Experiment 12

Aim: Program to Implementation and analysis Digital signature scheme using RSA

Roll No.	37
Name	Mikil Lalwani
Class	D15-B
Subject	Internet Security Lab
LO Mapped	LO2: Demonstrate Key management, distribution, and user authentication.

Aim:

Program to Implementation and analysis of Digital signature scheme using RSA.

Theory:

RSA encryption algorithm is a type of public-key encryption algorithm. To better understand RSA, let's first understand what is a public-key encryption algorithm.

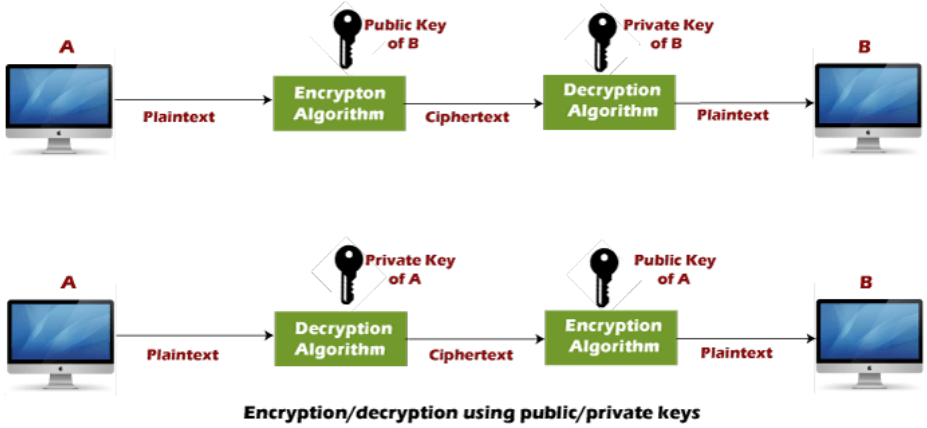
Public key encryption algorithm:

Public Key encryption algorithm is also called the Asymmetric algorithm. Asymmetric algorithms are those algorithms in which the sender and receiver use different keys for encryption and decryption. Each sender is assigned a pair of keys:

- **Public key**
- **Private key**

The **Public key** is used for encryption, and the **Private Key** is used for decryption. Decryption cannot be done using a public key. The two keys are linked, but the private key cannot be derived from the public key. The public key is well known, but the private key is secret and it is known only to the user who owns the key. It means that everybody can send a message to the user using the user's public key. But only the user can decrypt the message using his private key.

The Public key algorithm operates in the following manner:

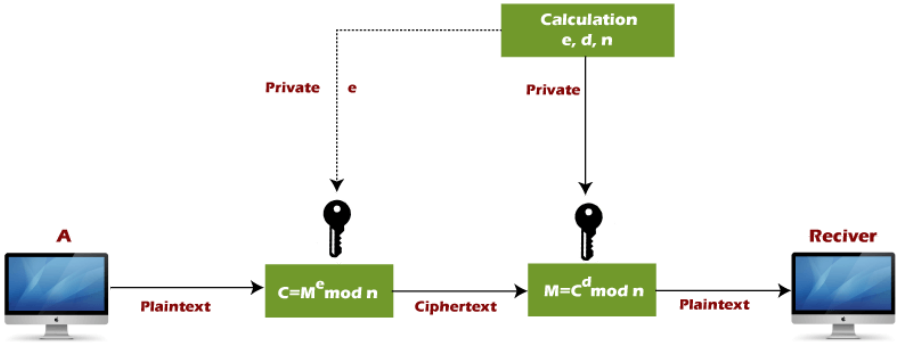


Encryption/decryption using public/private keys

- The data to be sent is encrypted by sender **A** using the public key of the intended receiver
- B decrypts the received ciphertext using its private key, which is known only to B. B replies to A encrypting its message using A's public key.
- A decrypts the received ciphertext using its private key, which is known only to him.

RSA encryption algorithm:

RSA is the most common public-key algorithm, named after its inventors **Rivest, Shamir, and Adelman (RSA).**



RSA

RSA algorithm uses the following procedure to generate public and private keys:

- Select two large prime numbers, p, and **q**.
- Multiply these numbers to find **n = p x q,** where **n** is called the modulus for encryption and decryption.
- Choose a number **e** less than **n**, such that n is relatively prime to **(p - 1) x (q -1).** It means that **e** and **(p - 1) x (q - 1)** have no common factor except 1. Choose "e" such that 1<e < φ (n), e is prime to φ (n),
  gcd (e,d(n)) =1
- If **n = p x q,** then the public key is <e, n>. A plaintext message **m** is encrypted using the public key <e, n>. To find ciphertext from the plain text following formula is used to get ciphertext C.
  $C = m^e \bmod n$
  Here**, m** must be less than **n**. A larger message (>n) is treated as a concatenation of messages, each of which is encrypted separately.
- To determine the private key, we use the following formula to calculate the d such that:
  $D_e \bmod \{(p - 1) \times (q - 1)\} = 1$
  Or
  $D_e \bmod \varphi(n) = 1$
- The private key is <d, n>. A ciphertext message **c** is decrypted using private key <d, n>. To calculate plain text **m** from the ciphertext c following formula is used to get plain text m.
  $m = c^d \bmod n$

Code:

```
def euclid(m, n):
    if n == 0:
        return m
    else:
        r = m % n
        return euclid(n, r)
```

```
def exteuclid(a, b):
	r1 = a
	r2 = b
	s1 = int(1)
	s2 = int(0)
	t1 = int(0)
	t2 = int(1)

	while r2 > 0:

		q = r1//r2
		r = r1-q * r2
		r1 = r2
		r2 = r
		s = s1-q * s2
		s1 = s2
		s2 = s
		t = t1-q * t2
		t1 = t2
		t2 = t

	if t1 < 0:
		t1 = t1 % a

	return (r1, t1)


p = 852
q = 634
n = p * q
Pn = (p-1)*(q-1)

key = []

for i in range(2, Pn):

	gcd = euclid(Pn, i)

	if gcd == 1:
		key.append(i)

e = int(313)

r, d = exteuclid(Pn, e)
if r == 1:
	d = int(d)
```

```
        print("decryption key is: ", d)

else:
        print("Multiplicative inverse for the given encryption key does not exist. Choose a
different encryption key ")

M = 999999

S = (M**d) % n

M1 = (S**e) % n

if M == M1:
        print("As M = M1, Accept the message sent by Alice")
else:
        print("As M not equal to M1, Do not accept the message sent by Alice ")
```

Output:

```
PS E:\Mikil\TE\SEM 5\Security Lab\Lab 12> python -u "e:\Mikil\TE\SE
decryption key is:  376906
As M not equal to M1, Do not accept the message sent by Alice
PS E:\Mikil\TE\SEM 5\Security Lab\Lab 12>
```

Conclusion:

Hence we have successfully analyzed and written a program to implement and analyze digital signature schemes using RSA.