Université de Paris – UFR de mathématiques et Informatique

Master 1 Cybersécurité – Cryptographie et applications

TD n°4

Chiffrement, déchiffrement et signature d'un message avec OpenSSL

Noms et prénom binôme 1 : GUARDIA Quentin
Noms et prénom binôme 2 : CLEMENTE Fabien
*Instructions : me transmettre votre TP par courriel à **Master2srs.dir@gmail.com**, en précisant dans le sujet « CRYPTO TP4 »*
Date limite de remise du TP : Lundi 30 novembre 13h00

Introduction :
OpenSSL est une boîte à outils cryptographiques implémentant des systèmes cryptographiques modernes.

Vous pourrez utiliser les instructions suivantes :

- *$openssl version* : pour obtenir la version d'openSSL
- *$openssl help* : pour obtenir de l'aide
- *$openssl genrsa [-des3] -out <rsaprivatekey.pem> <size>* : pour génèrer une paire de clés RSA de size bits. La valeur de size : 512bit, 1024bit, 2048bit, etc.
- *$openssl rsa -in <rsaprivatekey.pem> -text -noout*: pour visualiser le contenu du fichier rsaprivatekey.pem contenant la paire de clés en format texte. L'option *-noout* supprime l'affichage de la clé en format PEM.
- *$openssl rsa -in <rsaprivatekey.pem> -des3 -out <enc_rsaprivatekey.pem>* : pour chiffrer la clé privée avec l'algorithme DES3.
- *$openssl rsa -in <rsaprivatekey.pem>* ***-pubout*** *-out <rsapublic.pem>* : pour extraire de la partie publique et l'enregistrer dans un fichier à part.
- `$openssl enc <-algo> -in <file.txt> -out <encfile.bin>` : pour chiffrer le fichier file.txt avec l'algorithme symétrique spécifié *($openssl list-cipher-commands* pour avoir la liste des possibilités ou bien *$openssl enc --help*) en un fichier binaire chiffré : encfile.bin.
- *$openssl rsautl -encrypt* ***-pubin*** *-inkey rsapublic.pem -in file.txt -out encfile.bin* : pour chiffrer un fichier file.txt en un encfile.bin avec la clé publique.
- *$openssl rsautl -decrypt -inkey rsaprivate.priv -in file.bin -out fileclear.txt* : pour déchiffrer dans un fichier fic.dec
- *$openssl rsautl -sign –in file.txt -inkey rsaprivatekey.pem -in file.txt -out signature.bin* **:** pour la signature d'un message.
- `$openssl rsautl -verify -pubin -inkey rsapublickey.pem -in signautre.bin -out hashfile.txt` : pour la vérification de la signature.
- `$openssl dgst -<algo> -out <hashfile.txt> <file.txt>`: pour hacher un fichier. <algo> est l'algorithme de hachage (SHA1, MD5, etc.).
- `$openssl rand -out <clé.key> <numbits>`: pour générer un nombre aléatoire sur numbits.
- `$openssl enc -aes-128-cbc -salt -in <file.txt> -out <file.bin> -pass pass:password` : pour chiffrer un fichier avec l'algorithme AES utilisant le mot de passe *password*.
- `$openssl enc -aes-128-cbc -salt -d -in <fichier.bin> -out <fichier.txt> -pass file:/path/to/shared_key.pem` **:** pour déchiffrer un

fichier avec l'algorithme AES utilisant un mot de passe sauvegardé dans le fichier *shared_key.pem*.

***Attention** : **sous windows**, il faut modifier le chemin où est enregistré le fichier « openssl.cfg » en tapant en ligne de commande: Dans cet exemple, les fichiers ont été enregistrés sur la racine du disque C et le chemin absolu est alors « C:\OpenSSL-Win32 ». Il faut taper la commande ci-dessous :*
"C:\OpenSSL-Win32\openssl set OPENSSL_CONF=C:\OpenSSL-Win32\bin"

Exercice 1 :

Décompresser l'archive et recuperer les fichiers du répertoire « Openssl files »

- Chiffrer le fichier ″toto.txt″ avec l'algorithme Blowfish en mode CBC, avec un mot de passe de votre choix (par exemple: *master1*), le fichier chiffré portera le nom de ″toto.enc″.
  openssl enc -bf-cbc -in toto.txt -out toto.enc -pass pass:master1:

- Visualiser le contenu du fichier ″toto.enc″. De quel codage s'agit-il ?
  C'est un codage en binaire.

- Répéter la tâche 1 en ajoutant l'option ″***-base64***″ lors du chiffrement de fichier ″toto.txt″, et stocker le résultat dans le fichier ″toto.b64″. Visualiser le contenu du fichier ″toto.b64″. De quel codage s'agit-il ?
  openssl enc -base64 -bf-cbc -in toto.txt -out toto.b64 -pass pass:master1:
  C'est codé en base64.

- Déchiffrer un des fichiers précédents (″toto.b64″ ou ″toto.enc″)
  openssl enc -d -base64 -bf-cbc -in toto.b64 -out toto1.txt -pass pass:master1:
  openssl enc -d -bf-cbc -in toto.enc -out toto2.txt -pass pass:master1:

- Tentez de déchiffrer le fichier ″toto.enc″ ou ″toto.b64″ avec un mauvais mot de passe. Comment réagit OpenSSL ?
  OpenSSL indique l'erreur suivante :
  bad decrypt
  139986769809856:error:06065064:digital envelope routines:EVP_DecryptFinal_ex:bad decrypt:../crypto/evp/evp_enc.c:537:

Remarque: PEM (Privacy Enhanced Mail) n'est pas un système de chiffrement, mais un codage des fichiers binaires avec 64 caractères ASCII (base64). Ce codage est utilisé en particulier pour la transmission des fichiers binaires par courrier électronique.

Exercice 2 :

Le fichier ″encrypted_file.b64″ a été chiffré avec le système AES en mode CBC, la clé de 256 bits ayant été obtenue par un mot de passe stocker en base64 dans le fichier ″password.b64″.

- Le mot de passe codé en base 64, pourriez vous le décoder ?
  Oui il est possible de visualiser en clair le mot de passe dans un fichier codé en base64 au moyen de la commande suivante :
  openssl enc -base64 -d -in password.b64 -out password.txt

- Déchiffrer ensuite le fichier ″encrypted_file.b64″
  openssl enc -d -md md5 -base64 -aes-256-cbc -in encrypted_file.b64 -out encrypted_file.txt -pass file:password.txt

Exercice 3 :

- Générer votre paire de clé RSA de taille 1024 bits en la stockant dans un fichier « *rsaprivatekey.pem* » et ensuite visualiser le contenu du fichier avec *openssl rsa*.
  openssl genrsa -out rsaprivatekey.pem 1024
  openssl rsa -in rsaprivatekey.pem -text -noout

En fait, cette clé privée contient les éléments de l'algorithme RSA suivants :
Clé publique :

- *Modulus* est le modulo n
- *PublicExponent* est l'exposant public e (chiffrement)

Clé privée :

- *Modulus:* est le modulo n
- *PrivateExponent* est l'exposant prive d (déchiffrement)

Les autres éléments sont destines à faciliter les calculs (et sont prives) :

- *Prime1* est le nombre premier p
- *Prime2* est le nombre premier q (n=p.q)
- *Exponent1* est d mod (P-l)
- *Exponent2* est d mod (q-l)
- *Coefficient* est (q^{-l}) mod p

Il n'est pas prudent de laisser une paire de clef en clair, surtout avec l'accès physique et distant à la machine par une autre personne, qui risque de causer beaucoup de dégâts.

Pour cela, OpenSSL offre la possibilité de chiffrer la paire des clés.

- Protéger la clé précédemment générée avec chiffrement 3DES et une phrase de passe (*passphrase*) « *securite* » (*correspondant à la clé secrète utilisée par 3DES*). Visualiser le contenu avec une de ces commandes : ***more*** ou ***cat*** ou ***less***. (ss linux) ou

type (ss windows) Utiliser de nouveau la commande d'affichage de la clé RSA pour visualiser le contenu de la clé. Que constatez-vous ?
openssl rsa -in rsaprivatekey.pem -des3 -out enc_rsaprivatekey.pem
J'ai entré le mot de passe « mot de passe ». On visualise le contenu du fichier :
cat enc_rsaprivatekey.pem nous indique, avant le début de la clé :
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,E824F6ECA56DAA99

Ce qui permet à **openssl rsa** de savoir qu'il faut demander un mot de passe pour visualiser avec **openssl rsa -in rsaprivatekey.pem -text -noout**. Ainsi je peux toujours afficher le contenu de la clé avec cette commande, à condition de connaître le mot de passe.

- Extraire la partie publique de votre biclé (fichier précédent .pem) dans un fichier *rsapublickey.pem*, et ensuite utiliser la commande rsa pour visualiser cette dernière (attention vous devez préciser l'option ***–pubout*** puisque seule la partie publique est contenue dans le fichier rsapublickey.pem).
  openssl rsa -in rsaprivatekey.pem -pubout -out rsapublickey.pem

- Créez un document texte « *message.txt* » contenant la phrase confidentielle suivante:
  > Ceci est un message classé confidentiel !
  echo "Ceci est un message classé confidentiel!" >> message.txt

- Chiffrer le message avec votre clé publique dans un fichier « *message.enc* »
  openssl rsautl -encrypt -pubin -inkey rsapublickey.pem -in message.txt -out message.bin

Exercice 4 :

Le message/fichier *"**encrypted.bin**"* a été chiffré avec un mot de passe de 256 bits et l'algorithme aes-cbc. Ce mot de passe (*"**encrypted_sym_key.bin**"*) a été lui-même protégé par un chiffrement avec la clé publique. Par ailleurs, la clé privée (la paire de clés est stockée dans le fichier *"**rsaprivkey.pem**"*) a été chiffrée avec le mot de passe *"**ssic"***.
1. Saurez-vous déchiffrer le fichier *"**encrypted.bin**"* dans un fichier en clair « ***cleart_text.txt*** »?
Oui, cela est possible. Il faut commencer déchiffrer la clé privée :
openssl rsa -in rsaprivkey.pem -out dec_rsaprivkey.pem
En entrant « ssic » lorsque la pass phrase est demandée.
Puis, on déchiffre le mot de passe d'encrypted_text.bin, encrypted_sym_key.bin avec la clé que l'on vient d'obtenir :
openssl rsautl -decrypt -inkey dec_rsaprivkey.pem -in encrypted_sym_key.bin -out encrypted_sym_key.txt
Enfin, on déchiffre encrypted.bin
openssl enc -md md5 -d -aes-256-cbc -in encrypted.bin -out encrypted.txt -pass file:encrypted_sym_key.txt

2. Après le déchiffrement du fichier ″encrypted.bin″, calculer la signature de ce fichier en clair.
On crée les clés:
openssl genrsa -out sign_key.pem 1024
On crée la signature avec la clé obtenue:
openssl rsautl -sign -in encrypted.txt -inkey sign_key.pem -in encrypted.txt -out signature.bin

3. Quelles sont les opérations réalisées pour le calcul de la signature du fichier ?
Dans une premier temps le fichier est haché pour obtenir une empreinte. Puis l'empreinte est chiffrée avec la clé privée.

4. Quelles sont les étapes nécessaires pour la vérification de la signature ?
Il faut d'abord déchiffrer la signature avec la clé publique, calculer le hach du fichier en clair et comparer les deux éléments. S'ils sont identiques c'est que la signature est valide. Avec SSL, on peut se contenter d'obtenir la clé publique et utiliser la commande adéquate :
openssl rsa -in sign_key.pem -pubout -out public.pem
openssl rsautl -verify -pubin -inkey public.pem -in signature.bin