement, Bob décide de multiplier tous les messages entre eux : il calcule ainsi

$$m = m_1 m_2 \cdots m_\ell \mod n \quad \text{et} \quad s = s_1 s_2 \dots s_\ell \mod n\,.$$

Puis, il décide d'accepter la série de messages signés par Alice si et seulement si $s^e = m \mod n$.

Question 2.– Démontrer que si tous les messages ont bien été signés par Alice, alors Bob a raison d'accepter la série de signatures d'Alice.

Question 3.– Quantifier le gain de calcul de Bob en utilisant cette méthode.

Question 4.– Charlie sait que Bob utilise cette méthode pour vérifier les signatures d'Alice. Charlie intercepte une série $(m_1, s_1), \dots, (m_\ell, s_\ell)$ de messages signés par Alice (Charlie n'a donc pas choisi les messages). Comment peut-il intégrer un autre message m' à la série pour faire croire à Bob qu'Alice a également signé m' ?

Solutions de l'Exercice 4.

Solution Q1. Si n est de taille t bits, alors on doit effectuer $O(t)$ opérations arithmétiques (dans $\mathbb{Z}/n\mathbb{Z}$) pour calculer $m^d \mod n$, qui est la signature du message m.

Solution Q2. Si tous les messages ont été signés par Alice, alors on a :

$$s_i = m_i^d \mod n, \quad \forall i \in \{1, \dots, \ell\}.$$

Puis,

$$s^e = (s_1 \dots s_\ell)^e = (m_1^d \dots m_\ell^d)^e = m^{de} = m \mod n$$

Solution Q3. Avec la méthode usuelle, Bob aurait effectué ℓ élévations à la puissance e, soit $O(\ell t)$ opérations arithmétiques. Avec la vérification simultanée, il n'effectue plus qu'une élévation à la puissance e, et $\ell - 1$ multiplications, donc $O(\ell + t)$ opérations arithmétiques.

Solution Q4. L'idée est la suivante : Charlie peut modifier les messages et les signatures tant que les produits $m = m_1 \dots m_\ell \mod n$ et $s = s_1 \dots s_\ell$ restent constants. Si Charlie souhaite intégrer un message m' à la série de messages et prétendre qu'il a été signé, il peut donc simplement :

- transformer m_ℓ en $m'_\ell = m_\ell / m'$ et ajouter son message $m'_{\ell+1} = m'$ à la série de message (ainsi on a $m_1 \dots m_{\ell-1} m'_\ell m'_{\ell+1} = m \mod n$),
- ajouter une valeur de signature $s'_{\ell+1} = r$ quelconque à la série de signature, et définir une nouvelle signature $s'_\ell = s_\ell / r$, de sorte que le produits les éléments de la nouvelle série de signatures soit encore $s_1 \dots s_{\ell-1} s'_\ell s'_{\ell+1} = s \mod n$.

Exercice 5. $(\star)$ Signature ElGamal : réutilisation de l'aléa.

On s'intéresse au schéma de signature ElGamal, dans lequel le message est haché avant d'être signé. Plus précisément, si H est une fonction de hachage à valeurs dans le groupe $\mathbb{F}_p^\times$, l'algorithme de signature est donné dans l'Algorithme 6.

Algorithme 6 : Algorithme de signature d'ElGamal avec fonction de hachage

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

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 qu'**Alice réutilise le même aléa k pour toutes ses signatures**.

Question 1.– Soient $s = (b, c)$ et $s' = (b', c')$ les signatures de deux messages distincts m et m' (avec le même 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.

Solutions de l'Exercice 5.

Solution Q1. Comme k est le même pour toutes les signatures, on a

$$b \equiv g^k \equiv b' \mod p\,.$$

On obtient alors

$$(h - ab)c' \equiv (h - ab)(h' - ab)k^{-1} \equiv (h' - ab)(h - ab)k^{-1} \equiv (h' - ab)c \mod (p-1)$$

Solution Q2. On cherche la clé privée a d'Alice. Pour cela, on demande à Alice deux signatures $s = (b,c)$ et $s' = (b',c')$ quelconques (pour deux messages $\boldsymbol{m}$ et $\boldsymbol{m}'$ connus mais non-choisis). On a alors :

$$(h - ab)c' \equiv (h' - ab)c \mod (p-1) \quad \iff \quad a \equiv \frac{h'c - hc'}{b(c - c')} \mod (p-1)$$

Pour résumer :

1. On calcule $h = H(\boldsymbol{m})$ et $h' = H(\boldsymbol{m}')$.
2. On calcule l'inverse de $b(c - c')$ modulo $p - 1$ (s'il est inversible).
3. On retourne $a = \frac{h'c - hc'}{b(c-c')} \mod (p-1)$

Cette attaque fonctionne si b et $c - c'$ sont inversibles modulo $p - 1$, ce qui est vérifié avec bonne probabilité en les considérant comme des éléments aléatoires de $\mathbb{Z}/(p-1)\mathbb{Z}$.

Remarque. On pourrait également demander à Alice davantage de signatures pour augmenter la probabilité de réussite.

Exercice 6. 💻 (⋆⋆) Calcul rapide de $g^b b^c$ dans la vérification d'ElGamal.

Dans l'algorithme de vérification de la signature ElGamal, on doit calculer la valeur $\alpha^b b^c$, où α est un élément de $\mathbb{F}_p^\times$, et $(b, c) \in \{1, p-2\}$ peuvent être considérés comme aléatoires.

L'algorithme *square-and-multiply* permet de calculer une exponentiation dans un groupe cyclique d'ordre ℓ, en $\log_2(\ell)/2$ multiplications et $\log_2(\ell)$ carrés en moyenne (l'exposant est supposé aléatoire). Si b et c sont aléatoires, le calcul de $g^b b^c$ requiert donc, en moyenne, approximativement $\log_2(p)$ multiplications et $2\log_2(p)$ carrés.

Le but de cet exercice est de calculer $\alpha^b b^c$ sensiblement plus rapidement grâce à l'Algorithme 7.

Question 1.– Démontrer que pour tout $i = \ell - 1, \ldots, 0$, à la fin de la boucle **Pour (...)** de l'Algorithme 7, on a

$$x = \alpha^{\sum_{j=i}^{\ell-1} b_j 2^{j-i}} b^{\sum_{j=i}^{\ell-1} c_j 2^{j-i}}\,.$$

En déduire que l'algorithme est correct.

Question 2.– Compter le nombre moyen de carrés et le nombre moyen de multiplications effectués par l'Algorithme 7, lorsque b et c sont des entiers de ℓ bits tirés aléatoirement.

Algorithme 7 : Algorithme de calcul rapide de $a^b b^c$.

Entrée : $a \in \mathbb{F}_p^\times$, $b = \sum_{i=0}^{\ell-1} b_i 2^i$ et $c = \sum_{i=0}^{\ell-1} c_i 2^i \in \{1, \dots, p-2\}$
Sortie : $a^b b^c$

```
Calculer z ← ab.
Initialiser x ← 1.
Pour tout i allant de ℓ − 1 à 0 faire
    Calculer x ← x^2
    Si (b_i, c_i) = (1, 1)
        Calculer x ← xz
    Si (b_i, c_i) = (1, 0)
        Calculer x ← xa
    Si (b_i, c_i) = (0, 1)
        Calculer x ← xb
Retourner x.
```

Question 3.– Implanter l'Algorithme 7 et vérifier l'amélioration pratique qu'il procure, comparé aux calculs successifs de a^b et b^c par la méthode *square-and-multiply*.

Solutions de l'Exercice 6.

Solution Q1. Se démontre par induction. On note x_i la valeur de x en sortie de la i-ème boucle.

- C'est vrai à l'entrée dans la boucle (qu'on peut penser comme une étape $i = \ell$) : on a bien $x_\ell = 1 = a^0 b^0$.
- Supposons le résultat vrai à l'étape i, pour un certain $i \in \ell, \dots, 1$. Trois cas se présentent :
  — si $(b_i,c_i) = (1,1)$, alors la valeur de x_i devient

$$\begin{aligned} x_i = (x_{i+1})^2 z &= \left(a^{\sum_{j=i+1}^{\ell-1} b_j 2^{j-(i+1)}} b^{\sum_{j=i+1}^{\ell-1} c_j 2^{j-(i+1)}}\right)^2 ab \\ &= a^{1+\sum_{j=i+1}^{\ell-1} b_j 2^{j+1-(i+1)}} b^{1+\sum_{j=i+1}^{\ell-1} c_j 2^{j+1-(i+1)}} \\ &= a^{\sum_{j=i}^{\ell-1} b_j 2^{j-i}} b^{\sum_{j=i}^{\ell-1} c_j 2^{j-i}} \end{aligned}$$

  — si $(b_i,c_i) = (1,0)$, alors la valeur de x_i devient

$$\begin{aligned} x_i = (x_{i+1})^2 a &= \left(a^{\sum_{j=i+1}^{\ell-1} b_j 2^{j-(i+1)}} b^{\sum_{j=i+1}^{\ell-1} c_j 2^{j-(i+1)}}\right)^2 a \\ &= a^{1+\sum_{j=i+1}^{\ell-1} b_j 2^{j+1-(i+1)}} b^{\sum_{j=i+1}^{\ell-1} c_j 2^{j+1-(i+1)}} \\ &= a^{\sum_{j=i}^{\ell-1} b_j 2^{j-i}} b^{\sum_{j=i}^{\ell-1} c_j 2^{j-i}} \end{aligned}$$

  — si $(b_i,c_i) = (0,1)$, alors la valeur de x_i devient

$$\begin{aligned} x_i = (x_{i+1})^2 b &= \left(a^{\sum_{j=i+1}^{\ell-1} b_j 2^{j-(i+1)}} b^{\sum_{j=i+1}^{\ell-1} c_j 2^{j-(i+1)}}\right)^2 b \\ &= a^{\sum_{j=i+1}^{\ell-1} b_j 2^{j+1-(i+1)}} b^{1+\sum_{j=i+1}^{\ell-1} c_j 2^{j+1-(i+1)}} \\ &= a^{\sum_{j=i}^{\ell-1} b_j 2^{j-i}} b^{\sum_{j=i}^{\ell-1} c_j 2^{j-i}} \end{aligned}$$

La valeur retournée en fin d'algorithme est donc bien $x_0 = a^{\sum_{j=0}^{\ell-1} b_j 2^j} b^{\sum_{j=0}^{\ell-1} c_j 2^j} = a^b b^c$.

Solution Q2. En moyenne, on effectue un carré, et $3/4$ multiplications par boucle, auxquels il faut ajouter la multiplication pour obtenir $z = ab$. On obtient donc

$$\ell \text{ carrés} \quad \text{et} \quad \tfrac{3}{4}\ell + 1 \text{ multiplications.}$$

C'est bien mieux que les 2ℓ carrés et ℓ multiplications pour le calcul successif de a^b et b^c.

Solution Q3. Voir fichiers annexes.
