

# TP Cryptographie 2ème partie

Guillaume Bienkowski — Brincube

# TP

1. Fonctions de hachage
2. Signatures digitales
3. Certificats

# Récupération du document

Il vous faut ce PDF ouvert pour profiter des liens vers le Web.

Rendez-vous ici: <https://masterind4.github.io>

Et cliquez sur le lien de téléchargement de ce PDF:

**TP**

PDF du TP

[À récupérer ici](#)



# Méthodologie

Le TP se fera sur ordinateur avec un terminal et un navigateur.

Les réponses seront à fournir dans un formulaire situé ici:

<https://forms.gle/p4RirfG6Vas9Aaes6>

## **BIEN REMPLIR VOTRE NOM ET PRENOM**

Il faudra aussi y joindre les fichiers des certificats pour MQTT que vous générerez en fin de session.

# Fonctions de hachage

## Exercice 1

Rendez-vous sur <https://prometheus.io/download/>

Téléchargez la **version linux** `amd64` de l'outil `prometheus`

1. Vérifiez à l'aide de l'utilitaire `sha256sum <fichier téléchargé>` que la somme de contrôle est bonne.
2. Extrayez le contenu du fichier (via la commande `tar -xzf <fichier tgz>` ou votre explorateur de fichiers)
3. Placez vous dans le terminal dans le dossier créé ( `prometheus-2.xxxx` ) et lancez en ligne de commande l'utilitaire `./prometheus --version`

# Fonctions de hachage

## Exercice 2

Rendez-vous sur <https://masterind4.github.io/antivirus.html>

Téléchargez la version 1.0 de `Antivirus.exe`

### UTILISEZ LE MIROIR 1

1. Vérifiez à l'aide de l'utilitaire `md5sum` que la somme de contrôle est bonne
2. Lancez `Antivirus.exe` (pour le rendre exécutable, utilisez la commande `chmod +x Antivirus.exe`)

```
./<fichier antivirus>
```

# Fonctions de hachage

## On continue l'exercice

Retournez sur <https://masterind4.github.io/antivirus.html>

3. Téléchargez la version **MIROIR 2** de `Antivirus.exe`

4. Vérifiez à l'aide de l'utilitaire `md5sum` que la somme de contrôle est bonne

Lancez `Antivirus_miroir2.exe` (pour le rendre executable, utilisez la commande `chmod +x` comme précédemment)

Répondre aux questions suivantes dans le formulaire:

1. Que s'est-il passé?


2. Proposez vos idées pour que cela ne puisse plus se reproduire

# Signatures digitales

## Exercice 1

Rendez-vous sur <https://veracrypt.fr/en/Downloads.html>

1. Téléchargez la version pour linux debian 12 de veracrypt ainsi que son fichier de signature PGP:

-  **Linux:**
  - Generic Installers: [veracrypt-1.25.4-setup.tar.bz2](#) (41.5 MB) ([PGP Signature](#))
  - Linux Legacy installer for 32-bit CPU with no SSE2: [veracrypt-1.25.4-x86-legacy-setup.tar.bz2](#) (13.8 MB) ([PGP Signature](#))
  - Debian/Ubuntu packages:
    - Debian 11:
      - GUI: [veracrypt-1.25.4-Debian-11-amd64.deb](#) ([PGP Signature](#))
      - Console: [veracrypt-console-1.25.4-Debian-11-amd64.deb](#) ([PGP Signature](#))

2. Téléchargez la clé publique de Veracrypt sur le site idrix (tout en bas de la page de téléchargements)



## Vérification

Utilisez `gpg` pour:

1. **importer** ( `gpg --import <fichier de clé>` ) la clé publique que vous venez de télécharger (vérifiez le fingerprint)

Cette étape vous permet d'ajouter la clé **publique** de VeraCrypt dans votre outil GPG de vérification de signatures.

2. **vérifier** ( `gpg --verify <fichier de signature.sig>` ) la signature du paquet deb que vous venez de télécharger

# Certificats

**Exercice 1: Utiliser `openssl s_client` pour se connecter de façon sécurisée à un serveur tiers.**

1. Utiliser la commande `man s_client` pour vous renseigner sur l'outil `openssl s_client` et comment l'invoquer pour vous connecter à un serveur TLS (pressez `q` pour quitter)
2. Tenter une connection sur `letsencrypt.org` sur le port `443`, utiliser l'option `-showcerts` pour afficher les certificats renvoyés par le serveur en format PEM

Vérifier que la connection retourne bien `Verify return code: 0 (ok)`, qui signifie que la chaîne de certification est bien valide. Tapez `<Entrée>` pour sortir.

- Sauvegarder le **premier** certificat affiché par la commande précédente dans un fichier avec l'extension `.pem`

```
CONNECTED(00000000)
depth=2 C = US, O = Internet Security Research Group, CN = ISRG Root X1
verify return:1
depth=1 C = US, O = Let's Encrypt, CN = R3
verify return:1
depth=0 CN = lencr.org
verify return:1

-----
Certificate chain
 0 s:CN = lencr.org
  i:C = US, O = Let's Encrypt, CN = R3
-----BEGIN CERTIFICATE-----
MIIEbJCCA1agAwIBAgISBCGG0R9Ugg9lAvKt5Az74tMA0GCSqGSIb3DQEBCwUA
MDIx CzAjBgNVBAYTAVtMRyYwFAYDVQQKEw1MZXQncyBFbmNyeXBOMQswCQYDVQDD
EwJ3JmAEFwY2EwY2E1MjgyMTIwMjRlRmFwY2FwY2FwY2FwY2FwY2FwY2FwY2Fw
Cw4xlbmNyLm9yZzZBMGMGByGSM49AGEGCGSM49AWEHAQIABEDrws1t9zBnYwiiu
sDhIEZ9yWYb+lbgr3Q83qTsRAAYSSaWxARYzaPDJzNxf7T9RXsyN4qBo1ZQ1kz
/5hkMLJggjJUMIICYTA0BgNVHQ8BAfEBEAMCAB4AwHQYDVVR0LBBYwFAYIKwYBBQU
AwEGGCCGAQUFBwMCMAMaGA1UdEWEb/wQCMAAAwHQYDVVR0B0BBYEFB1+tsIN5/mdLGLL
u1i7zzHGh/qkMB8GA1UdIwQYMBABAFBQSw3wFbLrLA1Q0YfR52LFMLGfJGUCSSG
AQUFBwEBBEKwZARlBggrBgEFBQcwAYYVAHR0cDovL3IzLm8ubG9uY2F5Iub3JlMCIj
CCsGAQUFBzAChZodHRwOi8vcjMuS5S5Zw5jc15vcmcvMG8GA1UdEQR0MGACWxL
bmNyLm9yZyP1bGV0c2VUy3J5c3JXZjY29tG59zSXRzZW5jcnldC5vcmeC3d3dy5s
ZW5jc15vcmeCE3d3dy5sZXRzZW5jcnldC59jb22CE3d3dy5sZXRzZW5jcnldC5v
cmcvEwYDVVR0GBAwCjAIBgZngW8AgEwggEgBoR8BgEEAdZ5AgQCBII0BHIHxA08A
dQATU3d1P2d5G6ELMFsG/kA7Z9Pw/ThVQ0wY2FwY2FwY2FwY2FwY2FwY2FwY2FwY2
MEQICIGtNfAdnkvEDmbw59p7Eeb4muQspvosjgBEezSHOL3maIAOPK1R0Pe6nlqp
k:s0Ux5RBW0D24KRJ2446dsG3U5GX8AB2AEIw42vapkc0D+VvqAdQm20scUgHLVt0s
gdm7v6s52IRZAAABjBgE6YCAAAQDAECwQRtIhA38Aq12wh1+fgE8TP2mcyIbGk+
uAyRDQGEFC5i5HfAEiBRmZPJzUw5FY9Pjjz9LefoT3Qepa9N+yFDLLA960s+vJAN
BgkqhkiG9w0BAQsFAA0CAQEA6I4EmE8s1teqTmJumusQDPWpkjyspeFDW8J5ShUF
rX0d5XhC6pGPHtCR484PUKSLFDEPuw56xrcwFmDL02YUw1202YUw1202YUw1202Y
w8Knjg3q/eCSRqwnGyuljh3jckB05giLU/Qvvd2qoqET/esU+y+OwLjfJcakLMgv
fH/c5n7nbip/h9S50J6V/7Qmd/tAGhgk8+4AU+XLM0raqzTSyxy6TV1H5tb1ontg
JGQmIS1P3DPJba2F3vCR49PyoD1LzfzKfB9nHQw/PeZuMhEweWd2f00l+8/6lN3
0+sR71jwSHbj1rvDUqeYtozIJavdMAKt4A8p04pHCp9vw==
-----END CERTIFICATE-----
 1 s:C = US, O = Let's Encrypt, CN = R3
  i:C = US, O = Internet Security Research Group, CN = ISRG Root X1
-----BEGIN CERTIFICATE-----
MIIFfjCCAvGgAwIBAgIRAJERcErPDBinUjBwLiwnX1owDQYJKoZIhvcNAQELBQAw
TzELMAkGA1UEBmVxMmktAnBgNVBAoTIEludG9ybmV0IFNlY3VyaXR5IFJlc2Vh
cmNoe1Eedyb3wMRUwEYwYDVQQDEwUJU1JHIFJvb3QgWDEwHhcnMjAwOTA0MDAMDAw
whcnMjAwOTE1MTYwMDAwfjAQMzswCQYDVQGEwJVUzEwMBQGA1UEChMnVGV03Mg
RW5jcnldDELMAkGA1UEAxMCUjMwggEjMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEK
AoIBAQC7AhUozPagUNMEUyNVZLD+ILxmaZ6qoinXSqgtSu5xUyxr45r+XXIo9cP
R50UVTvJ36ooj1Z9YI8Qv10bvUw7ybJcWcXPNZ00ftz2nWqsgvbsCJWCWh+idx
```

**Copier coller ce texte dans  
un fichier "cert.pem"**

## On va explorer son contenu

1. Utiliser la commande `openssl x509 -in <fichier pem> -text` sur le fichier du certificat sauvegardé.
2. Récupérer la liste des DNS autorisés ( `Subject Alternative Name` ) par le certificat final.
3. Lister les contraintes basiques ( `Basic Constraints` ) sur le certificat, et expliquer ce que peut signifier cette contrainte.

## Exercice 2: Authentification par certificat client

Rendez-vous sur <https://test.mosquitto.org/ssl/>

1. Suivez les instructions pour créer une CSR d'un certificat à vous, et faites le signer par le rootCA de MQTT via leur interface (chercher "generate your own certificate" sur la page).

**Mettez bien votre nom quand le CN vous sera demandé** (le reste peut être rempli à votre guise)

```
Locality Name (eg, city) []:  
Organization Name (eg, company) [Internet Widgits Pty Ltd]:  
Organizational Unit Name (eg, section) []:  
Common Name (e.g. server FQDN or YOUR name) [] guillaume.bienkowski  
Email Address []:  
  
Please enter the following 'extra' attributes  
to be sent with your certificate request
```

2. Récupérez le rootCA de mosquitto:

wget <http://test.mosquitto.org/ssl/mosquitto.org.crt>

Vous allez vous retrouver avec 4 fichiers:

- Une clé **privée** (qui est restée sur votre PC): `client.key`
- Une CSR qui contient votre clé **publique** et que vous envoyez à Mosquitto pour récupérer un certificat signé: `client.csr`
- Un certificat client, signé par Mosquitto, qui va vous permettre de vous identifier chez eux: `client.crt`
- le RootCA de mosquitto, qui permet à votre PC d'authentifier le serveur mosquitto `mosquitto.org.crt`

Une fois les étapes précédentes faites, installez mqttx:

```
curl -LO https://www.emqx.com/en/downloads/MQTTX/v1.9.8/mqttx-cli-linux-x64  
chmod +x mqttx-cli-linux-x64
```

Et lancez le client MQTT pour publier des messages (de manière authentifiée):

Passez vos clés, certificat client, et CA générés et récupérés dans la ligne de commande.

```
./mqttx-cli-linux-x64 bench pub -h test.mosquitto.org -p 8884 -l mqttts \  
-v -t master4 -im 5000 -c 1 -m "Coucou c'est $USER" \  
--key client.key --cert client.crt --ca mosquitto.org.crt
```

J'enverrais des messages à intervalles réguliers

**Question bonus:** Tentez la connection au serveur mosquitto avec `openssl s_client` en passant les bons arguments pour vous authentifier avec le certificat client et votre clé. N'oubliez pas de passer le certificat root pour dire à Openssl de lui faire confiance.

(Aide: utilisez `openssl s_client --help` et cherchez les arguments à passer)



# Aide openssl

Pour se connecter à un serveur:

```
openssl s_client -connect HOST:PORT -showcerts  
# -showcerts demande à afficher les certificats au format PEM
```

Pour décrire un certificat x509 contenu dans un fichier au format PEM:

```
openssl x509 -text -in CERTFILE
```

# Merci à tous!

Rappel: formulaire <https://forms.gle/p4RirfG6Vas9Aaes6>