SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING(IoT and Cybersecurity Including BCT)

COURSE NAME : Fundamentals Of Cryptography

II YEAR / III SEMESTER

Unit III-

Topic : RSA Algorithm

RSA algorithm is an asymmetric cryptography algorithm.

Asymmetric actually means that it works on two different keys i.e. **Public Key** and **Private Key.**

As the name describes that the Public Key is given to everyone and the Private key is kept private.

An example of asymmetric cryptography:

1.A client (for example browser) sends its public key to the server and requests some data.
2.The server encrypts the data using the client's public key and sends the encrypted data.
3.The client receives this data and decrypts it.

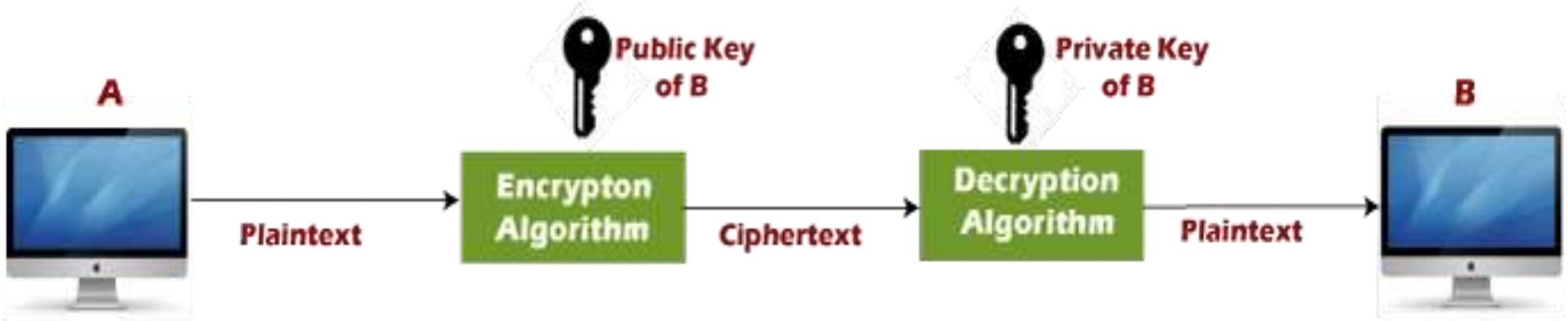
Since this is asymmetric, nobody else except the browser can decrypt the data even if a third party has the public key of the browser.

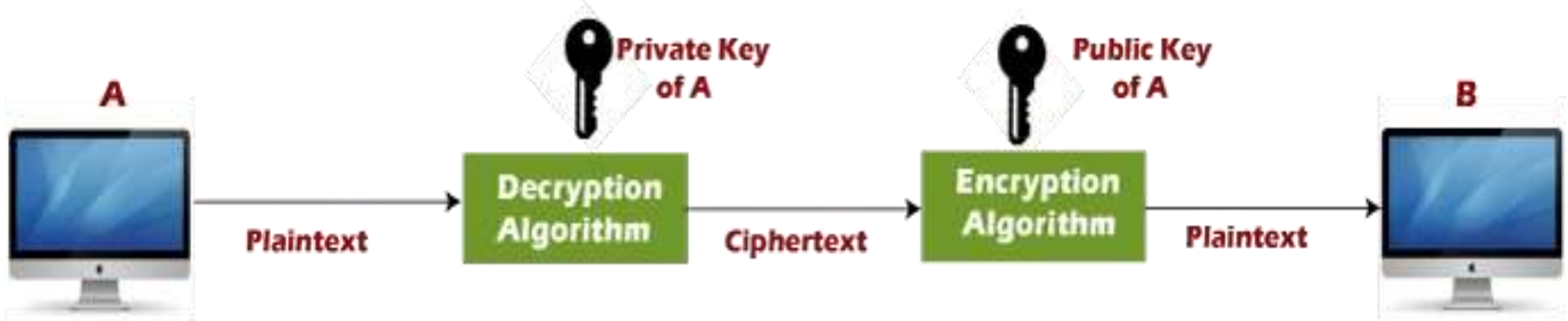
- The idea of RSA is based on the fact that it is difficult to factorize a large integer.
- The public key consists of two numbers where one number is a multiplication of two large prime numbers. And private key is also derived from the same two prime numbers.

So if somebody can factorize the large number, the private key is compromised.

Therefore encryption strength totally lies on the key size and if we double or triple the key size, the strength of encryption increases exponentially.

RSA keys can be typically 1024 or 2048 bits long, but experts believe that 1024-bit keys could be broken in the near future. But till now it seems to be an infeasible task.



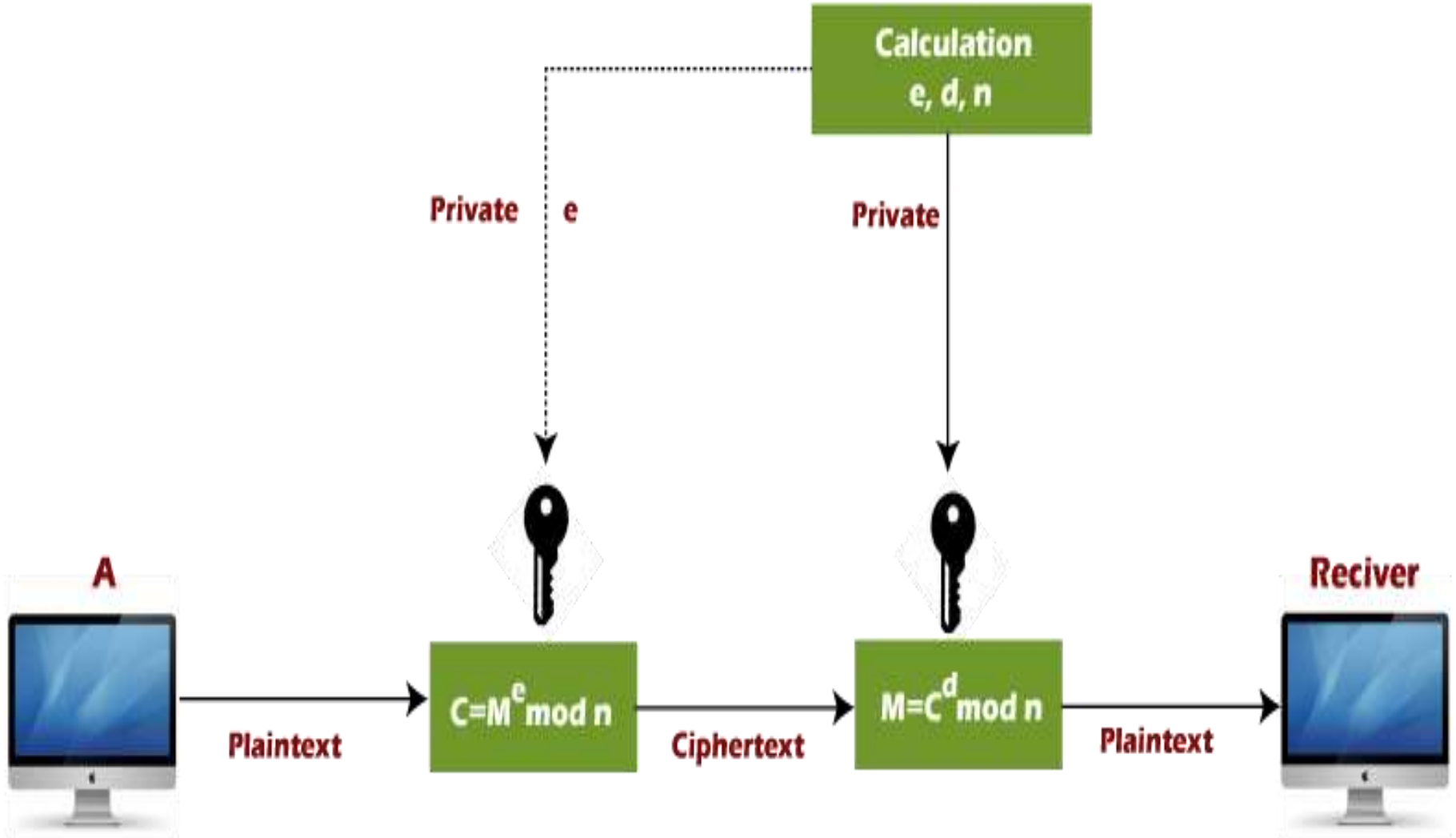


Encryption/decryption using public/private keys

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 user's public key. But only the user can decrypt the message using his private key.

- 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

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 primeφ(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 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.

$$m = c^d \bmod n$$

Example 1:

This example shows how we can encrypt plaintext 9 using the RSA public-key encryption algorithm. This example uses prime numbers 7 and 11 to generate the public and private keys.

Explanation:

Step 1: Select two large prime numbers, p, and **q**.

p = 7

q = 11

Step 2: Multiply these numbers to find **n = p x q,** where **n** is called the modulus for encryption and decryption.

First, we calculate

n = p x q

n = 7 x 11

n = 77

Step 3: Choose a number **e** less that **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.

Second, we calculate

φ (n) = (p - 1) x (q-1)

φ (n) = (7 - 1) x (11 - 1)

φ (n) = 6 x 10

φ (n) = 60
Let us now choose relative prime e of 60 as 7.
Thus the public key is <e, n> = (7, 77)
Step 4: A plaintext message **m** is encrypted using public key <e, n>. To find ciphertext from the plain text following formula is used to get ciphertext C.
To find ciphertext from the plain text following formula is used to get ciphertext C.
$C = m^e \bmod n$
$C = 9^7 \bmod 77$
C = 37
Step 5: The private key is <d, n>. To determine the private key, we use the following formula d such that:
$D_e \bmod \{(p - 1) \times (q - 1)\} = 1$
7d mod 60 = 1, which gives d = 43
The private key is <d, n> = (43, 77)
Step 6: 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.
$m = c^d \bmod n$
$m = 37^{43} \bmod 77$
m = 9
In this example, Plain text = 9 and the ciphertext = 37

Example 2:

In an RSA cryptosystem, a particular A uses two prime numbers, 13 and 17, to generate the public and private keys. If the public of A is 35. Then the private key of A is ……?.

Explanation:

Step 1: in the first step, select two large prime numbers, **p** and **q**.

p = 13

q = 17

Step 2: Multiply these numbers to find **n = p x q,** where **n** is called the modulus for encryption and decryption.

First, we calculate

n = p x q

n = 13 x 17

n = 221

Step 3: Choose a number **e** less that **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.

Second, we calculate

φ (n) = (p - 1) x (q-1)

φ (n) = (13 - 1) x (17 - 1)

φ (n) = 12 x 16

φ (n) = 192

g.c.d (35, 192) = 1

Step 3: To determine the private key, we use the following formula to calculate the d such that:

Calculate $\mathbf{d = d_e \bmod \varphi(n) = 1}$

$d = d \times 35 \bmod 192 = 1$

$d = (1 + k.\varphi(n))/e$ [let k =0, 1, 2, 3……]

Put k = 0

$d = (1 + 0 \times 192)/35$

$d = 1/35$

Put k = 1

$d = (1 + 1 \times 192)/35$

$d = 193/35$

Put k = 2

$d = (1 + 2 \times 192)/35$

$d = 385/35$

$d = 11$

The private key is <d, n> = (11, 221)

Hence, private key i.e. d = 11

Generating Public Key:

Select two prime no's. Suppose **P = 53 and Q = 59.**
Now First part of the Public key : n = P*Q = 3127.
We also need a small exponent say **e :**
But e Must be
An integer.
Not be a factor of Φ(n).
1 < e < Φ(n) [Φ(n) is discussed below],
Let us now consider it to be equal to 3.
Our Public Key is made of n and e

Generating Private Key:

We need to calculate Φ(n) :
Such that **Φ(n) = (P-1)(Q-1)**
so, Φ(n) = 3016
Now calculate Private Key, **d :**
d = (k*Φ(n) + 1) / e for some integer k
For k = 2, value of d is 2011.

Now we are ready with our – Public Key (n = 3127 and e = 3) and Private Key(d = 2011) Now we will encrypt **“HI”**:

Convert letters to numbers : H = 8 and I = 9

Thus **Encrypted Data c = $(89^e) \bmod n$**

Thus our Encrypted Data comes out to be 1394

Now we will decrypt **1394** :

Decrypted Data = $(c^d) \bmod n$

Thus our Encrypted Data comes out to be 89

8 = H and I = 9 i.e. "HI".

Below is the implementation of the RSA algorithm for

Method 1: Encrypting and decrypting small numeral values

```
# Python for RSA asymmetric cryptographic algorithm.
# For demonstration, values are
# relatively small compared to practical application
import math


def gcd(a, h):
    temp = 0
    while(1):
        temp = a % h
        if (temp == 0):
            return h
        a = h
        h = temp


p = 3
q = 7
n = p*q
e = 2
phi = (p-1)*(q-1)

while (e < phi):

    # e must be co-prime to phi and
    # smaller than phi.
    if(gcd(e, phi) == 1):
        break
    else:
        e = e+1
```

```
# Private key (d stands for decrypt)
# choosing d such that it satisfies
# d*e = 1 + k * totient

k = 2
d = (1 + (k*phi))/e

# Message to be encrypted
msg = 12.0

print("Message data = ", msg)

# Encryption c = (msg ^ e) % n
c = pow(msg, e)
c = math.fmod(c, n)
print("Encrypted data = ", c)

# Decryption m = (c ^ d) % n
m = pow(c, d)
m = math.fmod(m, n)
print("Original Message Sent = ", m)


# This code is contributed by Pranay Arora.
```

Outpu
t

```
Message data = 12.000000
Encrypted data = 3.000000
Original Message Sent = 12.000000
```

Advantages:

•**Security:** RSA algorithm is considered to be very secure and is widely used for secure data transmission.

•**Public-key cryptography:** RSA algorithm is a public-key cryptography algorithm, which means that it uses two different keys for encryption and decryption. The public key is used to encrypt the data, while the private key is used to decrypt the data.

•**Key exchange:** RSA algorithm can be used for secure key exchange, which means that two parties can exchange a secret key without actually sending the key over the network.

•**Digital signatures:** RSA algorithm can be used for digital signatures, which means that a sender can sign a message using their private key, and the receiver can verify the signature using the sender's public key.

•**Speed:** The RSA technique is suited for usage in real-time applications since it is quite quick and effective.

•**Widely used:** Online banking, e-commerce, and secure communications are just a few fields and applications where the RSA algorithm is extensively developed.

Disadvantages:

•**Slow processing speed:** RSA algorithm is slower than other encryption algorithms, especially when dealing with large amounts of data.

•**Large key size:** RSA algorithm requires large key sizes to be secure, which means that it requires more computational resources and storage space.

•**Vulnerability to side-channel attacks:** RSA algorithm is vulnerable to side-channel attacks, which means an attacker can use information leaked through side channels such as power consumption, electromagnetic radiation, and timing analysis to extract the private key.

•**Limited use in some applications:** RSA algorithm is not suitable for some applications, such as those that require constant encryption and decryption of large amounts of data, due to its slow processing speed.

•**Complexity:** The RSA algorithm is a sophisticated mathematical technique that some individuals may find challenging to comprehend and use.

•**Key Management:** The secure administration of the private key is necessary for the RSA algorithm, although in some cases this can be difficult.

•**Vulnerability to Quantum Computing:** Quantum computers have the ability to attack the RSA algorithm, potentially decrypting the data.