Université de Paris – UFR de mathématiques et Informatique
Master 1 informatique – UE Cryptographie et applications
TD n°3

Cryptographie moderne

Nom et prénom : GUARDIA Quentin
Date de remise de votre TD : Dimanche 18h00 par courriel
Email : master2srs.dir@gmail.com
Veuillez indiquer dans le sujet/objet de votre courriel : « CRYPTO TD3 »

Exercice 1 – Schéma de Feistel (DES)
1. Qu'est-ce qu'un schéma de Feistel ou SPN ? Citez un système cryptographique utilisant ce type de réseaux.

Les schémas Feistel et SPN offrent un chiffrement par bloc.
Dans un réseau Feistel, les données sont divisées en deux blocs. À chaque tour, on encode avec une clé un des blocs avant de faire un XOR bit-à-bit avec l'autre bloc initial. Puis on réitère à chaque tour en échangeant bloc initial et bloc à encoder avec une clé, avant le XOR. Le réseau de Feistel est utilisé par DES.
Un SPN divise les données en plusieurs blocs, qui sont permutés, substitués, et appliqués à des XOR bit-à-bit avec des clés. Le SPN est utilisé par AES.

Soit un schéma de Feistel à 3 rondes constitué des trois fonctions (non nécessairement bijectives) f_1, f_2 et f_3 de $\{0,1\}^3$ vers $\{0,1\}^3$ qui vont servir au chiffrement (elles constituent la clé de chiffrement) :

f_1 :	000	$\rightarrow$	110	f_2 :	000	$\rightarrow$	010	f_3 :	000	$\rightarrow$	111
	001	$\rightarrow$	100		001	$\rightarrow$	011		001	$\rightarrow$	010
	010	$\rightarrow$	111		010	$\rightarrow$	110		010	$\rightarrow$	110
	011	$\rightarrow$	000		011	$\rightarrow$	111		011	$\rightarrow$	110
	100	$\rightarrow$	110		100	$\rightarrow$	000		100	$\rightarrow$	000
	101	$\rightarrow$	010		101	$\rightarrow$	101		101	$\rightarrow$	101
	110	$\rightarrow$	001		110	$\rightarrow$	110		110	$\rightarrow$	100
	111	$\rightarrow$	101		111	$\rightarrow$	110		111	$\rightarrow$	001

Considérons le message m à six bits « **101110** ».

- m est découpé en deux parties L_0 et R_0 de longueur 3 (parties gauche et droite du message)
- A l'aide de la fonction f_1 de $\{0,1\}^n$ vers $\{0,1\}^n$, les parties L_0 et R_0 sont transformées lors d'une première ronde en :
  $L_1 = R_0$ et $R_1 = L_0 \oplus f_1(R_0)$ avec $\oplus$ l'opération de XOR binaire
- Ce schéma est réitéré deux fois de suite à l'aide de la fonction f_2 (ronde 2) puis à l'aide de la fonction f_3 (ronde 3).

2. Déterminer m', le chiffre du message m, en appliquant le schéma de Feistel à 3 rondes, sachant que :

$$\begin{aligned} L_1 &= R_0 \\ R_1 &= L_0 \oplus f(R_0) \end{aligned}$$

$L_0 = 101$ et $R_0 = 110$
$L_1 = 110$ et $R_1 = 101$ XOR $f_1(110) = 101$ XOR $001 = 100$
$L_2 = 100$ et $R_2 = 110$ XOR $f_2(100) = 110$ XOR $000 = 110$
$L_3 = 110$ et $R_3 = 100$ XOR $f_3(110) = 100$ XOR $100 = 000$
$m' = 110000$

3. Réaliser le déchiffrement du message m' sachant que :

$$\begin{aligned} R_0 &= L_1 \\ L_0 &= R_1 \oplus f(R_0) \end{aligned}$$

L_3 = 110 et R_3 = 000
L_2 = 000 XOR f_3(110) = 000 XOR 100 = 100 et R_2 = 110
L_1 = 110 XOR f_2(100) = 110 XOR 000 = 110 et R_1 = 100
L_0 = 100 XOR f_1(110) = 100 XOR 001 = 101 et R_0 = 110
m = 101110

Voir http://www.apprendre-en-ligne.net/crypto/blocs/feistel.html

Exercice 2 - Protocole Diffie-Hellman

Deux utilisateurs *Alice* (un client web) et *Bob* (serveur web) ne se connaissent pas et veulent partager un secret afin de chiffrer leurs communications futures. Le protocole de Diffie-Hellman permet de se mettre d'accord sur un secret à distance. Ce secret peut ensuite servir de clef privée dans un système cryptographique symétrique.

1. Expliquez les détails du fonctionnement du protocole de Diffie-Hellman.

Lorsqu'Alice et Bob initialisent leur connexion, ils se mettent d'accord sur sur deux clés publiques : **n**, un grand nombre premier et **g** un entier strictement positif inférieur à **n**.

Alice choisit secrètement un entier **a** et Bob **b**. Puis,
Alice calcule **A** tel que $A=g^a$ mod n
Bob calcule **B** tel que $B=g^b$ mod n

Alice et Bob s'échangent en clair **A** et **B**. Puis,
Alice calcule $K=B^a$ mod n
Bob calcule $K= A^b$ mod n
Les deux **K** sont égaux et c'est la clé secrète que seuls Alice et Bob ont pu calculer. Capturer les données envoyées en clair ne suffit pas pour retrouver la clé secrète.

2. On supposera qu'Alice sélectionne la valeur privée suivante : a=6 ; et les deux nombres premiers publiques suivants : n=23 et g=3.
Bob sélectionne lui la valeur privée suivante b=15.
Calculer la clé secrète symétrique Ks.

Par exponentiation modulaire :
Alice calcule A : a = 6 donc A = 3^6 mod 23 = 3^2*3^4 mod 23 = 9 * 12 mod 23 = 16
Bob calcule B : b = 15 donc B = 3^{15} mod 23 = $3^8*3^4*3^2*3^1$ mod 23 = 12

Alice et Bob s'échangent A et B
Alice calcule K: 12^6 mod 23 = 9
Bob calcule K : 16^{15} mod 23 = 9

3. Est-il totalement sûr ? Soit un attaquant *Charlie* placé entre *Alice* et *Bob*, interceptant le trafic et se faisant passer pour *Alice* auprès de *Bob* et pour *Bob* auprès d'Alice. *Charlie* peut-il pénétrer les communications entre *Alice* et *Bob* ? Comment s'appelle cette attaque ?

Le système n'est pas sûr. En effet, dans l'exemple donnée Charlie peut calculer la clé secrète. On parle alors d'attaque de type « Man in the middle » (attaque de l'homme du milieu en français).

Exercice 3 - hachage cryptographique

1. Quel(s) est (sont) le(s) service(s) de sécurité garantie(s) par l'utilisation d'une fonction de hachage seule sur un message m ?

Une fonction de hachage seule peut garantir :
- l'intégrité de **m** par checksum
- une authentification si une clé est concaténée à m, ce qui créerai donc un sceau
- une vérification de mot de passe en comparant deux empreintes, afin de ne pas manipuler des données en clair

2. Quelles sont les propriétés que doivent vérifier les fonctions de hachage cryptographiques pour pouvoir être utilisées dans le cadre d'applications cryptographiques ?

Une fonction de hachage est forcément :
- à sens unique : on ne peut pas deviner m à partir de son empreinte
- sans collision : m ne peut pas avoir la même empreinte qu'un autre message

3. Citez 3 exemples de fonctions de hachage les plus courantes et la taille (en bits) de l'empreinte générée.

La fonction MD5 produit une empreinte de 128 bits
La fonction SHA-1 produit une empreinte de 160bits
La fonction SHA-256 produit une empreinte de 256 bits

4. Donnez deux exemples d'applications utilisant les fonctions de hachage à sens unique dans lesquelles il est important que ces propriétés soient vérifiées. Pour chaque application, vous expliquerez en quoi ces propriétés interviennent

La signature numérique d'un message nécessite une fonction de hachage et sert à vérifier l'intégrité, l'authentification et la non-répudiation d'un message par son émetteur. Dans un premier temps l'émetteur calcule l'empreinte de son message et le chiffre avec sa clé privée afin d'obtenir une signature numérique. Puis il envoie au récepteur le message en clair avec la signature associée. Le récepteur déchiffre la signature à l'aide de la clé publique de l'émetteur. En cas d'échec, on sait que le message n'est pas d'Alice. Puis il compare le résultat à l'empreinte du message clair qu'il a reçu. Si cela ne fonctionne pas alors le message a été modifié depuis qu'Alice a signé ou alors le message n'est pas d'Alice. Dans cet exemple, on a besoin d'une fonction de hachage à sens unique pour assurer l'authentification et la non-répudiation de l'émetteur. La fonction doit être sans collision afin de garantir l'intégrité du message.

Une deuxième application est celle de l'authentification par nonce. Le client demande au serveur un nonce. Ce dernier lui renvoie une valeur aléatoire ou pseudo-aléatoire. Puis le client s'identifie grâce à son login, son propre nonce (cnonce) et le hcsh de la concaténation de nonce, cnonce et du mot de passe. Si le hachage n'était pas à sens unique alors toute personne interceptant les échanges pourrait accéder à des données personelles. D'où l'intérêt du sens unique. De même, si un autre client mal-intentionné ou non se connecte, il ne pourra pas y avoir d'interversion de données car le hach est forcément différent grâce à la propriété de non-collision et aux valeurs différentes du mot de passe et/ou du nonce.

5. En combinant une fonction de hachage H et une clé secrète K, il est possible de calculer un HMAC. Quel(s) est (sont) le(s) service(s) de sécurité garantie(s) par l'utilisation d'un HMAC sur un message m ? Expliquer comment est calculé ce HMAC sur un message m.

HMAC garantit l'authentification de l'émetteur et l'intégrité du message **m**.
L'émetteur envoie au récepteur le message en clair accompagné du hachage du message concaténé à la clé secrète partagée.

À son tour, le récepteur concatène le message clair à la clé secrète partagée, calcule le hach et compare avec le hach qui accompagnait le message clair.

Exercice 4 - chiffrement à clefs publiques
On appelle E une fonction de chiffrement à clef publique et D la fonction de déchiffrement associé.
On suppose qu'il existe une fonction de signature associé à E que l'on notera S. On notera VS la fonction de vérification de signature associé.
On suppose que toutes les personnes intervenant dans cet exercice ont chacune un couple (clef privée, clef publique) correspondant aux fonctions cités ci-dessus.
Par souci de simplification, on supposera que le même couple peut servir indifféremment aux opérations de chiffrement ou de signature.

1. Alice veut envoyer un message chiffré à Bob, avec quelle clef doit-elle le chiffrer ?
   A l'arrivée, quelle clef, Bob doit il utiliser pour déchiffrer le message ?

Alice doit chiffrer avec la clé publique (E) et Bob doit déchiffrer avec la clé privée (D)

2. Alice veut envoyer un message signé à Bob, avec quelle clef doit-elle le signer ?
   A l'arrivée, quelle clef, Bob doit-il utiliser pour vérifier la signature du message ?

Alice doit signer l'empreinte avec la clé privée (S) et Bob doit la déchiffrer avec sa clé publique (VS) pour la comparer à l'empreinte qu'il a obtenu.

3. Alice veut envoyer un message chiffré et signé à Bob, avec quelle clef doit-elle le chiffrer ? Le signer ?
   A l'arrivée, quelle clef, Bob doit-il utiliser pour déchiffrer le message ? Pour vérifier la signature ?

Alice chiffre le message avec la clé publique (E) et signe avec la clé privée (S). Bob déchiffre le message avec la clé privée (D) et vérifie la signature avec la clé publique (VS).

4. Alice veut envoyer un message chiffré et signé à Bob, Gérard, Jackie, Ahmed, ... (25 destinataires) avec quelle clef doit-elle le chiffrer ? Le signer ?

Alice chiffre le message avec la clé publique (E) et signe avec la clé privée (S). Les destinataires déchiffrent le message avec la clé privée (D) et vérifient la signature avec la clé publique (VS).

Exercice 5: RSA
1. Qu'est-ce que RSA ?

RSA est une méthode de chiffrement fonctionnant avec un algorithme de cryptographie asymétrique.

Voici comment l'employer.
Première étape, définir les clés publique et privée :
Il faut commencer par définir **p** et **q**, deux nombres premiers distincts.
Ensuite, on multiplie **p** et **q** pour obtenir **n**.
On calcule ø(n)=(p–1)(q-1) afin de trouver un entier **e**, compris entre 1 et ø(n) exclus tel que pgcd(e,ø(n))=1.
On détermine **d** tel que d*e mod ø(n) =1.
La clé publique est KU={e,n}
La clé privée est KR={d,n}

Deuxième étape, chiffrer le message :
D'abord, on doit convertir les valeurs du message en ASCII.

On découpe le message obtenu en blocs de même longueur, quitte à ajouter des zéros. Il faut juste veiller à ce que la valeur de chaque bloc soit inférieure à n.
On encrypte en calculant pour chaque bloc : $bloc^e \bmod n$

Dernière étape, déchiffrer le message :
Pour chaque bloc chiffré, on calcule le $bloc^d \bmod n$.
On convertit les valeurs ASCII en caractères.
Et voici le texte en clair.

Soit p=29 et q=37
Soit M un message en clair, M= "HELLO"
Soit C= (M), le message crypté de M

2. Calculer KU et KR, sachant que e=71

n=p*q=29*37=1073
ø(n)=28*36=1008
e=71
d*71 mod 1008 =1
Ce qui revient à l'identité de Bézout : d*71+k*1008=1. On va chercher d et k tels que l'équation soit vérifiée.
On applique l'algorithme d'Euclide :
1008=71*14+14 donc 14=1008-71*14
71=14*5+1 donc 1=71-14*5
On remonte l'algorithme :
1=71-(1008-71*14)*5
1=71-1008*5+71*70
1=71*71-1008*5
D'où 71*71 mod 1008=1
Donc d=71

KU={71,1073}
KR={71,1073}

3. Chiffrer le message M, sachant que selon le code ASCII:
   H=72, E=69, L=76, O=79

En ASCII, HELLO=7269767679
=726 976 767 900

Pour le premier bloc :
726^{71} mod 1073. Par exponentiation modulaire :
$726^{64}*726^{4}*726^{2}*726^{1}$ mod 1073 avec :
726^{1} mod 1073 = 726
726^{2} mod 1073 = 233
726^{4} mod 1073 = 233^{2} mod 1073 = 639
726^{8} mod 1073 = 639^{2} mod 1073 = 581
…
726^{64} mod 1073 = 639
Donc 726^{71} mod 1073= 639*639*233*726 mod 1073 = 581*233*726 mod 1073 = 436

De la même manière,
le deuxième bloc encrypté est 822

le troisième bloc encrypté est 825
le quatrième bloc encrypté est 552

Donc HELLO encrypté est 436822825552

4. Déchiffrer le message C. Expliquez comment avez-vous procédé.

On redivise le message chiffré en blocs de 3 chiffres :

436 822 825 552

En reprenant la méthode de l'exponentiation modulaire, on trouve :
436^{71} mod 1073=726
822^{71} mod 1073=976
825^{71} mod 1073=767
552^{71} mod 1073=900

72 69 76 76 79 (00)
H E L L O