TP : Le chiffrement RSA

L'algorithme de chiffrement *RSA* (du nom de ses inventeurs Rivest, Shamir, et Adelman qui l'ont imaginé en 1978) est la pierre angulaire de la cryptographie moderne, fournissant une méthode sécurisée pour échanger des données. On parle de **chiffrement asymétrique** : une clé publique est utilisée pour chiffrer, une clé privée pour déchiffrer.

Fonctionnement de l'algorithme

Génération des clés (publique et privée)

On commence par choisir deux nombres premiers p et q, par exemple :

$$p = 5 \qquad q = 11$$

On calcule ensuite $n = p \times q$ (appelé **module de chiffrement**) et $\phi(n) = (p-1) \times (q-1)$:

$$n = 55 \qquad \phi(n) = 40$$

On choisit ensuite un entier e premier avec $\phi(n)$ et strictement inférieur à $\phi(n)$, appelé **exposant de chiffrement**. Par exemple :

$$e = 7$$

On détermine d, l'*inverse modulaire* de e modulo $\phi(n)$, c'est à dire l'unique nombre entier strictement inférieur à $\phi(n)$ et vérifiant $e \times d = 1[\phi(n)]$. Ce nombre est appelé **exposant de déchiffrement**, et peut se calculer facilement avec l'algorithme d'Euclide étendu. Dans notre exemple :

$$d = 23$$

Les clés sont générées :

- la **clé publique** est le couple $(n, e) = (55, 7)$
- la **clé privée** est le couple $(n, d) = (55, 23)$

☞ *Une fois les clés générées, on peut jeter les entiers p et q à la poubelle !*

Chiffrement et Déchiffrement

Pour chiffer un entier $M < n$, on calcule $C = M^e[n]$.
À partir d'un entier chiffré C, on peut retrouver l'entier initial par le calcul $M = C^d[n]$.

☞ *Le résultat ci-dessus n'est pas magique : il est la conséquence du **petit théorème de Fermat**, une propriété mathématique liant congruences et nombres premiers*

Chiffrement du mot *RSA*, chaque lettre est associée à un entier M compris entre 1 (A) et 26 (Z) :

Chiffrement - Clé $(n, e) = (55, 7)$		
Lettre	Valeur M	Valeur chiffrée C
R	18	$18^7[55] = 17$
S	19	$19^7[55] = 24$
A	1	$1^7[55] = 1$

Déchiffrement - Clé $(n, d) = (55, 23)$		
Valeur chiffrée C	Valeur M	Lettre
17	$17^{23}[55] = 18$	R
24	$24^{23}[55] = 19$	S
1	$1^{23}[55] = 1$	A

☞ *Dans le déchiffrement, on peut voir le nombre $24^{23} = 55572324035428505185378394701824$: en réalité, 24^{23} n'est pas calculé, mais sa valeur modulo 55 est déterminée grâce à un algorithme d'exponentiation rapide ! En Python, on fera* `pow(24,23,55)` *plutôt que* `24**23 % 55`

Dans ce TP, on formera des groupes de 4 personnes : 2 seront chargées du chiffrement, et les deux autres du déchiffrement.

Partie A : Entraînement

Dans cette partie, on considère les entiers premiers $p = 13$ et $q = 17$.

1. Calculer les valeurs de n et $\phi(n)$.
2. Choisir un entier e inférieur strict à $\phi(n)$, et premier avec ce dernier.
3. Calculer l'entier d tel que $e \times d = 1[\phi(n)]$.
4. Déterminer les clés publique (n, e) et privée (n, d).
5. Chiffrer le message *RSA* avec la clé publique (n, e).

   Chaque lettre sera maintenant codée par sa valeur ASCII (A = 65, B = 66...).

   Donner le résultat sous la forme de trois octets écrits en héxadécimal.
6. Déchiffrer le message précédent avec la clé privée (n, d).

Partie B : Exemple complet

1. Choisir deux nombres premiers p et q tels que $p \times q < 65536$ (soit 2^{16}).
2. Calculer des clés publique et privée.
3. Choisir un mot et le chiffrer. Donner le résultat en héxadécimal, chaque lettre étant chiffrée par un nombre écrit sur 2 octets.
4. Écrire le chiffrement obtenu au tableau, ainsi que la clé publique utilisée.
5. Déchiffrer le résultat obtenu.

Partie C : Cracker RSA ?

1. Expliquer comment, à partir de la clé publique (n, e), on peut retrouver la clé privée (n, d).
2. Choisir un exemple au tableau et retrouver la clé privée correspondante, avant de déchiffrer le message associé.
3. Comment faire pour limiter la « découverte » de la clé privée ?

Partie D : Exemple avec des « grands » nombres premiers

1. Dans cette question, on choisit les nombres premiers $p = 999979$ et $q = 999983$.
   (a) Quelle est la valeur de n correspondante ?
   (b) Combien faut-il prévoir d'octets pour le chiffrement d'une lettre ? Est-ce un problème ?
2. Plutôt que de chiffrer toutes les lettres du mot, on préfère généralement chiffrer le mot (ou les données à transférer) de façon symétrique avec une clé K. On transmet alors le mot chiffré « simplement », et on ajoute au message la clé K chiffrée avec RSA.
   Pour l'exemple, on peut considérer un chiffrement symétrique bien connu : le XOR.
   (a) Choisir un nombre entier aléatoire K compris entre 0 et 255 (K « tient » sur un octet).
   (b) Choisir un mot à chiffrer ; donner la valeur binaire (sur 1 octet) de chacune de ses lettres en ASCII.
   (c) Pour chaque lettre L, calculer L XOR K afin d'obtenir le mot chiffré avec XOR.
   (d) Donner alors le message chiffré complet (chacune des lettres + clé K chiffrée avec RSA).
   (e) Si un mot contient des lettres identiques, qu'en est-il du mot chiffré ? Comment régler ce problème ?

3. Imaginons qu'un petit malin intercepte le message chiffré précédemment (mot chiffré avec XOR + clé K chiffrée avec RSA).

   Peut-il déchiffrer facilement ce message, sans connaître K ? Peut-il modifier ce message afin de transmettre une information erronée ?

Partie E : Authentification

RSA n'est pas seulement utilisé pour chiffrer des messages, mais aussi pour prouver l'identité de l'expéditeur et garantir l'intégrité du message grâce aux signatures numériques.
La méthode est assez simple :

- L'expéditeur calcule un *hash* du message qu'il transmet, puis chiffre ce *hash* avec sa **clé privée**.
- Il rajoute ce *hash* chiffré, appelé **signature**, au message transmis (par exemple, au début).
- Le destinataire déchiffre le *hash* transmis avec la clé publique. Il peut alors vérifier l'intégrité du message en recalculant le *hash* et en le comparant avec le *hash* déchiffré.
- Bien entendu, expéditeur et destinataire doivent utiliser la même fonction de *hash*, mais cette information n'a pas besoin d'être secrète.

Si entre temps, un petit malin a modifié le message, il n'est à priori par capable de fournir un *hash* correct (puisqu'il faut le chiffrer avec la clé privée), ce qui limite la transmission de données erronées.

Sans authentification, il serait facile pour un tiers de falsifier ou de copier un message : l'utilisation de signatures RSA empêche précisément ce type d'attaque et assure à la fois l'intégrité, l'authenticité et la non-répudiation. C'est cette sécurité qui permet, par exemple, la confiance dans les certificats SSL, le courrier électronique signé, ou les transactions bancaires électroniques.

☞ *Dans un navigateur WEB, si l'on se trouve sur une page sécurisée par HTTPS, il est possible de voir si l'algorithme utilisé est RSA. Il suffit d'afficher le certificat utilisé, ce qui permet d'accéder à la valeur du module n (si RSA est utilisé) ou encore à l'algorithme de chiffrement symétrique utilisé pour la signature.*