APLICACIONES CRIPTOGRÁFICAS

Implementación de Firma Digital y DSA

1 DE DICIEMBRE DE 2025

GRUPO 3
Ariel Elizalde – Alexis Troya – Mauricio López

Contenido

Firma Digital

1. Definición de la Tecnología DSA

El **Digital Signature Algorithm (DSA)** es un esquema de **criptografía de clave pública** propuesto en 1991 por el Instituto Nacional de Estándares y Tecnología (NIST) de EE. UU. y adoptado como el **Estándar de Firma Digital (DSS)** bajo la publicación FIPS 186.

Características Clave:

- **Propósito Exclusivo:** A diferencia de otros algoritmos asimétricos como RSA (que puede cifrar y firmar), DSA está diseñado **exclusivamente para la generación y verificación de firmas digitales**.
- **Base Matemática:** Su seguridad se fundamenta en la dificultad computacional del **Problema del Logaritmo Discreto (DLP)** en campos finitos, un problema que es inherentemente difícil de resolver.
- **Funcionalidad:** Proporciona dos servicios esenciales de seguridad de la información:
    1. **Autenticidad (o No Repudio):** Demuestra que el mensaje fue creado por el poseedor de la clave privada correspondiente, impidiendo al firmante negar su autoría.
    2. **Integridad:** Garantiza que el mensaje no ha sido alterado desde el momento en que se aplicó la firma.

El proceso de DSA involucra el uso de una **clave privada** para firmar un resumen del mensaje (generado por una función hash) y la **clave pública** para verificar esa firma.

2. Algoritmos Criptográficos Involucrados

Los siguientes algoritmos son fundamentales para la implementación de la firma digital con DSA, como se observa en el código y su contexto matemático:

Tipo de Algoritmo	Nombre	Función
Firma Asimétrica	**DSA (Digital Signature Algorithm)**	Algoritmo principal para generar el par de claves (pública/privada) y producir la firma.

Tipo de Algoritmo	Nombre	Función
Función Hash	**SHA-256** (Secure Hash Algorithm - 256 bits)	Se utiliza para calcular el *resumen* (*message digest*) del mensaje. El código firma este resumen, no el mensaje completo, asegurando la integridad.
Generación de Números Aleatorios (RNG)	**CSPRNG**	- Cryptographically Secure Pseudo-Random Number Generator - Usado para generar claves privadas y valores "k" en el proceso de firma - Crítico: Un RNG débil compromete toda la seguridad
Algoritmos de Generación de Claves		- Generación de parámetros de dominio (p, q, g) - Generación de pares de claves (privada/pública)}.

3. Protocolos Criptográficos Involucrados

DSA no es un protocolo de comunicación en sí mismo, sino un componente criptográfico que opera dentro de protocolos más grandes para proporcionar servicios de autenticación y seguridad:

1. FIPS / DSS (Digital Signature Standard) — definición normativa de DSA (generación de parámetros, formatos). (NIST FIPS 186 series).

2. RFC 6979 (Deterministic DSA/ECDSA) — procedimiento recomendado para generación determinística del nonce en DSA/ECDSA.

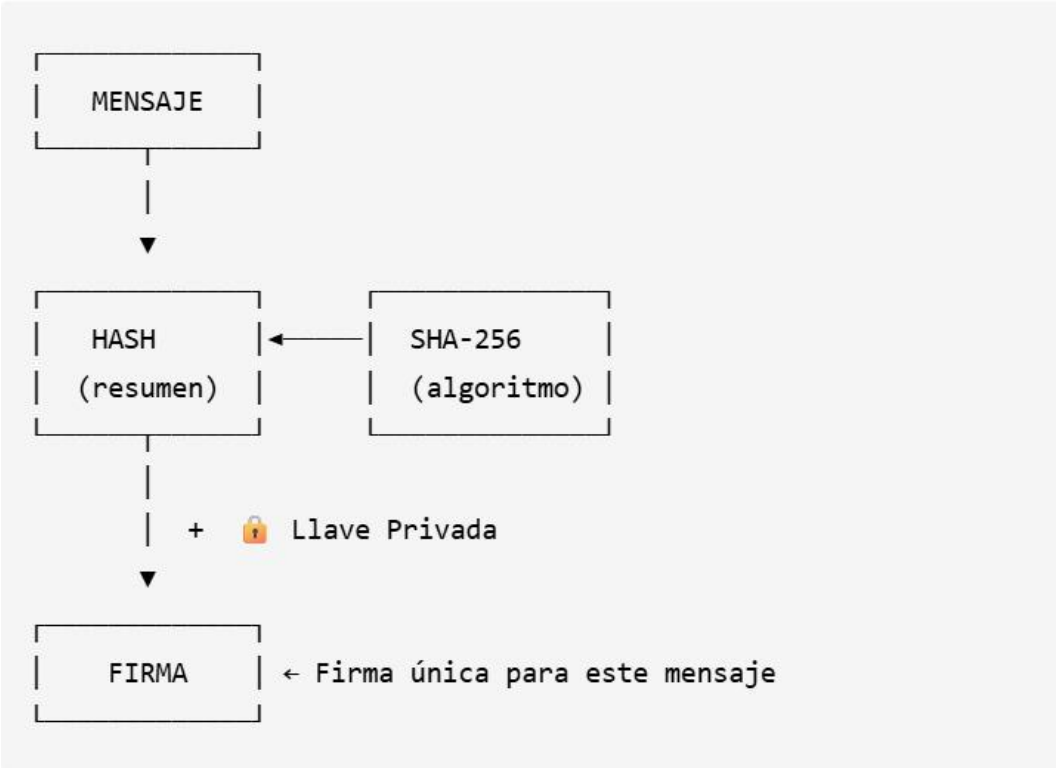
3. TLS (SSL/TLS) — en versiones históricas TLS/TLS 1.2 manejan DSA/DSS como uno de los algoritmos de firma aceptados (uso en certificados y en autenticación).

4. S/MIME / X.509 (PKI) — sistemas de firma de correo y certificados X.509 pueden contener claves DSA y usarlas para firmar/validar mensajes y certificados.

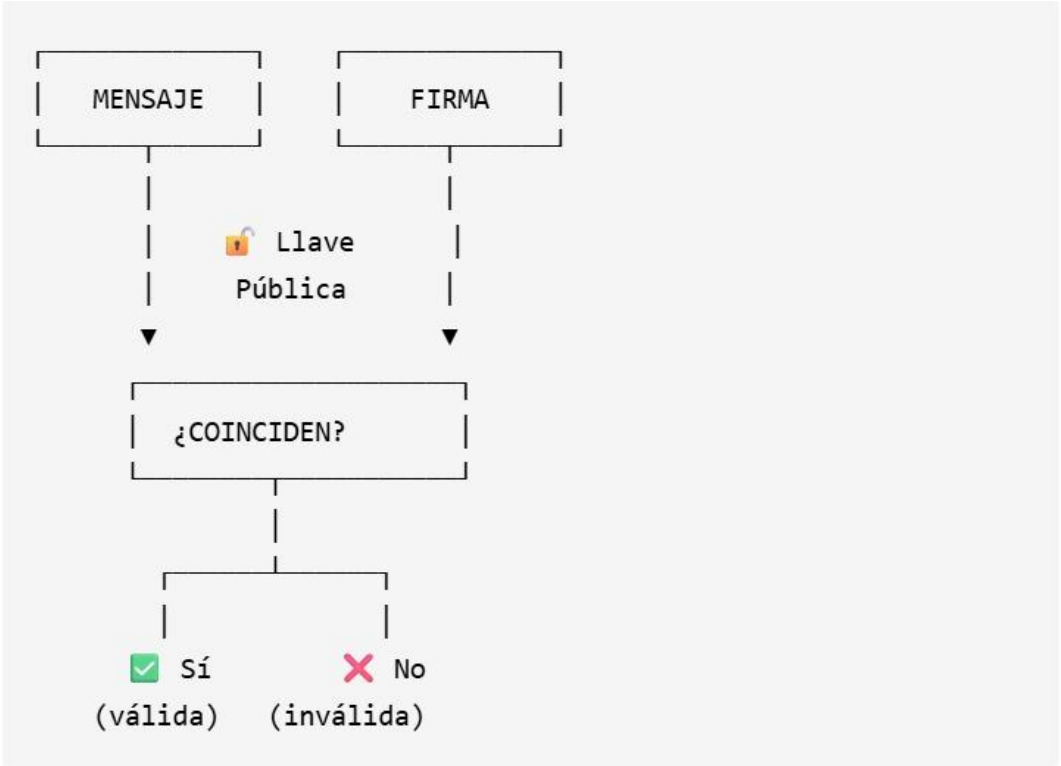
5. Implementaciones y librerías criptográficas (OpenSSL, cryptography, etc.) implementan DSA y/o sus variantes; en la práctica muchas migraciones han ido a ECDSA/EdDSA por eficiencia y tamaño de llave.

4. Diseño esquemático de su funcionamiento

Proceso de firma:



Proceso de validación:



5. Escenarios Frecuentes de Utilización

El DSA se utiliza en cualquier entorno donde la autenticación y la integridad sean críticas:

1. **Firma de Documentos Oficiales y Legales:** Utilizado por gobiernos y empresas para aplicar la firma digital con valor legal, garantizando la identidad del firmante y la inalterabilidad del contenido, como en sistemas de historial clínico digital o contratos electrónicos.
2. **Validación de Certificados Digitales (PKI):** Esencial en la Infraestructura de Clave Pública. Las Autoridades de Certificación (CA) firman los certificados X.509 utilizando DSA (o ECDSA/RSA) para establecer una cadena de confianza y validar la identidad de sitios web (HTTPS).
3. **Integridad de Software y Distribución de Código:** Utilizado por desarrolladores y distribuidores de software para firmar los ejecutables y las actualizaciones. Esto permite al usuario verificar que el *software* no ha sido manipulado por atacantes (malware) y que proviene de una fuente legítima.

6. Preguntas relacionadas

1. **Verdadero o Falso**

SHA-1 es seguro para crear nuevas firmas digitales en aplicaciones modernas y se recomienda su uso para garantizar la integridad de mensajes a largo plazo.

Respuesta:

Falso.

SHA-1 tiene vulnerabilidades prácticas (colisiones) y no se recomienda para nuevas firmas; usa SHA-256 o superiores (SHA-2 / SHA-3).

2. Opción Múltiple (Una respuesta)

¿Cuál de los siguientes esquemas de firma se recomienda para nuevas implementaciones de firma con RSA?

a) RSASSA-PKCS1-v1_5

b) RSA-PSS

c) RSA-OAEP

d) RSA-ES

Respuesta:

b) RSA-PSS.

RSA-PSS provee un esquema probabilístico más seguro para firmas que RSASSA-PKCS1-v1_5; RSA-OAEP es un esquema de cifrado (no firma).

3. Opción Múltiple (Múltiples respuestas)

Al verificar una firma digital en un entorno con PKI, ¿cuáles de las siguientes comprobaciones son relevantes? (Selecciona 3)

a) Calcular localmente el hash del mensaje y compararlo con lo que verifica la firma.

b) Confirmar que el certificado del firmante no esté revocado (OCSP/CRL).

c) Comprobar que la clave privada del firmante esté almacenada en un HSM.

d) Verificar la cadena de confianza del certificado hasta una CA confiable.

e) Asegurarse de que el mensaje fue cifrado con AES.

Respuestas correctas:

a), b) y d).

La verificación exige comprobar hash, estado de revocación y la cadena de confianza; la presencia de la clave privada en HSM es una práctica de seguridad pero no una comprobación durante la verificación, y el cifrado con AES no es requisito para la verificación de firma.

4. Emparejamiento

Relaciona cada algoritmo con su característica principal.

Columna A (Algoritmo) — Columna B (Característica)

DSA — a) Firma basada en curvas elípticas, tamaños de clave pequeños.

ECDSA — b) Estándar histórico de firma federal (originalmente FIPS 186).

Ed25519 — c) Proporciona resistencia y rendimiento con claves y firmas pequeñas (EdDSA).

RSA-PSS — d) Esquema de firma probabilístico recomendado para RSA.

Respuestas:

1–b, 2–a, 3–c, 4–d.

(Explicación breve: DSA es el algoritmo clásico del estándar; ECDSA es la versión sobre curvas; Ed25519 es EdDSA optimizado; RSA-PSS es el modo seguro para firmas RSA.)

5. Completar con una palabra

El estándar que define el formato para firmar objetos JSON en APIs y tokens (p. ej. tokens firmados en OAuth/OpenID) se denomina _____.

Respuesta: JWS.

JWS = JSON Web Signature, el formato que describe cómo aplicar firmas digitales a datos JSON (los JWT normalmente contienen un JWS como parte de su estructura).

Ejemplo de implementación práctica

firma_dsa_ejemplo.py > ...

```python
"""
Demostración de Firma Digital usando DSA (Digital Signature Algorithm)
======================================================================

Este ejemplo muestra cómo:
1. Generar un par de claves DSA (pública y privada)
2. Firmar un mensaje con la clave privada
3. Verificar la firma con la clave pública
"""

from cryptography.hazmat.primitives.asymmetric import dsa
from cryptography.hazmat.primitives import hashes
from cryptography.exceptions import InvalidSignature

print("=" * 70)
print("DEMOSTRACIÓN DE FIRMA DIGITAL DSA")
print("=" * 70)

# ============================================================================
# PASO 1: Generar claves DSA
# ============================================================================
print("\n📌 PASO 1: Generando par de claves DSA...")
private_key = dsa.generate_private_key(key_size=2048)
public_key = private_key.public_key()
print("   ✓ Clave privada generada (mantener en secreto)")
print("   ✓ Clave pública generada (compartir libremente)")


# ============================================================================
# PASO 2: Crear y firmar el mensaje
# ============================================================================
mensaje = b"Hola, esta es mi firma digital DSA"
print(f"\n📌 PASO 2: Firmando el mensaje...")
print(f"   Mensaje original: {mensaje.decode()}")

# Firmar el mensaje usando la clave privada
firma = private_key.sign(
    mensaje,
    hashes.SHA256()
)

print(f"   ✓ Firma generada ({len(firma)} bytes)")
print(f"   Firma (primeros 32 bytes en hex): {firma[:32].hex()}")

# ============================================================================
# PASO 3: Verificar la firma (mensaje original)
# ============================================================================
print("\n📌 PASO 3: Verificando la firma del mensaje original...")
try:
    public_key.verify(
        firma,
        mensaje,
        hashes.SHA256()
    )
    print("   ✓ ¡La firma es VÁLIDA! El mensaje no ha sido modificado.")
except InvalidSignature:
    print("   ❌ La firma NO es válida")

```

```python
# ============================================================================
# PASO 4: Intentar verificar con un mensaje alterado
# ============================================================================
print("\n📌 PASO 4: Probando con un mensaje ALTERADO...")
mensaje_alterado = b"Hola, esta es mi firma digital DSA modificada"
print(f"   Mensaje alterado: {mensaje_alterado.decode()}")

try:
    public_key.verify(
        firma,
        mensaje_alterado,
        hashes.SHA256()
    )
    print("   ✓ La firma es válida")
except InvalidSignature:
    print("   ❌ ¡La firma NO es válida! El mensaje fue modificado.")

# ============================================================================
# RESUMEN
# ============================================================================
print("\n" + "=" * 70)
print("RESUMEN:")
print("=" * 70)
print("• La firma digital garantiza la autenticidad e integridad del mensaje")
print("• Solo la clave privada puede crear la firma")
print("• Cualquiera con la clave pública puede verificar la firma")
print("• Si el mensaje cambia, la firma se invalida automáticamente")
print("=" * 70)

```

Referencias Bibliográficas (Formato IEEE)

[1] National Institute of Standards and Technology, *Digital Signature Standard (DSS)*, FIPS 186-5, 2023. [Online]. Disponible en: https://doi.org/10.6028/NIST.FIPS.186-5

[2] O. M. Semyonov, A. I. Chernykh, V. A. Evdokimov, y D. R. Tyncheva, "Eliminating Broadband Covert Channels in DSA-Like Signatures," en *Proc. IEEE Conf.*, 2021, pp. 1–6. doi: 10.1109/REDUNDANCY52534.2021.9606457

[3] M. Al-Haj y B. Tubaishat, "Using Digital Signature Standard Algorithm to Incorporate Non-invertibility in Private Digital Watermarking Techniques," en *Proc. ACIS Int. Conf. Softw. Eng., Artif. Intell., Netw. Parallel/Distrib. Comput.*, 2009, pp. 299–304. doi: 10.1109/SNPD.2009.89

[4] W. Li, S. Liu, y W. Liu, "Improved speed Digital Signature Algorithm based on modular inverse," en *Proc. Int. Conf. Inf. Sci. Technol.*, 2013, pp. 780–783. doi: 10.1109/MIC.2013.6758059

[5] H. Kaur, J. Singh, y J. Kaur, "Secure encryption with digital signature approach for Short Message Service," en *Proc. World Congr. Inf. Commun. Technol.*, 2012, pp. 1104–1108. doi: 10.1109/WICT.2012.6409184