Flow Diagram and Detailed Explanation of the Secure Sender Program

1. Key Generation and Serialization

- **Step 1: Generate RSA Key Pair**
  - The generate_rsa_keys() function generates a 2048-bit RSA key pair for secure communication.
  - **Mathematical Concept:** RSA key pair involves selecting two large prime numbers pp and qq, and computing: n=p×qn = p \times q ϕ(n)=(p−1)×(q−1)\phi(n) = (p - 1) \times (q - 1) The public key exponent ee is chosen (commonly 65537), and the private key exponent dd satisfies: e×d≡1mod ϕ(n)e \times d \equiv 1 \mod \phi(n)
- **Step 2: Serialize Keys**
  - The serialize_rsa_keys() function serializes the public and private keys into PEM format for transmission.

---

2. Secure Connection Establishment

- A TCP connection is established using the socket module, with the sender connecting to the receiver at localhost:65432.
- The sender receives the receiver's public RSA key.
- The sender sends its public RSA key to the receiver.

---

3. Chaotic Key Generation for AES

- The chaotic.key_generation() function generates an AES key using chaotic functions.
  - **Mathematical Insight:** Chaotic systems are sensitive to initial conditions, providing strong randomness.
  - The binary output is converted to bytes for AES.

---

4. AES Encryption

- **Mathematical Concept:** AES operates on 128-bit blocks.
  - **Encryption Formula:** C=Ek(P)C = E_k(P) Where:
    - CC is the ciphertext.
    - EkE_k is the AES encryption function using key kk.
    - PP is the plaintext.

- **PKCS7 Padding:** Ensures that the data length aligns with AES block size.
- The encrypt_message_with_aes() function applies AES-CBC encryption.

---

5. RSA Encryption of AES Key

- The AES key is encrypted using the receiver's public RSA key: Ck=ERSA(k)C_k = E_{RSA}(k)
- **Mathematical Concept:** Ck=kemod nC_k = k^e \mod n

---

6. Digital Signature Generation

- The sign_message() function signs the encrypted message using the sender's RSA private key.
- **Mathematical Concept:** RSA signature is computed as: S=H(M)dmod nS = H(M)^d \mod n
  Where H(M)H(M) is the hash of the message.

---

7. Data Transmission

- The encrypted AES key, IV, encrypted message, and signature are sent to the receiver.
- Lengths of the encrypted message and signature are sent first using struct.pack().

---

8. Flow Diagram

```
+------------------------+
| Generate RSA Key Pair  |
+-----------+------------+
            |
            v
+------------------------+
| Serialize RSA Keys     |
+-----------+------------+
            |
            v
+------------------------+
| Establish TCP Socket   |
+-----------+------------+
            |
            v
+------------------------+
| Receive Receiver RSA   |
| Send Sender RSA Key    |
+-----------+------------+
            |
            v
+------------------------+
| Generate Chaotic AES   |
+-----------+------------+
           |
           v
+------------------------+
| Encrypt Dataset with   |
| AES (CBC Mode)         |
+-----------+------------+
           |
           v
+------------------------+
| Encrypt AES Key with   |
| Receiver RSA Key       |
+-----------+------------+
           |
           v
+------------------------+
| Sign Encrypted Message |
+-----------+------------+
           |
           v
+------------------------+
| Send AES Key, IV,      |
| Encrypted Message,     |
| and Signature          |
+------------------------+
```

Summary

This program demonstrates secure data transmission by integrating RSA for key exchange, AES for data encryption, and digital signatures for data integrity. The use of chaotic key generation enhances the randomness of cryptographic keys, ensuring secure communication.

Flow Diagram and Detailed Explanation of the Secure Receiver Program

1. Key Generation and Serialization

- **Step 1: Generate RSA Key Pair**
  - The generate_rsa_keys() function generates a 2048-bit RSA key pair for secure communication.
  - **Mathematical Concept:** RSA key pair involves selecting two large prime numbers pp and qq, and computing: n=p×qn = p \times q ϕ(n)=(p−1)×(q−1)\phi(n) = (p - 1) \times (q - 1) The public key exponent ee is chosen (commonly 65537), and the private key exponent dd satisfies: e×d≡1mod ϕ(n)e \times d \equiv 1 \mod \phi(n)
- **Step 2: Serialize Keys**
  - The serialize_rsa_keys() function serializes the public and private keys into PEM format for transmission.

2. Secure Connection Establishment

- A TCP server is created using the socket module.
- The receiver binds to localhost:65432 and listens for a connection.
- The receiver sends its public RSA key to the sender and receives the sender's public RSA key.

3. Data Reception

- The recv_all() function ensures that the exact number of bytes is received for each data component.
- **Data Flow:**
    - Encrypted AES key (256 bytes)
    - Initialization vector (IV) (16 bytes)
    - Encrypted message length (4 bytes)
    - Signature length (4 bytes)
    - Encrypted message
    - Digital signature

4. RSA Decryption of AES Key

- The AES key is decrypted using the receiver's private RSA key: k=Ckdmod nk = C_k^d \mod n Where CkC_k is the encrypted AES key.

5. Signature Verification

- The verify_signature() function verifies the digital signature using the sender's public RSA key.
- **Mathematical Concept:** The verification process checks: H(M)=?Semod nH(M) \stackrel{?}{=} S^e \mod n Where H(M)H(M) is the hash of the message, and SS is the signature.

6. AES Decryption

- **Mathematical Concept:** AES decryption formula: P=Dk(C)P = D_k(C) Where:
    - PP is the plaintext.
    - DkD_k is the AES decryption function using key kk.
    - CC is the ciphertext.

- The decrypt_message_with_aes() function applies AES-CBC decryption and removes PKCS7 padding.

7. Data Restoration

- The decrypted message is saved to a CSV file as received_file.csv.

8. Flow Diagram

```
+-------------------------+
| Generate RSA Key Pair   |
+-----------+-------------+
            |
            v
+-------------------------+
| Serialize RSA Keys      |
+-----------+-------------+
            |
            v
+-------------------------+
| Create TCP Server       |
| Wait for Connection     |
+-----------+-------------+
            |
            v
+-------------------------+
| Send Receiver Public Key|
| Receive Sender Public Key|
+-----------+-------------+
            |
            v
+-------------------------+
| Receive AES Key, IV,    |
| Encrypted Message,      |
| and Signature           |
+-----------+-------------+
            |
            v
+-------------------------+
| Decrypt AES Key         |
+-----------+-------------+
            |
            v
+-------------------------+
| Verify Signature        |
+-----------+-------------+
            |
            v
+-------------------------+
| Decrypt Encrypted Message|
+-----------+-------------+
            |
            v
+-------------------------+
| Save Decrypted Data     |
| as CSV File             |
+-------------------------+
```

Summary

This program demonstrates secure data reception by integrating RSA for key exchange, AES for data decryption, and digital signatures for data integrity verification. The careful handling of message sizes and padding ensures secure and efficient communication.