Terna Engineering College

Computer Engineering Department

Class: TE **Sem.: VI**

Course: System Security Lab

PART A

(PART A: TO BE REFERRED BY STUDENTS)

Experiment No.02

A.1 Aim:

Write a program to implement the RSA algorithm and Digital Signature scheme using RSA / ElGamal.

A.2 Prerequisite:

Basic Knowledge of Asymmetric Key Cryptography.

A.3 Outcome:

After the successful completion of this experiment, students will be able to Analyze the various public-key cryptographic techniques and their applications.

A.4 Theory:

A. RSA Algorithm:

- **RSA** (**Rivest–Shamir–Adleman**) is one of the first public-key cryptosystems and is widely used for secure data transmission. In such a cryptosystem, the encryption key is public and it is different from the decryption key which is kept secret (private). In RSA, this asymmetry is based on the practical difficulty of the factorization of the product of two large prime numbers, the "factoring problem". The acronym RSA is made of the initial letters of the surnames of Ron Rivest, Adi Shamir, and Leonard Adleman, who first publicly described the algorithm in 1978. Clifford Cocks, an English mathematician working for the British intelligence agency Government Communications Headquarters (GCHQ), had developed an equivalent system in 1973, but this was not declassified until 1997.

- A user of RSA creates and then publishes a public key based on two large prime numbers, along with an auxiliary value. The prime numbers must be

kept secret. Anyone can use the public key to encrypt a message, but with currently published methods, and if the public key is large enough, only someone with knowledge of the prime numbers can decode the message feasibly.[2] Breaking RSA encryption is known as the RSA problem. Whether it is as difficult as the factoring problem remains an open question.

- RSA is a relatively slow algorithm, and because of this, it is less commonly used to directly encrypt user data. More often, RSA passes encrypted shared keys for symmetric key cryptography which in turn can perform bulk encryption-decryption operations at a much higher speed.

ALGORITHM:

1. Accept two prime numbers from the user (say p and q).
2. Calculate n = p * q.
3. Calculate Φ(n) as

   Φ (n) = (p - 1) * (q – 1).
4. Considering e * d = Φ(n) + 1, determine e and d where e and d are prime numbers.
5. To display information at sender as (e, n) and information at receiver as (d, n).
6. Check whether the user is sender or receiver.
7. If the user is the sender

   a. Get message M from the user.

   b. $C = M^e \bmod n$.

   c. Send ciphertext C to the receiver.

   d. Go to stop.
8. If the user is the receiver

   a. Get ciphertext C from the user.

   b. $M = C^d \bmod n$.

   c. Display plain text M to the receiver.

   d. Go to stop.
9. Ask whether the user wants to continue (yes or no?)

If yes, go to step 7.

Else go to stop.

- **EXAMPLE:**

- Choose p = 3 and q = 11
- Compute n = p * q = 3 * 11 = 33
- Compute ϕ(n) = (p - 1) * (q - 1) = 2 * 10 = 20
- Choose e such that 1 < e < ϕ(n) and e and n are co-prime. Let e = 7
- Compute a value for d such that (d * e) % ϕ(n) = 1. One solution is d = 3 [(3 * 7) % 20 = 1]
- Public key is (e, n) => (7, 33)
- Private key is (d, n) => (3, 33)
- The encryption of m = 2 is c = 27 % 33 = 29
- The decryption of c = 29 is m = 293 % 33 = 2

B. Digital Signature RSA :

Traditionally signature with a message is used to give evidence of **identity** and **intention** concerning that message. For years people have been using various types of signature to associate their identity and intention to the messages. Wax imprint, seal, and a handwritten signature are common examples. But when someone needs to sign a digital message, things turn different. In the case of signing a digital document, one cannot use any classical approach of signing, because it can be forged easily. Forger just needs to cut the signature and paste it with any other message. For signing a digital document one uses the **digital signature**.

Therefore, a digital signature is required not to be separated from the message and attached to another. That is a digital signature is required to be both message and signer dependent. For validating the signature anyone can verify the signature, so a digital signature is supposed to be verified easily.

A digital signature scheme typically consists of three distinct steps:

1. **Key generation:-** User compute their public key and corresponding private key.
2. **Signing:-** In this step, the user signs a given message with his/her private key.
3. **Verification:-** In this step user verify a signature for a given message and public key.

So the functionality provided by digital signature can be stated as follows:

Authentication:- Digital signature provides authentication of the source of the messages as a message is signed by the private key of the sender which is only known to him/her. Authentication is highly desirable in many applications.

Integrity:- Digital signature provides integrity as digital signature uniquely associated with the corresponding message. i.e. After signing a message cannot be altered if someone does it will invalidate the signature. There is no efficient method to change a message and its signature to produce a new message and valid signature without having a private key. So both sender and receiver don't have to worry about in transit alteration.

Non- repudiation:- For a valid signature sender of the message cannot deny having signed it.

RSA digital signature scheme

Suppose Alice want to send a *message**(m)** to* Bob. She can generate a digital signature using the RSA digital signature scheme [4] as follow:

Key Generation:-
She can generate a key for the RSA signature scheme:

1. Choose two distinct large prime numbers p and q.
2. Compute $n = pq$.
3. n is used as the modulus for both the public and private keys.
4. Compute $\phi(n) = (p - 1)(q - 1)$, where ϕ is Euler's totient function.
5. Choose an integer e such that $1 < e < \phi(n)$ and $gcd(e, \phi(n)) = 1$.
6. Compute $d = e^{-1} \bmod \phi(n)$.

Then the public key and private key of the user will be ***(e, n)*** and ***(d, n)*** respectively.
Now she has her public and private key. Now she can generate the signature of a message by encrypting it with her private key.
So she can generate signature corresponding to the message ***(m)*** as follow:

Signing: -

1. Represent the message ***m*** as an integer between ***0*** and ***n – 1***.
2. Sign message by raising it to the ***dth*** power modulo ***n***.

$S \equiv m^d \pmod n$

So ***S*** is the signature corresponding to message m. Now she can send message ***m*** along with the signature ***S*** to Bob.

Upon receiving the message and signature ***(m, S)***, Bob can verify the signature by decrypting it by Alice public key as follow:

Verification:-

1. Verify signature by raising it to the ***eth*** power modulo ***n***.

   $m' \equiv S^e \pmod n$

2. If ***m' = m (mod n)*** then the signature is valid otherwise not.

For a valid signature, both ***m*** and ***m'*** will be equal because:
$S \equiv m^d \pmod n$
$m' \equiv m^{de} \pmod n$
and
e is inverse of ***d***, i.e. $ed \equiv 1 \pmod{\Phi(n)}$.

So, by using above algorithm Alice can generate a valid signature ***S*** for her message ***m***, but there is a problem in above define scheme that is the length of the signature is equal to the length of the message. This is a disadvantage when a message is long.

Additional instructions for the RSA signature algorithm is as follows:

An RSA digital signature key pair consists of an RSA private key, which is used to compute a digital signature, and an RSA public key, which is used to verify a digital signature. An RSA digital signature key pair shall not be used for other purposes (e.g. key establishment).

An RSA public key consists of a modulus *n*, which is the product of two positive prime integers ***p*** and ***q*** (i.e., ***n = pq***), and a public key exponent ***e***. Thus, the RSA public key is the pair of values ***(n, e)*** and is used to verify digital signatures. The size of an RSA key pair is commonly considered to be the length of the modulus ***n*** in bits (***nlen***). The corresponding RSA private key consists of the same modulus ***n*** and a private key exponent ***d*** that depends on ***n*** and the public key exponent ***e***. Thus, the RSA private key is the pair of values ***(n, d)*** and is used to generate digital signatures. To provide security for the digital signature process, the two integers ***p*** and ***q***, and the private key exponent ***d*** shall be kept secret. The modulus n and the public key exponent ***e*** may be made known to anyone.

The Standard specifies three choices for the length of the modulus (i.e., *nlen*): 1024, 2048 and 3072 bits.

An approved hash function shall be used during the generation of key pairs and digital signatures. When used during the generation of an RSA key pair, the length in bits of the hash function output block shall meet or exceed the security strength associated with the bit length of the modulus ***n***. The security strength associated with the RSA digital signature process is no greater than the minimum of the security strength

associated with the bit length of the modulus and the security strength of the hash function that is employed. Both the security strength of the hash function used and the security strength associated with the bit length of the modulus ***n*** shall meet or exceed the security strength required for the digital signature process.

PART B

(PART B: TO BE COMPLETED BY STUDENTS)

(Students must submit the soft copy as per the following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case there is no Blackboard access available)

Roll No. 50	**Name:** AMEY THAKUR
Class: Comps TE B	**Batch:** B3
Date of Experiment: 11/02/2021	**Date of Submission:** 11/02/2021
Grade:	

B.1 The output of the RSA Algorithm with Digital Signature:

(add a snapshot of the output of RSA Algorithm with Digital Signature)

1. RSA Algorithm with Digital Signature

```
C:\Users\ameyt\Desktop>RSA.py
Enter Value of p: 17
Enter value of q: 19
Enter value of e: 23
Message: Amey Thakur
Encrypted Text: 278#29#271#315#280#50#111#279#27#110#228#
Decrypted Text: Amey Thakur
```

2. RSA Algorithm with Digital Signature

```
C:\Users\ameyt\Desktop>RSA.py
Enter the value of p = 17
Enter the value of q = 19
Enter the value of text = 23
n = 323 e = 5 t = 288 d = 173 cipher text = 245 decrypted text = 23
```

B.2 Source Code of RSA Algorithm with Digital Signature:

(Add source code of RSA Algorithm with Digital Signature)

1. **RSA.PY**

```
def gcd(a, b):
    if b==0:
        return a
    return gcd(b, a%b)


def m_inv(e, n):
    for _ in range(1,n):
        if (e*_)%n == 1:
        return _


 p = int(input("Enter Value of p: "))
q = int(input("Enter value of q: "))
e = int(input("Enter value of e: "))
n = p*q
phi = (p-1)*(q-1)


if gcd(e, phi)!=1:
    e = int(input('Enter different value for e: '))


 # e = 65537 #Fermat Number 2**2**4 + 1


d = m_inv(e, phi)
```

```
input_text = input("Message: ")

plaintext = ''

for _ in input_text:
        plaintext += str(ord(_))+'#'

plaintext = plaintext.split('#')[:-1]

ciphertext = ''
for _ in plaintext:
        ciphertext += str(int(_)**e % n)
        ciphertext += '#'

print('Encrypted Text: '+ciphertext)

decrypt = ''

ciphertext = ciphertext.split('#')[:-1]

for _ in ciphertext:
        decrypt += chr(int(_)**d % n)

print('Decrypted Text: '+decrypt)
```

2. **RSA.PY**

```
from decimal import Decimal


def gcd(a,b):
        if b==0:
                return a
        else:
                return gcd(b,a%b)
p = int(input('Enter the value of p = '))
q = int(input('Enter the value of q = '))
no = int(input('Enter the value of text = '))
n = p*q
t = (p-1)*(q-1)

for e in range(2,t):
        if gcd(e,t)== 1:
                break

for i in range(1,10):
        x = 1 + i*t
        if x % e == 0:
                d = int(x/e)
                break
ctt = Decimal(0)
ctt =pow(no,e)
```

```
ct = ctt % n


dtt = Decimal(0)

dtt = pow(ct,d)

dt = dtt % n


print('n = '+str(n)+' e = '+str(e)+' t = '+str(t)+' d = '+str(d)+' cipher text = '+str(ct)+' decrypted text = '+str(dt))
```

B.3 Question of Curiosity:

1. What is asymmetric key encryption?

Ans.

- ➔ Asymmetric Key Encryption is a form of Encryption where keys come in pairs. What one key encrypts, only the other can decrypt.
- ➔ Frequently (but not necessarily), the keys are interchangeable, in the sense that if key A encrypts a message, then B can decrypt it, and if key B encrypts a message, then key A can decrypt it. While common, this property is not essential to asymmetric encryption.
- ➔ Asymmetric cryptography, also known as public-key cryptography, uses public and private keys to encrypt and decrypt data. The keys are simply large numbers that have been paired together but are not identical (asymmetric). One key in the pair can be shared with everyone; it is called the public key. The other key in the pair is kept secret; it is called the private key. Either of the keys can be used to encrypt a message; the opposite key from the one used to encrypt the message is used for decryption. Only the user or computer that generates the key pair has the private key. The public key can be distributed to anyone who wants to send encrypted data to the holder of the private key. It's impossible to determine the private key with the public one. The two participants in the asymmetric encryption workflow are the sender and the receiver. First, the sender obtains the receiver's public key. Then the plaintext is encrypted with the asymmetric encryption algorithm using the recipient's public key, creating the ciphertext. The ciphertext is then sent to the receiver, who decrypts the ciphertext with his private key so he can access the sender's plaintext.

2. Differentiate between symmetric and asymmetric encryption algorithms.

Ans.

BASIS FOR COMPARISON	SYMMETRIC ENCRYPTION	ASYMMETRIC ENCRYPTION
Basic	Symmetric encryption uses a single key for both encryption and Decryption.	Asymmetric encryption uses a different key for encryption and decryption.
Performance	Symmetric encryption is fast in execution.	Asymmetric Encryption is slow in execution due to the high computational burden.
Algorithms	DES, 3DES, AES, and RC4.	Diffie-Hellman, RSA.
Purpose	Symmetric encryption is used for bulk data transmission.	The asymmetric encryption is often used for securely exchanging secret keys.

3. Explain different uses of the RSA algorithm.

Ans.

- ➔ RSA algorithm is safe and secure for its users through the use of complex mathematics.
- ➔ RSA algorithm is hard to crack since it involves factorization of prime numbers which are difficult to factorize.
- ➔ RSA algorithm uses the public key to encrypt data and the key is known to everyone, therefore, it is easy to share the public key.

4. List and explain different possible attacks on RSA.

Ans.

Possible Attacks on RSA:

1. Searching the Message Space
2. Guessing d
3. Cycle Attack
4. Common Modulus

5. Faulty Encryption
6. Low Exponent
7. Factoring the Public Key

Searching the Message Space:

➔ One of the seeming weaknesses of public-key cryptography is that one has to give away to everybody the algorithm that encrypts the data. If the message space is small, then one could simply try to encrypt every possible message block, until a match is found with one of the ciphertext blocks. In practice, this would be an insurmountable task because the block sizes are quite large.

Guessing d:

➔ Another possible attack is a known ciphertext attack. This time the attacker knows both the plaintext and ciphertext (they simply has to encrypt something). They then try to crack the key to discovering the private exponent, d. This might involve trying every possible key in the system on the ciphertext until it returns to the original plaintext. Once d has been discovered it is easy to find the factors of n. Then the system has been broken completely and all further ciphertexts can be decrypted.

➔ The problem with this attack is that it is slow. There is an enormous number of possible ds to try. This method is a factorizing algorithm as it allows us to factor n. Since factorizing is an intractable problem we know this is very difficult. This method is not the fastest way to factorize n. Therefore one is suggested to focus effort into using a more efficient algorithm specifically designed to factor n.

Cycle Attack:

➔ This attack is very similar to the last. The idea is that we encrypt the ciphertext repeatedly, counting the iterations until the original text appears. This number of re-cycles will decrypt any ciphertext. Again this method is very slow and for a large key, it is not a practical attack. A generalisation of the attack allows the modulus to be factored and it works faster the majority of the time. But even this will still have difficulty when a large key is used. Also the use of p^{--} strong primes aids the security.

➔ The bottom line is that the generalized form of the cipher attack is another factoring algorithm. It is not efficient, and therefore the attack is not good enough compared with modern factoring algorithms (e.g. Number Field Sieve).

Common Modulus:

- ➔ One of the early weaknesses found was in a system of RSA where the users within an organization would share the public modulus. That is to say, the administration would choose the public modulus securely and generate pairs of encryption and decryption exponents (public and private keys) and distribute them to all the employees/users. The reason for doing this is to make it convenient to manage and to write software for.

Faulty Encryption:

- ➔ Joye and Quisquater showed how to capitalise on the common modulus weakness due to a transient error when transmitting the public key. Consider the situation where an attacker, Malory, has access to the communication channel used by Alice and Bob. In other words, Malory can listen to anything that is transmitted and can also change what is transmitted. Alice wishes to talk privately to Bob but does not know his public key. She requests by sending an email, to which Bob replies. But during transmission, Malory can see the public key and decides to flip a single bit in the public exponent of Bob, changing (e,n) to (e',n).
- ➔ When Alice receives the faulty key, she encrypts the prepared message and sends it to Bob (Malory also gets it). But of course, Bob cannot decrypt it because the wrong key was used. So he lets Alice know and they agree to try again, starting with Bob re-sending his public key. This time Malory does not interfere. Alice sends the message again, this time encrypted with the correct public key.
- ➔ Malory now has two ciphertexts, one encrypted with the faulty exponent and one with the correct one. She also knows both these exponents and the public modulus. Therefore she can now apply the common modulus attack to retrieve Alice's message, assuming that Alice was foolish enough to encrypt the same message the second time.

Low Exponent:

- ➔ In the cycle attack section above, I suggested that the encrypting exponent could be chosen to make the system more efficient. Many RSA systems use e=3 to make encrypting faster. However, there is a vulnerability to this attack. If the same message is encrypted 3 times with different keys (that is same exponent, different moduli) then we can retrieve the message. The attack is based on the Chinese Remainder Theorem.

Factoring the Public Key:

➔ Factoring the public key is seen as the best way to go about cracking RSA.

5. Explain the digital signature scheme DSS.

Ans.

➔ The Digital Signature Standard is intended to be used in electronic funds transfer, software distribution, electronic mail, data storage and applications which require high data integrity assurance. The Digital Signature Standard can be implemented in software, hardware or firmware.

➔ The algorithm used behind the Digital Signature Standard is known as the Digital Signature Algorithm. The algorithm makes use of two large numbers which are calculated based on a unique algorithm which also considers parameters that determine the authenticity of the signature. This indirectly also helps in verifying the integrity of the data attached to the signature. The digital signatures can be generated only by the authorized person using their private keys and the users or public can verify the signature with the help of the public keys provided to them. However, one key difference between encryption and signature operation in the Digital Signature Standard is that encryption is reversible, whereas the digital signature operation is not.

B.4 Conclusion:

(Write an appropriate conclusion.)

The encryption and decryption time of the RSA algorithm is better than the ElGamal algorithm. Ciphertext RSA has fewer numbers than the ElGamal algorithm. The ElGamal algorithm has a ciphertext pair. Each encrypted plaintext will generate two ciphertext values. RSA algorithm and ElGamal algorithm are asymmetric algorithms which have different formulas for encryption and decryption. RSA algorithm is faster than the ElGamal algorithm. Regarding security, the ElGamal algorithm will be more challenging to solve than the RSA algorithm because ElGamal has a complicated calculation to solve discrete logarithms.

Furthermore, don't make the mistake of generalizing from RSA to conclude that any encryption scheme can be adapted as a digital signature algorithm. That kind of adaptation works for RSA and El Gamal, but not in general.