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: 4ž 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

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: entre 150 y 200 caracteres

- Contenido legible
- Sin caracteres especiales problemáticos

Listing 2.1: Contenido del archivo de prueba

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

2.3.2. Comandos utilizados

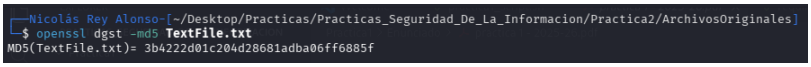
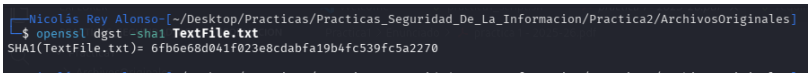
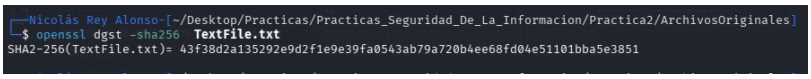
Listing 2.2: Cálculo de resúmenes con OpenSSL

```
# MD5
openssl dgst -md5 archivo.txt

# SHA-1
openssl dgst -sha1 archivo.txt

# SHA-256
openssl dgst -sha256 archivo.txt

# Formato con separadores
openssl dgst -sha256 -c archivo.txt

# Formato binario
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	Valor del hash
MD5	
SHA-1	
SHA-256	
SHA-512	
Whirlpool	

2.4.2. Verificación de tamaño

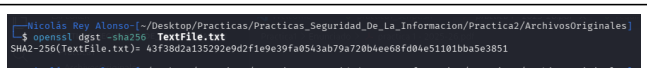
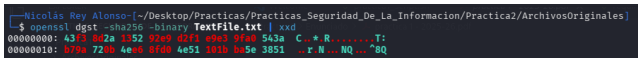
Los tamaños de los resúmenes coinciden con las especificaciones teóricas:

- 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:

Cuadro 2.3: Comparación de hash con cambio mínimo

Escenario	Hash SHA-256
Archivo original	
Con un carácter modificado	

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

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:

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

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

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

3.5.2. Cifrado con contraseña

Listing 3.2: Cifrado con contraseña y PBKDF2

```
openssl enc -aes-256-cbc -pbkdf2 -in archivo.txt \
    -out archivo.bin -pass pass:"contraseña"

# Con información de derivación
openssl enc -aes-256-cbc -pbkdf2 -in archivo.txt \
    -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:

$$\text{Tamaño\_cifrado} = 16 + \text{ceil}\left(\frac{\text{Tamaño\_original}}{T_{\text{bloque}}}\right) \times T_{\text{bloque}} \tag{3.2}$$

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

```
# Obtener clave e IV derivados
openssl enc -aes-256-cbc -pbkdf2 -in archivo.txt \
    -out archivo.bin -pass pass:"contraseña" -p

# Eliminar los 16 bytes de sal
dd if=archivo.bin of=archivo_sin_sal.bin bs=1 skip=16

# Descifrar con clave e IV
openssl enc -aes-256-cbc -d -in archivo_sin_sal.bin \
    -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:

$$C_i = E(K, P_i) \tag{4.1}$$

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

```
# Convertir a PGM
convert imagen.png imagen.pgm

# Separar cabecera y datos
head -n 3 imagen.pgm > cabecera.txt
tail -n +4 imagen.pgm > datos.bin

# Cifrado ECB (INSEGURO)
openssl enc -aes-256-ecb -in datos.bin -out datos_ecb.bin \
    -K 0123456789ABCDEF... -nosalt

# Reconstruir y convertir
cat cabecera.txt datos_ecb.bin > imagen_cifrada.pgm
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	*Se distinguen patrones de la imagen original*	Crítica
CBC	*Imagen completamente aleatoria*	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_operat 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

```
# Listar algoritmos disponibles
openssl dgst -list

# Diferentes formatos
openssl dgst -md5 archivo.txt
openssl dgst -sha256 -c archivo.txt          # Con separadores
openssl dgst -sha256 -binary archivo.txt     # Binario
```

A.2. Comandos de cifrado

Listing A.2: Comandos para cifrado simétrico

```
# Listar cifradores disponibles
openssl enc -list

# Cifrado básico
openssl enc -aes-256-cbc -in archivo.txt -out archivo.bin \
    -pass pass:"contraseña"

# Con información de derivación
openssl enc -aes-256-cbc -in archivo.txt -out archivo.bin \
    -pass pass:"contraseña" -pbkdf2 -p

```

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

A.3. Comandos para manipulación de archivos

Listing A.3: Comandos útiles

```
# Ver información del archivo
file archivo.bin
hexdump -C archivo.bin | head -n 5

# Eliminar primeros N bytes
dd if=entrada.bin of=salida.bin bs=1 skip=16

# Conocer tamaño
wc -c archivo.bin
```

Apéndice B

Configuración de ImageMagick

B.1. Instalación

En macOS:

```
brew install imagemagick
brew install ghostscript
```

En Linux:

```
apt install imagemagick
apt install ghostscript
```

B.2. Conversiones útiles

Listing B.1: Conversiones de imagen

```
# PNG a PGM
convert imagen.png imagen.pgm

# PGM a PNG
convert imagen.pgm imagen.png

# Crear imagen de prueba
convert -size 200x200 xc:red -fill blue \
    -draw 'rectangle 0,0 100,200' test.png
```