1.5em 0pt

(1)

RSA

December 2019

1 Abstract

Number theory plays a key role in cryptography, the subject of transforming information so that it cannot be easily recovered without special knowledge. The security of the message is directly linked the length of the prime number that has been chosen, but that comes with its own challenge of processing power and time taken by the encryption and decryption process. In this report I will be addressing RSA encryption, which is a block cipher. RSA is used for protect and secure our online life and tries to make sure that they are kept secure.The most important aspect of the RSA algorithm is the pair of keys, which have a particular mathematical form: An RSA key pair consists of three basic components: the modulus n, the public key component e (encryption), and the secret key component d (decryption). The pairs (e, n) and (d, n) then form a public and private key respectively.The implementation of its components(Prime numbers(generator), Greatest common divisor (function), inversion(function), phi and common factors (co-prime and relative prime) will be discussed in detail in this report. Public-key(asymmetric) and the limitation will also be addressed.

2 Introduction

In this report I will try to show the significance of cryptography in this modern age of data exchange and the challenges that comes with it in terms of data protection. Maths, particularly Number theory plays a vital role in the data protection (encrypting and decryption) public-key(asymmetric).RSA is a block cipher and these ciphers encrypt messages by changing each letter to a different letter, or each block of letters to a different block of letters and use asymmetric key distribution. In a block cipher instead of changing each and every key called mono alphabetic ciphers like shift cipher and Affine ciphers while block ciphers make changes on block by block which will add protection, as it can be permuttated and make it harder to know what it is, how it works. Number theory plays a vital role in cryptography in public key cryptosystem. Number theory is also important in public key cryptography,a type of cryptography invented in the 1970s.In public key cryptography, knowing how to encrypt does not also tell someone how to decrypt the message. The most widely used public key system,called the RSA cryptosystem, encrypts messages using modular exponentiation, where the modulus is the product of two large primes. Knowing how to encrypt requires that someone know the modulus and an exponent. (It does not require that the two prime factors of the modulus be known.)As far as it is known, knowing how to decrypt requires someone to know how to invert the encryption function, which can only be done in a practical amount of time when someone knows these two large prime factors. Here the key is how large the these prime numbers are as the strength of the encryption highly depend on the size of the prime numbers used.

Cryptology is defined as the study of cryptography or cryptanalysis. Cryptography is simply the process of communicating secretly through the use of ciphers, and cryptanalysis is the process of cracking or deciphering such secret communications. The report, to be written in LATEX, should contain:

which was published in 1978 by its inventors/discoverers, Ronald Rivest,Adi Shamir, and Leonard Adleman (see [Rive], [Elli]), and which by now has been implemented worldwide RSA is used for protect and secure our online life are kept secure. To name some of the protection that this digital world needs are Confidentiality protecting ones data from undesired person while Integrity tries to tackle that no one can edit or make changes to the data and Authentication is a way of making sure that the data or information came from a trusted source the last but not least is availability of data when it is needed.

3 Public key cryptography,asymmetric cryptosystems

The Asymmetric cryptosystems, in contrast to symmetric algorithms, do not use a secret key employed both for encryption and decryption of a message, but a pair of keys for each participant consisting of a public key E for encryption and a different, secret, key D for decryption. If the keys are applied to a message M one after another in sequence, then the following relation must hold:

$$D(E(M)) = M$$

For the sake of security of such a procedure it is necessary that a secret key D not be able to be derived from the public key E, or that such a derivation be infeasible on the basis of time and cost constraints is a way where the message is encrypted with a public and Decrypted by a private key. The concept of public-key encryption is simple and elegant, but has far-reaching consequences.

In this section of the report we will discuss public keys and their use in the overall role they play.If Alice wants to send a message to Bob, but they have not had previous contact and they do not want to take the time to send a courier with a key.Then the other option is use key sharing. Therefore, all information that Alice sends to Bob will potentially be obtained by the evil observer Eve. However, it is still possible for a message to be sent in such a way that Bob can read it but Eve cannot. The possibility of the present scheme,called a public key cryptosystem , was first publicly suggested by Diffie and Heilman in their classic paper (Diffie-Hellman).When a private key cryptosystem is used, two parties who wish to communicate in secret must share a secret key, any one who knows the key can both encrypt and decrypt messages. A requirement for the implementation of an asymmetric encryption system for the generation of digital signatures is that the association of D (M) and M can be reliably verified. The possibility of such a verification exists if the mathematical operations of encryption and decryption are commutative, that is, if their execution one after the other leads to the same, original, result regardless of the order in which they are applied: D (E (M)) = E (D (M)) = M

Protocol for key exchange à la Diffie–Hellman has three parts mentioned here under:

1. A chooses an arbitrary value $xAp1andsendsyA := a^xA$mod p as her public key to B.

2. B chooses an arbitrary value xB p 1 and sends $yB := a^xBmodp$ as his public key to A.

3. A computes the secret key $sA := y^xAB\, mod\, p$.

4. B computes the secret key $sB := y^xBA\, mod\, p$.

4 RSA Encryption

RSA as a Public Key System

As already mentioned in the RSA key pair consists of three basic components: the modulus n, the public key component e (encryption), and the secret key component d (decryption). The pairs (e, n) and (d, n) then make up a public and private key respectively. We will start by generating the modulus n as the product n = p*q of two prime numbers p and q. If $\phi(n) = (p1)(q1)$ then for a given n the public key component e can be chosen such that $e < \phi(n)$ and gcd $(e, \phi(n)) = 1$. The secret component d corresponding to n and e is obtained by calculating the inverse $d = e^{1mod\phi(n).}$

denotes the Euler phi function we shall see how the RSA procedure can be implemented both as an asymmetric cryptosystem scheme.Here is how the RSA algorithm works. Bob chooses two distinct large primes p and q and multiplies them together to form n = p*q. He also chooses an encryption exponent e such that

$$gcd(e,(p-1)(q-1)) = 1.$$

(n,e) are sent to Alice while holding p and q secret. On the other hand while Alice not needing to know the values of p and q can revert to the message m securely. Here We only need to break the message into blocks if message ¿ n, each block needs to be less than n.

and this is how you generate the cypher text

$$C = M^emodn.$$

Maybe the most interesting and important thing to understand is why $m = cd(modn)$. Euler's theorem tells us that if gcd(o, n) = 1, then = 1 (mod n). In our case,

$$\phi(n) = 4 > (pq) = (p - l)(< j - 1).$$

Suppose gcd(m,n) = 1. This is very likely the case; since p and q are large, m probably has neither as a factor. Since $d * e \equiv 1(mod < / > (tz))$, we can write $de \equiv 1 + k < j > (n)$ for some integer k. Therefore,

$$c^d \equiv m^{ed} \equiv m * (m^{\phi}(n))^k \equiv m * 1^k \equiv (modn)$$

we have shown that Bob can recover the message. If $gcd(m, n)^1$, Bob still recovers the message.

5 RSA Decryption

The plain text message can be quickly recovered from a ciphertext message when the decryption key d, an inverse of e modulo (p1)(q 1), is known. [Such an inverse exists because gcd(e,(p1)(q 1)) = 1.] To see this, note that if de 1 (mod (p1)(q 1)), there is an integer k such that de= 1+k(p1)(q 1). It follows that Cd (Me)d = Mde = M1+k(p1)(q1) (mod n).

By Fermat's little theorem [assuming that gcd(M,p) = gcd(M,q) = 1, which holds except in rare cases, which we cover in Exercise 28], it follows that Mp1 1 (mod p) and Mq1 1 (mod q). Consequently, Cd M ·(Mp1)k(q1) M ·1 = M(mod p) and Cd M ·(Mq1)k(p1) M ·1 = M(mod q). Because gcd(p,q) = 1, it follows by the Chinese remainder theorem that Cd M(mod pq).

$$Cd(Me)d = Mde = M1 + k(p1)(q1) * (modn).$$

6 GCD

GCD is used to verify that if the fuction has a inverse or not, this is done as follows The plaintext message can be quickly recovered from a ciphertext message when the decryption key d, an inverse of e

$$(p - 1)(q - 1)$$

, is known. [Such an inverse exists because

$$\gcd(e, (p - 1)$$

(q -1)) = 1.]

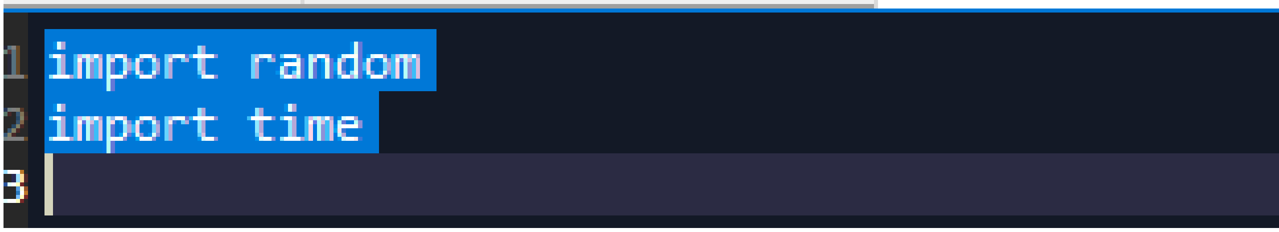
7 Prime numbers

- A review of relevant mathematics

To illustrate this mathematical principle with we will do a small example: We choose p = 7 and q = 11. Then we have n = 77 and $phi(n) = 2^2\Delta 3\Delta 5 = 60$. Then we calculate the gcd (e, Phi(n)) = 1, the least possible value for the key component e is 7, from which we derive the value d = 43 for the key component d, since $7\Delta 43 \equiv 301 \equiv 1 mod 60$. We can not test our RSA algorithm on a message "message" 5 is encrypted to $5^7 mod 77 = 47 and the decryption 47^4 3 mod 77 = 5$ thus getting the original message.

8 code

Implementation of the code with snip its of the code.

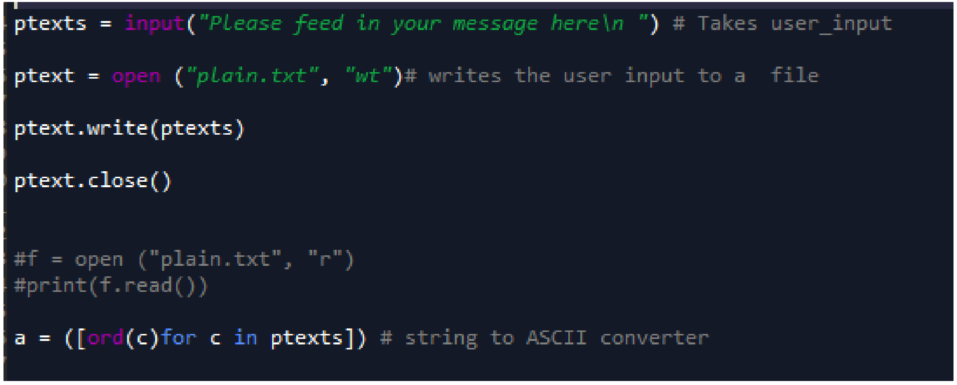


```
import random
import time

```

Figure 1: lib used

input.PNG input.PNG input.PNG input.PNG input.PNG input.PNG input.PNG input.PNG



```
ptexts = input("Please feed in your message here\n ") # Takes user_input

ptext = open ("plain.txt", "wt")# writes the user input to a  file

ptext.write(ptexts)

ptext.close()


#f = open ("plain.txt", "r")
#print(f.read())

a = ([ord(c)for c in ptexts]) # string to ASCII converter
```

Figure 2: user input

9 • Example runs of the program with different parameters.

=======================Prime number generator=====================

$$def prime_gen(num):$$

```
if num == 2: return
elif num ¡ 2: return []
s=list(range(3,num + 1,2))
root = num ** 0.5
half=(num+1)//2-1
i=0
m=3
while m ¡= root:
if s[i]:
j=(m*m-3)//2 this is a floor function
s[j]=0
while j¡half:
s[j]=0
j+=m i=i+1
m=2*i+3
return [2]+[x for x in s if x]
```

================testing for time ==================================

$$t0 = time.time()$$

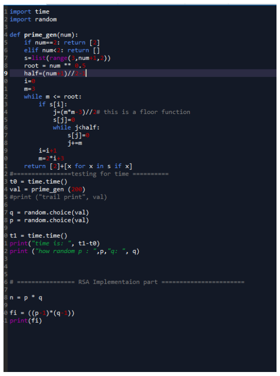
```
val = prime_gen(200)
print ("trail print", val)
q = random.choice(val)
p = random.choice(val)
t1 = time.time()
print("time is: ", t1-t0)
print ("how random p : ",p,"q: ", q)
```

==

Here is the result of the prime generator

$$defgcd(x, y):$$

```
while (y):
x,y = y, x
return x
def inv(x,i):
for x in range (1,i):
```



```python
import time
import random

def prime_gen(num):
    if num==2: return [2]
    elif num<2: return []
    s=list(range(3,num+1,2))
    root = num ** 0.5
    half=(num+1)//2-1
    i=0
    m=3
    while m <= root:
        if s[i]:
            j=(m*m-3)//2# this is a floor function
            s[j]=0
            while j<half:
                s[j]=0
                j+=m
        i=i+1
        m=2*i+3
    return [2]+[x for x in s if x]
#================testing for time ==========
t0 = time.time()
val = prime_gen (200)
#print ("trail print", val)

q = random.choice(val)
p = random.choice(val)

t1 = time.time()
print("time is: ", t1-t0)
print ("how random p : ",p,"q: ", q)



# =============== RSA Implementaion part ======================

n = p * q

fi = ((p-1)*(q-1))
print(fi)
```

Figure 3: liberary

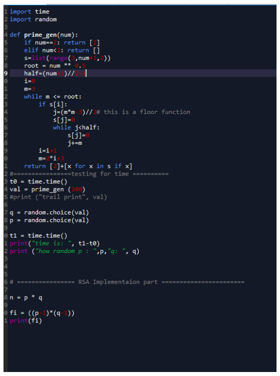
prime001.PNG

Figure 4: prime number generator

if (x * x)
return x
return None
def cprime(a):
p= []
for x in range (2,a):
if gcd (a,x)==1 and inv(x,phi)!= None:
p.append(x) for x in p:
if x == inv(x,phi):
p.remove(x)
return p

ﬄ$Thecodeasanattachment; wellcommented(sothatitiseasytofollow)$

import random



```python
import time
import random

def prime_gen(num):
    if num==2: return [2]
    elif num<2: return []
    s=list(range(3,num+1,2))
    root = num ** 0.5
    half=(num+1)//2-1
    i=0
    m=3
    while m <= root:
        if s[i]:
            j=(m*m-3)//2# this is a floor function
            s[j]=0
            while j<half:
                s[j]=0
                j+=m
        i=i+1
        m=2*i+3
    return [2]+[x for x in s if x]
#================testing for time ==========
t0 = time.time()
val = prime_gen (200)
#print ("trail print", val)

q = random.choice(val)
p = random.choice(val)

t1 = time.time()
print("time is: ", t1-t0)
print ("how random p : ",p,"q: ", q)



# ================ RSA Implementaion part =======================

n = p * q

fi = ((p-1)*(q-1))
print(fi)
```

Figure 5: prime number generator

import time

ptexts = input("Please feed in your message here") Takes user$_{i}nput$

ptext = open ("plain.txt", "wt") writes the user input to a file

ptext.write(ptexts)

ptext.close()

f = open ("plain.txt", "r")

print(f.read())

a = ([ord(c)for c in ptexts]) string to ASCII converter

if len(a) ¿ 0: for x in a: Reduced = ([x - 38]) enforcing two digit print (Reduced) else: print ("Input needed")

============Blocking and padding=====================

print ("Total number of words after removal of the space: :", Reduced)

strlen = len(a)

print ("this is the string length",strlen) for strlen is odd :

a.append("-6")

else :

tester (strlen)

print (strlen)

this is how one can append to the text in the file []

a.insert(5,32)

block = strlen/5

Prime number generator

def prime$_{g}en(num)$:

if num==2: return [2]

elif num¡2: return []

s=list(range(3,num+1,2))

root = num ** 0.5

half=(num+1)//2-1

i=0

m=3

```
while m ¡= root:
if s[i]:
j=(m*m-3)//2 this is a floor function
s[j]=0
while j¡half:
s[j]=0
j+=m i=i+1
m=2*i+3
return [2]+[x for x in s if x]
=================testing for time =========== t0 = time.time()
val = prime$_{g}en$(200)
print (”trail print”, val)
q = random.choice(val)
p = random.choice(val)
t1 = time.time()
print(”time is: ”, t1-t0)
print (”how random p : ”,p,”q: ”, q)
================= RSA Implementaion part ========================
n = p * q
phi = ((p-1)*(q-1))
print(phi)
+=============GCD calculator=================
def gcd(x, y):
while (y):
x,y = y, x
return x
def inv(x,i):
for x in range (1,i):
if (x * x)
return x
return None
def cprime(a):
p= []
for x in range (2,a):
if gcd (a,x)==1 and inv(x,phi)!= None:
p.append(x)
for x in p:
if x == inv(x,phi):
p.remove(x)
return p
val2 = cprime(phi)
e = random.choice(val2)
print (n)
```

step (i): Pick two prime numbers called p and q
step (ii): Multiply p * q to get n(Public key)
step (iii): calcultate phi (p-1)(q-1)

```
phi = ((p-1)*(q-1))
print (phi)
step (1): choosing your key e
e = 1 ¡ e ¡ phi
e = cprime(phi)
step (iv): choosing your decryption key d
formula
d *e(
d = ”
C = block**p
d = inv(e,phi)
print (”Public key is :”, e, n)
print(”Private key is :” ,d,n)
```

eavesdroppers

One solution to prevent eavesdroppers from accessing message contents is to encrypt it. This basically means to add a code to the message which changes it into a jumbled mess. If your code is sufphiciently complex, then the only people who will be able to access the original message are those who have access to the code.

The need for privacy and protection for personal data is hot issue in the 20Th century. RSA which is named after the last names of three MIT teachers (Rivest–Shamir–Adleman) is an algorithm used by modern computers to encrypt and decrypt messages. It is an asymmetric cryptography algorithm. Asymmetric makes use of two different keys for its purpose. One of the key can be given out as a public key while the other key is derived by the intended party only. This is also called public key cryptography, because one of the keys can be given to anyone.An algorithm is a sequence of doing things that could verified for the correctness. This data protection make use of two things one is encryption that is changing the original message into some other character, so even if the message ends up into the hands of unwanted entity they will not be able to acquire the original message. In this digital world where most of our personal and sensitive information is communicated over the wires where it is much easier for anyone with little know how about computer can access, change and falsify the message.

CIA. Confidentiality Integrity Authentication

What is RSA encryption? If one has a sensitive message to be sent to the other side of the country or continent, an email could be used the programming language

How it works make use User input make use of strings to ASCII make use of Blocking and padding make use of Prime number generator make use of GCD the Encryption sending and sharig the key decyphering

make use of

what is required (both hardware and software) the math behind it the chances of HOW the key is distributed distributed.net

- Conclusion, including a brief discussion on safety of the chosen method

RSA Labs holds competitions under the name RC5-56 for the public to break their codes and here are some of the results that were posted on the distributed.net website. This 56 key secret key was found with in the 47% of the 34 quadrillion keys. We know this method works! On 19 October 1997 at 1325 UTC, we found the correct solution for the RSA Labs 56-bit secret-key challenge (RC5-32/12/7). The key was 0x532B744CC20999, and it took us 250 days to locate.

Then, on 14 July 2002 at 0150 UTC we found the winning key for the RSA Labs 64-bit secret-key challenge (RC5-32/12/8). That key was 0x63DE7DC154F4D039 and took us 1,757 days to locate. As of 03 December 2002, we're now working on the 72-bit RSA Labs secret-key challenge (RC5-32/12/9).[source distributed.net]

10 Conclusion

If it was not for the time limitation it would have been educational and fun to look deeper into other cryptography methods and logic's.I might have been possible to make a much better and efficient code for the RSA as well.With faster Quantum computer on the rise no one is really sure what the future brings when it comes to data security at the personal and national level. For the time being existing protection might hold.More research needs to be done both on the hardware level and in the logic or Math level to ensure data security for the future. The other part being efficiency of codes used to implement this measures like BIG O for complexity of the code and memory demand comes other criteria like embedded on board chips for encryption and decryption has been around for a while now. As more and more people will be dependant on paperless communication in the future, a well studied and higher level of data security and safeguards needs to be in place, by the professional community.

[1]

References

[1] D. Adams. *The Hitchhiker's Guide to the Galaxy.* San Val, 1995.
- www.Distributed.net -Discrete Mathematics andIts Applications, Kenneth H. Rosen, SEVENTH EDITION -Cryptography in C and C++,MICHAELWELSCHENBACH TranslatedbyDAVIDKRAMER, SECOND EDITION -Overview of Cryptography -Hacker's Delight, Henry S. Warren, Jr., Second Edition -Introduction to Cryptography with Coding Theory,Lawrence C.Washington Wade Trappe SECOND EDITION

11 Appendix