Semaine 13 : Sécurité [Solutions]

1 Principes de base

a)
- — Destruction d'informations
- — Démenti d'action
- — Manipulation d'informations
- — Usurpation d'identité
- — Vol d'informations
- — Disponibilité
- — Responsabilité
- — Intégrité
- — Authentification
- — Confidentialité

b) Confidentialité et intégrité

c) Responsabilité (signature électronique)

2 Cryptographie asymétrique

[ ] Chaque participant dispose de deux clés, l'une secrète et l'autre publique et utilise sa clé secrète pour ~~chiffrer~~ les messages confidentiels.

[ ] La cryptographie à clés publiques permet de chiffrer des données, ~~mais pas~~ de leur appliquer des signatures numériques.

[ ] La cryptographie à clés publiques est ~~plus performante~~ que la cryptographie à clés symétriques.

[**X**] Des messages chiffrés avec une clé publique ne peuvent être déchiffrés que si l'on connaît la clé secrète correspondante. Retrouver cette clé secrète à partir de la clé publique est un problème pratiquement incalculable.

[ ] Des messages chiffrés avec une clé secrète peuvent être déchiffrés avec ~~n'importe quelle~~ clé publique.

[**X**] Chaque participant dispose de deux clés, l'une secrète et l'autre publique et utilise sa clé secrète pour déchiffrer les messages confidentiels.

3 One-Time Pad

a) et **b)**

```
  0100110001110011
⊕ 1001001001100101
= 1101111000010110
```

(rappel : addition sans retenue)

4 RSA – Confidentialité

Le « craquage » le plus immédiat du système RSA consiste à factoriser la seconde partie de la clé publique (le $n = p \cdot q$ du cours). Ici ceci est trivial : $n = 55 = 5 \cdot 11$.

On cherche alors la clé privée d du professeur par l'équation $d \cdot e = 1 \mod (p-1)(q-1)$, c.-à-d. ici $d \cdot 7 = 1 \mod 4 \cdot 10$ ou encore $7\,d = 1 \mod 40$. Ce qui se fait à l'aide de l'algorithme d'Euclide généralisé :

u	**v**	r	**t**
$(0, 40)$	$(1, 7)$	$\frac{40}{7} = 5$	$(-5, 5)$
$(1, 7)$	$(-5, 5)$	$\frac{7}{5} = 1$	$(6, 2)$
$(-5, 5)$	$(6, 2)$	$\frac{5}{2} = 2$	$(-17, 1)$

qui nous donne alors $d = -17$, c.-à-d. en modulo 40 : $d = 23$.

Reste à décrypter le message. Cela se fait par $M = C^d \mod n = 25^{23} \mod 55$:

$$25^{23} = (25^{16}) \cdot (25^4) \cdot (25^3) = (25^4)^4 \cdot 15 \cdot 5 = 15^4 \cdot 20 = 25 \cdot 20 = 5 \mod 55$$

Votre note est donc 5.

5 RSA – Responsabilité

a) Dans ce protocole, la signature et transmission confidentielle d'un message M consiste en l'encryptage avec la clé (publique) e_P du professeur (confidentialité) de la signature de M avec la clé (privée) d_A de l'assistant (responsabilité) : $S = (M^{d_A} \mod n_A)^{e_P} \mod n_P$.

Pour répondre à l'exercice, il faut donc commencer par trouver la clé privée de l'assistant. Pour cela on factorise $n_A = 15 = 3 \cdot 5$, puis l'on résout $d_A \cdot e_A = 1 \mod (p_A - 1)(q_A - 1)$, c.-à-d. $d_A \cdot 3 = 1 \mod 2 \cdot 4$ d'où facilement $d_A = 3$.

La signature d'un message M sera donc donnée par :

$$S = (M^3 \mod 15)^7 \mod 77$$

Pour le message $M = 4$ nous avons $(4^3 \mod 15)^7 \mod 77$. Or $4^3 = 16 \cdot 4 = 1 \cdot 4 = 4 \mod 15$, d'où

$$S = 4^7 = (4^3)^2 \cdot 4 = 64^2 \cdot 4 = 15 \cdot 4 = 60 \mod 77$$

et pour le message crypté $4^7 \mod 77 = 60$. Le message envoyé sera donc 60,60.

Notes :

— Le fait que le message et la signature sont les mêmes vient de ce que $4^3 = 4 \mod 15$.
Notez que $x^3 = x \mod 15$ a de nombreuses solutions (0, 1, 4, 5, 6, 9, 10, 11 et 14).

— Avec le protocole simplifié présenté ici/en cours, il est en toute rigueur inutile d'envoyer le message et la signature car avec la seule signature on garantit à la fois que c'est bien l'assistant qui a envoyé la note et on connaît cette note. Mais la réalité est plus subtile :

1. premièrement, le cadre pourrait être un peu différent : l'assistant pourrait avoir publié par ailleurs la note en question et on lui demande d'envoyer la signature pour prouver que c'est bien lui qui a publié cette même note : dans un tel cadre il y aurait bien communication à la fois de M et de S, indépendemment ;
2. de plus, en réalité on ne signe pas le message lui-même, mais un résumé (hash) du message ; il importe alors d'également communiquer le message lui-même.

Pour plus de détails sur les subtilités entre décryptage et signature, voir « *RSA Signing is Not RSA Decryption* » de l'Université de Cornell : `https://www.cs.cornell.edu/courses/cs5430/2015sp/notes/rsa_sign_vs_dec.php`

b) Pour le décodage il faut connaître la clé privée du professeur. On procède comme précédemment : $n_P = 77 = 7 \cdot 11$, puis l'on résout $d_P \cdot e_P = 1 \mod (p_P - 1)(q_P - 1)$, c.-à-d. $d_P \cdot 7 = 1 \mod 60$ d'où $d_P = 43$ à l'aide de l'algorithme d'Euclide généralisé :

u	**v**	r	**t**
$(0, 60)$	$(1, 7)$	$\frac{60}{7} = 8$	$(-8, 4)$
$(1, 7)$	$(-8, 4)$	$\frac{7}{4} = 1$	$(9, 3)$
$(-8, 4)$	$(9, 3)$	$\frac{4}{3} = 1$	$(-17, 1)$

qui nous donne alors $d = -17$, c.-à-d. en modulo 60 : $d = 43$.

Le professeur décode alors tout d'abord le message par

$$M = C^{d_P} \mod n_P = 41^{43} \mod 77 = 6$$

puis la signature par

$$S' = (S^{d_P} \mod n_P)^{e_A} \mod n_A = (41^{43} \mod 77)^3 \mod 15 = 6^3 \mod 15 = 6$$

La note codée est donc correctement authentifiée et vaut 6 (Remarque : l'égalité du message crypté et de la signature vient encore du fait que nous avons $6^3 = 6 \mod 15$).

c) Si l'on décode la signature comme précédemment on trouve : $S' = 27^{43} \mod 77 = 48$. Or par construction le chiffre obtenu (qui représente $M^{d_A} \mod n_A$) doit être plus petit que $n_A = 15$, ce qui n'est pas le cas ici. Donc un message signé 27 ne vient pas de l'assistant.

Remarque : Pour se faire passer pour l'assistant et envoyer la note voulue N, un élève doit résoudre $e_A(x) = N$, ce qui par hypothèse est difficile [1]. Si, en effet, l'élève est capable de trouver x tel que $e_A(x) = N$, c'est qu'il a craqué le système RSA (au moins pour ce cas précis).

1. évidemment pas dans le cadre de cet exercice !

Supposons donc que l'élève tricheur n'ait pas craqué le système [2] ; il ne peut donc pas falsifier la note qu'il veut. En revanche, il peut essayer d'être un peu plus malin que dans le cas ci-dessus et générer un S' qui soit compatible avec la clé de l'assistant, c.-à-d. un S' qui soit plus petit que 15. Mais il n'est alors pas du tout sûr du résultat.

Si par exemple il tire au hasard $S' = 7$ parmi $[1, 15]$, il transmettra alors au professeur $e_B(7) = 28$. Le professeur décodera alors : $S' = 28^{43} \mod 77 = 7$, puis $M = 7^3 \mod 15 = 13$ qui n'est clairement pas un message valide.

Il est évident que dans le cadre trivial de cet exercice, trouver *par hasard* un message valide dont le S' soit aussi valide est facile. Ceci est dû à la petitesse des messages et à la simplicité des clés. Dans la réalité, la chance est de 1 sur le nombre de valeurs possibles $(n_A - 1)$ puisque $x \mapsto x^{d_A} \mod n_A$ est une bijection de $[\![1, n_A]\!]$ dans lui-même. Imaginez le problème réel avec des clés de 56, 1024, 2048 bits ou plus... (et les tester toutes est beaucoup trop long et/ou coûteux à stocker : p.ex. $2^{1024} \simeq 10^{308}$, pour environ 10^{79} atomes dans l'Univers observable, agé d'environ $10^{23}\mu s$).

6 Synthèse

On souhaite envoyer de façon efficace, confidentielle et garantie (signée et intègre) le message suivant :

VIVE ICC !

a) *Pour chacun de ces adjectifs (efficace, confidentielle, signée, intègre), quelle technique peut on utiliser ?*

efficace : compression : Shannon-Fano ou Huffman

confidentielle : cryptage à clé secrète (One-Time PAD, DES, ...) ou publique (RSA, ...)

signée : c.-à-d. responsabilité : cryptographie à clé publique

intègre : cryptographie à clé secrète ou publique.

b) code de Huffman :

(espace)	C	I	V	E	!
2	2	2	2	1	1
00	01	100	101	110	111

Message comprimé : `101 100 101 110 00 100 01 01 00 111` = `101100101110001000010100111`

d) On utilise pour la suite un système RSA. On choisit $p = 97$, $q = 137$ et $d = 55$.

Quelle est notre clé publique ?

$n = 97 \times 137 = 13289$

2. hypothèse, encore une fois, difficilement vérifiée dans le cas trivial de cet exercice.

$m = 96 \times 136 = 13056 = 2^8 \times 3 \times 17$: ok pas de facteur commun avec $d = 5 \times 11$.

On trouve e par l'algorithme d'Euclide (à l'aide d'un programme ou d'une calculatrice) :

u	**v**	r	**t**
$(0, 13056)$	$(1, 55)$	$\frac{13056}{55} = 237$	$(-237, 21)$
$(1, 55)$	$(-237, 21)$	$\frac{55}{21} = 2$	$(475, 13)$
$(-237, 21)$	$(475, 13)$	$\frac{21}{13} = 1$	$(-712, 8)$
$(475, 13)$	$(-712, 8)$	$\frac{13}{8} = 1$	$(1187, 5)$
$(-721, 8)$	$(1187, 5)$	$\frac{8}{5} = 1$	$(-1899, 3)$
$(1187, 5)$	$(-1899, 3)$	$\frac{5}{3} = 1$	$(3086, 2)$
$(-1899, 3)$	$(3086, 2)$	$\frac{3}{2} = 1$	$(-4985, 1)$
$(3086, 2)$	$(-4985, 1)$	(stop)	

ainsi $e = -4985 = 8071 \mod 13056$ et notre clé publique est : $(8071, 13289)$

e) Notre destinataire à pour clé publique :

3337 s'écrit sur 12 bit. Nos messages sont des nombres entiers entre 0 et 3336, qui nécessitent donc aussi 12 bits. On découpe donc le message par tranches de 12 bits en s'assurant à chaque fois ne pas être au dessus de 3337 (auquel cas on ne garde que 11 bits et on décale).

On pourrait aussi décider de couper nos messages par tranches de 11 bits, ce qui nous assure des nombres inférieurs à `11111111111`= 2047.

- En tranches de combien de bits découpe-t-on notre message comprimé (question b)) ?
  Note : par convention, on remplit la fin du message de départ avec des espaces
- Encryptez le message

`101100101110  001000101001  110000000000`

2862 553 3072

$2862^{1929} = 896 \mod 3337$

$553^{1929} = 495 \mod 3337$

$3072^{1929} = 2805 \mod 3337$

$896 =$ `001110000000` (sur 12 bits)

$495 =$ `000111101111`

$2805 =$ `101011110101`

Le message crypté est donc : `001110000000000111101111101011110101`

e) *On souhaite de plus signer notre message...*

Il nous faut signer chacun de nos sous-messages (puis les encrypter).

Faisons le ici pour le premier : 2862.

Pour signer nous devons utiliser notre propre clé privée : $2862^{55} \mod 13289$

On trouve : 72.

Que l'on encrypte pour envoyer à notre destinataire : $72^{1929} = 1137 \mod 3337$

soit `010001110001` sur 12 bits.

Pour le paquet suivant :

$553^{55} \mod 13289 = 4984$

on atteint les limites techniques de ce qui a été présenté en cours puisque 4984 est supérieur à 3337 et ne peut donc pas être envoyé comme tel à notre destinataire.

En réalité, le mécanisme d'encryptage comprend justement un mécanisme de troncature et complétion (« padding ») des messages pour garantir qu'ils restent valides (= dans les bornes) dans tous les cas.

Ces méthodes ne sont pas triviales (et hors de propos de ce cours) car elles ne doivent aucunement trahir le message d'origine.

f) *[...] intégrité [...]*

La transmission cryptée du message et de sa signature sont suffisant pour garantir l'intégrité car à la réception on vérifie que $e_A(d_B(S(M)))$ est bien égal à M.

Et, par hypothèse, seul A a pu produire $S(M)$ à partir de M (sinon, de toutes façon le fraudeur peut alors régénérer ce qu'il veut en se faisant passer pour A et donc n'importe quel autre résumé généré par n'importe quel autre moyen similaire).

En clair : la signature encryptée dans un système RSA est elle-même un résumé certifié du message. (elle est même plus car elle authentifie aussi son auteur)

Pour plus de détails : voir la fin de la correction de l'exercice 5.

7 DES

Avant tout, calculons les deux fonctions systématiquement utilisées par notre système de codage :

$$\begin{aligned} f(K_1, Y) &= f(0,1,1,0,1,y_1,y_2,y_3,y_4) = \begin{pmatrix} 0\cdot y_1 \oplus 1\cdot y_2 \oplus 0\cdot y_4 \oplus 1 \\ 0\cdot y_4 \oplus 1\cdot y_2 \oplus 1\cdot y_1 \oplus 1 \oplus y_3 \oplus 0 \\ (0\oplus 1\oplus 1\oplus y_2)\cdot y_1 \\ (1\oplus 0\oplus y_4)\cdot y_3 \end{pmatrix} \\ &= \begin{pmatrix} y_2 \oplus 1 \\ y_1 \oplus y_2 \oplus y_3 \oplus 1 \\ y_1 \cdot y_2 \\ y_3 \cdot (y_4 \oplus 1) \end{pmatrix} \end{aligned}$$

$$f(K_2, Y) == f(1,0,1,1,0,y_1,y_2,y_3,y_4) = \begin{pmatrix} y_1 \oplus y_4 \\ y_2 \oplus y_3 \oplus y_4 \oplus 1 \\ y_1 \cdot y_2 \\ y_3 \cdot (y_4 \oplus 1) \end{pmatrix}$$

a) `0100110001110011` se décompose tout d'abord en sous-messages de taille 8 :
M_1 =`01001100` et M_2 =`01110011`. Puis l'on code chacun des sous-messages :

M_1 se décompose en deux blocs de 4 bits : B_0 =`0100`, B_1 =`1100`. La clé donne elle aussi deux sous-clés (de 5 bits chacune) : K_1 =`01101` et K_2 =`10110`. Le nouveau bloc B_2 est alors obtenu par :

$$\begin{aligned} B_2 &= B_0 \oplus f(K_1, B_1) \\ &= \texttt{0100} \oplus f(K_1, \texttt{1100}) \end{aligned}$$

or

$$f(K_1; 1,1,0,0) = \begin{pmatrix} 1\oplus 1 \\ 1\oplus 1\oplus 0\oplus 1 \\ 1\cdot 1 \\ 0\cdot 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

et donc

$$B_2 = \texttt{0100} \oplus \texttt{0110} = \texttt{0010}$$

puis :

$$\begin{aligned} B_3 &= B_1 \oplus f(K_2, B_2) \\ &= \texttt{1100} \oplus f(K_2, \texttt{0010}) \end{aligned}$$

or

$$f(K_2; 0,0,1,0) = \begin{pmatrix} 0\oplus 0 \\ 0\oplus 1\oplus 0\oplus 1 \\ 0\cdot 0 \\ 1\cdot 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

et donc

$$B_3 = \texttt{1100} \oplus \texttt{0001} = \texttt{1101}$$

d'où

$$C_1 = B_2\ B_3 = \texttt{00101101}$$

Et l'on recommence avec M_2, qui se décompose en deux blocs de 4 bits : B_0 =0111, B_1 =0011. Le nouveau bloc B_2 est alors obtenu par :

$$B_2 = \texttt{0111} \oplus f(K_1, \texttt{0011}) = \texttt{0111} \oplus \texttt{1000} = \texttt{1111}$$

puis :

$$B_3 = \texttt{0011} \oplus f(K_2, \texttt{1111}) = \texttt{0011} \oplus \texttt{0010} = \texttt{0001}$$

d'où

$$C_2 = B_2\ B_3 = \texttt{11110001}$$

Et finalement le message codé total :

$$C = C_1\ C_2 = \texttt{0010110111110001}$$

b) Pour décoder il s'agit évidemment d'appliquer le processus inverse. On commence donc par séparer le message codé en deux blocs de 8 bits : C_1 =01001100 et C_2 =01110011. Puis l'on décode chacun de ces sous-messages :

C_1 se décompose en deux blocs de 4 bits : B_2 =0100, B_3 =1100. La clé étant la même que pour la question précédente, nous avons à trouver B_1 par l'équation :

$$\begin{aligned} B_3 &= B_1 \oplus f(K_2, B_2) \\ \texttt{1100} &= B_1 \oplus f(K_2, \texttt{0100}) \end{aligned}$$

d'où :

$$B_1 = \texttt{1100} \oplus f(K_2, \texttt{0100}) = \texttt{1100} \oplus \texttt{0000} = \texttt{1100}$$

De même :

$$\begin{aligned} B_2 &= B_0 \oplus f(K_1, B_1) \\ \texttt{0100} &= B_0 \oplus f(K_1, \texttt{1100}) \end{aligned}$$

d'où :

$$B_0 = \texttt{0100} \oplus f(K_1, \texttt{1100}) = \texttt{0100} \oplus \texttt{0110} = \texttt{0010}$$

d'où la première partie du message :

$$M_1 = B_0\ B_1 = \texttt{00101100}$$

De même on décode la seconde partie du message : C_2 se décompose en deux blocs de 4 bits : B_2 =0111, B_3 =0011. D'où :

$$\begin{aligned} B_3 &= B_1 \oplus f(K_2, B_2) \\ \texttt{0011} &= B_1 \oplus f(K_2, \texttt{0111}) \end{aligned}$$

d'où :

$$B_1 = \texttt{1100} \oplus f(K_2, \texttt{0111}) = \texttt{0011} \oplus \texttt{1000} = \texttt{1011}$$

De même :

$$\begin{aligned} B_2 &= B_0 \oplus f(K_1, B_1) \\ \texttt{0111} &= B_0 \oplus f(K_1, \texttt{1011}) \end{aligned}$$

d'où :

$$B_0 = \texttt{0111} \oplus f(K_1, \texttt{1011}) = \texttt{0111} \oplus \texttt{1100} = \texttt{1011}$$

d'où la deuxième partie du message :

$$M_2 = B_0\ B_1 = \texttt{10111011}$$

D'où le message d'origine :

$$T = M_1\ M_2 = \texttt{0010110010111011}$$

8 PGP

Pour décoder un message RSA, on calcule $M = C^d \mod n$, c.-à-d. ici $58'423^{16'381} \mod 172'831$. C'est là où réside la difficulté de l'exercice.

$16'381$ se décompose en toutes les puissances de 2 depuis 2^{13} jusqu'à 2^0 sauf 2^1. Il nous faut donc calculer itérativement les carrés successifs de $58'423$ modulo $172'831$ puis faire leur produit sauf $58'423^2$. Un petit programme ou une bonne calculatrice nous donne :

i	1	2	3	4	5	...	13
$58'423^i \mod 172'831$	58'423	7'510	111'652	57'194	103'339	...	53'603

Un autre calcul (ou programme) nous permet de faire leur produit et de trouver

$$58'423^{16'381} \mod 172'831 = 142'857$$

qui est donc la clé utilisée pour crypter le message.

Pour le décodage, il faut alors faire la soustraction modulo 10 :

```
17237714237119365013215214405835485718277114346213327723
14285714285714285714285714285714285714285714285714285714
                  ---------------------------
03052000052405180309030500220121200004092400161509142019
```

que l'on traduit alors en séquence de lettres pour trouver :

CET_EXERCICE_VAUT_DIX_POINTS