

Secure Connection Flow

This document explains how the `sender`, `unsecureTunnel`, and `receiver` establish a secure communication channel using a Hybrid Encryption scheme (RSA + DES).

Components

1. **Receiver** (`receiver.py`): The server that holds the RSA Private Key and waits for messages.
2. **Unsecure Tunnel** (`unsecureTunnel.py`): A "Man-in-the-Middle" proxy that forwards traffic between Sender and Receiver. It can see the traffic but cannot decrypt the secure messages.
3. **Sender** (`sender.py`): The client that initiates the connection and sends encrypted messages.

Protocol Steps

1. Initialization (Receiver)

- **Action:** When `receiver.py` starts.
- **Operation:** It generates a fresh **RSA 2048-bit Key Pair** (Public Key and Private Key).
- **State:** The Receiver is now listening on port `65433`.

2. Connection & Public Key Exchange

- **Action:** `sender.py` connects to `unsecureTunnel.py` (port `65432`), which forwards the connection to `receiver.py`.
- **Handshake:**

3. Session Key Exchange (Hybrid Encryption)

- **Goal:** Establish a shared symmetric key for fast encryption (DES).
- **Action:**
- **Result:** Both Sender and Receiver now possess the same **DES Key**. The Tunnel only saw the encrypted blob and cannot derive the key.

4. Secure Messaging

- **Action:** User types a message in the Sender terminal.
- **Encryption (Sender):**
- **Forwarding:** The Tunnel logs the message `MSG: . . .` but sees only gibberish.
- **Decryption (Receiver):**

What is an IV (Initialization Vector)?

An **IV** is a per-message value used by CBC mode.

- **Why it exists:** It prevents identical plaintext messages encrypted with the same key from producing identical ciphertext.
- **Is it secret?** No. The IV is sent along with the ciphertext so the receiver can decrypt.
- **In this project:** The IV is the **first 8 bytes** of the MSG: payload (IV + Ciphertext).

Diagram

```

sequenceDiagram
    participant Sender
    participant KeyGen as key_generator.py
    participant Tunnel
    participant Receiver

    Note over Receiver: Generate RSA Key Pair
    Sender->>Tunnel: Connect
    Tunnel->>Receiver: Connect
    Receiver-->>Sender: Send RSA Public Key (PEM)

    Note over Sender,KeyGen: Generate DES session key (8 bytes)
    Sender->>KeyGen: generate_key_bytes(8)
    KeyGen-->>Sender: DES key
    Note over Sender: Encrypt DES Key with RSA Public Key
    Sender->>Receiver: KEY: <Encrypted DES Key>
    Note over Receiver: Decrypt DES Key with RSA Private Key

    Note over Sender: User types "Hello"
    Note over Sender,KeyGen: Generate per-message IV (8 bytes)
    Sender->>KeyGen: generate_key_bytes(8)
    KeyGen-->>Sender: IV
    Note over Sender: Encrypt "Hello" with DES Key + IV (DES-CBC)
    Sender->>Receiver: MSG: <IV + DES Ciphertext>
    Note over Receiver: Decrypt with DES Key + IV
    Note over Receiver: Print "Hello"

```

Notes / Limitations

- This demonstrates a **hybrid encryption** idea (RSA for key exchange + symmetric cipher for data).
- The Tunnel can see all traffic (public key, encrypted DES key, encrypted messages) but cannot decrypt without the RSA private key or DES session key.

