RSA Encryption/Decryption

Introduction

This document provides an explanation of the RSA (Rivest-Shamir-Adleman) encryption and decryption code. RSA is a widely used public-key cryptosystem that is used for secure communication and data encryption. The code provided here implements the RSA algorithm and includes functions for key generation, encryption, and decryption.

RSA Algorithm Overview

RSA is a public-key cryptosystem that uses two keys: a public key for encryption and a private key for decryption. The security of RSA is based on the difficulty of factoring large composite numbers into their prime factors.

The core steps of the RSA algorithm are as follows:

Key Generation:

- Select two large distinct prime numbers, p and q.
- Compute n = p*q, where n is used as the modulus for both the public and private keys.
- Calculate Euler's totient function, $\phi(n) = (p-1)(q-1)$.
- Choose a public exponent e such that $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$.
- Compute the private exponent d such that d is the modular multiplicative inverse of e modulo $\phi(n)$.

Encryption:

- Convert the plaintext message into numerical values, typically using a fixed mapping from characters to numbers.
- Split the numerical message into blocks and encrypt each block using the public key (n, e) with the formula ciphertext = (plaintext^e) % n.
- Combine the encrypted blocks to form the ciphertext.

Decryption:

- Split the ciphertext into blocks and decrypt each block using the private key d with the formula plaintext = (ciphertext^d) % n.
- Combine the decrypted blocks to recover the original plaintext message.

Code Explanation

Function: is_valid_input

This function checks if a given string contains only valid characters, which are uppercase letters ('A' to 'Z') and underscores ('_'). It returns true if the input is valid and false otherwise.

Function: is_prime

This function checks whether a given number is prime. It returns true if the number is prime and false otherwise.

Function: gcd

This function calculates the greatest common divisor (GCD) of two numbers using Euclid's algorithm.

Function: mod_pow

This function computes modular exponentiation (base^exp) % mod efficiently using the square-and-multiply method.

Function: mod_inverse

This function calculates the modular multiplicative inverse of a number a modulo m. It returns the inverse if it exists, otherwise 0.

Function: letters_to_num

This function converts a string of letters into a numerical value. It treats lowercase letters as uppercase and uses a mapping where 'A' corresponds to 1, 'B' to 2, and so on, with '_' representing a gap.

Function: num_to_letters

This function converts a numerical value back into a string of letters. It reverses the process of letters_to_num.

Function: encrypt

This function encrypts a plaintext message using the public key (n, e). It breaks the message into blocks, converts them to numerical values, and applies modular exponentiation to each block.

Function: decrypt

This function decrypts a ciphertext message using the private key d. It reverses the encryption process, performing modular exponentiation with the private key to recover the original plaintext.