Intro:

The program implements the RSA (Rivest-Shamir-Adleman) algorithm, which is a public-key cryptosystem that is widely used for secure data transmission. The RSA algorithm involves three steps: key generation, encryption, and decryption. Additionally it provides signature generation and verification. This process is crucial for ensuring the authenticity and integrity of messages in a communication system.

RSA Key Generation:

The key generation process is implemented in the `generate(p, q)` function. Here, `p` and `p` are two prime numbers that are provided as arguments.

- The function initially computes `n` by multiplying `p` and `q`. This `n` will be used as the modulus for both the public and private keys.
- Next, it computes `phi` which is Euler's totient function `(p-1)*(q-1)`.
- Then it chooses an integer e such that `1 < e < phi` and `gcd(a, b) == 1`. This e is the public key exponent.
- Finally, it computes d which is the multiplicative inverse of e modulo phi. This d is the private key exponent.

The function returns the tuple `(e, d, mod, phi)`, where `e` and `n` form the public key, and `d` and `n` form the private key.

An example of how to run the script from the command line for key generation:

```
$python3 rsa_program.py --generate-key --p 61 --q 53
```

Encryption:

- The `encrypt_text(plaintext_ascii, e, n)` function encrypts the ASCII representation of the plaintext using the public key.
- It applies modular exponentiation to ASCII representation of the plaintext using the public exponent **e** and modulus **n**.
- The function returns a list of ciphertext values in hexadecimal format.
- This function is called when the **--encrypt** argument is provided.

An example of how to run the script from the command line for encryption:

```
$python3 rsa_program.py --encrypt plaintext.txt --public-key public_key.txt
--output ciphertext.txt
```

RSA Decryption:

- The `decrypt_text(ciphertext, d, n)` function decrypts the ciphertext using the private key.
- It applies modular exponentiation using the private exponent **d** and modulus **n**.
- The function returns the decrypted plaintext.
- This function is called when the **--decrypt** argument is provided.

An example of how to run the script from the command line for decryption:

```
$python3 rsa_program.py --decrypt ciphertext.txt --private-key private_key.txt
--output plaintext.txt
```

Signing a Message

Signing a message is implemented in the **sign** section of the driver code. The purpose of signing a message is to allow the receiver to verify that the message came from the claimed sender (authenticity) and that the message has not been tampered with (integrity).

To sign a message, the sender first converts the message into its ASCII representation. Then, the sender encrypts the ASCII message using their own private key, resulting in the signature. This process is essentially the same as the encryption process, but uses the sender's private key instead of the receiver's public key.

An example of how to run the script from the command line for signature creation:

```
$python3 rsa_program.py --sign ThisIsMySign --private-key private_key.txt
--signature signature.txt
```

Verifying a Signature

Verifying a signature is implemented in the **verify** section of the driver code. The purpose of verifying a signature is to check the authenticity and integrity of a received message.

To verify a signature, the receiver first reads the signature from a file and converts it from **hexadecimal** back to integers. The receiver then decrypts the signature using the sender's public key, resulting in the original ASCII message. This process is essentially the same as the decryption process, but uses the **sender's public key** instead of the **receiver's private key**. Finally, the receiver checks if the decrypted message matches the original message. If they match, the signature is valid; otherwise, it's invalid.

An example of how to run the script from the command line for signature verification:

```
$python3 rsa_program.py --verify ThisIsMySign --signature signature.txt
--public-key public_key.txt
```