TADJER Badr | TRAN Leo | ARRADI Naoufal
M1-APP-LS1

TP3

CHIFFREMENT ASYMÉTRIQUE

1. **Quelle est la version du paquet OpenSSL de votre système ? Où se trouvent les certificats des différents CA (Certification Authority) sur votre environnement ?**

Sur MacOS, OpenSSL se nomme LibreSSL

```
➜  TP3 openssl version
LibreSSL 2.8.3
```

Les certificats se situent dans le répertoire /etc/ssl/certs

2. **Chiffrement asymétrique avec RSA :**
    a. **Générer un couple de clés (publique, privée) pour Alice et sauvegardez-le dans un fichier cle.pem. Quel est le codage utilisé dans ce fichier ?**

openssl genrsa -out cle.pem 2048
Le fichier est codé en base64

    b. **Comment extraire la clé publique sauvegardée dans cle.pem afin de la stocker dans pub.pem ?**

openssl rsa -pubout -in cle.pem -out pub.pem

    c. **Bob, qui est en possession du fichier pub.pem, veut chiffrer un message secret et l'envoyer à Alice en utilisant la clé publique d'Alice. Il envoie à Alice le message chiffré. Quelle est la commande à utiliser ?**

openssl rsautl -encrypt -in message.txt -out encrypted_message.txt -inkey pub.pem -pubin

    d. **Quelle est la commande qui permet à Alice de déchiffrer le secret en utilisant sa clé privée ?**

openssl rsautl -decrypt -in encrypted_message.txt -inkey cle.pem

LES CERTIFICATS ÉLECTRONIQUES

1) C'est quoi un certificat électronique et c'est quoi son utilité ?

Un certificat électronique est un certificat qui certifie qu'une clé publique appartient bien à la personne qui prétend être le propriétaire légitime de cette clé. L'utilité d'un certificat électronique, est d'authentifier une personne et peut également servir pour chiffrer les échanges.

2) Récupération, visualisation et transcodage de certificats

a) Saisir le script get-cert.sh suivant :

```
➜  TP3 cat get-cert.sh
#!/bin/sh
# usage: get-cert.sh remote.host.name [port]
REMHOST=$1
REMPORT=${2:-443}
echo |\
openssl s_client -connect ${REMHOST}:${REMPORT} 2>&1 |\
sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p'
```

b) Expliquer le rôle de la commande s_client, de la commande openssl et de la commande sed.

s_client : connexion à distance vers un hôte en utilisant SSL/TLS.

openssl : boîte à outils pour la cryptographie et les communications sécurisée.

sed : la commande sed permet le traitement de fichier,
Ex : supprimer des lignes, afficher le contenu, afficher un certain contenu du fichier, etc…

c) Exécuter ce script afin de récupérer le certificat d'un site Internet de votre choix, puis visualiser ce certificat (qui est au format PEM).

```
➜  TP3 ./get-cert.sh www.efrei.fr > efrei.pem
➜  TP3 cat efrei.pem
-----BEGIN CERTIFICATE-----
MIICwjCCAaqgAwIBAgIJAIfz0BMAhhEBMA0GCSqGSIb3DQEBCwUAMBkxFzAVBgNV
BAMMDmRuMTEwNC5taXMub3ZoMB4XDTE5MDMxNTEzNTgyMFoXDTI5MDMxMjEzNTgy
MFowGTEXMBUGA1UEAwwOZG4xMTA0Lm1pcy5vdmgwggEiMA0GCSqGSIb3DQEBAQUA
A4IBDwAwggEKAoIBAQDMCxJPdOJ31KeNHOg25wqobnqcaTLmy26rdc4T6W/T3Z7A
nXGr2TYPXmSHeS8PpZUqRGV6Is1oJ4CWRtMf/MyQnbY+iYHkyWkToMekJdJQVlaT
j/YomUv7hG5RMwdfU/wS3/3GxZ8iKS+GYWxddS6JnT4k5OqVCvaA5NgzDasVcwWD
JZXXcEizhOWRG+0vbTU0MJ84eZeGSAdcLGT7iyWD07oPCRtuB9afUAJ4Ud+YRhXt
QbrTyzZYmcqahQkeDaAAtAjGQ47UqPoauqvkWolJ7n067yHwhRsGyob8PWEUQo/G
m3K7ieKSqsu6ORXe/cpZY1Z1oX1E/5sSbdUaHZE5AgMBAAGjDTALMAkGA1UdEwQC
MAAwDQYJKoZIhvcNAQELBQADggEBAD+S5VLutj7kYsZeCvKlItoEj7NdocYCY0MB
1N17cmIsTsMKhOiXHwDl+qFR/UUCMce5BycJRshYLLkLME2Q4FrqIx6RFRw7Nwud
EeTM3OmJibi+/Qt4k6PPc6wqHIXHYojLDC7KZ0S+mwqd5AOhTXTSBIKYDPbN/uWZ
th2xif+FWM0ONznNBZZdjZ+EYczdOP6P96d2KsOPzJmILUc3GXqFYRvL+39tu9dp
CFKCb6U+7Fln4bzT9j3PHFdCUPBAGsDUZ02yhpK3LWLgAfc+5iclnznWtNUkIwAj
MWQMRO2oNLTIa8YyShPEfP8OIOKf0EFxjzAJBixP1IqCv1gli1A=
-----END CERTIFICATE-----
```

d) Convertir ce certificat du format PEM au format DER. C'est quoi la différence entre ces deux formats ? quels sont les autres formats possibles ?

openssl x509 -in efrei.pem -out efrei.der -outform DER
Les fichiers au format PEM comportent un entête et un pied de page, les données entres sont en base 64.
Les fichiers au format DER, contiennent des certificats codés en binaire.
Les autres formats sont PEM, DER, P7B, PFX.

3) Création d'un certificat x509 auto-signé : créer un couple de clés publique/privée, puis créer une requête de certificat. Utiliser ensuite ces deux informations pour générer un certificat auto signé.

Créer un couple de clés publique/privée : (cle.pem)

openssl genrsa -out cle.pem 2048

Requête de certificat et certificat auto-signé : (ca.crt)

openssl req -new -x509 -key cle.pem -out ca.crt -days 1095

TESTER LES LIAISONS TLS/SSL

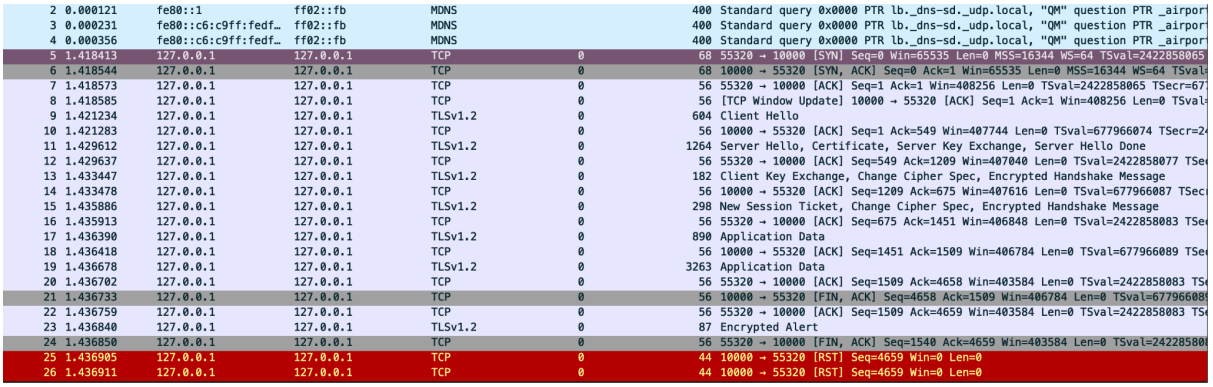
Le paquet OpenSSL embarque avec lui un serveur SSL. Nous allons l'utiliser pour comprendre le fonctionnement du protocole SSL.

1) **Commencer par lancer le serveur SSL d'OpenSSL (commande s_server) sur le port 10000 en utilisant le certificat autosignée que vous avez genere dans l'etape precedente ainsi que la cle**

openssl s_server -key cle.pem -cert ca.crt -accept 10000 -www

```
badr@MacBook-Pro-de-Badr.local:/Users/badr/Documents/LYCE
E_IUT_EI/3_EFREI/M1/Semestre_8/Reseaux/TP/TP3 git:(main*)
 $ openssl s_server -key cle.pem -cert ca.crt -accept 100
00 -www

Using auto DH parameters
Using default temp ECDH parameters
ACCEPT
```

2) Lancer Wireshark pour capturer le trafic, puis tester l'accès à ce serveur à partir d'un navigateur web graphique ou bien textuel (lynx). Analyser tout l'échange SSL et expliquer toutes les étapes opérationnelles du protocole SSL.



2	0.000121	fe80::1	ff02::fb	MDNS	400	Standard query 0x0000 PTR lb._dns-sd._udp.local, "QM" question PTR _airport
3	0.000231	fe80::c6:c9ff:fedf…	ff02::fb	MDNS	400	Standard query 0x0000 PTR lb._dns-sd._udp.local, "QM" question PTR _airport
4	0.000356	fe80::c6:c9ff:fedf…	ff02::fb	MDNS	400	Standard query 0x0000 PTR lb._dns-sd._udp.local, "QM" question PTR _airport
5	1.418413	127.0.0.1	127.0.0.1	TCP	0	68 55320 → 10000 [SYN] Seq=0 Win=65535 Len=0 MSS=16344 WS=64 TSval=2422858065
6	1.418544	127.0.0.1	127.0.0.1	TCP	0	68 10000 → 55320 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=16344 WS=64 TSval=
7	1.418573	127.0.0.1	127.0.0.1	TCP	0	56 55320 → 10000 [ACK] Seq=1 Ack=1 Win=408256 Len=0 TSval=2422858065 TSecr=677
8	1.418585	127.0.0.1	127.0.0.1	TCP	0	56 [TCP Window Update] 10000 → 55320 [ACK] Seq=1 Ack=1 Win=408256 Len=0 TSval=
9	1.421234	127.0.0.1	127.0.0.1	TLSv1.2	0	604 Client Hello
10	1.421283	127.0.0.1	127.0.0.1	TCP	0	56 10000 → 55320 [ACK] Seq=1 Ack=549 Win=407744 Len=0 TSval=677966074 TSecr=24
11	1.429612	127.0.0.1	127.0.0.1	TLSv1.2	0	1264 Server Hello, Certificate, Server Key Exchange, Server Hello Done
12	1.429637	127.0.0.1	127.0.0.1	TCP	0	56 55320 → 10000 [ACK] Seq=549 Ack=1209 Win=407040 Len=0 TSval=2422858077 TSe
13	1.433447	127.0.0.1	127.0.0.1	TLSv1.2	0	182 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
14	1.433478	127.0.0.1	127.0.0.1	TCP	0	56 10000 → 55320 [ACK] Seq=1209 Ack=675 Win=407616 Len=0 TSval=677966087 TSec
15	1.435886	127.0.0.1	127.0.0.1	TLSv1.2	0	298 New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
16	1.435913	127.0.0.1	127.0.0.1	TCP	0	56 55320 → 10000 [ACK] Seq=675 Ack=1451 Win=406848 Len=0 TSval=2422858083 TSe
17	1.436390	127.0.0.1	127.0.0.1	TLSv1.2	0	890 Application Data
18	1.436418	127.0.0.1	127.0.0.1	TCP	0	56 10000 → 55320 [ACK] Seq=1451 Ack=1509 Win=406784 Len=0 TSval=677966089 TSe
19	1.436678	127.0.0.1	127.0.0.1	TLSv1.2	0	3263 Application Data
20	1.436702	127.0.0.1	127.0.0.1	TCP	0	56 55320 → 10000 [ACK] Seq=1509 Ack=4658 Win=403584 Len=0 TSval=2422858083 TS
21	1.436733	127.0.0.1	127.0.0.1	TCP	0	56 10000 → 55320 [FIN, ACK] Seq=4658 Ack=1509 Win=406784 Len=0 TSval=67796608
22	1.436759	127.0.0.1	127.0.0.1	TCP	0	56 55320 → 10000 [ACK] Seq=1509 Ack=4659 Win=403584 Len=0 TSval=2422858083 TS
23	1.436840	127.0.0.1	127.0.0.1	TLSv1.2	0	87 Encrypted Alert
24	1.436850	127.0.0.1	127.0.0.1	TCP	0	56 55320 → 10000 [FIN, ACK] Seq=1540 Ack=4659 Win=403584 Len=0 TSval=24228580
25	1.436905	127.0.0.1	127.0.0.1	TCP	0	44 10000 → 55320 [RST] Seq=4659 Win=0 Len=0
26	1.436911	127.0.0.1	127.0.0.1	TCP	0	44 10000 → 55320 [RST] Seq=4659 Win=0 Len=0

```
s_server -key cle.pem -cert ca.crt -accept 10000 -www
Secure Renegotiation IS supported
Ciphers supported in s_server binary
TLSv1/SSLv3:ECDHE-ECDSA-CHACHA20-POLY1305TLSv1/SSLv3:ECDHE-RSA-CHACHA20-POLY1305
TLSv1/SSLv3:DHE-RSA-CHACHA20-POLY1305TLSv1/SSLv3:ECDHE-RSA-AES256-GCM-SHA384
TLSv1/SSLv3:ECDHE-ECDSA-AES256-GCM-SHA384TLSv1/SSLv3:ECDHE-RSA-AES256-SHA384
TLSv1/SSLv3:ECDHE-ECDSA-AES256-SHA384TLSv1/SSLv3:ECDHE-RSA-AES256-SHA
TLSv1/SSLv3:ECDHE-ECDSA-AES256-SHA    TLSv1/SSLv3:DHE-RSA-AES256-GCM-SHA384
TLSv1/SSLv3:DHE-RSA-AES256-SHA256     TLSv1/SSLv3:DHE-RSA-AES256-SHA
TLSv1/SSLv3:GOST2012256-GOST89-GOST89TLSv1/SSLv3:DHE-RSA-CAMELLIA256-SHA256
TLSv1/SSLv3:DHE-RSA-CAMELLIA256-SHA   TLSv1/SSLv3:GOST2001-GOST89-GOST89
TLSv1/SSLv3:AES256-GCM-SHA384         TLSv1/SSLv3:AES256-SHA256
TLSv1/SSLv3:AES256-SHA                TLSv1/SSLv3:CAMELLIA256-SHA256
TLSv1/SSLv3:CAMELLIA256-SHA           TLSv1/SSLv3:ECDHE-RSA-AES128-GCM-SHA256
TLSv1/SSLv3:ECDHE-ECDSA-AES128-GCM-SHA256TLSv1/SSLv3:ECDHE-RSA-AES128-SHA256
TLSv1/SSLv3:ECDHE-ECDSA-AES128-SHA256TLSv1/SSLv3:ECDHE-RSA-AES128-SHA
TLSv1/SSLv3:ECDHE-ECDSA-AES128-SHA    TLSv1/SSLv3:DHE-RSA-AES128-GCM-SHA256
TLSv1/SSLv3:DHE-RSA-AES128-SHA256     TLSv1/SSLv3:DHE-RSA-AES128-SHA
TLSv1/SSLv3:DHE-RSA-CAMELLIA128-SHA256TLSv1/SSLv3:DHE-RSA-CAMELLIA128-SHA
TLSv1/SSLv3:AES128-GCM-SHA256         TLSv1/SSLv3:AES128-SHA256
TLSv1/SSLv3:AES128-SHA                TLSv1/SSLv3:CAMELLIA128-SHA256
TLSv1/SSLv3:CAMELLIA128-SHA           TLSv1/SSLv3:ECDHE-RSA-RC4-SHA
TLSv1/SSLv3:ECDHE-ECDSA-RC4-SHA       TLSv1/SSLv3:RC4-SHA
TLSv1/SSLv3:RC4-MD5                   TLSv1/SSLv3:ECDHE-RSA-DES-CBC3-SHA
TLSv1/SSLv3:ECDHE-ECDSA-DES-CBC3-SHA TLSv1/SSLv3:EDH-RSA-DES-CBC3-SHA
TLSv1/SSLv3:DES-CBC3-SHA
---
Ciphers common between both SSL end points:
ECDHE-ECDSA-AES128-GCM-SHA256 ECDHE-RSA-AES128-GCM-SHA256 ECDHE-ECDSA-CHACHA20-POLY1305
ECDHE-RSA-CHACHA20-POLY1305 ECDHE-ECDSA-AES256-GCM-SHA384 ECDHE-RSA-AES256-GCM-SHA384
ECDHE-ECDSA-AES256-SHA      ECDHE-ECDSA-AES128-SHA      ECDHE-RSA-AES128-SHA
ECDHE-RSA-AES256-SHA        AES128-GCM-SHA256           AES256-GCM-SHA384
AES128-SHA                  AES256-SHA
---
New, TLSv1/SSLv3, Cipher is ECDHE-RSA-AES128-GCM-SHA256
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-RSA-AES128-GCM-SHA256
    Session-ID:
    Session-ID-ctx: 01000000
    Master-Key: 00D5CCD528AB0A205A1C9B7A271039F074B278FB9F61887796CB844F1D5014A4753305BB08577AFF7BA3CD0DC10243AE
    Start Time: 1647588799
    Timeout   : 7200 (sec)
    Verify return code: 0 (ok)
```

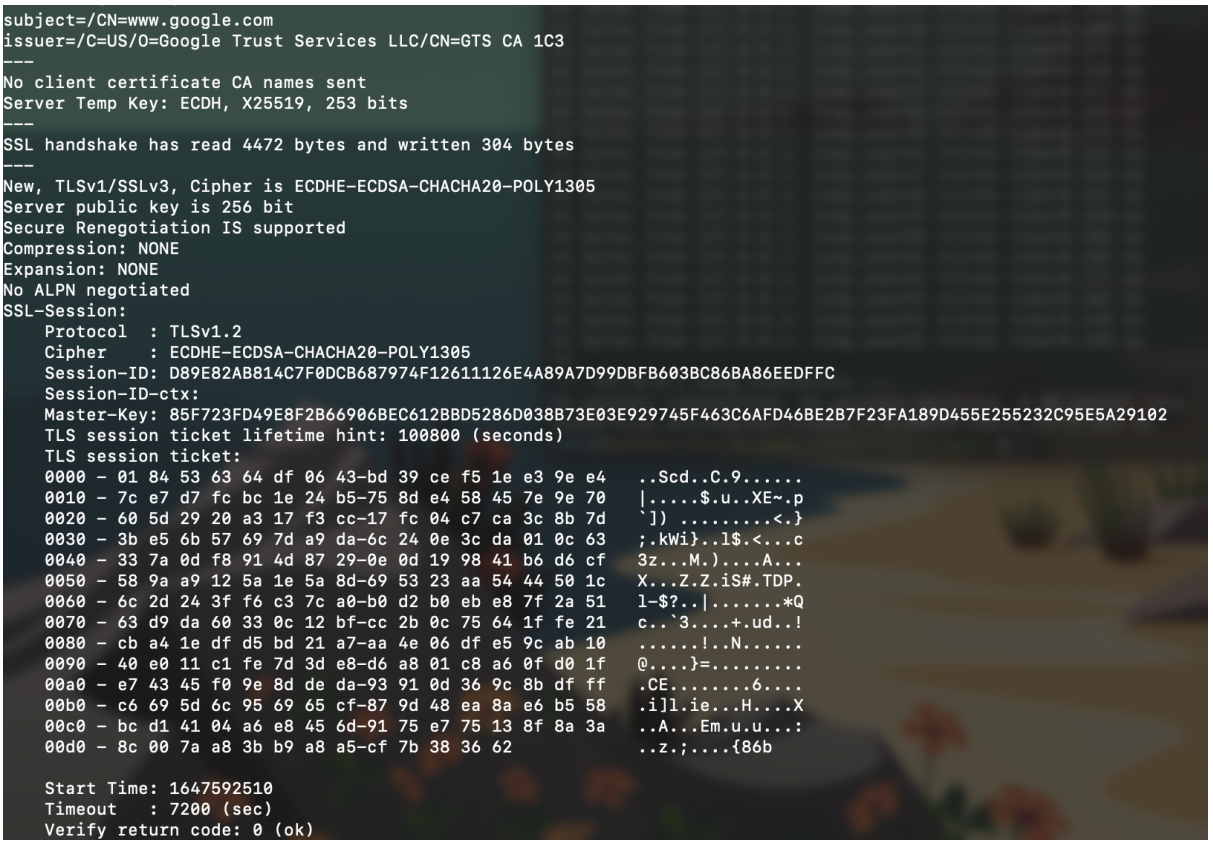
3) **C'est quoi la notion de suite de chiffrement présente dans le message Client Hello ? Est-ce que l'ordre des propositions de ces suites de chiffrement est important ? Afficher la liste des suites de chiffrement d'OpenSSL.**

```
Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
Cipher Suite: TLS_CHACHA20_POLY1305_SHA256 (0x1303)
Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca9)
Cipher Suite: TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca8)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
Cipher Suite: TLS_RSA_WITH_AES_128_GCM_SHA256 (0x009c)
Cipher Suite: TLS_RSA_WITH_AES_256_GCM_SHA384 (0x009d)
Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)
Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)
```

4) **Afficher la trace d'une session SSL avec le client de test incorporé dans openssl. C'est quoi le premier message chiffré ? à quoi sert-il ? (Faire une petite recherche pour répondre à cette question).**

5) **En utilisant OpenSSL, accéder à un site SSL sur Internet. Quelle est la suite de chiffrement choisie ?**

openssl s_client -crlf -connect www.google.com:443 -servername www.google.com



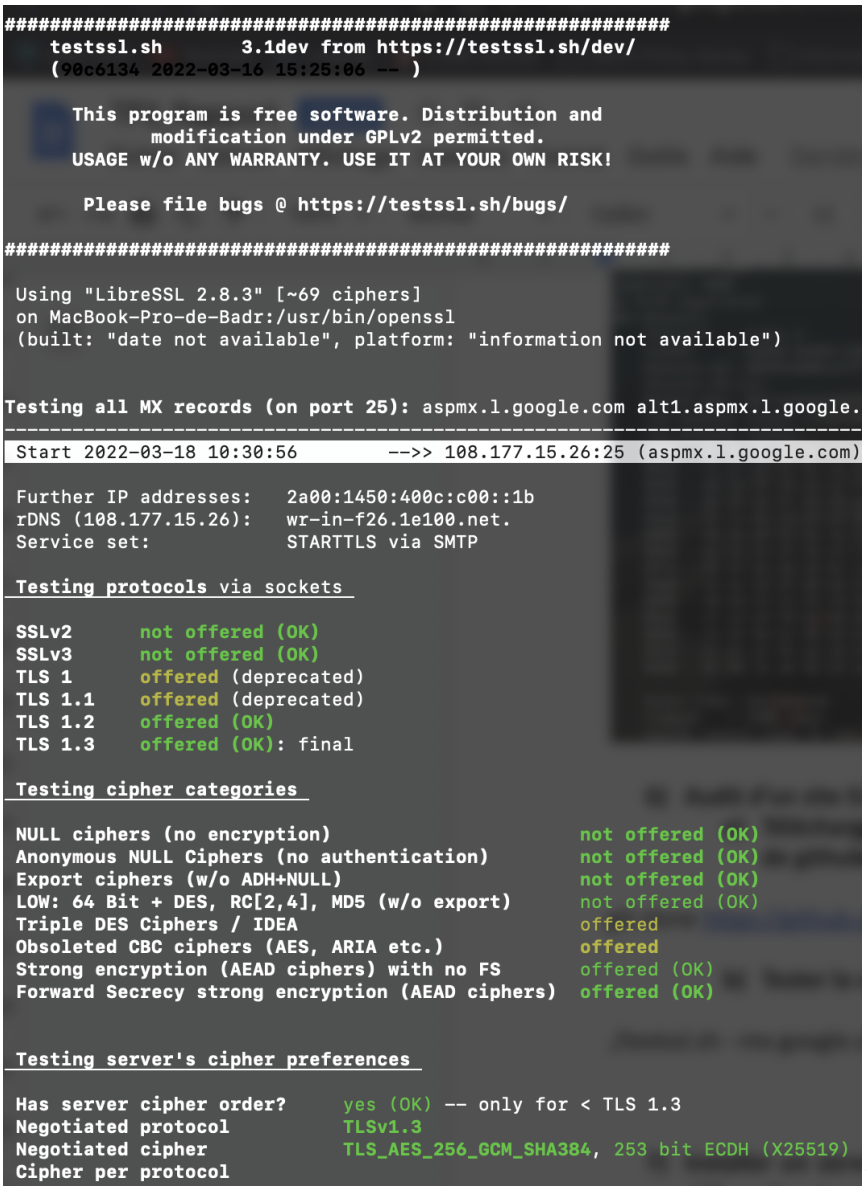
```
subject=/CN=www.google.com
issuer=/C=US/O=Google Trust Services LLC/CN=GTS CA 1C3
---
No client certificate CA names sent
Server Temp Key: ECDH, X25519, 253 bits
---
SSL handshake has read 4472 bytes and written 304 bytes
---
New, TLSv1/SSLv3, Cipher is ECDHE-ECDSA-CHACHA20-POLY1305
Server public key is 256 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-ECDSA-CHACHA20-POLY1305
    Session-ID: D89E82AB814C7F0DCB687974F12611126E4A89A7D99DBFB603BC86BA86EEDFFC
    Session-ID-ctx:
    Master-Key: 85F723FD49E8F2B66906BEC612BBD5286D038B73E03E929745F463C6AFD46BE2B7F23FA189D455E255232C95E5A29102
    TLS session ticket lifetime hint: 100800 (seconds)
    TLS session ticket:
    0000 - 01 84 53 63 64 df 06 43-bd 39 ce f5 1e e3 9e e4   ..Scd..C.9......
    0010 - 7c e7 d7 fc bc 1e 24 b5-75 8d e4 58 45 7e 9e 70   |.....$.u..XE~.p
    0020 - 60 5d 29 20 a3 17 f3 cc-17 fc 04 c7 ca 3c 8b 7d   `]) .........<.}
    0030 - 3b e5 6b 57 69 7d a9 da-6c 24 0e 3c da 01 0c 63   ;.kWi}..l$.<...c
    0040 - 33 7a 0d f8 91 4d 87 29-0e 0d 19 98 41 b6 d6 cf   3z...M.)....A...
    0050 - 58 9a a9 12 5a 1e 5a 8d-69 53 23 aa 54 44 50 1c   X...Z.Z.iS#.TDP.
    0060 - 6c 2d 24 3f f6 c3 7c a0-b0 d2 b0 eb e8 7f 2a 51   l-$?..|.......*Q
    0070 - 63 d9 da 60 33 0c 12 bf-cc 2b 0c 75 64 1f fe 21   c..`3....+.ud..!
    0080 - cb a4 1e df d5 bd 21 a7-aa 4e 06 df e5 9c ab 10   ......!..N......
    0090 - 40 e0 11 c1 fe 7d 3d e8-d6 a8 01 c8 a6 0f d0 1f   @....}=.........
    00a0 - e7 43 45 f0 9e 8d de da-93 91 0d 36 9c 8b df ff   .CE........6....
    00b0 - c6 69 5d 6c 95 69 65 cf-87 9d 48 ea 8a e6 b5 58   .i]l.ie...H....X
    00c0 - bc d1 41 04 a6 e8 45 6d-91 75 e7 75 13 8f 8a 3a   ..A...Em.u.u...:
    00d0 - 8c 00 7a a8 3b b9 a8 a5-cf 7b 38 36 62            ..z.;....{86b

    Start Time: 1647592510
    Timeout   : 7200 (sec)
    Verify return code: 0 (ok)
```

6) **Audit d'un site SSL :**
    a) **Télécharger le script testssl.sh. De préférence, faites-le en le clonant à partir de github comme c'est indiqué sur le site de téléchargement.**

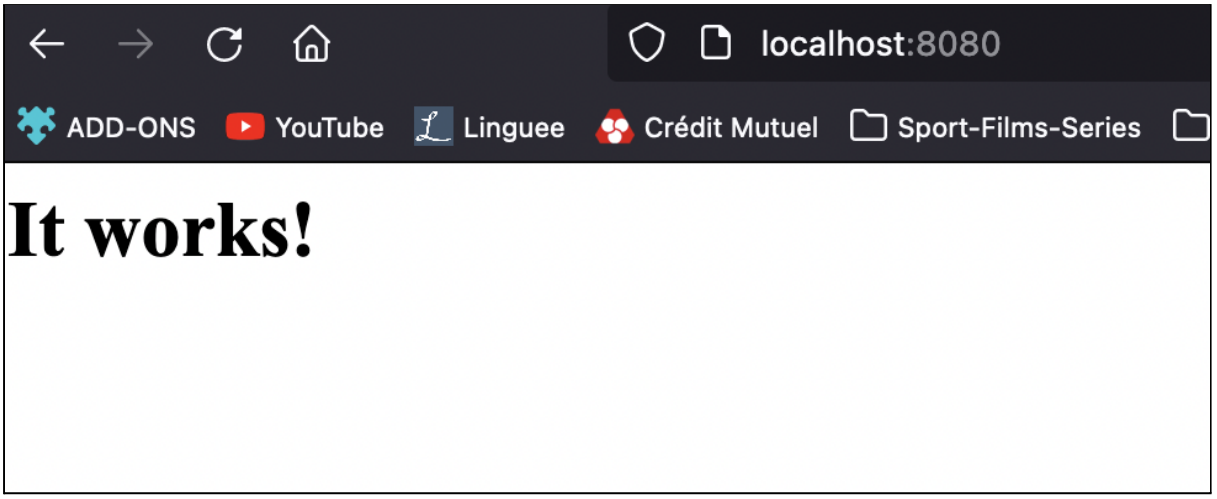
git clone https://github.com/drwetter/testssl.sh.git

b) Tester la robustesse d'un serveur web en utilisant ce script

./testssl.sh --mx google.com



7) **Installer un serveur Apache, Activer le module ssl ainsi que le site virtuel ssl par défaut. Ensuite tester sa robustesse avec le script testssl.sh. Faites les changements nécessaires pour durcir la configuration de votre site.**



./testssl.sh --mx localhost:8080