Roll. No. A016	Name: Varun Khadayate
Class B.Tech CsBs	Batch: 1
Date of Experiment: 10-09-2022	Subject: Cryptology

Aim

To implement RSA Algorithm.

Theory

The pioneering paper by Diffie and Hellman [DIFF76b] introduced a new approach to cryptography and, in effect, challenged cryptologists to come up with a crypto- graphic algorithm that met the requirements for public-key systems. A number of algorithms have been proposed for public-key cryptography. Some of these, though initially promising, turned out to be breakable.[4]

One of the first successful responses to the challenge was developed in 1977by Ron Rivest, Adi Shamir, and Len Adleman at MIT and first published in 1978 [RIVE78].[5] The Rivest-Shamir-Adleman (RSA) scheme has since that time reigned supreme as the most widely accepted and implemented general-purpose approachto public-key encryption.

The **RsA** scheme is a cipher in which the plaintext and ciphertext are integersbetween 0 and $n - 1$ for some n. A typical size for n is 1024 bits, or 309 decimal digits. That is, n is less than 2^{1024}. We examine RSA in this section in some detail,beginning with an explanation of the algorithm. Then we examine some of the com- putational and cryptanalytical implications of RSA.

RSA makes use of an expression with exponentials. Plaintext is encrypted in blocks, with each block having a binary value less than some number n. That is, the block
size must be less than or equal to $\log_2(n) + 1$; in practice, the block size is i bits, where 2^i 6 n … 2^{i+1}. Encryption and decryption are of the following form, for some plaintext block M and ciphertext block C.

$$C = M^e \bmod n$$
$$M = C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$$

Both sender and receiver must know the value of n. The sender knows the value of e, and only the receiver knows the value of d. Thus, this is a public-key en- cryption algorithm with a public key of $PU = \{e, n\}$ and a private key of $PR = \{d, n\}$. For this algorithm to be satisfactory for public-key encryption, the following require- ments must be met.

1. It is possible to find values of e, d, and n such that $M^{ed} \bmod n = M$ for all M 6 n.
2. It is relatively easy to calculate $M^e \bmod n$ and $C^d \bmod n$ for all values of M 6 n.
3. It is infeasible to determine d given e and n.

For now, we focus on the first requirement and consider the other questions later. We need to find a relationship of the form

$$M^{ed} \bmod n = M$$

The preceding relationship holds if e and d are multiplicative inverses modulo $\phi(n)$, where $\phi(n)$ is the Euler totient function. It is shown in Chapter 8 that for p, q prime, $\phi(pq) = (p - 1)(q - 1)$. The relationship between e and d can be expressed as

$$ed \bmod \phi(n) = 1 \qquad \textbf{(9.1)}$$

This is equivalent to saying

$$ed \equiv 1 \bmod \phi(n)$$
$$d \equiv e^{-1} \bmod \phi(n)$$

That is, e and d are multiplicative inverses mod $\phi(n)$. Note that, according to the rules of modular arithmetic, this is true only if d (and therefore e) is relatively prime to $\phi(n)$. Equivalently, $\gcd(\phi(n), d) = 1$. See Appendix R for a proof that Equation (9.1) satisfies the requirement for RSA.

We are now ready to state the RSA scheme. The ingredients are the following:

p, q, two prime numbers	(private, chosen)
$n = pq$	(public, calculated)
e, with $\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$	(public, chosen)
$d \equiv e^{-1} \pmod{\phi(n)}$	(private, calculated)

The private key consists of $\{d, n\}$ and the public key consists of $\{e, n\}$. Supposethat user A has published its public key and that user B wishes to send the message$M$ to A. Then B calculates $C = M^e \bmod n$ and transmits C. On receipt of this cipher- text, user A decrypts by calculating $M = C^d \bmod n$.

Figure 9.5 summarizes the RSA algorithm. It corresponds to Figure 9.1a: Alice generates a public/private key pair; Bob encrypts using Alice's public key; and Alice decrypts using her private key. An example from [SING99] is shown in Figure 9.6.For this example, the keys were generated as follows.

1. Select two prime numbers, $p = 17$ and $q = 11$.
2. Calculate $n = pq = 17 * 11 = 187$.
3. Calculate $\phi(n) = (p - 1)(q - 1) = 16 * 10 = 160$.
4. Select e such that e is relatively prime to $\phi(n) = 160$ and less than $\phi(n)$; we choose $e = 7$.
5. Determine d such that $de \equiv 1 \pmod{160}$ and $d < 160$. The correct value is $d = 23$, because $23 * 7 = 161 = (1 * 160) + 1$; d can be calculated using the extended Euclid's algorithm.

Code

```
from utils import *
import random

primes = [i for i in range(100, 500) if is_prime(i)]
p = random.choice(primes)
q = random.choice(primes)

n = p*q
k = (p-1)*(q-1)

for e in range(2, k):
    if gcd(e, k) == 1:
        break
public_key = (n, e)
_, b, _ = extended_gcd(e, k)
if b < 0:
    b = b + k
private_key = (n, b)

while True:
    ch = int(input('Welcome to RSA Encryption and Decryption Program Made by
Varun Khadayate..\n [*] Press 1 for Encryption \n [*] Press 2 for Decryption
\n [*] Press 0 to exit..\n \nYour Choice:: '))

    if ch == 1:
        print("\n=============================================")
        print("              !!!!Encryption!!!!             ")
        message = input('Enter the text to be encrypted: ')
        encrypted = encrypt(public_key, message)
        print(f'Encrypted message: {"".join(str(s) for s in encrypted)}')
        print("\n=============================================\n\n")

    elif ch == 2:
        print("\n=============================================")
        print("               !!!Decryption!!!              ")
        decrypted = decrypt(private_key, encrypted)
        print(f'Enter the text to be decrypted: {"".join(str(s) for s in
encrypted)}')
        print(f'Decrypted message: {"".join(str(s) for s in decrypted)}')
        print("\n=============================================\n\n")

    elif ch == 0:
        print("\n=============================================")
        print("     Thank You for using the Software ;)     ")
        print("                 Exiting Now.                ")
        print("=============================================")
        exit()
```

Output

```
PS E:\College-Codes\Fourth Year\SEM VII> & C:/Users/varun/AppData/Local/Programs/Python/Python310/python.exe "e:/College-Codes/Fourth Year/SE
M VII/CT/RSA_Practical_5.py"
Welcome to RSA Encryption and Decryption Program Made by Varun Khadayate..
 [*] Press 1 for Encryption
 [*] Press 2 for Decryption
 [*] Press 0 to exit..

Your Choice:: 1

==========================================
             !!!!Encryption!!!!
Enter the text to be encrypted: Varun Khadayate 0810
Encrypted message: 14619927814140744362711415087221342111179627814176042781489077278141204898772372263218753691886563218

==========================================


Welcome to RSA Encryption and Decryption Program Made by Varun Khadayate..
 [*] Press 1 for Encryption
 [*] Press 2 for Decryption
 [*] Press 0 to exit..

Your Choice:: 2

==========================================
              !!!Decryption!!!
Enter the text to be decrypted: 14619927814140744362711415087221342111179627814176042781489077278141204898772372263218753691886563218
Decrypted message: Varun Khadayate 0810

==========================================


Welcome to RSA Encryption and Decryption Program Made by Varun Khadayate..
 [*] Press 1 for Encryption
 [*] Press 2 for Decryption
 [*] Press 0 to exit..

 Your Choice:: 0

 ==========================================
      Thank You for using the Software ;)
                Exiting Now.
 ==========================================
```

Conclusion

Hence, we were able to perform RSA Algorithm.