Période 4

RSA

Devoir Maison

Les exercices à rendre se trouvent sur `ataraxy.info/Modelisation` dans le fichier `DM_NOMINATIF.pdf` de la période 4. Chaque page correspond à un étudiant ; reportez-vous à la table des matières de ce fichier pour déterminer la page correspondant à votre DM.

Proposition

Soient $p < q$ deux nombres premier, e un nombre premier avec $\varphi = (p-1)(q-1)$ et d l'inverse de e modulo φ. Pour tout $x \in \mathbb{Z}$,

$$x^{de} \equiv_{pq} x$$

Démonstration. Puisque d est l'inverse de e modulo φ on en déduit $de \equiv_{\varphi} 1$. D'une autre manière, il existe $k \in \mathbb{Z}$ tel que $de = 1 + k(p-1)(q-1)$. Dans ce cas $x^{de} = x \times \left(x^{(p-1)(q-1)}\right)^k$.

Si x n'est pas multiple de p et de q alors le corollaire du petit théorème Fermat implique que $x^{p-1} \equiv_p 1$ soit encore $x^{(p-1)(q-1)} \equiv_p 1$. De même $x^{(p-1)(q-1)} \equiv_q 1$. Le lemme chinois implique alors que $x^{(p-1)(q-1)} \equiv_{pq} 1$ et donc $x^{de} \equiv_{pq} x$.

Si x est multiple de p mais pas de q alors le corollaire au petit théorème de Fermat implique que $x^{q-1} \equiv_q 1$ et donc $x^{k(p-1)(q-1)} \equiv_q 1$. Par définition il existe $\alpha \in \mathbb{Z}$ tel que $x^{k(p-1)(q-1)} = \alpha q + 1$. En multipliant par x on a $x^{1+k(p-1)(q-1)} = \alpha x q + x$ mais $xq \equiv_{pq} 0$ car x est multiple de p. Dans ce cas $x^{de} \equiv_{pq} x$.

Si x est multiple de q mais pas de p c'est le même raisonnement.

Si x est multiple de p et de q alors il est nul modulo pq et trivialement $x^{de} \equiv_{pq} x (\equiv_{pq} 0)$. □

Définition

Soient $p < q$ deux nombres premiers. Notons $n = pq$ et $\varphi(n) = (p-1)(q-1)$

Les données suivantes définissent le cryptosystème de RSA de base n.

Espace de clefs. $\mathcal{K}_{p,q} = \{\texttt{Les entiers e premier à } \varphi(n)\}$

Fonction de chiffrement. Quelque soit $e \in \mathcal{K}_{p,q}$ et $x \in \mathbb{Z}$, $C_e(x) \equiv_n x^e$

Fonction de déchiffrement. Quelque soit $e \in \mathcal{K}_{p,q}$ et $x \in \mathbb{Z}$, $D_e(x) \equiv_n x^{e^{-1}}$ où e^{-1} désigne l'inverse de e modulo $\varphi(n)$.

La proposition précédente justifie que la propriété de déchiffrement est satisfaite.

Dans la pratique la clef de chiffrement, la donnée (n, e), est appelé la **clef publique** et la clef de déchiffrement, la donnée (n, e^{-1}) est appelé la **clef privée**.

Prenons par exemple $p = 3$ et $q = 11$. Dans ce cas, $n = 33$ et $\varphi(33) = 20$. Le nombre $e = 13$ est un entier premier à 20 donc $(33, 13)$ est une clef publique. Chiffrons le message *PIKACHU*

Message	P	I	K	A	C	H	U
Codage	15	08	10	00	02	07	20
$x^{13} \bmod 33$	9	17	10	0	8	13	14

Ainsi le message chiffré est $9-17-10-0-8-13-14$.

Imaginons avoir reçu le message $8-0-29-0-14-8-31$ avec la même clef de chiffrement. L'inverse de 13 modulo 20 s'obtient en appliquant l'algorithme d'Euclide étendue. On trouve $13^{-1} \equiv_{20} 17$ Déchiffrons alors le message.

Message	08	00	29	00	09	14	08	31
$x^{17} \bmod 33$	02	00	17	00	15	20	02	04
Décodage	C	A	R	A	P	U	C	E

La sécurité de ce cryptosystème réside dans le fait que même si on connait la clef publique (n, e) il est difficile de trouver l'inverse de e car on ne connait pas φ (p et q en fait). Bien sur ce cryptosystème sera

sûre avec de grande valeur pour p et q. Ainsi si vous savez qu'un message a été codé avec $(9068370463, 17)$ il sera difficile (du moins sans ordinateur) de trouver la clef privée.

Le paquetage est "automatique" avec cette méthode.

Si l'entier n est compris entre 26 et 2525 on travaillera par paquet de 2, s'il est entre 252526 et 25252525 on travaillera par paquet de 3 etc.

Prenons par exemple $p = 509$ et $q = 691$ alors $n = 351719$: on chiffrera par paquet de 3. Prenons $e = 7$ qui est bien premier avec $\varphi(351719) = 350520$.

Message	A	T	T	R	A	P	E	Z	L	E	S	T	O	U	S
Codage	00	19	19	17	00	15	04	25	11	04	18	19	14	20	18
Paquetage	1919			170015			42511			41819			142018		
$x^7 \bmod 351719$	27519			87444			106946			327923			329813		

et le message chiffré est $27519 - 87444 - 106946 - 327923 - 329813$.

Échange et signature

Madame A souhaite transmettre un message suivant le protocole RSA à monsieur B. Monsieur B a crié sur tous les toits que sa clef publique est (n_B, e_B) et a secrètement gardé sa clef privée (n_B, d_B). D'ailleurs madame A a fait la même chose avec sa clef publique (n_A, e_A) et privée (n_A, d_A).

Madame A code son message claire M avec la clef publique de monsieur B et publie sur son mur FB le message crypté. Tout le monde peut le voir mais seul monsieur B qui dispose de la clef de déchiffrement peut retrouver le message initial M.

Monsieur B voudrait être sûre que le message qu'il reçoit provient bien de madame A et pas d'un pirate qui se ferait passer pour elle. Madame A va alors signer son message avant de le chiffrer et de l'envoyer. Elle applique au message claire M sa clef privée (n_A, d_A) puis ensuite la clef publique (n_B, e_B) de monsieur B. En recevant le message, monsieur B va appliquer sa clef privée et obtenir un message encore chiffré. Il va alors appliquer la clef publique (n_A, e_A) de A qui est à la seule qui a pu utiliser sa clef privée pour signer ce message, pour retrouver le message claire M.

Exponentiation modulaire rapide

Bien que cette méthode de chiffrement soit sûre elle est soumise à une contrainte : le calcul des puissances modulaires, communément appelé **exponentiation modulaire**.

Reprenons l'exemple du déchiffrement introduit plus tôt. Avec une calculatrice on a $1919^7 = 9.5835149e + 22$. Cet arrondi rend le déchiffrement impossible [1]. Il existe cependant une méthode rapide pour réaliser ces calculs sans arrondi ni problème du à la gestion de mémoire de la machine : c'est l'algorithme d'exponentiation modultaire rapide.

Cet algorithme permet de calculer très rapidement x^n modulo m. L'idée est d'écrire l'entier n en binaire (base 2).

Écrivons $n = \sum_{i=0}^{k} a_i 2^i$ où pour tout i, $a_i \in \{0, 1\}$. Dans ce cas, $x^n = \prod_{i=0}^{k} x^{2^i}$. L'avantage est que $(x^{2^i})^2 = x^{2^{i+1}}$. D'où l'algorithme.

```
Écrire n en binaire : n = Σ_{i=0}^{k} a_i 2^i
Pour i de 0 à k
        Si i=0
             poser x[0] = x modulo m
        Sinon
             x[i] = x[i-1]*x[i-1] modulo m
        Fin Si
Fin pour
res = 1
Pour i de 0 à k
        Si a_i=1
             res = res*x[i] modulo m
        Fin Si
Fin pour
Renvoyer res
```

1. Et il s'agit d'un cas très simple

Par exemple calculons 19^{19} modulo 2017.

Ecriture binaire. La première étape consiste à écrire 19 en binaire ce qui se fait par division euclidienne successive. On trouve $19 = (10011)_2$.

Calcul des puissances de puissance de 2. On représente la situation dans un tableau

k	0	1	2	3	4
2^k					
mod					

où la dernière ligne est la seconde ligne modulo $m = 2017$. Le tableau va jusqu'à $k = 4$ car c'est la plus grande puissance de 2 qui apparait dans l'écriture binaire de 19. On l'initialise de la sorte

k	0	1	2	3	4
2^k	19				
mod	19				

On prend la valeur de la dernière ligne de la colonne k que l'on met au carré et que l'on inscrit dans à la seconde ligne de la colonne $k+1$ et on calcul la réduction modulo m à la dernière ligne.

k	0	1	2	3	4
2^k	19	361			
mod	19	361			

On itère le processus.

k	0	1	2	3	4
2^k	19	361	130321		
mod	19	361	1233		

k	0	1	2	3	4
2^k	19	361	130321	1520289	
mod	19	361	1233	1488	

k	0	1	2	3	4
2^k	19	361	130321	1520289	2214144
mod	19	361	1233	1488	1495

Conclusion. Les puissances de 2 apparaissant dans l'écriture binaire de 19 sont 0, 1 et 4 on a donc $19^{19} \equiv_{2017} 19 \times 361 \times 1495 \equiv_{2017} 808 \times 1495 \equiv_{2017} 1794$