Cryptographie, RSA

1 Exercices

Exercice 1.1 (RSA jouet)**.**
On prend $p = 17$ et $q = 13$ donc $n = 221$.

(a) Déterminer $\varphi(n)$.
(b) Vérifier qu'on peut utiliser $e = 7$ comme exposant de chiffrement. Calculer l'exposant de déchiffrement d.
(c) Chiffrer $M = 3$. Déchiffrer $C = 198$.
(d) Pour p et q quelconque, estimer la complexité des opérations de calcul des clefs, chiffrement, déchiffrement.

Exercice 1.2 (RSA $e = 3$)**.**
(a) Expliquer quel est l'intérêt de choisir $e = 3$ comme clef publique si on ne se préoccupe pas de la sécurité.
(b) Supposons que $e = 3$ soit utilisé pour vérifier des signatures, prenons pour exemple jouet $n = 587 \times 383$. Comment peut-on calculer efficacement une signature s correspondant à un nombre $m = s^e \pmod n$ sans connaitre la factorisation de n si s est inférieur à $n^{1/3}$? Par exemple pour $m = 205379$.
(c) Pouquoi vaut-il mieux choisir $e = 257$ ou $e = 65537$ que $e = 3$?

Exercice 1.3 (Diffie-Hellmann, secret commun)**.**
Exemple jouet :
Alice et Bob choisissent de travailler dans $\mathbb{Z}/19\mathbb{Z}$ et d'utiliser $g = 2$ qui est un générateur de $(\mathbb{Z}/19\mathbb{Z})^*$.

(a) Alice choisit $a = 7$ et Bob choisit $b = 13$. Donner A et B puis le secret commun.
(b) Que se passe-t-il si Alice choisit $a = 25$? À quelle valeur maximale pour a et b Alice et Bob peuvent-ils se restreindre ?
(c) Déterminer la table de toutes les puissances de 2 dans $\mathbb{Z}/19\mathbb{Z}$.
(d) Alice et Bob choisissent deux autres entiers a et b et s'envoient $A = 4$ et $B = 17$. En utilisant la table, déterminer les valeurs de a et b. Quelle est la valeur du secret commun ?

Sécurité :
On a vu qu'une personne qui connaît les entiers A et B (par exemple en espionnant les échanges entre Alice et Bob) pouvait calculer le secret commun si on travaille dans $\mathbb{Z}/19\mathbb{Z}$. Pour espérer sécuriser le secret commun, il faut travailler dans $\mathbb{Z}/p\mathbb{Z}$ avec p un nombre premier plus grand.

(a) Commençons par essayer avec $p = 65537$ et $g = 3$.
  (i) Vérifier que 3 est un générateur de $(\mathbb{Z}/p\mathbb{Z})^*$.
  (ii) Si Alice choisit $a = 12345$, combien doit-elle effectuer de multiplications pour calculer A par l'algorithme de la puissance rapide ?
  (iii) À quelle valeur maximale pour a et b Alice et Bob peuvent-ils se restreindre ?
  (iv) Quelle est la taille de la table des puissances de 3 modulo p ? Comparer avec la question (ii). La sécurité du secret commun vous semble-t-elle suffisante ?

(*b*) On prend maintenant un nombre premier dont l'écriture en base 2 comporte exactement 1024 bits, et tel que $g = 2$ est un générateur de $(\mathbb{Z}/p\mathbb{Z})^*$. La sécurité du secret commun vous semble-t-elle suffisante ?

Exercice 1.4 (Attaque active contre Diffie-Hellman)**.**
On pourra travailler avec des valeurs explicites sur un exemple jouet de groupe $(\mathbb{Z}/19\mathbb{Z})^*$ avec comme générateur $g = 2$. Alice et Bob veulent échanger un secret commun, ils choisissent chacun une clef secrète a, b et envoient g^a et g^b. Charles intercepte les messages et envoie à la place aux deux g^e où e est sa propre clef secrète. Comment Charles doit-il ensuite modifier les message émis par Alice qu'il intercepte pour les retransmettre à Bob ?

Exercice 1.5 (Attaque contre RSA (module partagé))**.**
Alice et Bob décident de recevoir des messages cryptés en utlisant le système RSA. Ils publient donc chacun leur clef publique.

On va étudier une attaque qu'un espion peut exploiter si Alice et Bob utilisent tous les deux la même valeur de n. On suppose donc que la clef publique d'Alice est (n, e_A), et celle de Bob (n, e_B). On suppose que Catherine envoie une même information m à Alice et à Bob, donc envoie le message crypté $c_A = m^{e_A} \pmod n$ à Alice et le message crypté $c_B = m^{e_B} \pmod n$ à Bob. Un espion Daniel intercepte les deux messages cryptés.

Dans l'exercice, on prendra $n = 4897$ pour pouvoir faire des calculs à la calculatrice.

(*a*) Dans cette question on suppose qu'on connaît la factorisation de n : $n = 59 \times 83$. Expliquer pourquoi on peut prendre $e_A = 71$ et $e_B = 227$. Déterminer la clef privée d'Alice.
(*b*) Déterminer une identité de Bézout entre 71 et 227.
(*c*) En déduire deux entiers u et v tels que $m = c_A^u c_B^v \pmod n$
(*d*) Daniel intercepte $c_A = 1846$ et $c_B = 487$. Déterminer m sans utiliser la factorisation de n.
(*e*) Expliquer pourquoi Daniel ne peut pas utiliser la factorisation de n dans une attaque réelle contre RSA.

Exercice 1.6 (Attaque RSA par itération)**.**
(*a*) Exemple jouet : on prend $n = 77$ et une clé publique $c = 7$, le message original est $a = 2$. Retrouver le message original par itération de la fonction de cryptage sur le message crypté.
(*b*) En général, quelles sont les valeurs de c vulnérables à une attaque par itération ?

2 TP

Exercice 2.1 (Vérification machine des calculs)**.**
Vérifier les résultats des exercices du TD.

Exercice 2.2 (Générer une paire de clefs)**.**
Générer deux grands nombres premiers p et q au hasard puis une paire de clefs, en utilisant par exemple les fonctions `nextprime` et `randint` de Xcas ou le test de Miller-Rabin si votre langage préféré n'a pas de test de primalité.

Exercice 2.3 (Codage et décodage d'un message (sur PC))**.**
On transforme une chaine de caractères en une liste d'entiers et réciproquement (avec `asc` et `char` en Xcas, ou l'application répétée de `ord` et `chr` en Python). Pour le moment on code caractère par caractère, sans s'inquiéter de la sécurité du codage. Pour coder/décoder une liste `l` d'entiers, on peut utiliser `pow(l,c,n)` en Xcas et Python.

(*a*) En utilisant la paire de clefs de l'exercice précédent, coder un message puis décoder ce message pour vérifier.

(*b*) Décoder le message authentifié situé à l'URL
`http://www-fourier.univ-grenoble-alpes.fr/~parisse/mat249/rsa1`

Exercice 2.4 (Attaque simple)**.**
On a vu que le codage monoalphabétique n'est pas une bonne idée, une attaque possible étant la recherche de fréquences, ici on peut utiliser une attaque encore plus simple : la personne souhaitant décoder un message codé avec une clef publique sans en connaitre la clef secrète calcule la liste des $a^c \pmod n$ pour les 256 valeurs possibles de a et compare au message.

Décoder de cette manière le message situé à l'URL
`http://www-fourier.univ-grenoble-alpes.fr/~parisse/mat249/rsa2`

Exercice 2.5 (Padding aléatoire)**.**
Pour parer à l'attaque précédente, on augmente le nombre de valeurs possibles de a pour que le calcul de la liste de toutes les puissances possibles de a soit trop long.

Plusieurs stratégies sont possibles, l'une d'elle consiste à ajouter à a un multiple aléatoire de 256. Comment la personne qui recoit un message crypté retrouvera-t-elle le message en clair ? Implémenter cette méthode.

Exercice 2.6 (Groupement de lettres)**.**
On peut aussi grouper par paquets de x caractères et on associe à un groupe de caractères l'entier correspondant en base 256. Par exemple, si on prend des groupes de $x = 3$ caractères, "ABC" devient `65*256^2+66*256+67` car le code ASCII de A, B, C est respectivement 65, 66, 67.

(*a*) Donner une condition reliant n et x pour que le décodage redonne le message original.
(*b*) Choisir une paire de clefs vérifiant cette condition pour $x = 3$ (calculatrices avec entiers représentés par des flottants) ou $x = 8$ (autres).
(*c*) Écrire un programme de codage et de décodage avec groupement (on commencera par compléter le message original par des espaces pour qu'il soit un multiple de 8 caractères, en Xcas et Python). L'instruction `len` permet de connaitre la taille d'une chaîne de caractères, (En Xcas, on pourra utiliser la fonction `convert(.,base,256)`d'écriture en base 256).

Exercice 2.7 (Sécurité du codage)**.**
(*a*) Vérifier sur l'exemple de l'exercice 1.1 et du 2.2 que la connaissance de $\varphi(n)$ et de n permet de calculer p et q par résolution d'une équation de degré 2.
(*b*) Si on connait seulement c et d, peut-on retrouver $\varphi(n)$?
(*c*) La sécurité du codage repose donc sur la difficulté de factoriser n. Tester sur des entiers de taille croissante le temps nécessaire au logiciel pour factoriser p et q. Une valeur de n de taille 128 bits, 512 bits, 1024 bits parait-elle suffisante ?

Exercice 2.8 (Sécurité du codage)**.**
Le choix de c et de d est aussi important. Pour le comprendre, prenons $p = 11$ et $q = 13$.

(*a*) Représenter pour différentes valeurs de c les points $(a, a^c \pmod n)$. Plus le dessin obtenu est aléatoire, plus il sera difficile à une personne mal intentionnée de déchiffrer un message sans connaitre la clef. (En Xcas, on pourra utiliser les instructions `seq` pour générer une suite de terme général exprimée en fonction d'une variable formelle, et `scatterplot(l)` qui représente le nuage de points donné par une liste `l` de couples de coordonnées. En Python, on peut utiliser l'instruction `plot` de `matplotlib`).
(*b*) Observer en particulier les cas où c n'est pas premier avec $\varphi(n)$ (comment voit-on que RSA ne fonctionne pas ?) et également le cas $c = 3$.

Exercice 2.9 (Attaque par les restes chinois)**.**
Une personne souhaite envoyer le même message x à trois destinataires différents, ayant chacun leur propre clef publique $c = 3, N_1$, $c = 3, N_2$ et $c = 3, N_3$ avec $c = 3$ pour les 3 destinataires.

Il envoie donc $y_1 = x^3 \pmod{N_1}$, $y_2 = x^3 \pmod{N_2}$ et $y_3 = x^3 \pmod{N_3}$. Une personne mal intentionnée arrive à intercepter y_1, y_2 et y_3. En appliquant les restes chinois, elle peut en déduire x.

Par exemple, retrouver x sans chercher à factoriser les clefs pour

```
46693373016 mod 180711261397, (-111575037168) mod 840724735099,

(-18270191368) mod 372130013641
```

Exercice 2.10 (Attaque RSA).
On suppose qu'on connait un couple de clé secrète/publique
`9666444569588462909530283637832811667582471504662603`3 pour 65537 pour n valant
`14853687637916039711650329532817378529646911522656401`13
En déduire la factorisation de n.

Exercice 2.11 (Attaque RSA par fraction continue).
Cette attaque fonctionne si la clef privée d est petite et si p et q sont du même ordre de grandeur :

$$q < p < 2q.$$

Le principe consiste à calculer les réduites de e/n. Soit $k \in]0, d[$ tel que

$$ed = 1 + k\varphi(n) = 1 + k(n + 1 - p - q) = kn + 1 + k(1 - p - q)$$

On divise par dn :

$$\frac{e}{n} = \frac{k}{d} + \frac{1 + k(1 - p - q)}{dn}$$

donc :

$$\left|\frac{e}{n} - \frac{k}{d}\right| = \frac{k(p + q - 1) - 1}{dn} \leq \frac{k(p + q)}{nd}.$$

(*a*) En déduire que $\left|\frac{e}{n} - \frac{k}{d}\right| \leq \frac{3}{\sqrt{n}}$.
Si $3/\sqrt{n} < 1/(2d^2)$, alors les résultats connus sur les fractions continues permettent de conclure que k/d est une réduite de e/n. On calcule ces réduites en utilisant les résultats intermédiaires de l'algorithme d'Euclide étendu et on teste si $m^{de} = m \pmod{n}$.

(*b*) Mettre en oeuvre cette attaque, par exemple pour

```
n=241217702326110088055199747587228944702906245353 41
c=707896313355520595017418380434002615912172060722 9
```

Exercice 2.12 (Polllard=rho).
Programmer l'algorithme de Pollard-rho pour chercher un facteur de taille au plus environ 10 digits d'un entier.