

Sécurité avancée des systèmes

M1-ESGI-REIMS

PRÉ-REQUIS

Environnement

Poste de travail W/L/M

Software

VScode, Git, Python3

Virtualisation

1 VM Windows 10

1 VM Kali Linux

Conteneurisation

Docker, Docker-compose,
Docker-desktop

Objectifs du module

Chiffrement SSL / TLS

Cyber Threat Intelligence

Par feu

Man In The Middle



Objectifs de la séance

Chiffrement SSL/TLS



CM

Être capable de comprendre
les différents mécanismes du
chiffrement SSL.TLS

TP

Être capable de distinguer,
générer & piloter les formats
de chiffrements & hashage

TP

Être capable de mettre en
place un serveur web avec la
protection SSL/TLS

TP

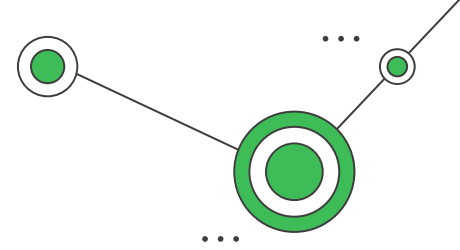
Être capable de proposer et
démontrer des solutions pour
protéger un serveur WEB et
des certificats

Introduction

...



Chiffrement VS. Hashage



Le chiffrement est la transformation d'une information en clair en une information chiffrée, incompréhensible, mais que l'on peut déchiffrer avec une clé pour obtenir l'information en clair originale.

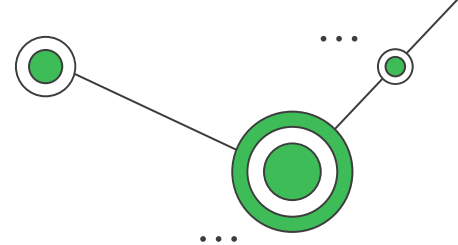
- Certificat HTTPS
- Ransomware

Le hashage est la transformation d'une information en clair en une information hachée, incompréhensible, de longueur fixe, généralement plus courte, représentant la chaîne d'origine et non déchiffrable.

- Mot de passe dans une BDD
- Checksum



Chiffrement symétrique VS. Chiffrement asymétrique



Chiffrement symétrique

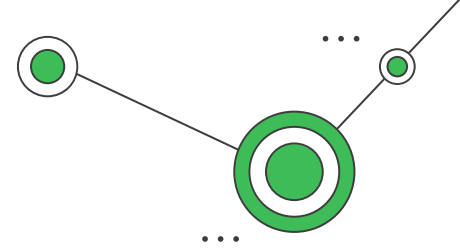
- Utilise une seule clé pour le chiffrement et le déchiffrement
- Utilisé pour la protection de données en masse
- Ransomwares
- AES, DES, 3DES, RC4...

Chiffrement asymétrique

- Utilise la clé public du destinataire pour le chiffrement et le destinataire déchiffre avec sa clé privée.
- Utilisé pour l'échange de clés secrètes.
- SSH, SFTP, HTTPS, Mails PGP
- Diffie-Hellman, RSA...



Base



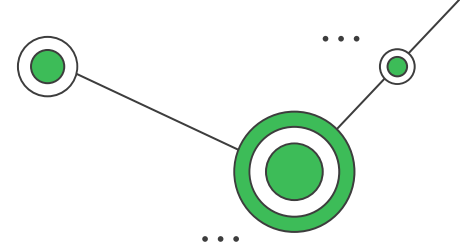
La cryptographie / le chiffrement est essentielle à la sécurité des systèmes d'information

Sans elle, un attaquant peut écouter les communications IP

- Requêtes HTTP avec les données d'un formulaire
- Requêtes Telnet de connexion à un routeur
- Requêtes FTP d'échange de document



Historique Chiffre de César



Chiffrement par décalage

- Consiste à substituer chaque lettre d'un message par la lettre de l'alphabet située à une distance fixée.

Par exemple, si la distance est 5, la lettre A est remplacée par la lettre F, la lettre B par G, et la lettre Z par E.

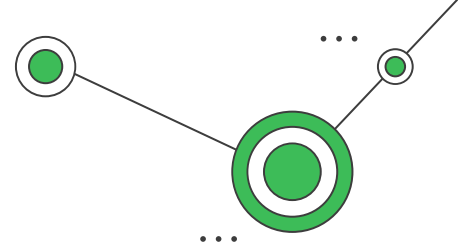
- MASTERESGI devient THZALYLZNP avec un décalage de 7

Vulnérabilités principale

- BRUTE FORCE



Historique Chiffre de Vigenère



Reprend en partie le principe de substitution, mais en variant la distance de décalage au cours du chiffrement en utilisant un mot ou une phrase comme clé. Chaque lettre de la clé correspond à sa position dans l'alphabet.

- Le chiffrement consiste à additionner chaque lettre du message avec la lettre de la clé en dessous, modulo 26 ;
- Le déchiffrement consiste à soustraire chaque lettre du message chiffré avec la lettre de la clé en dessous, modulo 26.

Par exemple, la lettre E du message en clair est d'abord associée à la lettre K de la clé et devient J dans le texte chiffré. Puis E est associée à l et devient M...

- MASTERESGI chiffré avec la clé ANATARES devient MNSMEIIKGV

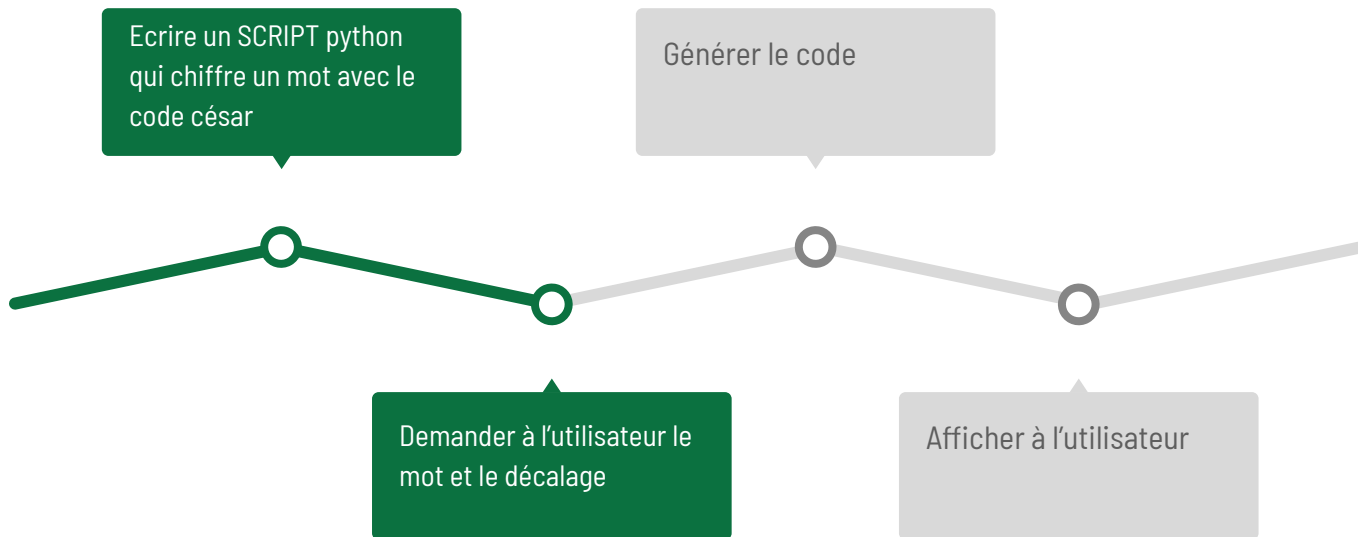
Vulnérabilités principale

- ANALYSE FREQUENTIELLE (Français : RSTLN E)



A vous de jouer

Génération d'un code César



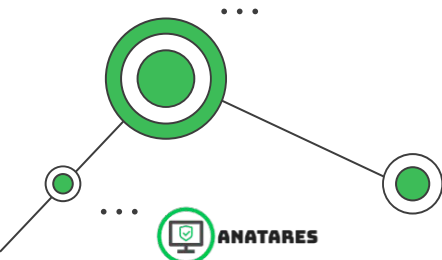
Sous dossier GIT : J1/Introduction



Solution

Mot : MASTERESGI

Résultat avec un shift = 7
thzalylnp

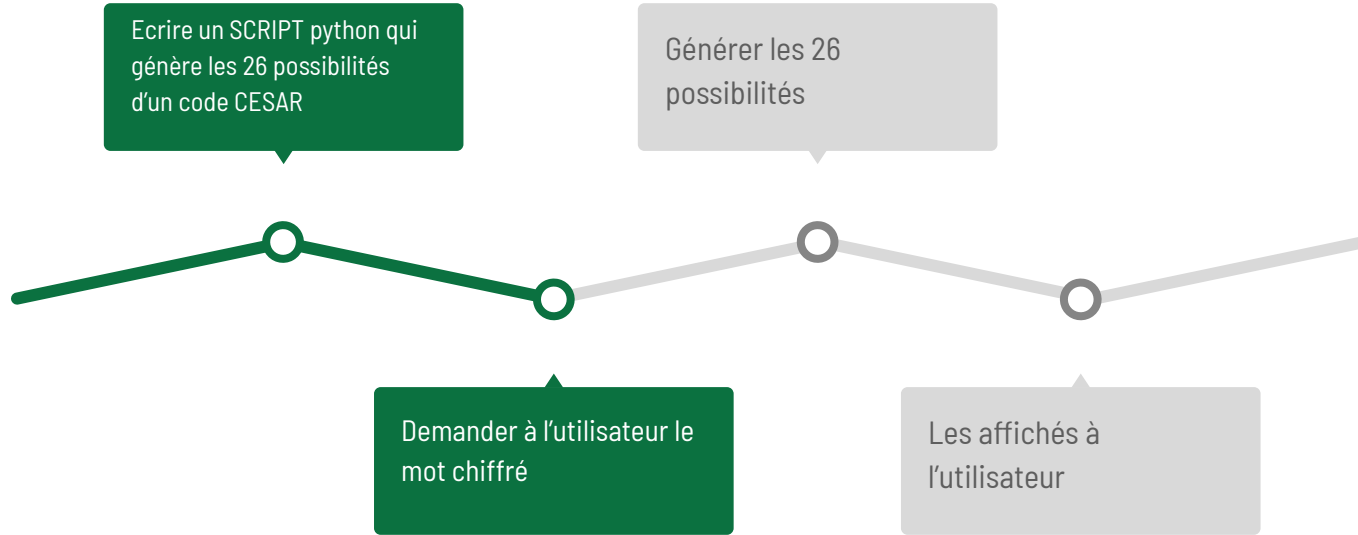


```
import time
letters = "abcdefghijklmnopqrstuvwxyz"
print("Please enter your string > ")
string_to_encode = input()
print("Please enter your shift > ")
cipher_shift = input()
string_to_encode = string_to_encode.lower()
cipher_shift = int(cipher_shift)
string_encrypted = ""
start_time = time.time()
for character in string_to_encode:
    position = letters.find(character)
    new_position = (position + cipher_shift) % 26
    if character in letters:
        string_encrypted = string_encrypted + letters[new_position]
    else:
        string_encrypted = string_encrypted + character
print("Cipher shift > " + str(cipher_shift))
print("Message      > " + str(string_encrypted))
print("--- %s seconds ---" % (time.time() - start_time))
```



A vous de jouer

Brute force d'un code César



Sous dossier GIT : J1/Introduction



Solution

Mot chiffré : THZALYLZNP

Cipher shift > 6

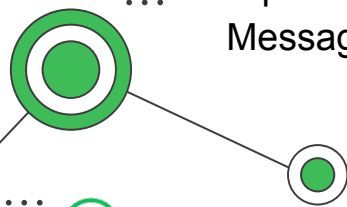
Message > nbtufsftjh

Cipher shift > 7

Message > masteresgi

Cipher shift > 8

Message > lzrsdqdrfh



```
import time
letters = "abcdefghijklmnopqrstuvwxyz"
print("Please enter your encoded string > ")
encoded_string = input()
start_time = time.time()
x = 0
while x < 26:
    x = x + 1
    string_to_decrypt = encoded_string.lower()
    cipher_shift = int(x)
    string_decrypted = ""
    for character in string_to_decrypt:
        position = letters.find(character)
        new_position = position - cipher_shift
        if character in letters:
            string_decrypted = string_decrypted + letters[new_position]
        else:
            string_decrypted = string_decrypted + character
    print("Cipher shift > " + str(cipher_shift))
    print("Message      > " + str(string_decrypted))
print("--- %s seconds ---" % (time.time() - start_time))
```

Chiffrement SSL / TLS

...



Historique

Le protocole SSL a connu 3 versions

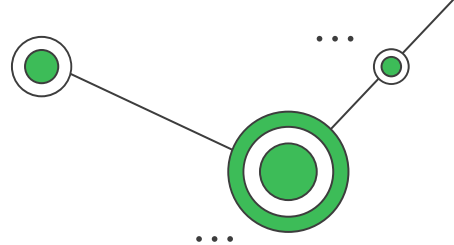
- SSL 1.0 en 1994
- SSL 2.0 en 1995
- SSL 3.0 en 1996

Le protocole TLS a connu 4 versions

- TLS v1.0 en 1999
- TLS v1.1 en 2006
- TLS v1.2 en 2008
- TLS v1.3 en 2018

TLS est aujourd'hui le successeur de SSL

Fonctionnement



L'utilisateur tente d'accéder à un site web qui utilise TLS, le client (navigateur de l'utilisateur) commence par envoyer une demande de mise en place de connexion sécurisée par TLS au serveur.

Le serveur renvoie alors son certificat contenant sa clé publique, ses informations (nom de la société, adresse postale, pays, e-mail de contact...) et une signature numérique qui permettent de l'authentifier.

Le certificat est un certificat numérique X.509 délivré par une autorité de certification (AC). Ce certificat peut être vu comme une carte d'identité numérique. Il permet d'identifier une entité et peut également être utilisé pour chiffrer des échanges.

Une fois le certificat envoyé, le client vérifie si celui-ci est valide. Pour cela, il vérifie la signature numérique du certificat du serveur en utilisant les clés publiques contenues dans les certificats des autorités de certifications intégrés par défaut dans le navigateur.

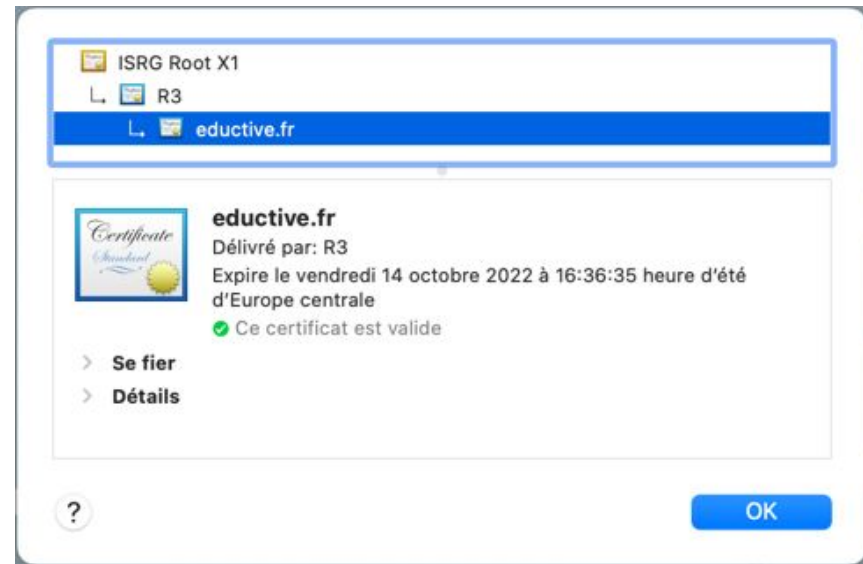
Si la validité du certificat est confirmée, le client génère une clé de chiffrement symétrique ou « clef de session » ou « secret » à partir de la clé publique contenue dans le certificat du serveur puis transmet cette clé de session au serveur.





Exemple d'un certificat

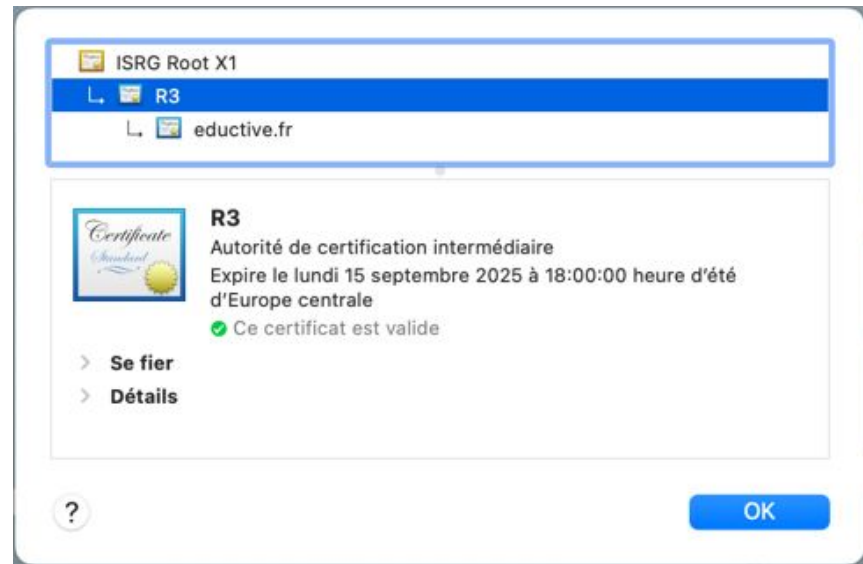
Certificat de base





Exemple d'un certificat

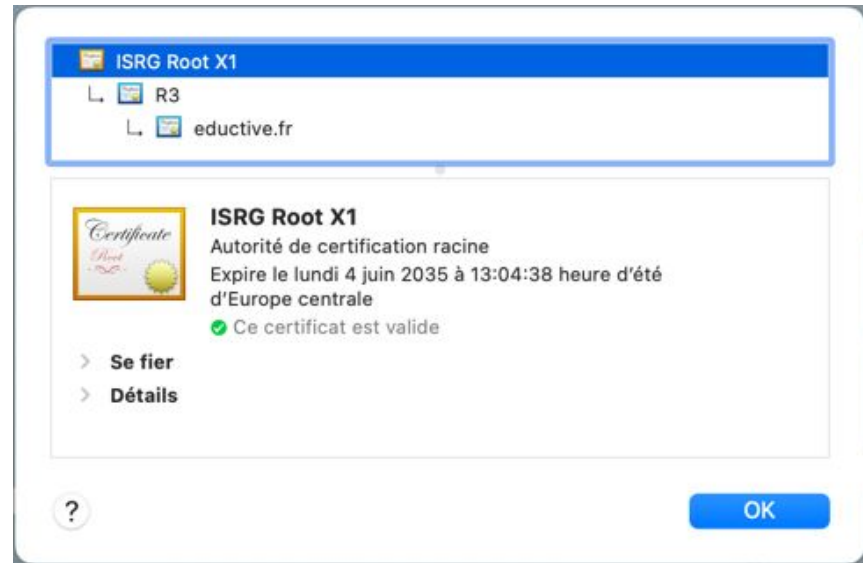
Certificat intermediaire



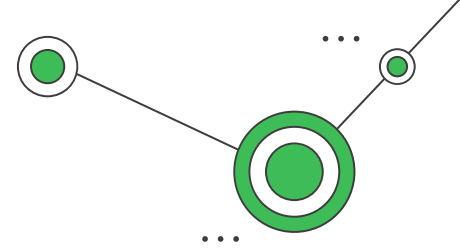


Exemple d'un certificat

Certificat racine



Résumé



L'autorité intermédiaire (Let's Encrypt, Digicert, Gandi...) certifiée par l'autorité racine (ISRG, USERTrust) délivre un certificat pour un site internet.

- Si un maillon de la chaîne est cassé / invalidé, le certificat de base est invalidé

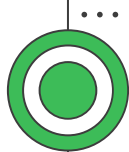
L'utilisateur discute avec le certificat de base

- Clé privée sur le serveur web
- Clé publique dans le certificat

Processus

- L'utilisateur chiffre les échanges avec la clé publique
- Le serveur déchiffre les échanges avec sa clé privée





A quoi ressemble un certificat TLS ?

```
-----BEGIN ENCRYPTED PRIVATE KEY-----
MIIFHABJbqkqhkiG9w0BBQowFADABgkqhkiG9w0BBQowFQI70K9tuzY3IECAqQA
HB0GCGWGSAAFIaWCGKqKk10DeZIDRJMngQPy5BQvIqSCBNC4+D/n7Fzq7p/gu/A
3vf9Xw39GTRsPF54Cy+NBQWupW17w3ge5aPrXtyq1c5b41GVG/aCBKXf/1ENh9
7xdrIZpNBSUS+5/q5ab7JQLG/G1BRNPaOGI2K0314eUjXkx8VahwxbxhAS2Jj
1Qj+FxHAb1J9E4LE2Tr93AHhZ2ZYbURCjneB5+I+wp1B1k8vblIMd4oAp/a1z8
Bk2q10c4S0VntV9YFFYhnlhLd8Spv6bUV8NAqo320QwhaDnx2EIDJIED+pj3xQBAa
PF91mdG0N3vRnLkPzRi4AAERaf5xaG28UXbaYmeJVau+1dvULjueB6HEZ2/cMn8
o61pAQSpQoRn0B6boeNVS1DCA8Nb3EUBD/budnfl1TpvIyQ/+1YHy/qhRqnlOkpt
bdhA23k/xxMAkaIG7UN9xu+2amK09H6NkaP76a486nuL+Th18qTxenruX2w890
y+ThVIGK+SUwgHYqjJj7ro67mD1E6RXGd35k1RtgvKDa7fuVXA217z1geap0Me
7L5nIP1o1+xx3eCjroY8G0XnL/tyZ3/G8x0YqT4qqB6K057X1VoXrnLbSHUwd1zr
QcQetWjRtW2kaGZVMDoygP/Vnoxy5IK0gZc55khlkahl+ubwv9mujH6AFy+AYa
84vQqkaQaa3syQ0k31FMcQh1M3jp8tnpKNQ5ad9Rc3L48Nu7Bzx09C7PsyKbDM9
wGz43Nfe/U11j1p/5b9cV4y/Te2VHJxyy0182x1kt6UKazPH+rqWew7QY1XJV1S
RoaKCGqFwV11cHwvK1k1MjyBOLwQUKx8PNkYvJ5X1C1TvvvrbB/zqT1G1Y1004
ppW6AR2nUWPN20xn28KFH22hN5zWe522u2tIq0KqK7Qd8yL3FF100CQmLx53v1
qQm5w1nkv244qjpdKyltW3eFlhTNIzyK2PJoosri6p8GpMKDp1171xhK9/LaH
Ha+n1jKxcp9+XH16Pa3a+Agd8v7zLeSBo0M1Be1GuoGwI0MYHz3o2+HF48pCq5W
IEVaY/vncQv8TzwaAC2CaGgUY1nBA48/eV29G03ypp153KJGUuvH00XnXgaC+djq
2aXFNQwhbo47r1gaD3n78Q6cYocKfttYp+RG7XoqX1qJaha6R3mU1QX93RE60e
PF/Afh4DUYv2ERjgk+q2f10DCaCD5RI6Mp6v8y7F02d9j3D+boFdaVz68Dq
m3Qv5e5PTJdhdF2qkaB2NGc9W11nqNG0oXx38+Mxe1nnTAS5aXXRoqi/R6T03HfV
M0AMhQoch1cFg3F40asw8GvLa5390e5vn0x6JD6J217yW68b80NDnB3j8112GaDO
yWpKb9FQSD7051o0J011NM25Pq/1I2/TL8oanJEBhNY01MWN7C2B5yG650uapl
F4o8UX1nG0P0HrdYmC0Gqky7jK20GMOF61urn21JpLtkcSkq2v0VRP3qA10Pz
boIIBhjPuStaULIA0u2qD0Uj5597x+FS8kPnz1FBKStPct069RLG0baG988Mo1GX
+BoYPMhk6fT52QAuH0CJnqbnXxalP8y8vLCCkpyh9j91qKQqxxGmAS5Y2Mn
t2Hz2Kgt03a1j3Gc2punoVnKzA==
-----END ENCRYPTED PRIVATE KEY-----
```

```
-----BEGIN CERTIFICATE-----
MIIDcjCCAlOCCQCw58gV8g2KGjANBgkqhkiG9w0BAQsFADB7MQswCQYDVQQGEwJG
UjESMBAGAEUECAwJR1JBTkQgRVNUMQ4wDAYDVQQHDAVSRU1NUzERMA8GA1UECgwI
RURVQ1RJVKUxEDA0BgNVBASMB00xLUVVTR0kxIzAhBgkqhkiG9w0BCQEFwFgnvbnRh
Y3RAYW5hdGFyZXMuY29tMB4XDTEyMDgxNTA2MzcyMl0xMDTIzMDgxNTA2MzcyMl0w
ezELMAkGA1UEBhMCRI1EjEjAQBgNVBAGMCUDsQU5EIEVTVDEOMAwGA1UEBwwFUKVJ
TVMxETAPBgNVBAoMCEVEVUNUSVZFMRADGyYDVQLDAAdNMS1FU0dJMSMwIQYJKoZI
hvcNAQKBFRjb250YWN0QGUFYXRhcmVzLmNvbTCCASIdDQYJKoZIhvcNAQEBBQAD
ggEPADCCAQoCggEBAMJ8LjWzHPv6ghpw/B+hfrjRyeKeCQFFb/mVO00Yxwv3WpmMU
MZUGmMiUUEKYHq+kp6QPCYHHauZb6QmnQylkjA1+O1JoWMMXES8ltCF02169q98j
KuMfnPJ0n9J82GCAaSp2KBQwoLLrU/gHfPoUs6N03qp20hroRkG3wydo9Y13UpQst
k+0fZD/gNa7mcmwmdRA7+dMjKCO4K8gPC2dqy9dF/A+F+50kYDxRPPjtRN6dEn52
6xZotaWmtWmNt0m0Id2HJEt+I1CKpy4LB33pps+00HxsHn9vKYLsStgOP/EyqWX2
/YwVrfI34agAcekJWym5G2YC1Cp15F8aniOwI8CAwEAATANBgkqhkiG9w0BAQsF
AAOCAQEATKLs7QhGg5Z2DkXxETJ3rpTxsD/+UCx5DGDd1ldpJdU0a1yFbKJzXJv
S9w++yus4b511p7aJ8/FE2mKutvjuBgaWnHAEKXU00VyZx+QZWEBV7+wLhH2CJ4
koKZrGV14coTP9tTH83n3akVDA1tKQUGVBsAHw2zc+wPo7Y07+SY0Z00i+L1oo89
T7URUXjz2R1fAa/wuH0GbWAEeCvXbh0s603kuDglr1z1+aCa6AmT4e5DeVWYasbM
KdT8xWFmTU0t0Hb2BWAwlcoThz/52KgHpd6/bMp2nh20WJUEoskxXNDznUlrwOh+
by78jwIF/NUnanaIVHV4akAvkuruKq==
-----END CERTIFICATE-----
```



Certificat auto-signé

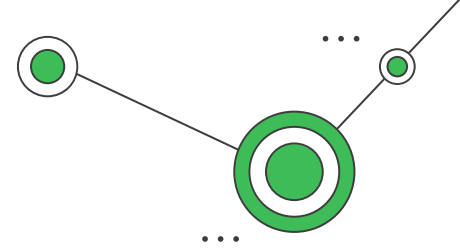
Gratuit contrairement à un certificat avec une chaîne valide (Env. 500 euros)

Très utilisé en entreprise

En général, un certificat auto-signé peut être une option appropriée lorsqu'il est utilisé pour un site interne (intranet) ou pour tester un site Web avant qu'il ne soit disponible pour le grand public.



Format des certificats



Quelle est la différence entre les certificats enregistrés sous .p7b, .pfx, .p12, .pem, .der, .crt ou .cer ?

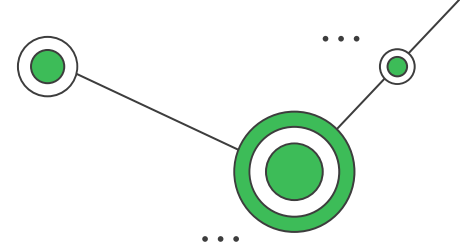
- Il faut savoir que l'extension du fichier du certificat SSL/TLS n'est pas importante. Un certificat SSL/TLS peut être enregistré au format txt (qui est également le plus courant pour la plupart des serveurs Linux, Apache, Unix et autres) ou binaire (Java, Microsoft Server).

Un certificat est enregistré dans un fichier de texte codé en Base64. Le fichier est lisible dans un éditeur de texte

- Il comporte les indications `--BEGIN` et `--END CERTIFICATE`.



Format PEM



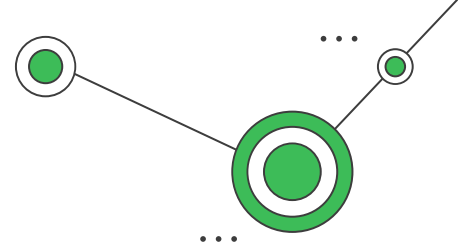
Le format le plus utilisé valable pour les certificats et les clés privées. La plupart des serveurs (par exemple Apache) utilisent la clé privée et le certificat dans les fichiers séparés. Des certificats PEM sont au format texte codé en Base64. PEM est un fichier codé en Base64 utilisant des caractères ASCII.

Les extensions sont généralement .cer, .crt, .pem ou .key (pour la clé privée).

Utilisé par les serveurs Apache et autres serveurs sous Unix/Linux OS.



Format DER



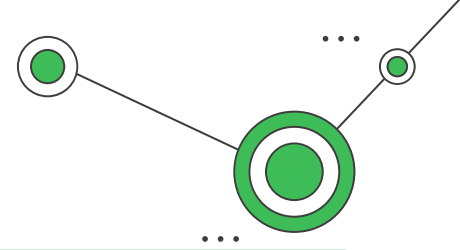
DER est un format de certificat binaire. Ce n'est pas un fichier texte et par conséquent, il ne peut pas être édité, ouvert et copié en tant que texte en Base64. Tous les types de certificats et les clés privées peuvent être sauvegardés au format DER.

Les extensions sont généralement .cer ou .der.

Utilisé par les plates-formes Java.



Format P7B/PKCS#7



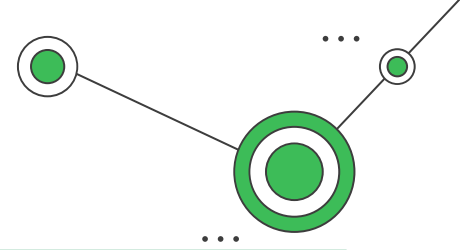
Le format P7B est un format basé sur le Base64. Les extensions sont .p7b ou .p7c.

Un fichier P7B contient le certificat et des certificats de chaîne (certificats intermédiaires) sans la clé privée.

Utilisé par les plate-formes Java Tomcat.



Format PFX/P12/PKCS#12



Le format PKCS#12 ou .pfx/P12 est un format binaire et contient le certificat (et son intermédiaire) et la clé privée. Le fichier .pfx est protégé par un mot de passe.

Les extensions sont .pfx et .p12.

Utilisé sous Windows pour importer et exporter le certificat avec la clé privée.



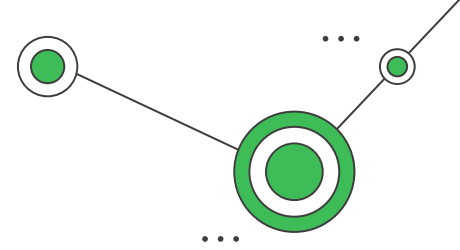
Un outil sous windows pour la gestion des certificats

XCA - X - Certificate and Key
management

<https://hohnstaedt.de/xca/>



OPENSSL

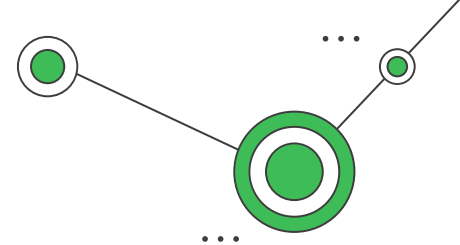


La bibliothèque OpenSSL est une implantation libre des protocoles SSL et TLS qui donne accès à :

- Une bibliothèque de fonctionnalité écrite en C permettant de réaliser des applications client/serveur sécurisées s'appuyant sur SSL/TLS,
- Un ensemble d'exécutables cmd permettant :
 - la forge de clés RSA, DSA (pour les signature)
 - la création de certificat X509 (identification)
 - le calcul d'empreinte (MD5, SHA, RIPEMD160, ...)
 - le chiffrement et le déchiffrement (RSA, DES, IDEA, RC2, RC4, Blowfish, ...)
 - la réalisation de de tests de clients et serveurs SSL/TLS
 - la signature et le chiffrement de courriers (S/MIME).



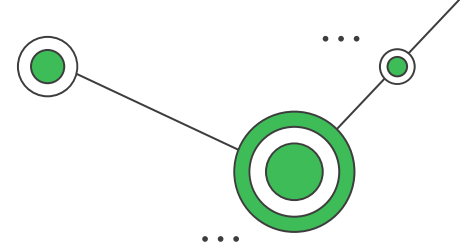
Générer un certificat CRT



```
openssl req -newkey rsa:2048 -new -x509 -days 365 -out domain.crt
```



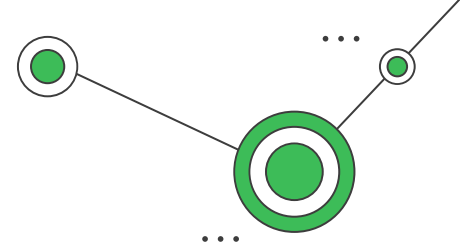
Générer un certificat DER



```
openssl x509 -in domain.crt -outform der -out domain.der
```

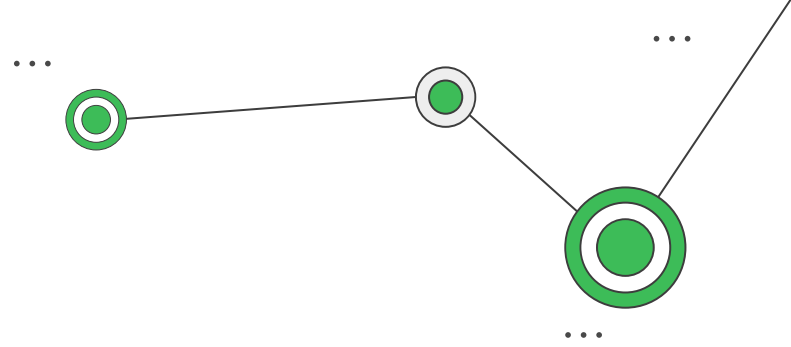


Convertir un certificat en PFX



```
openssl pkcs12 -inkey domain.key -in domain.crt -export -out domain.pfx
```

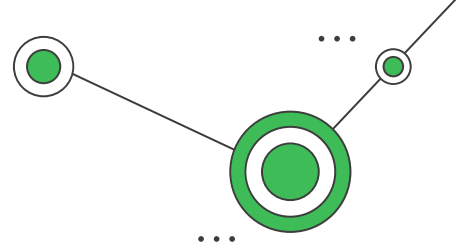




TP 1

Être capable de distinguer, générer & piloter les
formats de chiffrements & hashage

TP 1



RSA - OpenSSL

- Forgez vos clefs RSA.
- Chiffrer un fichier en RSA.
- Envoyez votre clé publique à votre voisin. Celui-ci vous enverra la sienne.
- Déchiffrez le fichier reçu

SHA - OpenSSL & Python

- Générer en python un fichier texte avec la liste des hash en SHA de toutes les chaînes de caractère de longueur 4 (Maj.

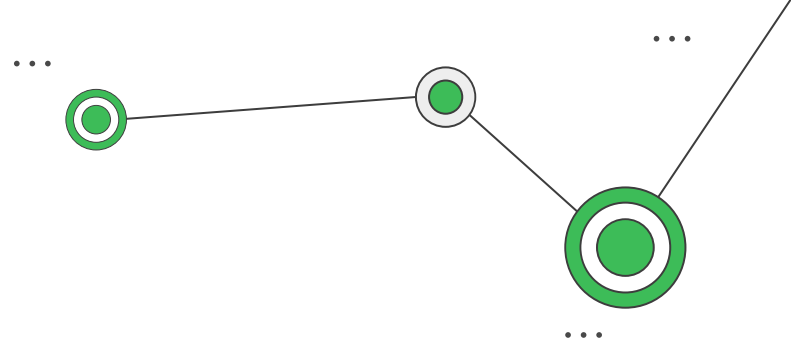
AES - OpenSSL & Python

- Générez une clef AES.
- Chiffrez le fichier SHA généré précédemment avec puis déchiffrez-le.

BONUS

- Écrire un script python qui prend tous les fichiers d'un dossier et qui les chiffre et déchiffre en AES.
- ATTENTION à bien spécifier le bon dossier...

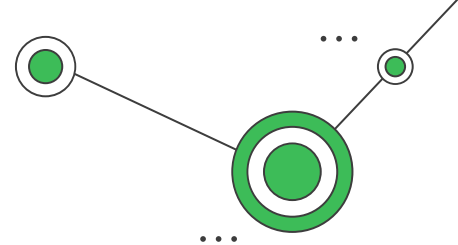




TP 2

Être capable de mettre en place un serveur web
avec la protection SSL/TLS

TP 2



SSL

- Générer un certificat autosigné pour un serveur WEB

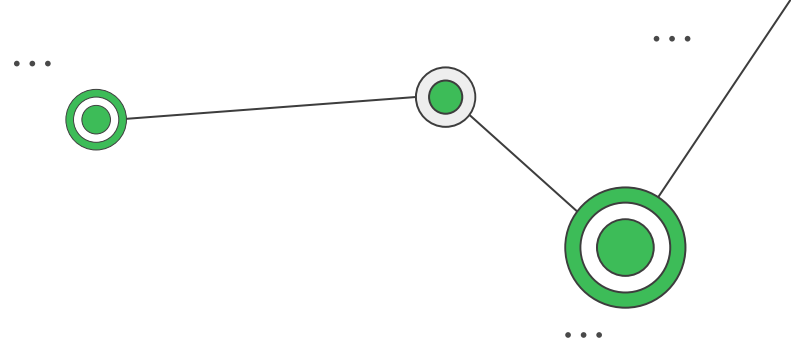
Web

- Créer une page HTML de base
- Monter et configurer un serveur web (Windows : IIS, Linux : Apache2, Docker : Apache2)
- Lui appliquer le certificat auto-signé
- Lui appliquer les sécurité basiques
 - ServerSignature
 - ServerTokens
 - Désactivation de « Index of / »
 - Paramétrage des LOG

BONUS

- Ecrire un script python qui au lancement nous dit dans combien de temps le certificat de plusieurs domaine expire

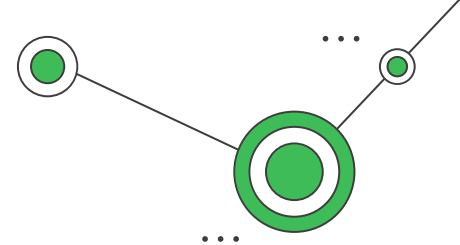




TP 3

Être capable de proposer et démontrer des solutions pour protéger un serveur WEB et des certificats

TP 3



PSSI

- Proposer un texte à l'exigence "Serveur WEB" d'une PSSI.
- Proposer un texte à l'exigence "Certificats & Chiffrement" d'une PSSI

Bonus

- Créer un script python qui brute force un formulaire HTML (anatares.com/admin)



Merci pour votre attention

Edouard LEBAS
ANATARES
edouard.lebas@anatares.com

