14/10/25

Redes de computadoras II

Algoritmos de encriptación

Alumno: Jared López Toledo
Profesor: Ing. Carlos Mijangos Jiménez

Contenido

Algoritmos de encriptación

Este documento integra la información de tres algoritmos fundamentales en criptografía y procesamiento de datos: RSA para cifrado asimétrico, MD5 para funciones hash, y Base64 para codificación. Cada algoritmo cumple propósitos específicos y opera bajo principios matemáticos diferentes.

RSA (Rivest-Shamir-Adleman)

Definición y Propósito

RSA es un algoritmo de cifrado asimétrico diseñado para garantizar la confidencialidad y autenticidad de la información. Sus aplicaciones principales incluyen:

- Cifrado seguro de datos
- Creación de firmas digitales
- Establecimiento de conexiones seguras (HTTPS/SSL)
- Transacciones bancarias y comunicaciones protegidas

Fundamentos Técnicos

Generación de Claves

```
# Proceso matemático de generación de claves
p = 61  # Número primo grande
q = 53  # Número primo grande
n = p * q  # Módulo público (3233)
φ(n) = (p-1) * (q-1)  # Función totient de Euler (3120)
e = 17  # Exponente público (coprimo con φ(n))
d = 2753  # Exponente privado (inverso modular de e mod φ(n))
```

Proceso de Cifrado/Descifrado

```
# Cifrado: C = M^e mod n
mensaje = 65
texto_cifrado = pow(mensaje, e, n)  # 2790

# Descifrado: M = C^d mod n
texto_original = pow(texto_cifrado, d, n)  # 65
```

Mecanismo de Seguridad

La seguridad de RSA se basa en la dificultad computacional de factorizar números enteros grandes en sus factores primos. Esta "función trampa" permite que multiplicar dos números primos grandes sea computacionalmente fácil, pero revertir este proceso sea extremadamente difícil.

Características Clave

- Par de claves: Pública (compartida) y Privada (secreta)
- Reversibilidad condicional: Solo con la clave correspondiente
- Base matemática: Teoría de números y aritmética modular

MD5 (Message Digest 5)

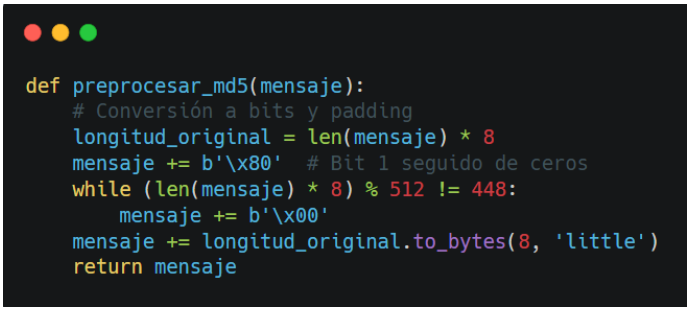
Definición y Propósito

MD5 es un algoritmo de función hash criptográfica que genera una "huella digital" única de datos. Sus usos incluyen:

- Verificación de integridad de archivos
- Detección de modificaciones en datos
- Validación de descargas

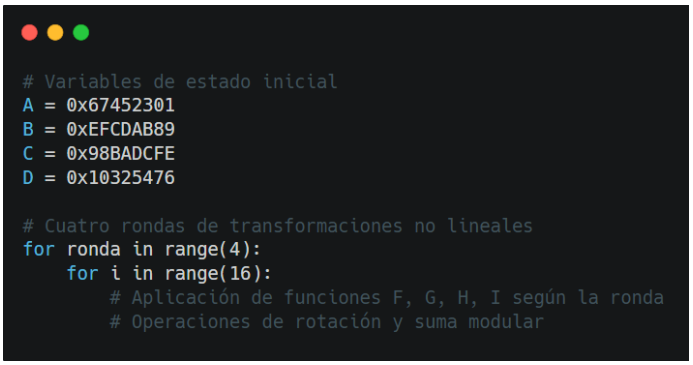
Proceso Técnico

Preprocesamiento



```python
def preprocesar_md5(mensaje):
    # Conversión a bits y padding
    longitud_original = len(mensaje) * 8
    mensaje += b'\x80'  # Bit 1 seguido de ceros
    while (len(mensaje) * 8) % 512 != 448:
        mensaje += b'\x00'
    mensaje += longitud_original.to_bytes(8, 'little')
    return mensaje
```

Procesamiento Principal



```
# Variables de estado inicial
A = 0x67452301
B = 0xEFCDAB89
C = 0x98BADCFE
D = 0x10325476

# Cuatro rondas de transformaciones no lineales
for ronda in range(4):
    for i in range(16):
        # Aplicación de funciones F, G, H, I según la ronda
        # Operaciones de rotación y suma modular
```

Características del Hash

- Tamaño fijo: 128 bits (32 caracteres hexadecimales)
- Determinismo: Misma entrada → mismo hash
- Avalancha: Pequeños cambios → hash completamente diferente
- Irreversibilidad: No se puede reconstruir la entrada desde el hash

Estado de Seguridad Actual

MD5 se considera criptográficamente vulnerable debido a:

- Colisiones demostradas (diferentes entradas producen mismo hash)
- Ataques eficientes de preimagen
- Recomendación: Usar alternativas más seguras como SHA-256

Base64

Definición y Propósito

- Base64 es un sistema de codificación (no cifrado) diseñado para representar datos binarios en formato de texto ASCII. Aplicaciones principales:
- Adjuntar archivos en correos electrónicos
- Almacenar datos binarios en formatos XML/JSON
- Transmitir datos binarios a través de protocolos texto

Proceso de Codificación

Algoritmo de Conversión

```python
def codificar_base64(datos):
    # 1. Conversión de datos binarios a secuencia de bits
    bits = ''.join(format(byte, '08b') for byte in datos)

    # 2. División en grupos de 6 bits
    grupos = [bits[i:i+6] for i in range(0, len(bits), 6)]

    # 3. Mapeo a tabla Base64
    tabla = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"
    resultado = ''.join(tabla[int(grupo, 2)] for grupo in grupos)

    # 4. Agregar padding si es necesario
    return resultado + '=' * ((4 - len(resultado) % 4) % 4)
```

Ejemplo Práctico

```
Texto: "Man"
Bytes: [77, 97, 110] → Binario: 01001101 01100001 01101110
Grupos de 6 bits: 010011 010110 000101 101110
Decimal: 19 22 5 46
Base64: T W F u → "TWFu"
```

Tabla de Caracteres Base64

Rango	Caracteres	Valores
0-25	A-Z	025
26-51	a-z	26-51
52-61	0-9	52-61
62	+	62
63	/	63
Padding	=	Relleno

Características Esenciales

- Reversibilidad completa: Codificación ↔ Decodificación
- Sin seguridad: No protege información
- Expansión de datos: ~33% de overhead
- Propósito: Compatibilidad y transporte seguro

Análisis Comparativo

Aspecto	RSA	MD5	Base64
Tipo	Criptografía asimétrica	Función hash	Codificación
Seguridad	Alta (con claves adecuadas)	Vulnerable	Ninguna
Reversibilidad	Condicional (con clave)	Irreversible	Totalmente reversible
Entrada/Salida	Texto/Texto	Cualquier dato/Hash 128-bit	Binario/texto ASCII
Propósito Principal	Confidencialidad/Autenticación	Integridad	Compatibilidad
Estado Actual	Seguro (claves >=2048 bits)	Obsoleto para seguridad	Ampliamente usado

Conclusión

Estos tres algoritmos representan categorías distintas en el procesamiento de datos:

- RSA proporciona seguridad mediante matemática avanzada
- MD5 (históricamente) ofrecía verificación de integridad
- Base64 facilita la compatibilidad entre sistemas

La comprensión de sus diferencias fundamentales especialmente la distinción entre cifrado, hash y codificación es crucial para implementar soluciones de seguridad apropiadas en el contexto moderno.