Filières:

- *Génie des Systèmes Communicants et Sécurité Informatique (GSCSI)*
- *Ingénierie Informatique, Intelligence Artificielle et Confiance Numérique (3IACN)*

Année universitaire : 2024-2025

Semestre : S2

Elément du Module : Cryptographie

Série de Travaux Pratiques :

Sécurité et Cryptographie avec OpenSSL : Chiffrement, Signature et Gestion des Certificats

1. Introduction

OpenSSL est une **boîte à outils logicielle** open source gratuite qui offre un large éventail de fonctionnalités de cryptographie et de sécurité pour les communications et les applications informatiques. Il est largement utilisé dans le monde entier pour sécuriser les communications sur Internet, les serveurs web, les applications mobiles et bien plus encore.

Voici quelques points clés à savoir sur ***OpenSSL*** :

- **Fonctionnalités** : Il offre une large gamme de fonctionnalités liées à la cryptographie, notamment :
  - ✓ Chiffrement et déchiffrement de données symétriques et asymétriques
  - ✓ Génération et gestion de clés cryptographiques
  - ✓ Hachage et vérification d'intégrité de données
  - ✓ Authentification et validation de certificats numériques
  - Prise en charge des protocoles SSL/TLS et DTLS pour la sécurité des communications réseau
  - ✓ Bibliothèque logicielle pouvant être intégrée à d'autres applications
- **Outils en ligne de commande** : ***OpenSSL*** fournit un ensemble d'outils en ligne de commande qui permettent aux utilisateurs d'exécuter ces différentes tâches.
- **Bibliothèque logicielle** : Au cœur d'***OpenSSL*** se trouve une bibliothèque logicielle appelée ***libssl***. Cette bibliothèque est utilisée par de nombreuses applications logicielles pour ajouter des fonctionnalités de sécurité, comme par exemple le serveur web Apache avec le module ***mod_ssl***.

- **Open Source** : ***OpenSSL*** est un logiciel libre et open-source, ce qui signifie que son code source est librement disponible et peut être inspecté, modifié et redistribué.
- **Utilisations courantes d'OpenSSL** :
  - ✓ Sécurisation des serveurs web (HTTPS)
  - ✓ Protection des communications email (SMTP/POP3/IMAP)
  - ✓ Chiffrement des données stockées
  - ✓ Authentification des utilisateurs et des appareils
  - ✓ Création de signatures numériques
  - ✓ Validation de certificats SSL/TLS

Si vous débutez avec ***OpenSSL***, il existe de nombreuses ressources disponibles pour vous aider à en savoir plus :

- Le site web officiel d'***OpenSSL*** https://www.openssl.org/ propose une documentation complète, y compris des guides et des manuels.
- De nombreux tutoriels et formations en ligne peuvent vous guider pas à pas dans l'utilisation des fonctionnalités d'***OpenSSL***.

2. Etapes d'installation d'OpenSSL

Openssl présente une interface en ligne de commande. Cette utilitaire sécuritaire s'installe sur différents systèmes d'exploitation (Linux, OS, Windows,etc). Par la suite, l'installation d'***OpenSSL*** dépend de votre système d'exploitation. Voici les étapes générales pour les systèmes les plus courants :

Windows :

a) Téléchargez la dernière version d'***OpenSSL*** pour Windows à partir de https://www.openssl.org/.

b) Exécutez le fichier d'installation et suivez les instructions à l'écran.

c) Acceptez les termes de la licence et choisissez un emplacement d'installation.

d) Cliquez sur "Installer" pour démarrer l'installation.

e) Une fois l'installation terminée, vous pouvez vérifier la version d'***OpenSSL*** installée en ouvrant une invite de commande et en tapant la commande openssl version.

Linux (Ubuntu) :

a) Ouvrez une fenêtre de terminal.

b) Vérifiez si ***OpenSSL*** est déjà installé en tapant la commande openssl version.

c) Si ***OpenSSL*** n'est pas installé, installez-le en utilisant la commande suivante :

d) sudo apt install openssl

e) Vous pouvez également installer une version spécifique d'***OpenSSL*** en utilisant la commande apt-cache search openssl.

MacOS:

a) Téléchargez la dernière version d'OpenSSL pour macOS à partir de https://stackoverflow.com/questions/35129977/how-to-install-latest-version-of-openssl-mac-os-x-el-capitan.

b) Décompressez le fichier téléchargé.

c) Ouvrez une fenêtre de terminal et accédez au dossier décompressé.

d) Tapez la commande suivante pour installer ***OpenSSL*** :

e) ./configure --prefix=/usr/local

f) make

g) sudo make install

h) Vous pouvez vérifier la version d'***OpenSSL*** installée en tapant la commande openssl version.

Remarques :

- Assurez-vous de télécharger la version d'***OpenSSL*** qui correspond à votre architecture système (32 bits ou 64 bits).
- Si vous rencontrez des problèmes lors de l'installation, veuillez consulter la documentation officielle d'***OpenSSL*** https://www.openssl.org/docs/.
- Il est important de noter que les instructions ci-dessus ne sont que des indications générales. Les étapes spécifiques peuvent varier en fonction de votre système d'exploitation et de votre configuration.

3. Utilisationd'openSSl

Une fois ***OpenSSL*** installé, vous pouvez commencer par exécuter la commande :

openssl version

Cette commande affiche des informations sur la version d'OpenSSL installée sur votre système. Voici les informations généralement affichées :

- La version majeure et mineure d'OpenSSL
- La date de publication de la version
- Le numéro de build
- La plateforme pour laquelle OpenSSL a été compilé
- Les options de configuration avec lesquelles OpenSSL a été compilé
- Les modules et algorithmes disponibles

La commande **openssl ?** est une commande générique qui affiche une liste des commandes ***OpenSSL*** disponibles. Elle est utile pour obtenir une vue d'ensemble des fonctionnalités offertes par ***OpenSSL*** et pour découvrir les différentes options et arguments disponibles pour chaque commande.

Voici quelques points importants à noter à propos de la commande openssl ? **:**

- **Syntaxe :** openssl ?
- **Sortie :** La sortie de la commande openssl ? est une liste de toutes les commandes OpenSSL disponibles, avec une brève description de chacune d'elles.
- **Exemple de sortie :**

Available commands:

asn1	ASN.1 parser
ca	Certification Authority functions
ciphers	List available ciphers
cms	Cryptographic Message Syntax
dh	Diffie-Hellman key generation
dgst	Digest generation
engine	Engine management
errstr	Error string functions
genrsa	Generate an RSA ke
help	Display this help mess ge
md5	MD5 message digest
nid	Display a t of NIDs
pkcs12	PKCS#12 key and certificate management
pkcs7	PKCS#7 functions
rand	Generate random data
req	Certificate Request functions
rsa	RSA key operations
s_client	TLS/SSL client
s_server	TLS/SSL server
smime	S/MIME message handling
speed	Test the speed of various algorithms
ts	Time Stamping Protocol

verify	Verify a certificate
version	Display the OpenSSL version number
x509	X.509 certificate functions

4. Utilisations de la commande openssl ? :

- **Découvrir les fonctionnalités d'OpenSSL:** Si vous êtes nouveau sur ***OpenSSL***, la commande openssl ? est un excellent point de départ pour découvrir les différentes fonctionnalités offertes par l'outil.
- **Obtenir de l'aide sur une commande spécifique:** Si vous connaissez le nom de la commande ***OpenSSL*** que vous souhaitez utiliser, mais que vous avez besoin d'aide pour comprendre ses options et arguments, vous pouvez utiliser la commande openssl ? suivie du nom de la commande. Par exemple, pour obtenir de l'aide sur la commande openssl genrsa, vous pouvez utiliser la commande suivante :

openssl genrsa ?

Ensuite vous pouvez commencer à l'utiliser pour effectuer diverses tâches de sécurité, par exemples :

- Générer une paire de clés publique et privée
- Créer un certificat ***SSL***
- Chiffrer et déchiffrer des fichiers
- Hacher des données
- Vérifier l'intégrité de fichiers

NB : Pour en savoir plus sur l'utilisation d'***OpenSSL***, veuillez consulter la documentation officielle https://www.openssl.org/docs/ ou l'un des nombreux tutoriels disponibles en ligne.

Openssl est une boite à outils de chiffrement comportant deux bibliothèques :

• ***Libcrypto*** : implémentation des algorithmes de chiffrement

• **Libssl** : implémente les protocole ***SSL/TLS***

Syntaxe générale d'une commande d'OpenSSL

La syntaxe générale pour utiliser les outils en ligne de commande d'OpenSSL est la suivante :

openssl <commande> [options] [arguments]

- ***commande*** : qui spécifiée soit la génération des clés, soit chiffrer un message, soit signer un message ou déchiffrer un message ou autres.
- ***Options_commande*** : spécifiée le type par exemple d'algorithme de chiffrement qu'on utilise par exemple le chiffrement des en mode cbc ou aes avec une clé de **128** bits etc.
- ***Arguments_commande*** : chiffré un fichier en le plaçant dans un répertoire par exemple et ainsi de suite.

Exemple:

openssl genrsa -out key.pem 2048

Explication:

- ***openssl*** : Le nom du programme
- ***genrsa*** : La commande pour générer une paire de clés RSA
- ***-out key.pem*** : L'option pour spécifier le nom du fichier de sortie pour la clé privée
- ***2048*** : L'argument pour spécifier la taille de la clé en bits

Voici quelques exemples d'options et d'arguments courants :

- ***-help*** : Affiche l'aide pour la commande
- ***-in*** : Spécifie le fichier d'entrée
- ***-out*** : Spécifie le fichier de sortie
- ***-d*** : Déchiffre les données
- ***-e*** : Chiffre les données
- ***-v*** : Affiche des informations détaillées sur l'opération

Remarques :

- L'ordre des options et des arguments est important. Veuillez consulter la documentation de la commande pour connaître l'ordre correct.
- Vous pouvez utiliser plusieurs options et arguments en même temps.
- Les options et les arguments peuvent être abrégés.

5. Génération de clés et de certificats numériques

Si je veux générer une clé aléatoire en base **64** de taille **128** octets, alors je tape la commande :

- openssl rand -base64 128

```
c:\OpenSSL\bin>openssl rand -base64 128
F9Ieg+l7bPntETKqV8NLd8wkuKC4h4kt8KVFdXZ0YYiIYo7+D/f+Cmvl/V0K/JSy
zKeaHWoua2Akb2sXW6JuO6DMuEqzBo/crgq04K+fhxEYA+uuRYsC2QuBcKiqh+uS
Ol1kl46ZLSXcIPA4v0E/UDY17s3wTsKLDReU0/XJDkw=
```

Pour générer 128 bits de manière aléatoire et en hexadécimal on tape la commande :

- openssl rand -hex 32

```
c:\OpenSSL\bin>openssl rand -hex 32
0615d92bd3300413f53775fc016ea368007c81e67a3274312d9b9c322242e908
```

6. Codage et décodage en base 64

Table 1: The Base 64 Alphabet

Value	Encoding	Value	Encoding	Value	Encoding	Value	Encoding
0	A	17	R	34	i	51	z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	+
12	M	29	d	46	u	63	/
13	N	30	e	47	v		
14	O	31	f	48	w	(pad)	=
15	P	32	g	49	x		
16	Q	33	h	50	y		

- Je crée un fichier **textOriginal.txt** et je le place dans le dossier **tp_openssl**
- Je me place dans le dossier et je lance ***openssl***.
- Pour coder le fichier en base 64 on utilise la commande :

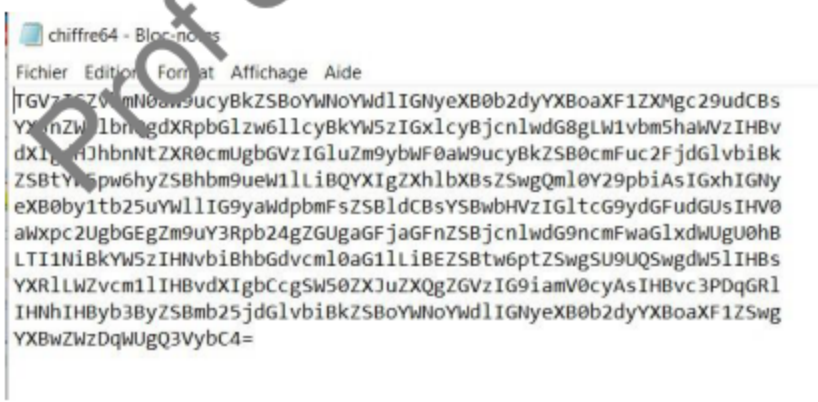
openssl enc -a -in fichierOriginal.txt -out chiffre64

-a : pour spécifier la base 64

-in : qu'est ce je veux chiffrer ou coder c.à.d pour rediriger la commande vers l'objectif c.à.d le fichier qu'on souhaite coder ou chiffrer

-out pour rediriger la sortie

Taper la commande exit pour revenir au répertoire de travail et vous taper la com- mande dir. Vous allez remarquer qu'un fichier de nom chiffre64 est créé. Si on l'ouvre, on remarque que le fichier est crypté en base 64



Pour décoder ce fichier on tape la commande :

- **openssl enc -d -a -in chiffre64**

```
C:\Users\hraou\Desktop\tp_openssl>openssl enc -d -a -in chiffre64
Les fonctions de hachage cryptographiques sont largement utilis├®es dans les crypto -monnaies pour trans
mettre les informations de transaction de mani├¿re anonyme. Par exemple, Bitcoin , la crypto-monnaie ori
ginale et la plus importante, utilise la fonction de hachage cryptographique SHA-256 dans son algorithme
. De m├¬me, IOTA, une plate-forme pour l' Internet des objets , poss├¿de sa propre fonction de hachage c
ryptographique, appel├®e Curl.
```

Chiffrement symétrique

6.1. Les principales commandes pour le cryptage symétrique

OpenSSL propose plusieurs commandes pour le cryptage symétrique. Voici les plus courantes :

Chiffrement et déchiffrement de fichiers :

- ***openssl enc*** : Chiffre un fichier
- ***openssl dec*** : Déchiffre un fichier

Exemples :

- **Chiffrer un fichier avec l'algorithme AES-256 et le mot de passe "*secret*"**

openssl enc -aes-256-cbc -in fichier.txt -out fichier.enc -pass pass:secret

- **Déchiffrer le fichier chiffré avec le mot de passe "*secret*"**

openssl dec -aes-256-cbc -in fichier.enc -out fichier.txt -pass pass:secret

Génération de clés symétriques :

- ***openssl genrsa*** : Génère une paire de clés RSA (asymétrique)
- ***openssl rand*** : Génère une clé symétrique aléatoire

Exemples :

- **Générer une clé symétrique AES-256 de 32 octets**

openssl rand -base64 32 > key.txt

- **Générer une paire de clés RSA de 2048 bits**

openssl genrsa -out key.pem 2048

Remarques :

- Il est important de choisir un algorithme de cryptage et une taille de clé adaptés à vos besoins de sécurité.
- Le mot de passe utilisé pour le cryptage doit être fort et difficile à deviner.
- Il est important de stocker les clés symétriques de manière sécurisée

Avant de chiffrer avec les algorithmes de chiffrement symétrique qui sont à la base d'une seule clef pour chiffrer et déchiffrer un message, nous allons générer un mot de passe aléatoire de

8 octets en format hexadécimal en utilisant OpenSSL. Pour cela, on utilise la commande suivante :

- openssl rand -hex 8 : clé générée est : 71dc0c3629b56c8c

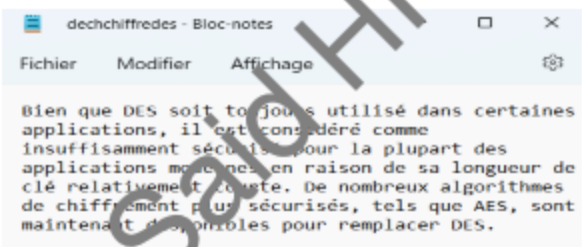
```
c:\OpenSSL\bin>openssl rand -hex 8
71dc0c3629b56c8c
```

Pour chiffrer un message avec l'algorithme de chiffrement DES avec la clé bonjour, on utilise la commande :

- **openssl enc -des-ede-cbc -in fichierOriginal.txt -out chiffredes**

Pour déchiffrer ce fichier « ***chiffredes*** » on utilise la commande :

- **openssl enc -d -des-ede-cbc -in chiffredes -out dechchiffredes**



Pour chiffrer/déchiffrer avec AES ayant une clé de 256 et implémenté en mode cbc on utilise la commande :

- **openssl enc -aes-256-cbc -in fichierOriginal.txt -out cryptAES-256**

```
c:\OpenSSL\bin>openssl enc -aes-256-cbc -in fichierOriginal.txt -out cryptAES-256
enter AES-256-CBC encryption password:
Verifying - enter AES-256-CBC encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
```

- **openssl enc -d -aes-256-cbc -in chiffreaes256**

```
c:\OpenSSL\bin>openssl enc -d -aes-256-cbc -in cryptAES-256 -out dechchiffreAES256
enter AES-256-CBC decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
```

On peut également chiffrer un fichier avec un algorithme de chiffrement symétrique en spécifiant le nombre d'itérations qui précise le nombre de clés dérivées.

- **openssl enc -aes-256-cbc -salt –in fichierOriginal.txt -out fichierchiffreAESIter.enc -iter 10000**
- **openssl enc -aes-256-cbc -d -in fichierchiffreAESIter.enc -out fichierDecriter.txt -iter 10000**

3. Chiffrement asymétrique

Pour chiffrer avec les algorithmes de chiffrements asymétriques, on utilise deux clés ; une privée et une autre publique. Donc, on génère premièrement la clé privée par la commande :

- **openssl genrsa -out clePr.pem 1024**
- **openssl genrsa -out clefPrive.key 2048**

Seulement, vous remarquez que nous avons sauvegardé la clé privée sur notre ordinateur, ce qui peut entrainer une usurpation de la clé. La solution c'est générer la clé privée toute en la chiffrant. Pour cela on utilise la commande suivante :

- **openssl enc -aes-256-cbc -in clefPrive.key -out clefPrive.key.enc**

Au niveau de cette commande on utilise un mot de passe pour chiffrer et protéger ainsi la clé privée générée.

De plus au niveau de **RSA**, on calcul ***N=p*q*** qu'on appelle au niveau de openssl par modulus. Pour faire ce calcul, on utilise la commande :

- **openssl rsa -modulus -in clefPrive.key**

Si je ne veux pas afficher la clé privée, je tape la commande :

- **openssl rsa -noout -modulus -in clefPrive.key**

Après, on va générer une clé publique à partir de la clé privée. Pour cela, on utilise la commande :

- **openssl rsa -pubout -in clefPrive.key -out clefPublique.key**

Une fois que cela est bien établit, je peux chiffrer et déchiffrer. Pour cela on utilise la commande

Mot de passe : 71dc0c3629b56c8c

Commande pour générer une clé privée et la chiffré avec **AES128**

- **openssl genrsa -aes128 -out alice_private.pem 1024**

Pour voir les détails de la clé, taper la commande :

- **openssl rsa -in alice_private.pem -noout -text**

Après avoir générer la clé privée et la sécuriser en la chiffrant avec un algorithme de chiffrement symétrique, il faut générer la clé Publique associée à la clé privée. Pour cela taper la commande:

- **openssl rsa -in alice_private.pem -pubout > alice_public.pem**

Pour afficher les détails de la clé publique, taper la commande :

- **openssl rsa -in alice_public.pem -pubin -text -noout**

Pour chiffrer un message avec la clé publique, utiliser la commande :

- **openssl pkeyutl -encrypt -inkey alice_public.pem -pubin -in messageSecret.txt -out fichier_secret.enc**

Pour déchiffrer, il faut utiliser la clé privée qui correspond à la clé publique. Pour ce faire, on utilise la commande :

- **openssl rsautl -decrypt -inkey clePr.pem -in chiffrersa**

Fonction de hachage

Les fonctions de hachage sont utilisées pour le contrôle d'intégrité. On peut également utiliser une fonction de hachage avec clé (***MAC***) pour éviter toute attaques de l'homme de milieu(***MIM***).

Hachage de données :

- openssl dgst : Calcule le hachage d'un fichier ou d'une chaîne de caractères

Exemple :

Calculer le hachage SHA-256 du fichier "fichier.txt"

openssl dgst -sha256 fichier.txt

Donc pour hacher sous ***openssl***, on utilise la commande :

- **openssl dgst textOriginal.txt**

On remarque que cette commande à utiliser par défaut l'algorithme SHA256 pour calculer le condensé de notre message.

On peut spécifier l'algorithme, Pour cela on utilise la commande :

- **openssl dgst -md5 textOriginal.txt**
- **openssl dgst -sha1 textOriginal.txt**
- **openssl dgst -sha512 textOriginal.txt**

On peut également utiliser la ***HMAC***, c'est une fonction de hachage avec clé. Pour cela j'ajoute une clé.

- **openssl dgst -sha1 -hmac "salam" textOriginal.txt**

Signature électronique

On va premièrement hacher le message(document), puis on va chiffrer l'empreinte avec une clé privée (clé d'Alice). On obtient la signature numérique. Après, on va transmettre le

message et la signature à Bob. Bob reçoit le couple (message, signature). On recalcule le hash du message reçu et on déchiffre l'empreinte de document reçu avec la clé publique d'Alice et on compare avec la signature reçue.

Pour réaliser tout ça sous ***openssl***, on suit les étapes suivantes :

On utilise l'algorithme **DSA**(DSA : Digital Signature Algorithm, est un algorithme de signature numérique standardisé par le NIST aux États-Unis) pour générer les clés, nous tapons la commande :

- **openssl dsaparam -genkey -out maclePr.pem 1024**

Après on va créer la clé publique associée à la clé privée qu'on a générée. Pour cela, on utilise la commande ci-dessous :

- **openssl dsa -pubout -in maclePr.pem -out pubCle.pem**

Une fois ceci fait, je peux donc signer :

- **openssl dgst -sha256 -sign maclePr.pem -out Signature.sign textOriginal.txt**

On va calculer l'empreinte avec SHA256 en la signant avec la clé privée que j'ai générée. Le résultat c'est **Signature.sign**. Finalement ce que je veux signer c'est **textOriginal.txt**.

Pour vérifier la signature. On a besoin de la fonction de hachage, de la clé publique correspondant à la clé privée qu'on a utilisée lors de la signature et de document qu'on a signé. Pour faire la vérification, on tape la commande

- **openssl dgst -sha256 -verify pubCle.pem -signature Signature.sign textOriginal.txt**

Certificat Numérique

A. **Générer une clé privée RSA :**
   - **openssl genrsa -out clePriveCertificat.key 2048**
B. **Créer une demande de signature de certificat (CSR) :**
   - **openssl req -new -key clefPriveCertificat.key -out request.csr**

Vous serez invité à fournir des informations sur votre organisation, telles que le nom de l'entreprise, le pays, la ville, etc.

C. **Envoyer le CSR à une autorité de certification (CA) pour signature ou auto-signer le certificat:**
   i. **Pour auto-signer le certificat :**

- **openssl x509 -req -days 365 -in request.csr -signkey private.key -out certificate.crt**

   ii. **Pour envoyer le CSR à une autorité de certification tierce (CA), vous devez fournir le CSR au CA, qui vous enverra un certificat signé correspondant. Vous devrez ensuite fusionner votre certificat avec le certificat du CA. Par exemple:**

- **openssl x509 -req -days 365 -in request.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out certificate.crt**

où "ca.crt" et "ca.key" sont les fichiers du certificat et de la clé privée de l'autorité de certification, respectivement.

iii. Facultatif : vérifier le contenu du certificat :

- **openssl x509 -text -noout -in certificate.crt**

Ces étapes vous permettent de générer un certificat numérique avec une clé privée et une demande de signature de certificat. Si vous avez besoin d'un certificat signé par une autorité de certification tierce, vous devrez envoyer le CSR à une autorité de certification pour signature.