Département d'informatique
Faculté des sciences exactes et des sciences de la nature et de la vie
Université Mohamed Khider, Biskra

Année universitaire 2019/2020.

Sécurité Informatique — Série d'exercices N 1 (L3) (Solutions)

Exercice 1

[Challal 2016]

1. Le chiffrement symétrique
   - √ **Garantie la confidentialité du message chiffré**
   - ◯ Utilise une paire de clés publique/privée
   - ◯ Assure la non répudiation
   - √ **Repose sur la confidentialité de la clé utilisée**
   - ◯ Repose sur la confidentialité de l'algorithme de chiffrement utilisé
2. RSA
   - √ **Est un système cryptographique asymétrique**
   - √ **Repose sur la difficulté de factoriser un grand nombre en ses facteurs premiers**
   - ◯ Repose sur la difficulté du logarithme discret
   - √ **Permet de faire une signature digitale**
   - ◯ Ne peut pas être utilisé pour assurer la confidentialité
3. Pour garantir la non-répudiation de l'origine, on peut utiliser
   - √ **Un chiffrement RSA avec la clé privée de l'émetteur**
   - ◯ Un MAC
   - √ **Une signature digitale**
   - ◯ Un chiffrement RSA avec la clé publique de l'émetteur
   - ◯ Un échange de clé Diffie-Hellman

Exercice 2

[Avoine *et al.* 2010] Un groupe de n personnes souhaite utiliser un système cryptographique pour s'échanger deux à deux des informations confidentielles. Les informations échangées entre deux membres du groupe ne devront pas pouvoir être lues par un autre membre.

1. Le groupe décide d'utiliser un système de chiffrement symétrique. Quel est le nombre minimal de clés symétriques nécessaires ?

   Nombre de clés = n(n-1)/2

2. Le groupe décide ensuite de remplacer ce système par un système asymétrique. Quel est le nombre minimal de couples de clés asymétriques nécessaires pour que chaque membre puisse envoyer et recevoir des informations chiffrées et/ou signées ?

   Nombre de couples de clés = n

3. Bob souhaite envoyer des informations chiffrées et signées à Alice. Quelles clés Bob doit-il utiliser ?

   Pour le chiffrement : Bob utilise la clé publique de Alice. Pour la signature : Bob utilise sa clé privée

4. Le groupe décide finalement d'utiliser un système hybride pour le chiffrement. Donner les raisons qui ont poussé ce groupe à utiliser un tel système.

   Pour profiter des avantages des deux classes cryptographiques et optimiser les performances.

Exercice 3

[Avoine *et al.* 2010] Bob qui utilise souvent la messagerie sécurisée de son entreprise, vient de perdre sa clés privée, mais dispose encore de la clé publique correspondante.

1. Peut-il encore envoyer des courriers électroniques chiffrés ? **(OUI)** En recevoir ? **(NON)**
2. Peut-il encore signer les courriers électroniques qu'il envoie ? **(NON)** Vérifier les signatures des courriers électroniques qu'il reçoit ? **(OUI)**
3. Que doit-il faire pour être de nouveau capable d'effectuer toutes les opérations mentionnées ci-dessus ?
   (Il doit regénérer un nouveau couple de clés pubique et privée. Il faut en plus, diffuser la nouvelle clé publique).

Exercice 4

[Challal 2016] On souhaite concevoir un protocole d'échange sécurisé. Ce protocole doit garantir les services de sécurité suivants :

— Confidentialité de l'échange

— Intégrité des données échangées

— Non-répudiation de l'émetteur

On suppose que la confidentialité est assurée grâce à un chiffrement symétrique.
Initialement, les deux parties communicantes ne partagent aucun secret.
Ecrire une spécification de ce protocole en utilisant la notation suivante :

— $S ==> D$: M (S envoie M à D)

— $\{M\}_K$: M chiffré/déchiffré avec K

— SK_X : Clé privée de X

— PK_X : Clé publique de X

— K_{AB} : Clé symétrique partagée entre A et B

— $H(M)$: code de hachage de M

— $H_{-K}(M)$: code MAC de M

— . (point) : Concaténation entre deux messages

Solution de l'exercice 4 :

Phase 1 : Partage sécurisé de la clé symétrique (clé secrète)

A ==> B : {Kab} PKb . {H(Kab)} SKa

(ou bien : B ==> A : {Kab} PKa . {H(Kab)} SKb

Phase 2 : Echage sécurisé de données

A ==> B : {M} Kab . {H(M)} SKa

B ==> A : {M} Kab . {H(M))} SKb

La confidentialité est assurée grace au chiffrement symétrique.

L'intégrité des données est assurée par la signature numérique plus particulièrement, le hachage.

La non répudiation de l'émetteur est également garantie par la signature qui utilise la clé privée de l'émetteur .

Exercice 5

[Challal 2016] Dans cet exercice, vous allez implémenter le protocole que vous avez conçu dans l'exercice précédent en utilisant OpenSSL.

Cet atelier nécessite un jeu de rôles. Une partie (élève ou binôme) jouera le rôle de l'émetteur (source), et une autre partie jouera le rôle du récepteur (destinataire).

Pensez à inverser les rôles et refaire l'exercice.

La Source souhaite transmettre un message stocké dans un fichier `message.txt` à une Destination tout en assurant sa confidentialité, intégrité et non-répudiation.

Cet atelier pratique sera réalisé avec OpenSSL. Vous pouvez utiliser l'outil déjà installé dans les distribution Linux. Sous Windows, vous pouvez installer `xampp` qui fournira cet outil. Dans ce cas, pensez à rajouter dans la variable d'environnement `PATH` le chemin vers le répertoire qui contient `openssl.exe` : ....`xampp\apache\bin`

Voici ci-après des commandes `openssl` qui permettent de mettre en oeuvre les différentes étapes du protocole.

Pensez à analyser chaque commande avant son utilisation

La liste n'est pas exhaustive. Néanmoins, vous pouvez changer certains paramètres pour répondre aux besoins nécessaires pour mettre en œuvre votre protocole.

Indice :

— Générer la clé privée RSA de la source :
`openssl genrsa -out src_rsa.pem -passout pass:srcpasswd -des 512 src_rsa.pem` va contenir la clé privée de la source protégée par le mot de passe `srcpasswd`

— Extraction de la clé publique à partir de la clé privée vers "src_rsa_pub.pem" :
`openssl rsa -in src_rsa.pem -passin pass:srcpasswd -out src_rsa_pub.pem -pubout`
Ici l'option `-pubout` permet d'extraire la clé publique (par défaut c'est la clé privée qui est extraite)
L'option `passin` et `passout` c'est pour protéger les fichiers, `pass:xxxx` permet de faire une protection par mots de passe.

— Chiffrer un secret avec la clé publique du destinataire :
`openssl rsautl -in secret.txt -out secret.crypt -inkey dest_rsa_pub.pem -pubin -encrypt`

— Chiffrement d'un message avec un secret (clé symétrique) en utilisant l'algorithme DES-CBC :
`openssl enc -des-cbc -in message.txt -out message.crypt -pass file:secret.txt`

— Calcul de condensat (code de hashage) avec MD5 :
`openssl dgst -md5 -binary -out message.crypt.dgst message.crypt`

— Chiffrement du condensat (code de hashage) avec la clé privée de la source "src_rsa.pem" :
`openssl rsautl -in message.crypt.dgst -out message.crypt.dgst.sign -sign -inkey src_rsa.pem`

— Déchiffrement de l'empreinte (code de hashage) du message vers dgst1 :
`openssl rsautl -in message.crypt.dgst.sign -out dgst1 -pubin -inkey src_rsa_pub.pem`

— Déchiffrement d'un secret avec la clé privée du destinataire :
`openssl rsautl -decrypt -in secret.crypt -out secret.txt -inkey dest_rsa.pem`

— Déchiffrement du "message.crypt" à l'aide de secret qui se trouve dans "secret.txt" :
`openssl enc -in message.crypt -out message.txt -pass file:secret.txt -d -des-cbc`