

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

Escuela de Ingeniería Informática

Práctica 2:
Algoritmos de Resumen y Cifrado Simétrico

Asignatura: Seguridad de la Información

Curso: 4º de Grado en Ingeniería Informática

Autor:

Nicolás Rey Alonso

`nicolas.rey101@alu.ulpgc.es`

Fecha: Febrero 2026

“La criptografía es la ciencia y el arte de proteger la información.”

Índice general

1. Introducción	3
1.1. Objetivo de la práctica	3
2. Generación y Comprobación de Resúmenes	4
2.1. Descripción teórica	4
2.2. Algoritmos utilizados	4
2.3. Metodología experimental	4
2.3.1. Archivo de prueba	4
2.3.2. Comandos utilizados	5
2.4. Resultados	5
2.4.1. Resúmenes obtenidos	5
2.4.2. Verificación de tamaño	7
2.4.3. Sensibilidad del hash	7
3. Cifrado Simétrico de Documentos	9
3.1. Descripción teórica	9
3.2. Algoritmos de cifrado simétrico	9
3.2.1. Cifradores de bloque	9
3.2.2. Cifradores de flujo	10
3.3. Modos de operación	10
3.4. Derivación de claves: PBKDF2	10
3.5. Metodología experimental	11
3.5.1. Archivo de prueba	11
3.5.2. Cifrado con contraseña	11
3.6. Resultados	11
3.6.1. Cifrado con diferentes algoritmos	11
3.6.2. Análisis de overhead	11
3.6.3. Cifrado con clave e IV explícitos	12
4. Demostración de la Peligrosidad del Modo ECB	13

4.1. Fundamento teórico: El problema del modo ECB	13
4.1.1. Problemas de seguridad	13
4.2. Experimento: Cifrado de imágenes	13
4.2.1. Procedimiento	13
4.2.2. Comandos utilizados	14
4.2.3. Comparación de modos	14
4.3. Resultados obtenidos	15
4.3.1. Imágenes generadas	15
4.4. Conclusiones sobre seguridad	15
5. Conclusiones	16
5.1. Resumen de aprendizajes	16
5.2. Recomendaciones de seguridad	16
5.3. Problemas encontrados y soluciones	16
5.4. Trabajos futuros	17
A. Comandos OpenSSL Utilizados	19
A.1. Comandos de hash	19
A.2. Comandos de cifrado	19
A.3. Comandos para manipulación de archivos	20
B. Configuración de ImageMagick	21
B.1. Instalación	21
B.2. Conversiones útiles	21

Capítulo 1

Introducción

1.1. Objetivo de la práctica

El objetivo principal de esta práctica es utilizar *OpenSSL* para realizar operaciones criptográficas utilizando:

- Algoritmos de resumen (*hash*)
- Algoritmos de cifrado simétrico
- Diferentes modos de operación
- Derivación de claves a partir de contraseñas

Capítulo 2

Generación y Comprobación de Resúmenes

2.1. Descripción teórica

Los algoritmos de resumen (*hash*) son funciones criptográficas que transforman un documento de cualquier tamaño en una cadena de longitud fija, denominada *hash* o *resumen*.

2.2. Algoritmos utilizados

Cuadro 2.1: Algoritmos de resumen utilizados

Algoritmo	Bits	Bytes	Observaciones
MD5	128	16	Obsoleto
SHA-1	160	20	Débil
SHA-256	256	32	Recomendado y usado en la actualidad
SHA-512	512	64	Muy seguro
Whirlpool	512	64	Alternativa a SHA-512

2.3. Metodología experimental

2.3.1. Archivo de prueba

Se utilizará un archivo de texto plano con las siguientes características:

- Tamaño: 170 caracteres-

- Contenido legible.
- Sin caracteres especiales.

Listing 2.1: Contenido del archivo de prueba

```
1 Lorem ipsum dolor sit amet, consectetur adipiscing elit.  
2 Vivamus bibendum iaculis ante, quis sagittis eros eleifend  
3 iaculis. Sed egestas consequat feugiat. Lorem aliquam.
```

Listing 2.2: Contenido del archivo de prueba modificado

```
1 Lorem ipsum dolor sit amet, consectetur adipiscing elit.  
2 Vivamus bibendum iaculis ante, quis sagittis eros eleifend  
3 iaculis. Sed egestas consequat feugiat. Lorem aliquas.
```

2.3.2. Comandos utilizados

Listing 2.3: Cálculo de resúmenes con OpenSSL

```
1 # MD5  
2 openssl dgst -md5 archivo.txt  
3  
4 # SHA-1  
5 openssl dgst -sha1 archivo.txt  
6  
7 # SHA-256  
8 openssl dgst -sha256 archivo.txt  
9  
10 # Formato con separadores  
11 openssl dgst -sha256 -c archivo.txt  
12  
13 # Formato binario  
14 openssl dgst -sha256 -binary archivo.txt > hash.bin
```

2.4. Resultados

2.4.1. Resúmenes obtenidos

Cuadro 2.2: Resúmenes calculados para el archivo de prueba

Algoritmo

MD5

```
Nicolás Rey Alonso-[~/Desktop/Practicas/Practicas_Seguridad_De_La_Informacion/Practica2/ArchivosOriginales]
$ openssl dgst -md5 TextFile.txt
MD5(TextFile.txt)= 3b4222d01c204d28681adba06ff6885f
```

MD5 con formato de salida con separadores

```
Nicolás Rey Alonso-[~/Desktop/Practicas/Practicas_Seguridad_De_La_Informacion/Practica2/ArchivosOriginales]
$ openssl dgst -md5 -c TextFile.txt
MD5(TextFile.txt)= 3b:42:22:d0:1c:20:4d:28:68:1a:db:a0:6f:f6:88:5f
```

MD5 en formato binario

```
Nicolás Rey Alonso-[~/Desktop/Practicas/Practicas_Seguridad_De_La_Informacion/Practica2/ArchivosOriginales]
$ openssl dgst -md5 -binary TextFile.txt | xxd
00000000: 3b42 22d0 1c20 4d28 681a dba0 6ff6 885f ;B" .. M(h ... o _
```

SHA-1

```
Nicolás Rey Alonso-[~/Desktop/Practicas/Practicas_Seguridad_De_La_Informacion/Practica2/ArchivosOriginales]
$ openssl dgst -sha1 TextFile.txt
SHA1(TextFile.txt)= 6fb6e68d041f023e8cdabfa19b4fc539fc5a2270
```

SHA-1 con formato de salida con separadores

```
Nicolás Rey Alonso-[~/Desktop/Practicas/Practicas_Seguridad_De_La_Informacion/Practica2/ArchivosOriginales]
$ openssl dgst -sha1 -c TextFile.txt
SHA1(TextFile.txt)= 6f:b6:e6:8d:04:1f:02:3e:8c:da:bf:a1:9b:4f:c5:39:fc:5a:22:70
```

SHA-1 en formato binario

```
Nicolás Rey Alonso-[~/Desktop/Practicas/Practicas_Seguridad_De_La_Informacion/Practica2/ArchivosOriginales]
$ openssl dgst -sha1 -binary TextFile.txt | xxd
00000000: 6fb6 e68d 041f 023e 8cda bfa1 9b4f c539 o.....>....0.9
00000010: fc5a 2270 .Z"p
```

SHA-256

```
Nicolás Rey Alonso-[~/Desktop/Practicas/Practicas_Seguridad_De_La_Informacion/Practica2/ArchivosOriginales]
$ openssl dgst -sha256 TextFile.txt
SHA2-256(TextFile.txt)= 43f38d2a135292e9d2f1e9e39fa0543ab79a720b4ee68fd04e51101bba5e3851
```

SHA-256 con formato de salida con separadores

```
Nicolás Rey Alonso-[~/Desktop/Practicas/Practicas_Seguridad_De_La_Informacion/Practica2/ArchivosOriginales]
$ openssl dgst -sha256 -c TextFile.txt
SHA2-256(TextFile.txt)= 43:f3:8d:2a:13:52:92:e9:d2:f1:e9:e3:9f:a0:54:3a:b7:9a:72:0b:4e:e6:8f:d0:4e:51:10:1b:ba:5e:38:51
```

SHA-256 en formato binario

```
Nicolás Rey Alonso-[~/Desktop/Practicas/Practicas_Seguridad_De_La_Informacion/Practica2/ArchivosOriginales]
$ openssl dgst -sha256 -binary TextFile.txt | xxd
00000000: 43f3 8d2a 1352 92e9 d2f1 e9e3 9fa0 543a C..*.R.....T:
00000010: b79a 720b 4ee6 8fd0 4e51 101b ba5e 3851 ..r.N...NQ...^8Q
```

SHA-512

```
Nicolás Rey Alonso-[~/Desktop/Practicas/Practicas_Seguridad_De_La_Informacion/Practica2/ArchivosOriginales]
$ openssl dgst -sha512 TextFile.txt
SHA2-512(TextFile.txt)= 9d83571a2b70a5ac918445d4340cfc0403518631f546c04870ef6377739aca47a01e3f8c5d026314280755c0a3d6a5f5fe5446edef0fb59079ac8e5384bab478
```

SHA-512 con formato de salida con separadores

```
Nicolás Rey Alonso-[~/Desktop/Practicas/Practicas_Seguridad_De_La_Informacion/Practica2/ArchivosOriginales]
$ openssl dgst -sha512 -c TextFile.txt
SHA2-512(TextFile.txt)= 9d:83:57:1a:2b:70:a5:ac:91:84:45:d4:34:0c:fc:04:03:51:86:31:f5:46:c0:48:70:ef:63:77:73:9a:ca:47:a0:1e:3f:8c:5d:02:63:14:28:07:55:c0:a3:d6:a5:f5:fe:54:46:ed:ef:0f:b5:90:79:ac:8e:53:84:bab:47:8
```

SHA-512 en formato binario

Cuadro 2.2: Resúmenes calculados para el archivo de prueba (cont.)

Algoritmo

```

Nicolás Rey Alonso-[~/Desktop/Practicas/Practicas_Seguridad_De_La_Informacion/Practica2/ArchivosOriginales]
$ openssl dgst -sha512 -binary TextFile.txt | xxd
00000000: 9d83 571a 2b70 a5ac 9184 45d4 340c fc04  ..W.+p....E.4...
00000010: 0351 8631 f546 c048 70ef 6377 739a ca47  .Q.1.F.Hp.cws..G
00000020: a01e 3f8c 5d02 6314 2807 55c0 a3d6 a5f5  ..?.]..c(..U.....
00000030: fe54 46ed ef0f b590 79ac 8e53 84ba b478  .TF.....y..S...x

```

Whirlpool

```

Nicolás Rey Alonso-[~/Desktop/Practicas/Practicas_Seguridad_De_La_Informacion/Practica2/ArchivosOriginales]
$ openssl dgst -whirlpool -c TextFile.txt
WHIRLPOOL(TextFile.txt)= 29d2a395274ab59ea25fabe0124cb9664fc503b06ca6cddb7f5130248b427bf0e8f760b2048ad46e45de44721dbeb38b0eaf9a4bab31a01c211a2f0c5dadf34

```

Whirlpool con formato de salida con separadores

```

Nicolás Rey Alonso-[~/Desktop/Practicas/Practicas_Seguridad_De_La_Informacion/Practica2/ArchivosOriginales]
$ openssl dgst -whirlpool -c TextFile.txt
WHIRLPOOL(TextFile.txt)= 29:d2:a3:95:27:4a:b5:9e:a2:5f:ab:e0:01:24:cb:96:64:fc:50:3b:06:ca:6c:db:7f:51:30:24:8b:42:7f:bf:0e:8f:76:0b:20:48:ad:46:e4:5d:e4:47:21:db:eb:38:b0:ea:f9:a4:ba:b3:1a:01:c2:11:a2:f0:c5:da:df:34

```

Whirlpool en formato binario

```

Nicolás Rey Alonso-[~/Desktop/Practicas/Practicas_Seguridad_De_La_Informacion/Practica2/ArchivosOriginales]
$ openssl dgst -whirlpool -binary TextFile.txt | xxd
00000000: 29d2 a395 274a b59e a25f abe0 0124 cb96  )... 'J'... ..$..
00000010: 64fc 503b 06ca 6cdb 7f51 3024 8b42 f7bf  d.P; ..l..Q0$.B..
00000020: 0e8f 760b 2048 ad46 e45d e447 21db eb38  ..v. H.F.].G! ..8
00000030: b0ea f9a4 bab3 1a01 c211 a2f0 c5da df34  .....4

```

2.4.2. Verificación de tamaño

Los tamaños de los resúmenes coinciden con las especificaciones teóricas:

- MD5: 32 caracteres hexadecimales (128 bits)
- SHA-1: 40 caracteres hexadecimales (160 bits)
- SHA-256: 64 caracteres hexadecimales (256 bits)
- SHA-512: 128 caracteres hexadecimales (512 bits)
- Whirlpool: 128 caracteres hexadecimales (512 bits)

2.4.3. Sensibilidad del hash

Se realizó una prueba de sensibilidad cambiando un único carácter del archivo original:

Figura 2.1: Diferencias tras cambiar un único carácter (comparación de resúmenes)

```

Nicolás Rey Alonso-[~/Desktop/Practicas/Practicas_Seguridad_De_La_Informacion/Practica2/Entrega]
$ cat $TEXT0_RESUMEN
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus bibendum iaculis ante, quis sagittis eros eleifend iaculis. Sed egestas consequat feugiat. Lorem aliquam.

Nicolás Rey Alonso-[~/Desktop/Practicas/Practicas_Seguridad_De_La_Informacion/Practica2/Entrega]
$ openssl dgst -sha256 "$TEXT0_RESUMEN"
SHA2-256(..ArchivosOriginales/TextFile.txt)= 43f38d2a135292e9d2f1e9e39fa0543ab79a720b4ee68fd04e51101bba5e3851

Nicolás Rey Alonso-[~/Desktop/Practicas/Practicas_Seguridad_De_La_Informacion/Practica2/Entrega]
$ vim $TEXT0_RESUMEN

Nicolás Rey Alonso-[~/Desktop/Practicas/Practicas_Seguridad_De_La_Informacion/Practica2/Entrega]
$ cat $TEXT0_RESUMEN
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus bibendum iaculis ante, quis sagittis eros eleifend iaculis. Sed egestas consequat feugiat. Lorem aliquas.

Nicolás Rey Alonso-[~/Desktop/Practicas/Practicas_Seguridad_De_La_Informacion/Practica2/Entrega]
$ openssl dgst -sha256 "$TEXT0_RESUMEN"
SHA2-256(..ArchivosOriginales/TextFile.txt)= 28e7849c288d254f85a09597e00aba72c45114481c51e73867a83ec0742ba93f

```

Conclusión: A pesar de cambiar un único carácter, el hash es completamente diferente.

Capítulo 3

Cifrado Simétrico de Documentos

3.1. Descripción teórica

El cifrado simétrico es una técnica criptográfica en la que se utiliza la misma clave tanto para cifrar como para descifrar. Los elementos principales son:

- **Texto claro:** Documento original sin cifrar
- **Algoritmo de cifrado:** Función matemática que transforma el texto
- **Clave:** Información secreta que controla la transformación
- **Texto cifrado:** Resultado ilegible del proceso
- **Vector de inicialización (IV):** Valor aleatorio para ciertos modos

3.2. Algoritmos de cifrado simétrico

3.2.1. Cifradores de bloque

Cuadro 3.1: Cifradores de bloque empleados

Algoritmo	Tamaño bloque	Tamaño clave	Estado
AES-256	128 bits	256 bits	Recomendado
3DES/TDES	64 bits	168 bits	Deprecado

3.2.2. Cifradores de flujo

Cuadro 3.2: Cifradores de flujo empleados

Algoritmo	Tamaño clave	Observaciones
RC4	Variable	Débil, deprecado
ChaCha20	256 bits	Moderno, recomendado

3.3. Modos de operación

Cuadro 3.3: Modos de operación de cifrado en bloque

Modo	Descripción	Seguridad
ECB	Cada bloque cifra de forma independiente	Inseguro
CBC	Encadenamiento de bloques con IV	Seguro
OFB	Flujo de retroalimentación de salida	Seguro
CTR	Modo contador	Seguro
GCM	Authenticated encryption	Muy seguro

3.4. Derivación de claves: PBKDF2

El estándar PKCS#5 define los siguientes métodos de derivación:

- **PBKDF1:** Primera versión, limitada
- **PBKDF2:** Versión mejorada, recomendada

PBKDF2 genera una clave a partir de una contraseña utilizando:

$$DK = \text{PBKDF2}(P, S, c, dkLen) \quad (3.1)$$

Donde:

- P : Contraseña
- S : Sal (salt) - mínimo 8 bytes
- c : Número de iteraciones (por defecto 10000)
- $dkLen$: Longitud deseada de la clave

3.5. Metodología experimental

3.5.1. Archivo de prueba

Se utiliza un archivo de texto pequeño (31-81 caracteres):

Listing 3.1: Archivo para cifrado simétrico

```
1 Este es un texto de prueba para cifrado simétrico.
```

3.5.2. Cifrado con contraseña

Listing 3.2: Cifrado con contraseña y PBKDF2

```
1 openssl enc -aes-256-cbc -pbkdf2 -in archivo.txt \  
2     -out archivo.bin -pass pass:"contraseña"  
3  
4 # Con información de derivación  
5 openssl enc -aes-256-cbc -pbkdf2 -in archivo.txt \  
6     -out archivo.bin -pass pass:"contraseña" -p
```

3.6. Resultados

3.6.1. Cifrado con diferentes algoritmos

Se realizaron cifrados con los siguientes algoritmos:

Cuadro 3.4: Resultados del cifrado simétrico

Algoritmo	Modo	Tamaño	Descifrado
AES-256	CBC	[Tamaño] bytes	✓ OK
AES-128	CTR	[Tamaño] bytes	✓ OK
3DES	CBC	[Tamaño] bytes	✓ OK
3DES	OFB	[Tamaño] bytes	✓ OK
RC4	-	[Tamaño] bytes	✓ OK
ChaCha20	-	[Tamaño] bytes	✓ OK
AES-256	GCM	[Tamaño] bytes	✓ OK

3.6.2. Análisis de overhead

El tamaño de los archivos cifrados incluye:

- **Sal (Salt):** Primeros 8 bytes (formato: “Salted__”)
- **Contenido cifrado:** Datos cifrados
- **Padding (si aplica):** Relleno para completar bloques (en modos CBC)

Ecuación general:

$$\text{Tamaño_cifrado} = 16 + \text{ceil} \left(\frac{\text{Tamaño_original}}{T_{\text{bloque}}} \right) \times T_{\text{bloque}} \quad (3.2)$$

Donde T_{bloque} es el tamaño del bloque (128 bits para AES, 64 bits para 3DES).

3.6.3. Cifrado con clave e IV explícitos

Se demostró cómo descifrar utilizando la clave y el IV en lugar de la contraseña:

Listing 3.3: Cifrado con información de derivación

```
1 # Obtener clave e IV derivados
2 openssl enc -aes-256-cbc -pbkdf2 -in archivo.txt \
3     -out archivo.bin -pass pass:"contraseña" -p
4
5 # Eliminar los 16 bytes de sal
6 dd if=archivo.bin of=archivo_sin_sal.bin bs=1 skip=16
7
8 # Descifrar con clave e IV
9 openssl enc -aes-256-cbc -d -in archivo_sin_sal.bin \
10     -out archivo_descifrado.txt -K <clave_hex> -iv <iv_hex>
```

Capítulo 4

Demostración de la Peligrosidad del Modo ECB

4.1. Fundamento teórico: El problema del modo ECB

El modo ECB (*Electronic Codebook*) es uno de los modos de operación más simples e inseguros. Su funcionamiento es:

$$C_i = E(K, P_i) \quad (4.1)$$

Donde cada bloque de texto plano P_i se cifra independientemente con la misma clave K .

4.1.1. Problemas de seguridad

1. **Determinismo:** Bloques idénticos producen criptogramas idénticos
2. **Revelación de patrones:** Patrones en el texto plano se revelan en el cifrado
3. **Análisis de frecuencia:** Posible análisis estadístico
4. **Imagen de Tux:** El ejemplo clásico demuestra visualmente estas debilidades

4.2. Experimento: Cifrado de imágenes

4.2.1. Procedimiento

Se utilizó una imagen de colores sólidos (PNG de 20-50 KB) para demostrar los problemas de ECB:

1. Conversión de la imagen a formato PGM
2. Separación de cabecera y datos binarios
3. Cifrado de los datos con AES-256-ECB
4. Reconstrucción de la imagen cifrada
5. Conversión de vuelta al formato original
6. Comparación con modo CBC

4.2.2. Comandos utilizados

Listing 4.1: Cifrado ECB de imágenes

```
1 # Convertir a PGM
2 convert imagen.png imagen.pgm
3
4 # Separar cabecera y datos
5 head -n 3 imagen.pgm > cabecera.txt
6 tail -n +4 imagen.pgm > datos.bin
7
8 # Cifrado ECB (INSEGURO)
9 openssl enc -aes-256-ecb -in datos.bin -out datos_ecb.bin \
10     -K 0123456789ABCDEF... -nosalt
11
12 # Reconstruir y convertir
13 cat cabecera.txt datos_ecb.bin > imagen_cifrada.pgm
14 convert imagen_cifrada.pgm imagen_cifrada.png
```

4.2.3. Comparación de modos

Cuadro 4.1: Comparación de modos de operación

Modo	Resultado	Seguridad
ECB	<i>Se distinguen patrones de la imagen original</i>	Crítica
CBC	<i>Imagen completamente aleatoria</i>	Segura

4.3. Resultados obtenidos

4.3.1. Imágenes generadas

Se generaron las siguientes imágenes:

- **Original:** Imagen de colores sólidos clara
- **ECB con AES-256:** Los colores originales son vagamente visibles
- **ECB con 3DES:** Patrones más pronunciados debido a bloques de 64 bits
- **CBC con AES-256:** Imagen completamente aleatoria

4.4. Conclusiones sobre seguridad

- **NUNCA usar ECB:** Incluso con claves fuertes, revela patrones
- **Usar CBC, CTR, GCM:** Estos modos son seguros
- **Tamaño de bloque:** Bloques mayores (128 bits) ocultan mejor los patrones
- **IV aleatorio:** Esencial en modos que requieren IV

Capítulo 5

Conclusiones

5.1. Resumen de aprendizajes

A través de esta práctica se han consolidado los siguientes conceptos:

1. **Algoritmos de resumen:** Importancia de la sensibilidad y tamaño del hash
2. **Cifrado simétrico:** Uso correcto de algoritmos y modos de operación
3. **Derivación de claves:** PBKDF2 como estándar seguro
4. **Modos seguros vs. inseguros:** Errores comunes en criptografía

5.2. Recomendaciones de seguridad

Para aplicaciones reales:

1. **Resúmenes:** Usar SHA-256 o superior (SHA-3)
2. **Cifrado simétrico:** Preferir AES con modo CBC, CTR o GCM
3. **Derivación de claves:** PBKDF2 con mínimo 100,000 iteraciones
4. **Sal:** Mínimo 16 bytes de sal aleatoria
5. **IV:** Aleatorio y único para cada cifrado en modos que lo requieran

5.3. Problemas encontrados y soluciones

- **Provider Legacy:** Fue necesario activarlo para usar algoritmos antiguos (MD5, RC4)

- **Formatos binarios:** Importancia de usar `-binary` en OpenSSL
- **Imágenes comprimidas:** No funcionan bien con el experimento ECB

5.4. Trabajos futuros

1. Estudiar cifrado asimétrico (RSA, ECDSA)
2. Implementar funciones hash criptográficas personalizadas
3. Analizar vulnerabilidades de padding (Padding Oracle)
4. Estudiar Side-channel attacks en criptografía

Bibliografía

- [1] OpenSSL Documentation, <https://www.openssl.org/docs/man3.2/>, 2024.
- [2] RSA Laboratories, *PKCS #5: Password-Based Cryptography Specification*, RFC 2898, 2000.
- [3] NIST Federal Information Processing Standards Publication 197, *Specification for the Advanced Encryption Standard (AES)*, 2001.
- [4] Kelm, R., Turan, M. S., *PBKDF2 Implementation in OpenSSL*, 2023.
- [5] Wikipedia, *Block cipher mode of operation*, https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation, 2024.
- [6] Wikipedia, *Electronic Codebook (ECB) - Tux the Linux Penguin*, https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation#ECB, 2024.
- [7] Rivest, R., *The MD5 Message-Digest Algorithm*, RFC 1321, 1992.
- [8] FIPs 180-4: Secure Hash Standard, National Institute of Standards and Technology, 2015.

Apéndice A

Comandos OpenSSL Utilizados

A.1. Comandos de hash

Listing A.1: Comandos para resúmenes

```
1 # Listar algoritmos disponibles
2 openssl dgst -list
3
4 # Diferentes formatos
5 openssl dgst -md5 archivo.txt
6 openssl dgst -sha256 -c archivo.txt          # Con separadores
7 openssl dgst -sha256 -binary archivo.txt     # Binario
```

A.2. Comandos de cifrado

Listing A.2: Comandos para cifrado simétrico

```
1 # Listar cifradores disponibles
2 openssl enc -list
3
4 # Cifrado básico
5 openssl enc -aes-256-cbc -in archivo.txt -out archivo.bin \
6     -pass pass:"contraseña"
7
8 # Con información de derivación
9 openssl enc -aes-256-cbc -in archivo.txt -out archivo.bin \
10     -pass pass:"contraseña" -pbkdf2 -p
11
```

```
12 # Descifrado
13 openssl enc -aes-256-cbc -d -in archivo.bin -out archivo.txt \
14     -pass pass:"contraseña" -pbkdf2
```

A.3. Comandos para manipulación de archivos

Listing A.3: Comandos útiles

```
1 # Ver información del archivo
2 file archivo.bin
3 hexdump -C archivo.bin | head -n 5
4
5 # Eliminar primeros N bytes
6 dd if=entrada.bin of=salida.bin bs=1 skip=16
7
8 # Conocer tamaño
9 wc -c archivo.bin
```

Apéndice B

Configuración de ImageMagick

B.1. Instalación

En macOS:

```
1 brew install imagemagick
2 brew install ghostscript
```

En Linux:

```
1 apt install imagemagick
2 apt install ghostscript
```

B.2. Conversiones útiles

Listing B.1: Conversiones de imagen

```
1 # PNG a PGM
2 convert imagen.png imagen.pgm
3
4 # PGM a PNG
5 convert imagen.pgm imagen.png
6
7 # Crear imagen de prueba
8 convert -size 200x200 xc:red -fill blue \
9     -draw 'rectangle 0,0 100,200' test.png
```