Université Côte d'Azur — L3 Info, L3 Math-Info
Codes, cryptographie et calcul symbolique — 2025-26

TP n° 9

Hachage et signature avec OpenSSL

Nous avons déjà exploré les possibilités de la bibliothèque OpenSSL en ce qui concerne le *chiffrement à clef secrète* ou le *chiffrement à clef publique*. Aujourd'hui, nous allons utiliser ses potentialités en matière de *fonctions de hachage* et de *signature*. D'après les cours n° 8 et 9, on sait qu'il y a deux façons de signer : soit par un MAC en utilisant une *fonction de hachage* et une *clef secrète*, soit en utilisant les mécanismes de *clefs privée/publique* comme dans DSA. Une large part de ce TP sera donc consacrée à la *génération de clefs*.
Parmi les *commandes standard* d'OpenSSL rappelées ci-dessous, vous avez sans doute déjà une idée de celles que nous allons utiliser aujourd'hui :

```
  Standard commands
asn1parse         ca                ciphers           cmp
cms               crl               crl2pkcs7         dgst
dhparam           dsa               dsaparam          ec
ecparam           enc               engine            errstr
fipsinstall       gendsa            genpkey           genrsa
help              info              kdf               list
mac               nseq              ocsp              passwd
pkcs12            pkcs7             pkcs8             pkey
pkeyparam         pkeyutl           prime             rand
rehash            req               rsa               rsautl
s_client          s_server          s_time            sess_id
skeyutl           smime             speed             spkac
srp               storeutl          ts                verify
version           x509
```

On rappelle le pointeur sur le site officiel d'OpenSSL :
http://www.openssl.org

Fonctions de hachage

En anglais, les fonctions de hachage cryptographiques s'appellent *Message Digest Algorithm*, c'est donc la commande `dgst` d'OpenSSL qui sera utilisée ici.

Exercice 1)

1. Commencez par explorer les possibilités de la commande `dgst` avec son option `-help`. Listez les principales fonctions de hachage dont celles vues au Cours n° 8, notamment MD5, SHA1, SHA256, SHA3 ...
2. Calculez l'empreinte d'un fichier quelconque par les différentes fonctions de hachage. Variez les options avec `-r` et `-c`. Sachant que la sortie est codée en hexadécimal, trouvez pour chacune la taille (en binaire) des empreintes ce qui vous permettra de vérifier que certaines sont bien de la taille prévue :

MD5	SHA1	SHA256	SHA512	RIPEMD160

Quelle est la fonction de hachage utilisée par défaut ? Y a-t-il une limite à la taille du fichier passé en entrée ?

3. Considérons la fonction SHA3 aussi appelée KECCAK qui autorise une variation de la taille des empreintes. Quelles sont toutes les tailles d'empreintes possibles pour cette fonction récente ?
4. Quelle est la différence avec le résultat obtenu par les commandes `md5sum` ou bien `md5` d'UNIX et celles d'OPENSSL utilisant MD5 ? Testez également les commandes `shasum`, `sha1sum`, `sha256sum` d'UNIX, si elles sont disponibles.

Exercice 2)

1. L'usage des fonctions de hachage en mode HMAC permet de *signer* une empreinte à partir d'une clef secrète partagée préalablement entre deux personnes. Authentifiez un document quelconque avec une clef de votre choix. Quelle est la fonction de hachage utilisée par défaut ?

   Vérifiez qu'une modification de la clef (ou son absence sans utiliser HMAC) entraîne bien une modification de l'empreinte.

   Variez les fonctions de hachage utilisées par le HMAC en les passant en paramètres.
2. Mettez-vous d'accord avec votre binôme afin de partager une clef secrète K (clef d'authentification). Envoyez-lui un message M accompagné de son empreinte signée $\text{HMAC}_K(M)$, afin qu'il authentifie et vérifie l'intégrité de votre message. Faites de même avec le sien. Est-il nécessaire de se mettre d'accord sur une fonction de hachage en particulier ?

Signature RSA

Seuls des documents de taille réduite peuvent être signés en utilisant RSA. La parade consiste à calculer au préalable une *empreinte* avec la commande `dgst` et à la signer à la place d'un document de taille plus conséquente. Si cela ne suffit pas, les documents peuvent également être *compressés*.

Exercice 3)

1. Générez un couple RSA clef privée/clef publique dans deux fichiers `pem`. On rappelle qu'habituellement, une clef privée est chiffrée dès sa création. Diffusez votre clef publique, ce qui revient ici à l'envoyer à votre binôme.
2. Commencez par calculer l'empreinte quelconque d'un document puis, à l'aide de la commande standard `rsautl` [1], procédez à la signature de l'empreinte avec votre clef privée (sans l'en-tête !). Visualisez votre signature en hexadécimal. Quelle est la taille de votre signature RSA en binaire ?
3. Votre binôme doit pouvoir vérifier votre signature. Pour cela, outre votre clef publique, il doit connaître le document d'origine dont l'empreinte a été signée. Précisez-lui la fonction de hachage utilisée pour qu'il puisse recalculer l'empreinte. Enfin, envoyez-lui votre signature RSA de l'empreinte.

1. Cette commande est en passe d'être remplacée par la commande standard `pkeyutl` qui n'a pas exactement les mêmes fonctionalités.

4. Vérifiez la validité de la signature reçue de votre binôme avec l'option `-verify` de la commande `rsautl`. Vous utiliserez bien entendu sa clef publique.

   Attention ! La vérification neutralise l'effet de la signature mais c'est à vous de comparer la sortie de cette commande et l'empreinte recalculée par vos soins à partir du document transmis (avec le bon algorithme).

Exercice 4)

Il existe une autre manière de procéder, testez-la seul (sans recourir à votre binôme) mais attention de ne pas vous tromper entre vos clefs publique et privée ...

1. Une fois que le couple des clefs RSA est généré, on peut calculer directement à partir d'un fichier la signature de son empreinte. Pour cela, on utilise la commande `dgst` avec l'option `-sign` afin d'introduire sa clef privée :
   ```
   openssl dgst -sign clefPrivee.pem -out signature fichier.txt
   ```
2. Utilisez l'option `-verify` afin de vérifier si la signature est correcte. Comme n'importe quel tiers qui voudrait effectuer cette vérification, vous n'avez besoin que de la clef publique et du document d'origine. Si la signature est conforme, quel message apparaît ?

Signature DSA

Exercice 5)

1. Générez un jeu de paramètres DSA grâce à la commande `dsaparam`. Utilisez ensuite la commande `gendsa` pour en extraire la clef privée.

   On peut aussi tout engendrer d'un seul coup avec l'option `genkey` de la commande `dsaparam`.
2. Grâce à l'option `-text`, visualisez les paramètres de votre clef privée DSA. Identifiez chacun des 5 paramètres puis considérez le module q correspond au nombre d'éléments du sous-groupe. Quelle est la taille de q une fois converti de l'hexadécimal au binaire ?
3. Avec la commande `dsa`, vous pouvez à présent isoler la clef publique correspondante à votre clef privée DSA. Identifez les 4 paramètres (p, q, α, β) vus en cours.
4. Une fois le jeu de clefs DSA généré, utilisez votre clef privée DSA pour signer l'empreinte d'un document. Pour ce faire, procédez comme à l'Exercice 4 avec la commande `dgst`. Notez que c'est la clef privée elle-même qui renseigne sur l'algorithme de signature à utiliser.

   Visualisez la signature en hexadécimal pour en connaître sa taille. Quelle est la taille attendue *a priori* ?
5. Vérifiez avec votre propre clef publique la validité de la signature comme n'importe quel tiers pourrrait le faire. Quel message apparaît ?