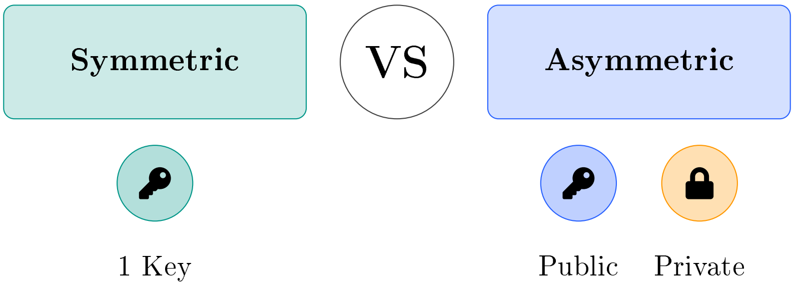
Cryptography

Symmetric vs Asymmetric



Symmetric

VS

Asymmetric

1 Key

Public

Private

Practical Demonstration

File Encryption and Decryption

Author: Ahmed Dinari
Date: November 2025
Course: Computer Security

Contents

1 Introduction

Cryptography is the art of protecting information by transforming it into an unreadable format for anyone who does not possess the decryption key.

Objectives of Cryptography

- **Confidentiality**: Only authorized persons can read the message
- **Integrity**: The message has not been modified
- **Authentication**: The sender's identity is verified
- **Non-repudiation**: The sender cannot deny having sent the message

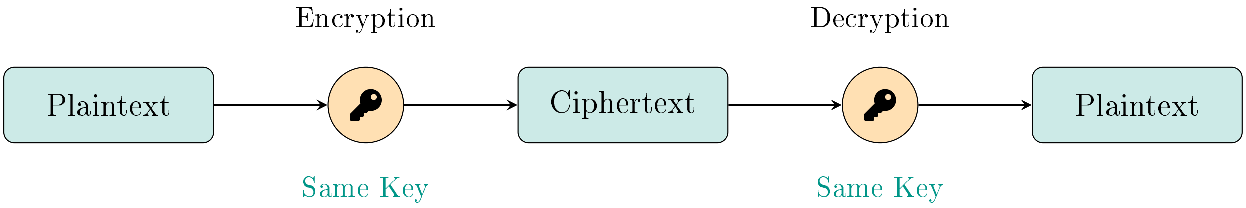
There are two main families of encryption:

1. **Symmetric Encryption** (AES, DES, 3DES)
2. **Asymmetric Encryption** (RSA, ECC, DSA)

2 Symmetric Encryption

2.1 Principle

Symmetric encryption uses **a single key** to both encrypt AND decrypt data.



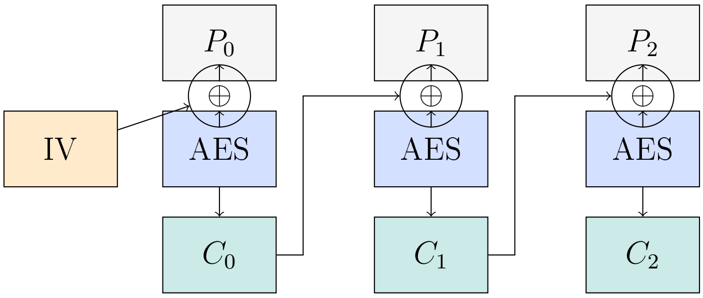
2.2 AES Algorithm (Advanced Encryption Standard)

AES is the most widely used symmetric algorithm today.

Property	**AES-128**	**AES-192**	**AES-256**
Key Size	128 bits	192 bits	256 bits
Number of Rounds	10	12	14
Block Size	128 bits	128 bits	128 bits
Security	High	Very High	Maximum

2.3 CBC Mode (Cipher Block Chaining)

In our demonstration, we use **CBC** mode:



- **IV** (Initialization Vector): Random vector for the first block
- Each block depends on the previous block ⇒ more security

2.4 Python Code - Symmetric Encryption

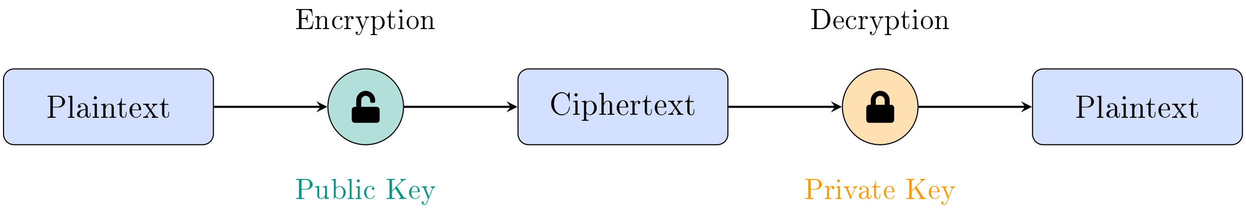
```
from cryptography.hazmat.primitives.ciphers import Cipher,
    algorithms, modes
import os

# Generate a 256-bit AES key
key = os.urandom(32)

# Generate a random IV
iv = os.urandom(16)

# Create AES-CBC cipher
cipher = Cipher(algorithms.AES(key), modes.CBC(iv))

# Encrypt
encryptor = cipher.encryptor()
ciphertext = encryptor.update(padded_data) + encryptor.finalize()

# Decrypt
decryptor = cipher.decryptor()
plaintext = decryptor.update(ciphertext) + decryptor.finalize()
```

3 Asymmetric Encryption

3.1 Principle

Asymmetric encryption uses **two different keys**:

- **Public Key**: Shared with everyone, used to **encrypt**
- **Private Key**: Kept secret, used to **decrypt**



3.2 RSA Algorithm

RSA (Rivest-Shamir-Adleman) is based on the difficulty of factoring large prime numbers.

RSA Mathematics

Key Generation:

1. Choose two large prime numbers p and q
2. Calculate $n = p \times q$
3. Calculate $\phi(n) = (p-1)(q-1)$
4. Choose e such that $\gcd(e, \phi(n)) = 1$ (usually $e = 65537$)
5. Calculate $d = e^{-1} \mod \phi(n)$

Keys:

- Public key: (n, e)
- Private key: (n, d)

Operations:

$$\begin{aligned} \text{Encryption: } C &= M^e \mod n \\ \text{Decryption: } M &= C^d \mod n \end{aligned}$$

3.3 OAEP Padding

For more security, RSA uses **OAEP** (Optimal Asymmetric Encryption Padding):

- Adds randomness to the message
- Protects against chosen-ciphertext attacks
- Uses hash functions (SHA-256)

3.4 Python Code - Asymmetric Encryption

```python
from cryptography.hazmat.primitives.asymmetric import rsa, padding
from cryptography.hazmat.primitives import hashes

# Generate a 2048-bit RSA key pair
```

```
private_key = rsa.generate_private_key(
    public_exponent=65537,
    key_size=2048
)
public_key = private_key.public_key()

# Encrypt with public key
ciphertext = public_key.encrypt(
    message,
    padding.OAEP(
        mgf=padding.MGF1(algorithm=hashes.SHA256()),
        algorithm=hashes.SHA256(),
        label=None
    )
)

# Decrypt with private key
plaintext = private_key.decrypt(ciphertext, padding.OAEP(...))
```

4 Comparison: Symmetric vs Asymmetric

Criteria	**Symmetric**	**Asymmetric**
Number of Keys	1 (same key)	2 (public/private)
Speed	Fast ★★★	Slow ★
Key Size	128-256 bits	2048-4096 bits
Key Exchange	Difficult	Easy
Max Data Size	Unlimited	190 bytes (RSA-2048)
Algorithms	AES, DES, 3DES	RSA, ECC, DSA
Use Cases	Files, Disks	Signatures, HTTPS

4.1 Advantages and Disadvantages

Symmetric

Advantages:

- ✓ Very fast
- ✓ Efficient for large volumes
- ✓ Short keys

Disadvantages:

- × Key exchange is difficult
- × Complex key management

Asymmetric

Advantages:

- ✓ Secure key exchange
- ✓ Digital signatures
- ✓ Non-repudiation

Disadvantages:

- × Slow
- × Long keys
- × Size limit

5 Practical Demonstration

5.1 Scenario

In our demonstration, we:

1. **Created a confidential file**: `original_file.txt`
2. **Encrypted the file** with AES (symmetric) ⇒ `encrypted_file.txt`
3. **Decrypted the file** ⇒ `decrypted_file.txt`

5.2 Results

Original File

```
============================================
CONFIDENTIAL FILE
============================================
Name: Ahmed Dinari
Subject: Security Lab Work
Date: November 2025
```

Encrypted File (Unreadable)

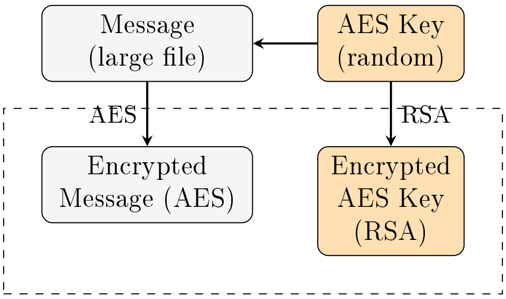
```
tlJzhikFecZWMIfhqcXY12iKbn/AfPiXSa5dvu
onT2zmEflP0N6/TlZV8SU3+11CrClSc7oZkEZr
55XG8Xkl2wQy...
```

Decrypted File (Same as Original)

```
============================================
CONFIDENTIAL FILE
============================================
Name: Ahmed Dinari
Subject: Security Lab Work
Date: November 2025
```

6 Hybrid Encryption (Real-World Usage)

In practice, both methods are combined:



Sent to recipient

Example: HTTPS (TLS/SSL)

1. The server sends its **RSA public key**
2. The client generates a **random AES key**
3. The client **encrypts the AES key** with RSA and sends it
4. All data is **encrypted with AES**

⇒ Combining the **speed of AES** with the **security of RSA**!

7 Conclusion

Key Takeaways

🔑 **Symmetric**: 1 key, fast, for files

🔒 **Asymmetric**: 2 keys, secure, for key exchange

🛡 **Hybrid**: Combines both (HTTPS, Email)

Thank you for your attention!

Questions?