Seguridad en Redes
Practica 2.1

David Antuña Rodríguez
Javier Carrión García

1 OpenSSL

1.1 Cifrado de bloque

Los textos generados no son iguales porque la clave de cifrado del algoritmo es distinta a pesar de que las contraseñas coincidan.

```
usuario_vms@pto1103:~/SeR$
usuario_vms@pto1103:~/SeR$ cmp cipher1.bin cipher2.bin
cipher1.bin cipher2.bin son distintos: byte 9, línea 1
usuario_vms@pto1103:~/SeR$
usuario_vms@pto1103:~/SeR$ cmp cipher1.bin cipher3.bin
cipher1.bin cipher3.bin son distintos: byte 9, línea 1
usuario_vms@pto1103:~/SeR$
usuario_vms@pto1103:~/SeR$ cmp cipher2.bin cipher3.bin
cipher2.bin cipher3.bin son distintos: byte 9, línea 1
usuario_vms@pto1103:~/SeR$
```

Figure 1.1.1 : Comparación de los ficheros generados.

Por defecto OpenSSL utiliza salt al derivar la clave de cifrado para evitar los ataques por diccionario. Al utilizar la opción no salt se elimina el componente aleatorio de la clave, la salt, por lo que la misma clave da lugar al mismo texto cifrado.

```
usuario_vms@pto1103:~/SeR$
usuario_vms@pto1103:~/SeR$ cmp cipher1no.bin cipher2no.bin
usuario_vms@pto1103:~/SeR$ cmp cipher1no.bin cipher3no.bin
usuario_vms@pto1103:~/SeR$ cmp cipher2no.bin cipher3no.bin
usuario_vms@pto1103:~/SeR$ _
```

Figure 1.1.2 : Comparación de los cifrados sin salt.

Al cifrar o descifrar con OpenSSL se pueden utilizar las siguientes opciones.

- **-p**: muestra la key e iv usados para cifrar/descifrar.

```
usuario_vms@pto1103:~/SeR$ openssl enc -aes-128-ecb -p -in texto1.txt -out cipher1_p.bin
enter aes-128-ecb encryption password:
Verifying - enter aes-128-ecb encryption password:
salt=B6CCBC9A182C9D12
key=3A5A79414042BE53E13D03A75B2155A5
usuario_vms@pto1103:~/SeR$
```

Figure 1.1.3 : Cifrado utilizando la opcion -p.

```
usuario_vms@pto1103:~/SeR$ openssl aes-128-ecb -d -p -in cipher1_p.bin -out texto1_p.txt
enter aes-128-ecb decryption password:
salt=B6CCBC9A182C9D12
key=3A5A79414042BE53E13D03A75B2155A5
usuario_vms@pto1103:~/SeR$
```

Figure 1.1.4 : Descifrado utilizando la opcion -p.

- **-P**: muestra la key e iv generados por no se llega a cifrar/descifrar.
- **-pass**: si no se quiere que openSSL solicite la clave por consola se puede utilizar esta opcion para indicar el origen de la misma.

```
usuario_vms@ptoll03:~/SeR$
usuario_vms@ptoll03:~/SeR$ openssl enc -aes-128-ecb -pass file:pass.txt -in texto1.txt -out cipher1_pass.bin
usuario_vms@ptoll03:~/SeR$ openssl aes-128-ecb -d -pass file:pass.txt -in cipher1_pass.bin -out cipher1_pass.txt
usuario_vms@ptoll03:~/SeR$
```

Figure 1.1.5 : Cifrado/descifrado utilizando la opcion -pass.

- **-S**: permite especificar un salt, debe ser una cadena de digitos hexadecimales.

```
usuario_vms@ptoll03:~/SeR$
usuario_vms@ptoll03:~/SeR$ openssl enc -aes-128-ecb -p -S 6789 -in texto1.txt -out cipher1_S.bin
enter aes-128-ecb encryption password:
Verifying - enter aes-128-ecb encryption password:
salt=6789000000000000
key=A3C457761BE9D0AF56FC6F05948309F9
usuario_vms@ptoll03:~/SeR$ openssl aes-128-ecb -d -p -S 6789 -in cipher1_S.bin -out cipher1_S.txt
enter aes-128-ecb decryption password:
salt=6789000000000000
key=A3C457761BE9D0AF56FC6F05948309F9
usuario_vms@ptoll03:~/SeR$
```

Figure 1.1.6 : Cifrado/descifrado utilizando la opcion -S.

- **-K**: permite especificar la key a utilizar, debe ser una cadeba de digitos hexadecimales. Obliga a utlizar tambien la opción-iv, porque el iv se genera al generar la key.
- **-iv**: permite especificar el iv a utilizar, debe ser una cadeba de digitos hexadecimales. Si se especifica se ignora el iv que genere el proceso de creacion de la key, salvo que se hayan utilizado opciones especiales de generacion de la key.

```
usuario_vms@ptoll03:~/SeR$
usuario_vms@ptoll03:~/SeR$ openssl enc -aes-128-ecb -p -K 12345 -iv 6789 -in texto1.txt -out cipher1_Kiv.bin
warning: iv not use by this cipher
salt=98EC9E57397F0000
key=12345000000000000000000000000000
usuario_vms@ptoll03:~/SeR$ openssl aes-128-ecb -p -d -K 12345 -iv 6789 -in cipher1_Kiv.bin -out cipher1_Kiv.txt
warning: iv not use by this cipher
salt=98DCCC796A7F0000
key=12345000000000000000000000000000
usuario_vms@ptoll03:~/SeR$
```

Figure 1.1.7 : Cifrado/descifrado utilizando las opciones -K y -iv.

El receptor recibe la salt en el propio mensaje cifrado, figura 1.1.8 , y obtiene el iv al generar la clave con dicha salt y la contraseña privada que ha de conocer.

```
usuario_vms@ptoll03:~/SeR$ xxd cipher1_salt.bin
0000000: 5361 6c74 6564 5f5f b6d6 7fb9 22f5 3afc  Salted__....".:.
0000010: 6160 b22b 2b24 7ff8 99a6 d970 5ecc 9445  a`.++$.....p^..E
```

Figure 1.1.8 : Salt en el texto cifrado.

1.2 Modos de bloque

La diferencia es que en ecb podemos discernir los contornos de la figura y en cbc no se ve nada. Esto ocurre porque ecb no oculta los patrones del fichero origen en el fichero cifrado y cbc si.

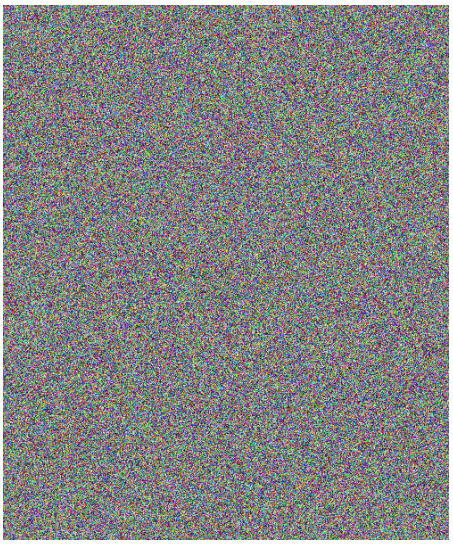

Figure 1.2.1 : Cifrado con cbc.

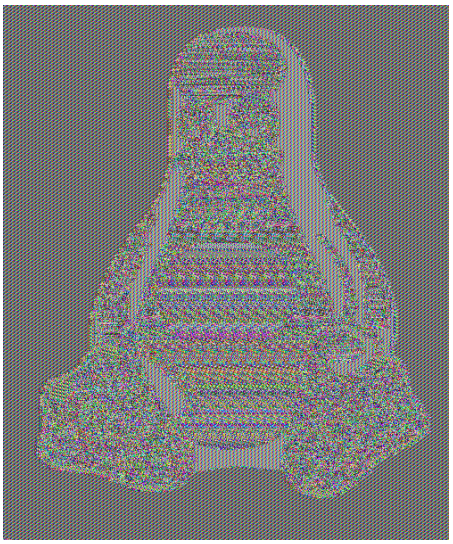

Figure 1.2.2 : Cifrado con ecb.

1.3 Cifrado de flujo

El texto en claro es **Attack at dawn**, la clave es **Secre** para el algoritmo compilado o **5365637265** para OpenSSL. Ambas claves son la misma pero a OpenSSL hay que proporcionarsela en hexadecimal y al compilado se le da la cadena.

```
usuario_vms@pto1102:~/SeR$ ./rc4 "Attack at dawn" "Secre"
14246F1CCFE55713C92202F6D31F
usuario_vms@pto1102:~/SeR$ echo -ne "Attack at dawn" | openssl enc -rc4-40 -K 5365637265 | xxd -u -p
14246F1CCFE55713C92202F6D31F
```

Figure 1.3.1 : Cifrado con rc4-40.

1.4 Funciones hash y HMAC

Se ha realizado el resumen de */etc/services* que tiene un tamaño de 19Kb, para ello hemos utilizado los algoritmos: md5, sha1 y sha256.

```
usuario_vms@pto1102:~/SeR$ ls -lh /etc/services
-rw-r--r-- 1 root root 19K may 14  2012 /etc/services
```

Figure 1.4.1 : Tamaño del fichero */etc/services*.

```
usuario_vms@ptol102:~/SeR$
usuario_vms@ptol102:~/SeR$ openssl dgst -md5 /etc/services
MD5(/etc/services)= 22e9e61234a27da688971ba9377ec731
usuario_vms@ptol102:~/SeR$
```

Figure 1.4.2 : Hash utilizando md5.

```
usuario_vms@ptol102:~/SeR$
usuario_vms@ptol102:~/SeR$ openssl dgst -sha1 /etc/services
SHA1(/etc/services)= 2376fab6b4405a625189fdb39d4cccf7f42cdd2d
usuario vms@ptol102:~/SeR$
```

Figure 1.4.3 : Hash utilizando sha1.

```
usuario_vms@ptol102:~/SeR$
usuario_vms@ptol102:~/SeR$ openssl dgst -sha256 /etc/services
SHA256(/etc/services)= 34701cc2257605ef104743891d6e621e645187295bde9544eeb2bd8f357a362e
usuario vms@ptol102:~/SeR$
```

Figure 1.4.4 : Hash utilizando sha256.

El valor del HMAC depende de la clave utilizada y la función de hash escogidas. Se han obtenido los siguientes HMAC del fichero */etc/services*.

```
usuario_vms@ptol102:~/SeR$
usuario_vms@ptol102:~/SeR$ openssl dgst -md5 -hmac seguridad /etc/services
HMAC-MD5(/etc/services)= 53d97373c4352ff054637f1c89d56285
usuario_vms@ptol102:~/SeR$
```

Figure 1.4.5 : HMAC utilizando md5.

```
usuario_vms@ptol102:~/SeR$
usuario_vms@ptol102:~/SeR$ openssl dgst -sha1 -hmac seguridad /etc/services
HMAC-SHA1(/etc/services)= 12274d5762f1f4f9c793608cf8edfaa5e89809b3
usuario_vms@ptol102:~/SeR$
```

Figure 1.4.6 : HMAC utilizando sha1.

```
usuario_vms@ptol102:~/SeR$
usuario_vms@ptol102:~/SeR$ openssl dgst -sha256 -hmac seguridad /etc/services
HMAC-SHA256(/etc/services)= 2e2caae0db347870222e658aea16f13be99fdb6145f5a520df860a7b73023cf4
usuario_vms@ptol102:~/SeR$
```

Figure 1.4.7 : HMAC utilizando sha256.

2 GnuPG

2.1 Cifrado y descifrado

Se ha cifrado el fichero */etc/services* utilizado la clave seguridad con **batch mode**, modo por bloques.

```
usuario_vms@ptol102:~/SeR$ cp /etc/services $HOME/SeR/services
usuario_vms@ptol102:~/SeR$ ls
services
usuario_vms@ptol102:~/SeR$ gpg2 --version
gpg (GnuPG) 2.0.25
libgcrypt 1.5.0
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Home: ~/.gnupg
Algoritmos disponibles:
Clave pública: RSA, ELG, DSA
Cifrado: 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH, CAMELLIA128,
         CAMELLIA192, CAMELLIA256
Resumen: MD5, SHA1, RIPEMD160, SHA256, SHA384, SHA512, SHA224
Compresión: Sin comprimir, ZIP, ZLIB, BZIP2
usuario_vms@ptol102:~/SeR$ gpg2 --symmetric --cipher-algo aes256  -a services
```

Figure 2.1.1 : Cigrado con clave simetrica en gpg2.

2.2 Funciones hash

Se ha calculado el hash de */etc/services* con varios algorimos.

```
usuario_vms@ptol102:~/SeR$ gpg2 --print-md md5  /etc/services
gpg: anillo `/home/usuario_vms/.gnupg/secring.gpg' creado
/etc/services: 22 E9 E6 12 34 A2 7D A6  88 97 1B A9 37 7E C7 31
```

Figure 2.2.1 : Hash utilizando md5.

```
usuario_vms@ptol102:~/SeR$ gpg2 --print-md sha1 /etc/services
/etc/services: 2376 FAB6 B440 5A62 5189  FDB3 9D4C CCF7 F42C DD2D
```

Figure 2.2.2 : Hash utilizando sha1.

```
usuario_vms@ptol102:~/SeR$ gpg2 --print-md sha256 /etc/services
/etc/services: 34701CC2 257605EF 10474389 1D6E621E 64518729 5BDE9544 EEB2BD8F
               357A362E
```

Figure 2.2.3 : Hash utilizando sha256.