Cramer-Shoup Cryptosystem Documentation

Overview

This code implements the Cramer-Shoup cryptosystem, a public-key encryption scheme that provides security against chosen-ciphertext attacks. Our implementation works with a simple alphabet of English letters and an underscore (_).

The code includes:

Key Generation

- **Group Setup**:
  A large cyclic group G of prime order q is chosen, typically represented by a large prime number p. Two distinct random generators g1 and g2 of the group are selected.
- **Secret Key Generation**:
  Five random values are chosen from the set {0,...,q−1} to form the secret key: (x1,x2,y1,y2,z), where x1,x2,y1,y2,z are the secret values used for encryption and decryption.
- **Public Key Generation**:
  The following public parameters are computed:
  - c=g1^x1 * g2^x2 mod p
  - d=g1^y1 * g2^y2 mod p
  - h=g1^z mod p

  The public key is:

  (p,g1,g2,c,d,h)

  and the secret key is:

  (x1,x2,y1,y2,z)

Encryption

1. **Message Conversion**:
   The message mmm is converted into an integer $m \in G$. Each character in the message is mapped to a corresponding integer from the alphabet.
2. **Random Value Selection**:
   A random integer k is chosen from the set $\{0,\ldots,q-1\}$
3. **Computation of Ciphertext Components**:
   - u1=g1^k mod p
   - u2=g2^k mod p
   - e=h^k mod p (the message encrypted with the random exponentiation)
   - α=H(u1,u2,e) mod p, where H is a cryptographic hash function (e.g., SHA-256).
   - v=c^k* d^(kα) mod p

   The ciphertext (u1,u2,e,v) is generated and can be transmitted.

Decryption

- **Hash Calculation**:
  The hash is computed as: α=H(u1,u2,e)mod p

  α=H(u1,u2,e) mod p
- **Ciphertext Validation**:
  The validity of the ciphertext is checked by verifying the equation: u1^x1u2^x2 *(u1^y1 * u2^ y2)^α mod p=v

  If this check fails, decryption is aborted, and the ciphertext is rejected.
- **Plaintext Recovery**:
  If the ciphertext is valid, the plaintext message m is recovered as follows:
- m = e / (u1^z)

  The decryption stage correctly decrypts any properly-formed ciphertext, since:

  u1^z = g1^(k*z) and m= e/(h^k)

The original message is then successfully decrypted.

Helper Functions

mod_exp(base, exp, mod)

Performs modular exponentiation to compute .

Parameters:

- **base**: The base integer
- **exp**: The exponent integer
- **mod**: The modulus integer

Returns:

- Result of the modular exponentiation

mod_inv(a, p)

Computes the modular inverse of *a* under modulo *p*.

hash_function(*values)

Computes a hash value by combining the input values into a string and hashing it using SHA-256.

Parameters:

- **values**: Variable-length input values to be hashed

Returns:

- A large integer representation of the hash

CramerShoup Class

Represents the Cramer-Shoup cryptosystem with methods for key generation, encryption, and decryption.

Attributes:

- ***alphabet***: The set of characters supported by the cryptosystem
- ***char_to_int***: A mapping of characters to integers for encoding plaintext
- ***int_to_char***: A mapping of integers to characters for decoding ciphertext

__init__(self, alphabet)

Initializes the **CramerShoup** instance with the given alphabet.

Parameters:

- **alphabet**: A string representing the set of characters

---

generate_keys(self, p)

Generates public and private keys for the cryptosystem.

Parameters:

- **p**: A large prime number defining the group

Returns:

- **public_key**: A tuple used for encryption
- **private_key**: A tuple used for decryption

---

encrypt(self, public_key, plaintext)

Encrypts a plaintext message using the public key.

Parameters:

- **public_key**: The public key tuple
- **plaintext**: A string message to encrypt

Returns:

- **ciphertext**: A tuple representing the encrypted message

Process:

1. Validates the plaintext characters against the alphabet
2. Converts the plaintext into numerical representation
3. Generates a random value **r**
4. Computes components of the ciphertext

---

decrypt(self, private_key, public_key, ciphertext)

Decrypts a ciphertext message using the private key.

Parameters:

- **private_key**: The private key tuple
- **public_key**: The public key tuple
- **ciphertext**: A tuple representing the encrypted message

Returns:

- **plaintext**: The decrypted string message

Process:

1. Computes using the hash function
2. Validates the ciphertext
3. Decrypts to retrieve the numerical representation of the plaintext
4. Converts numerical values back to characters