**Lee\_Report\_PA4**

**Cryptography Section W01 Summer Semester 2024 CO**

**Raehyeong Lee**

**2024-07-17**

**Overview**

This project implements the RSA algorithm, a widely used asymmetric cryptographic technique, to encrypt and decrypt text. The RSA algorithm involves generating a pair of keys (public and private) using prime numbers, encrypting data with the public key, and decrypting it with the private key. The project also includes functions to handle string inputs by converting them to their ASCII codes, applying the RSA algorithm, and converting the result back to text. This approach demonstrates the practical application of RSA for secure communication.

**Functions**

**1. isPrime(number)**

This function checks if a given number is prime.

* Takes an integer as input.
* Returns True if the number is prime, otherwise False.

**2. primeGenerator()**

This function generates a list of prime numbers within a specified range.

* Returns a list of prime numbers between 100 and 1000.

**3. mod(A, M)**

This function computes the modular inverse of A modulo M.

* Takes two integers, A and M, as input.
* Returns the modular inverse of A modulo M.

**4. KeyGen(primeList)**

This function generates the public and private keys for RSA encryption.

* Takes a list of prime numbers as input.
* Selects two distinct primes from the list to compute the modulus and totient.
* Chooses a public exponent that is coprime with the totient.
* Computes the private exponent as the modular inverse of the public exponent.
* Returns the public and private keys as tuples.

**5. RSA\_Algorithm(M, key)**

This function performs RSA encryption or decryption.

* Takes an integer M (the message or ciphertext) and a key (a tuple containing the exponent and modulus) as input.
* Returns the result of M raised to the power of the exponent modulo the modulus.

**6. StrToAscii(s)**

This function converts a string to its ASCII representation.

* Takes a string as input.
* Returns a list of ASCII values corresponding to each character in the string.

**7. asciiToStr(a)**

This function converts a list of ASCII values back to a string.

* Takes a list of ASCII values as input.
* Returns the corresponding string.

**RSA Algorithm Application**

The following steps outline the application of the RSA algorithm to encrypt and decrypt text:

1. **Key Generation**

* Generate a list of prime numbers using primeGenerator().
* Use KeyGen(primeList) to generate the public and private keys.

1. **Text Conversion and Encryption**

* Convert the input string to its ASCII representation using StrToAscii(s).
* Encrypt each ASCII value using RSA\_Algorithm(M, Public\_Key).

1. **Decryption and Text Conversion**

* Decrypt each encrypted ASCII value using RSA\_Algorithm(M, Private\_Key).\
* Convert the decrypted ASCII values back to text using asciiToStr(a).

**Conclusion**

This project successfully implements the RSA algorithm to encrypt and decrypt text. The functions ensure the correct processing of prime numbers, key generation, and RSA operations, while the text converter allows for easy handling of string inputs. The resulting ciphertext and decrypted text demonstrate the effectiveness of the RSA algorithm in securing data. Future work could involve optimizing key generation, implementing more robust error handling, and exploring different key sizes for enhanced security.

**Project Code**

import random

from math import gcd

def isPrime(number):

    if number <= 1:

        return False

    for i in range(2, int(number\*\*0.5) + 1):

        if number % i == 0:

            return False

    return True

def primeGenerator():

    p = [int for int in range(100, 1001) if isPrime(int)]

    return p

def mod(A, M):

    m1, x1, x2 = M, 0, 1

    if M == 1:

        return 0

    while A > 1:

        Q = A // M

        M, A = A % M, M

        x1, x2 = x2 - Q \* x1, x1

    if x2 < 0:

        x2 += m1

    return x2

def KeyGen(primeList):

    prime1, prime2 = random.sample(primeList, 2)

    print("Chosen Prime: ",prime1,",",prime2)

    modulus = prime1 \* prime2

    phN = (prime1 - 1) \* (prime2 - 1)

    public\_exp = random.choice([int for int in range(2, phN) if gcd(int, phN) == 1])

    private\_exp = mod(public\_exp, phN)

    Public\_Key = (public\_exp, modulus)

    Private\_Key= (private\_exp, modulus)

    print("Public Key: ", Public\_Key)

    print("Private Key: ", Private\_Key)

    return Public\_Key, Private\_Key

def RSA\_Algorithm(M, key):

    exp, N = key

    return pow(M, exp, N)

def StrToAscii(s):

    return [ord(character) for character in s]

def asciiToStr(a):

    return ''.join(chr(character) for character in a)

p = primeGenerator()

Public\_Key, Private\_Key = KeyGen(p)

String = input("String: ")

ascii = StrToAscii(String)

encrypt = [RSA\_Algorithm(code, Public\_Key) for code in ascii]

decrypt = [RSA\_Algorithm(code, Private\_Key) for code in encrypt]

decryption = asciiToStr(decrypt)

print("Encrypt :", encrypt)

print("Decrypt :", decryption)

**Output**

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**