Week 6: Hybrid Encryption

HYBRID ENCRYPTION

Hybrid encryption is a **cryptographic design pattern** that combines the strengths of **symmetric** and **asymmetric** encryption. It's used in real-world secure systems like **HTTPS, PGP, TLS**, etc.

Why Hybrid?

RSA (Asymmetric): Secure but slow. Ideal for small data like keys.
AES (Symmetric): Fast and secure. Great for encrypting large files/messages.
So we **encrypt the AES key with RSA**, and **encrypt the actual data with AES**.

Real-World Analogy

Imagine sending a **locked treasure chest**:

- You use **AES** to lock the chest (fast, secure).
- You put the **key to the chest inside a small vault**.
- You then lock this vault using **RSA (Bob's public key)**.
- Only Bob, with his **RSA private key**, can open the vault, get the AES key, and unlock the chest.

Hybrid Encryption: Full Workflow

Let's say Alice wants to send Bob a secure message.

Encryption Process (Sender side)

Step 1: Generate a symmetric key

- A random **AES key** is generated (128/256 bits).
- This key is used to encrypt the actual data.

```
AES_KEY = Random()
```

Step 2: Encrypt data using AES

- AES is used in a secure mode like **EAX**, **GCM**, or **CBC + HMAC**.
- Produces: `ciphertext`, `nonce`, and `authentication tag`.

```
ciphertext = AES_encrypt(data, AES_KEY)
```

Step 3: Encrypt AES key using RSA

- Bob's **public key** is used to encrypt the AES key securely using **RSA + OAEP**

enc_AES_KEY = RSA_encrypt(AES_KEY, Bob's public key)

Step 4: Send the bundle

- Send the following securely:
  - Encrypted AES key `enc_AES_KEY`
  - AES-encrypted `ciphertext`
  - AES `nonce` and `tag` (needed for decryption)

Send = { enc_AES_KEY, ciphertext, nonce, tag }

Decryption Process (Receiver side)

Step 1: Decrypt AES key

- Bob uses his **private key** to decrypt `enc_AES_KEY` to recover the original AES key.

```
AES_KEY = RSA_decrypt(enc_AES_KEY, Bob's private key)
```

Step 2: Decrypt message

- Using AES key + nonce + tag, Bob decrypts the message.

```
plaintext = AES_decrypt(ciphertext, AES_KEY, nonce, tag)
```

Security Analysis

Advantages:

- **Performance**: Large data is encrypted fast with AES.
- **Security**: Key exchange is secure due to RSA.
- **Scalability**: You don't have to pre-share AES keys.

Building blocks used:

- **AES (Advanced Encryption Standard)**: Fast symmetric block cipher.
- **RSA (Rivest–Shamir–Adleman)**: Public-key encryption.
- **PKCS#1 OAEP**: Padding scheme to prevent attacks on RSA.
- **EAX/GCM mode**: Provides **confidentiality + authenticity** (integrity).