Introduction to Cryptography

January 23, 2026

Introduction

Encryption is a fundamental technique used to protect sensitive information from unauthorized access. This experiment explores symmetric and asymmetric encryption, hashing, and digital signatures using standard cryptographic tools to understand how confidentiality, integrity, and authentication are achieved in real-world security systems.

Requirements

- Kali Linux / Linux OS
- OpenSSL
- sha256sum
- Terminal access
- Text editor

Learn Symmetric vs Asymmetric Encryption

Symmetric Encryption

- Uses the same key for encryption and decryption.
- Fast and suitable for encrypting large amounts of data.
- Common example: AES (Advanced Encryption Standard).

Asymmetric Encryption

- Uses a pair of keys: a public key and a private key.
- Slower compared to symmetric encryption.
- Mainly used for secure key exchange and authentication.
- Common example: RSA (Rivest–Shamir–Adleman).

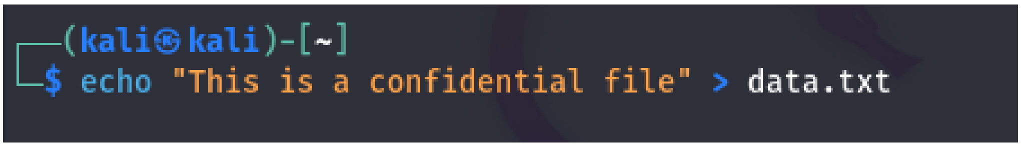
Conclusion

Real-world security systems use both symmetric and asymmetric encryption together, known as hybrid encryption, to achieve efficiency and strong security.

Procedure

Step 1: Create a Sample File

```
echo "This is a confidential file" > data.txt
```



```
┌──(kali㉿kali)-[~]
└─$ echo "This is a confidential file" > data.txt
```

Figure 1: Creating a Sample Text File

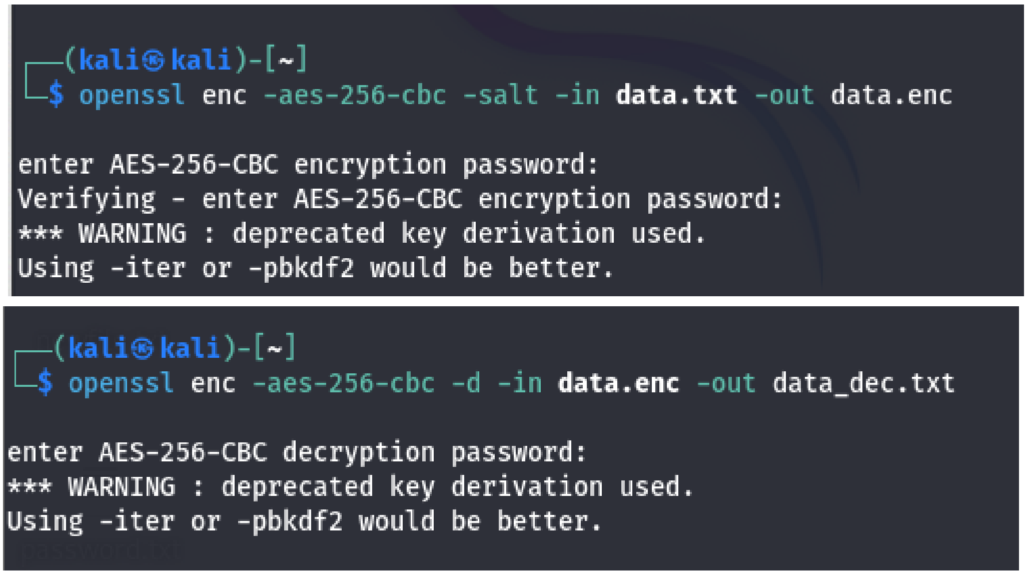
Step 2: AES File Encryption

Encryption

```
openssl enc -aes-256-cbc -salt -in data.txt -out data.enc
```

Decryption

```
openssl enc -aes-256-cbc -d -in data.enc -out data_dec.txt
```



```
┌──(kali㉿kali)-[~]
└─$ openssl enc -aes-256-cbc -salt -in data.txt -out data.enc

enter AES-256-CBC encryption password:
Verifying - enter AES-256-CBC encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
```

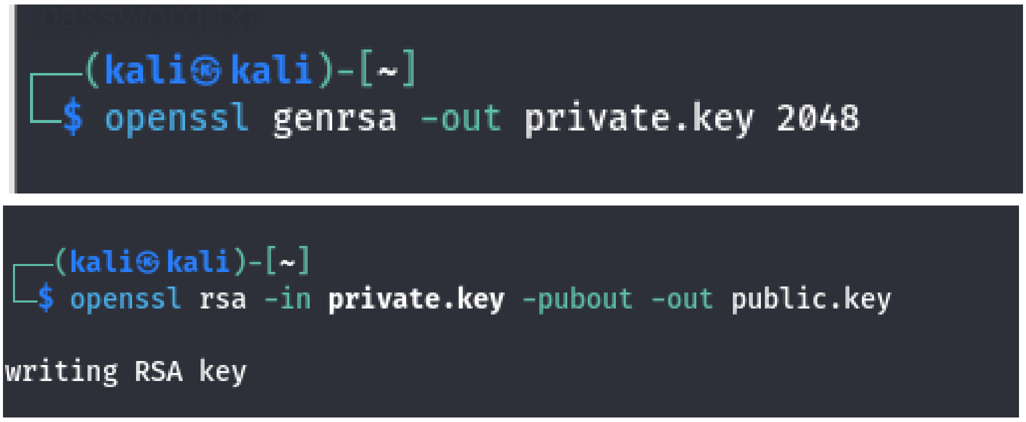
```
┌──(kali㉿kali)-[~]
└─$ openssl enc -aes-256-cbc -d -in data.enc -out data_dec.txt

enter AES-256-CBC decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
```

Figure 2: AES Encryption and Decryption

Step 3: RSA Key Generation

```
openssl genrsa -out private.key 2048
openssl rsa -in private.key -pubout -out public.key
```



```
┌──(kali㉿kali)-[~]
└─$ openssl genrsa -out private.key 2048
```

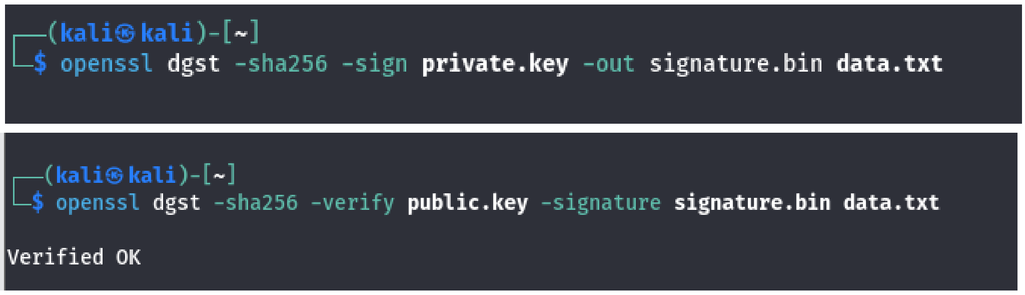
```
┌──(kali㉿kali)-[~]
└─$ openssl rsa -in private.key -pubout -out public.key

writing RSA key
```

Figure 3: RSA Key Generation

Step 4: Digital Signature

```
openssl dgst -sha256 -sign private.key -out signature.bin data.
   txt
openssl dgst -sha256 -verify public.key -signature signature.bin
   data.txt
```



```
┌──(kali㉿kali)-[~]
└─$ openssl dgst -sha256 -sign private.key -out signature.bin data.txt
```

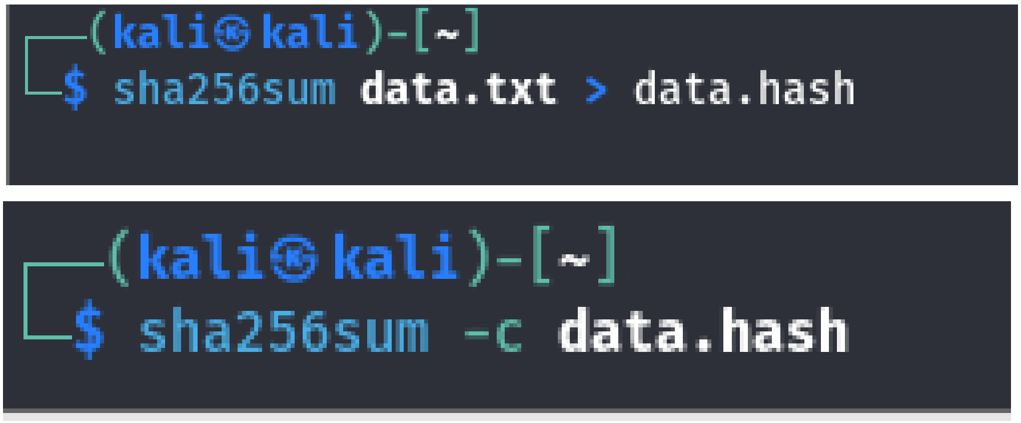
```
┌──(kali㉿kali)-[~]
└─$ openssl dgst -sha256 -verify public.key -signature signature.bin data.txt

Verified OK
```

Figure 4: Digital Signature Verification

Step 5: Hashing and Integrity Verification

```
sha256sum data.txt > data.hash
sha256sum -c data.hash
```



```
┌──(kali㉿kali)-[~]
└─$ sha256sum data.txt > data.hash
```

```
┌──(kali㉿kali)-[~]
└─$ sha256sum -c data.hash
```

Figure 5: SHA-256 Hash Verification

Understand Digital Signatures

Purpose

- To verify the authenticity of the sender.
- To ensure data integrity.
- To provide non-repudiation.

Process

- The file is first hashed using a cryptographic hash function.
- The generated hash is encrypted using the sender's private key.
- The receiver decrypts the hash using the sender's public key to verify the signature.

Applications

- Software updates
- Digital certificates
- Secure email communication

Compare Encryption Algorithms

Algorithm	Type	Speed	Security	Use Case
AES	Symmetric	Fast	Very High	File and disk encryption
RSA	Asymmetric	Slow	High	Secure key exchange
DES	Symmetric	Fast	Weak	Deprecated
ECC	Asymmetric	Fast	Very High	Mobile devices and IoT

Real-World Usage of Encryption

- **HTTPS:** Uses RSA or ECC for secure key exchange and AES for encrypting data in transit.
- **VPN:** Uses AES encryption along with digital certificates to secure network communication.
- **Email (PGP):** Uses RSA for encryption and hashing to ensure confidentiality and integrity.
- **Cloud Storage:** Uses AES encryption to protect data at rest.

Cryptography Experiment Report

Experiment Title	Cryptography: Encryption, Hashing, and Digital Signatures
Aim	To study and implement cryptographic techniques including symmetric encryption, asymmetric encryption, hashing, and digital signatures using standard tools.
Tools Used	Kali Linux / Linux OS, OpenSSL, sha256sum, Terminal
Cryptographic Techniques	• Symmetric Encryption (AES) • Asymmetric Encryption (RSA) • Digital Signatures • Hashing (SHA-256)
Procedure Summary	• Created a sample text file. • Encrypted and decrypted the file using AES. • Generated RSA public and private keys. • Created and verified a digital signature. • Generated and verified SHA-256 hash for integrity.
Observations	• AES encryption securely protected file data. • RSA key pair generation was successful. • Digital signatures verified authenticity and integrity. • Hash values changed when file content was altered.
Result	All cryptographic operations were performed successfully using OpenSSL and Linux utilities.
Conclusion	The experiment demonstrates how cryptography ensures confidentiality, integrity, and authentication in secure communication systems.
Real-World Applications	HTTPS, VPNs, Secure Email (PGP), Cloud Storage