

# **ATME COLLEGE OF ENGINEERING**

**13<sup>th</sup> KM Stone, Bannur Road, Mysore - 560 028**



# **A T M E**

**College of Engineering**

## **DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

### **- CYBER SECURITY**



## **LECTURE NOTES**

**COURSE: CRYPTOGRAPHY AND NETWORK SECURITY**

**COURSE CODE: BCY602**

**SEMESTER: VI**

**COURSE COORDINATOR: Mrs. Suhasini**

**(ACADEMIC YEAR 2025-26)**

## INSTITUTIONAL MISSION AND VISION

### **Objectives**

- To provide quality education and groom top-notch professionals, entrepreneurs and leaders for different fields of engineering, technology and management.
- To open a Training-R & D-Design-Consultancy cell in each department, gradually introduce doctoral and postdoctoral programs, encourage basic & applied research in areas of social relevance, and develop the institute as a center of excellence.
- To develop academic, professional and financial alliances with the industry as well as the academia at national and transnational levels.
- To develop academic, professional and financial alliances with the industry as well as the academia at national and transnational levels.
- To cultivate strong community relationships and involve the students and the staff in local community service.
- To constantly enhance the value of the educational inputs with the participation of students, faculty, parents and industry.

### **Vision**

Development of academically excellent, culturally vibrant, socially responsible and globally competent human resources.

### **Mission**

- To keep pace with advancements in knowledge and make the students competitive and capable at the global level.
- To create an environment for the students to acquire the right physical, intellectual, emotional and moral foundations and shine as torch bearers of tomorrow's society.
- To strive to attain ever-higher benchmarks of educational excellence.

## Module – 2

### Chapter – 1

#### Pseudorandom Number Generators (PRNGs)

##### **Linear Congruential Generator (LCG)**

A widely used method for pseudorandom number generation is the **linear congruential generator (LCG)**, introduced by **Lehmer**. It is defined by the recurrence relation:

$$X_{n+1} = (aX_n + c) \mod m$$

where **m** is the *modulus*,

**a** the *multiplier*,

**c** the *increment*, and

**X<sub>0</sub>** the *initial seed*.

All parameters are integers, and the output lies in the range  $0 \leq X_n < m$ .

The quality of the generated sequence depends heavily on the selection of **a**, **c**, **m**, and **X<sub>0</sub>**.

For example, if **a = c = 1**, the sequence is trivial and lacks randomness. Similarly, with **a = 7**, **c = 0**, **m = 32**, **X<sub>0</sub> = 1**, the sequence only cycles through a few values (period = 4).

For generating a good sequence, **m** should be very large, typically close to  $2^{31}$ , the maximum positive integer for 32-bit systems. One of the best-known parameter choices is: **a = 16807**, **c = 0**, **m =  $2^{31} - 1$** .

This combination produces a long sequence and passes standard quality tests.

According to **PARK88**, a good pseudorandom number generator should satisfy three properties:

**T1:** It should have a **full period**, i.e., generate all numbers from 0 to m–1 before repeating.

**T2:** The sequence should **appear random**.

**T3:** The algorithm must be **efficiently implementable** on **32-bit machines**.

The **linear congruential method** with well-chosen parameters like the one above meets all three conditions. Despite this, the method is **not truly random**. Once **X<sub>0</sub>** is known, the rest of the sequence is **completely deterministic**.

In cryptographic contexts, this determinism poses a **serious weakness**. If an attacker knows the algorithm and a few values like **X<sub>0</sub>**, **X<sub>1</sub>**, **X<sub>2</sub>**, **X<sub>3</sub>**, they can use the equations:

$$X_1 = (aX_0 + c) \mod m$$

$$X_2 = (aX_1 + c) \mod m$$

$$X_3 = (aX_2 + c) \mod m$$

to solve for  $a$ ,  $c$ , and  $m$ , thus fully compromising the generator.

To enhance unpredictability, techniques such as using the internal system clock can be applied. For instance, restarting the generator with a new seed every  $N$  numbers, using the current clock time (mod  $m$ ), or adding the clock value to each generated number (mod  $m$ ), can make the sequence non-reproducible, even if the algorithm is known.

## Blum Blum Shub (BBS) generator

It is a widely used algorithm for generating secure pseudorandom bits, introduced by Blum and Blum (1986).

The procedure is as follows.

- Choose two large prime numbers,  $p$  and  $q$ , that both have a remainder of 3 when divided by 4. That is,

$$p \equiv q \equiv 3(\text{mod } 4)$$

For example, the prime numbers 7 and 11 satisfy  $7 \equiv 11 \equiv 3(\text{mod } 4)$ .

- Let  $n = p * q$ . Next, choose a random number  $s$ , such that  $s$  is relatively prime to  $n$ ; this is equivalent to saying that neither  $p$  nor  $q$  is a factor of  $s$ . Then the BBS generator produces a sequence of bits  $B_i$  according to the following algorithm:

```
x0 = s2 mod n
for i = 1 to ∞
  xi = (xi-1)2 mod n
  Bi = xi mod 2
```

Thus, the least significant bit is taken at each iteration.

The BBS is referred to as a **cryptographically secure pseudorandom bit generator (CSPRBG)** because it passes the **next-bit test**. A generator passes this test if **no polynomial-time algorithm** can predict the **(k+1)<sup>th</sup> bit** given the first **k bits** with probability better than  $\frac{1}{2}$ . In simple terms, the output is **unpredictable**.

The **security** of the BBS generator is based on the **difficulty of factoring  $n$**  into its two prime components  **$p$  and  $q$** . If an attacker knows only  **$n$** , but not  **$p$  and  $q$** , then reconstructing the sequence becomes **computationally infeasible**.

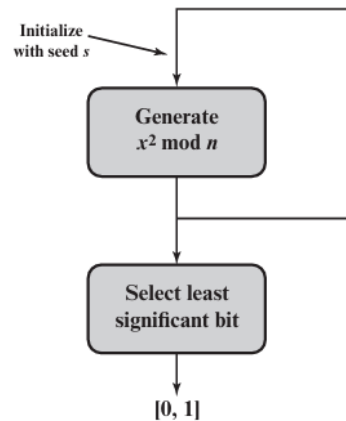


Figure 8.3 Blum Blum Shub Block Diagram

## CHAPTER 2

### PUBLIC-KEY CRYPTOGRAPHY AND RSA

**Public-key cryptography is asymmetric**, involving the use of two separate keys, in contrast to symmetric encryption, which uses only one key.

#### **Terminology Related to Asymmetric Encryption**

##### **Asymmetric Keys**

Two related keys, a public key and a private key, that are used to perform complementary operations, such as encryption and decryption or signature generation and signature verification.

##### **Public Key Certificate**

A digital document issued and digitally signed by the private key of a Certification Authority that binds the name of a subscriber to a public key. The certificate indicates that the subscriber identified in the certificate has sole control and access to the corresponding private key.

##### **Public Key (Asymmetric) Cryptographic Algorithm**

A cryptographic algorithm that uses two related keys, a public key and a private key. The two keys have the property that deriving the private key from the public key is computationally infeasible.

##### **Public Key Infrastructure (PKI)**

A set of policies, processes, server platforms, software and workstations used for the purpose of administering certificates and public-private key pairs, including the ability to issue, maintain, and revoke public key certificates.

## Principles of Public-Key Cryptosystems

The concept of public-key cryptography evolved from an attempt to attack two of the most difficult problems associated with symmetric encryption.

### **The first problem is that of key distribution.**

As we have seen, key distribution under symmetric encryption requires either

1. Two communicants already share a key, which somehow has been distributed to them or
2. The use of a key distribution center.

Whitfield Diffie, one of the discoverers of public-key encryption reasoned that this second requirement negated the very essence of cryptography: the ability to maintain **total secrecy** over your own communication.

**The second problem is digital signatures.** The electronic messages and documents would need the equivalent of signatures used in paper documents. That is digital message had been sent by a particular person? This is a somewhat broader requirement than that of authentication, Diffie and Hellman achieved breakthrough by coming up with a method that addressed both problems and was radically different from all previous approaches to cryptography.

## Public-Key Cryptosystems

Asymmetric algorithms rely on one key for encryption and a different but related key for decryption. These algorithms have the following important characteristic:

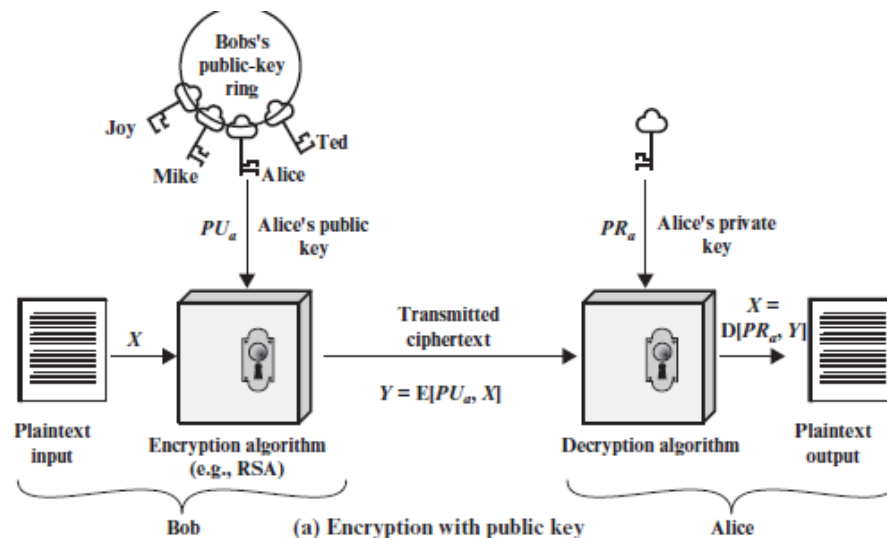
- It is computationally infeasible to determine the decryption key given only knowledge of the cryptographic algorithm and the encryption key.
  - Either of the two related keys can be used for encryption, with the other used for decryption.
- Anyone knowing the public key can encrypt messages or verify signatures, but **cannot** decrypt messages or create signatures, thanks to some clever use of number theory.

A public-key encryption scheme has six ingredients

- **Plaintext:** This is the readable message or data that is fed into the algorithm as input.
- **Encryption algorithm:** The encryption algorithm performs various transformations on the plaintext.
- **Public and private keys:** This is a pair of keys that have been selected so that if one is used for encryption, the other is used for decryption. The exact transformations performed by the algorithm depend on the public or private key that is provided as input.

- **Ciphertext:** This is the scrambled message produced as output. It depends on the plaintext and the key. For a given message, two different keys will produce two different ciphertexts.
- **Decryption algorithm:** This algorithm accepts the ciphertext and the matching key and produces the original plaintext.

### Public key cryptography for providing confidentiality (secrecy)



The essential steps are the following.

1. Each user generates a pair of keys to be used for the encryption and decryption of messages.
2. Each user places one of the two keys in a public register or other accessible file. This is the public key. The companion key is kept private. As Figure 9.1a suggests, each user maintains a collection of public keys obtained from others.
3. If Bob wishes to send a confidential message to Alice, Bob encrypts the message using Alice's public key.
4. When Alice receives the message, she decrypts it using her private key. No other recipient can decrypt the message because only Alice knows Alice's private key.

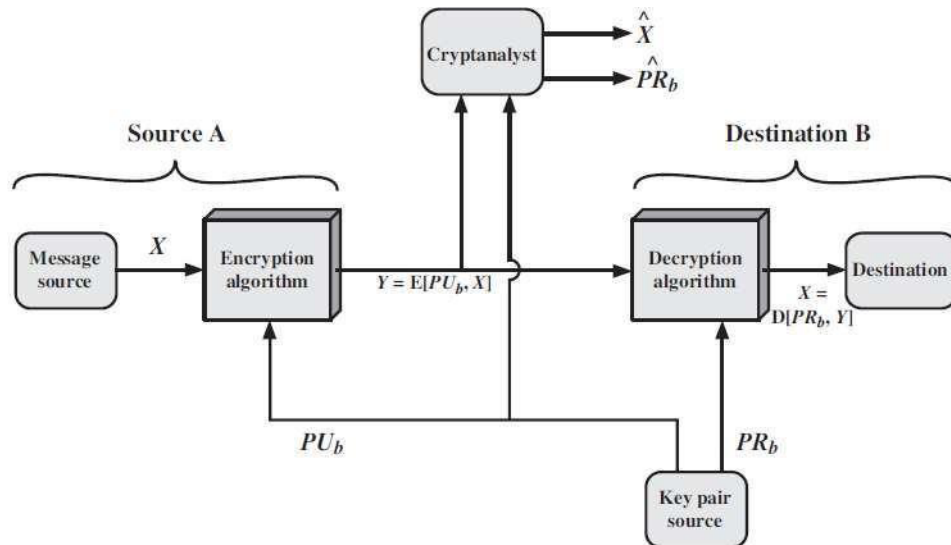


Figure 9.2 Public-Key Cryptosystem: Secrecy

There is some source A that produces a message in plaintext  $X = [X_1, X_2, \dots, X_M]$ . The  $M$  elements of  $X$  are letters in some finite alphabet. The message is intended for destination B.

B generates a related pair of keys: a public key,  $PU_b$ , and a private key,  $PR_b$ .

$PR_b$  is known only to B, whereas  $PU_b$  is publicly available and therefore accessible by A.

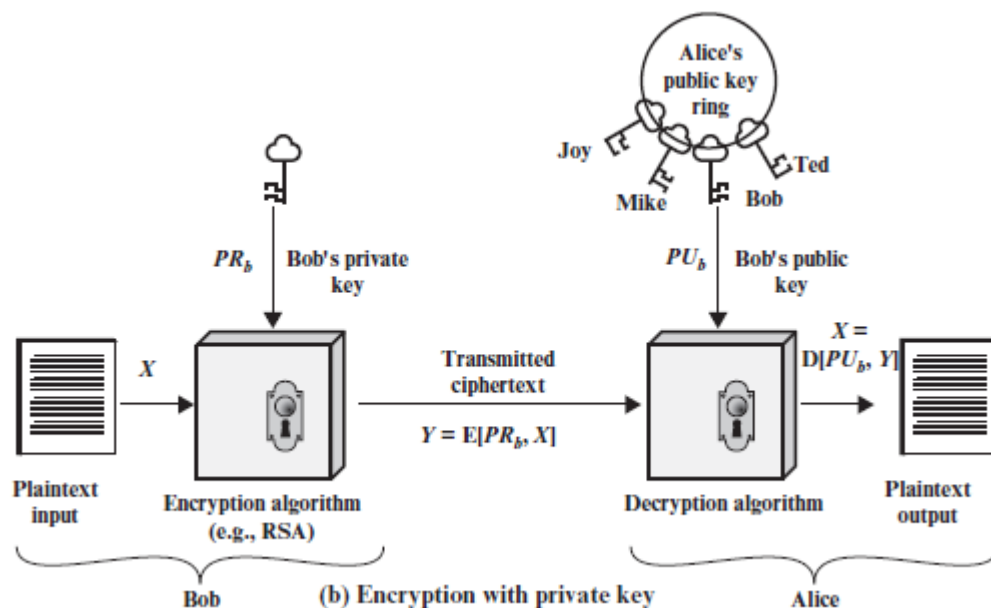
With the message  $X$  and the encryption key  $PU_b$  as input, A forms the ciphertext  $Y = [Y_1, Y_2, \dots, Y_N]$ :

$$Y = E(PU_b, X)$$

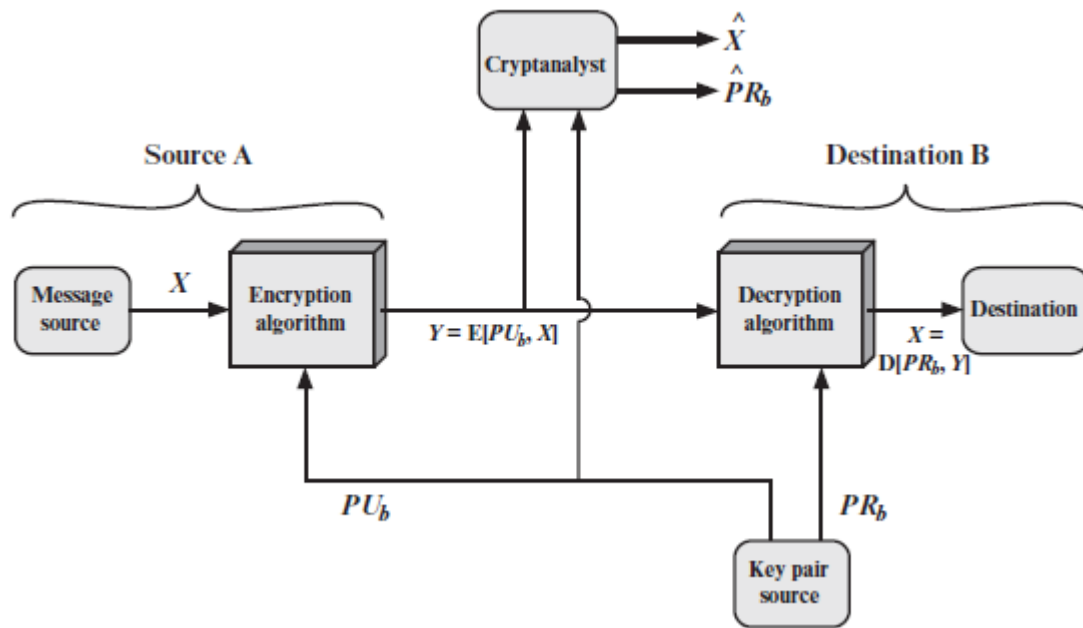
The intended receiver, in possession of the matching private key, is able to invert the transformation:

$$X = D(PR_b, Y)$$

## Public key cryptography for proving Authentication







Public-key schemes can be used for either secrecy or authentication, or both (as shown here). There is some source A that produces a message in plaintext  $X$ . The message is intended for destination B.

- B generates a related pair of keys: a public key,  $PU_b$ , and a private key,  $PR_b$ .  $PR_b$  is known only to B, whereas  $PU_b$  is publicly available and therefore accessible by A.
- With the message  $X$  and the encryption key  $PU_b$  as input, A forms the ciphertext  $Y = E(PU_b, X)$ .
- The intended receiver, in possession of the matching private key, is able to invert the transformation:  $X = D(PR_b, Y)$ .
- An adversary, observing  $Y$  and having access to  $PU_b$ , but not having access to  $PR_b$  or  $X$ , must attempt to recover  $X$  and/or  $PR_b$ . This provides confidentiality.
- Public-key encryption can also provide authentication:

$$Y = E(PR_a, X); X = D(PU_a, Y)$$

- To provide both the authentication function and confidentiality have a double use of the public key scheme (as shown here):

$$Z = E(PU_b, E(PR_a, X))$$

$$X = D(PU_a, D(PR_b, Z))$$

## RSA

It is the most common public key algorithm.

- This RSA name is get from its inventors first letter (Rivest (R), Shamir (S) and Adleman (A)) in the year 1977.
- The RSA scheme is a block cipher in which the plaintext & ciphertext are integers between 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  $2^{1024}$ .

### Description of the Algorithm:

- RSA algorithm uses an expression with exponentials.
- In RSA 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  $\log_2(n)$
- **RSA** uses two exponents 'e' and 'd' where e→public and d→private.
  - Encryption and decryption are of following form, for some PlainText 'M' and CipherText block 'C'.

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

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

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

Both sender and receiver must know the value of n.

The sender knows the value of 'e' & only the receiver knows the value of 'd' thus this is a public key encryption algorithm with a.

Public key PU={e, n}

Private key PR={d, n}

### Requirements:

The RSA algorithm to be satisfactory for public key encryption, the following requirements must be met:

1. It is possible to find values of e, d n such that " $M^{ed} \bmod n = M$ " for all  $M < n$
2. It is relatively easy to calculate " $M^e \bmod n$ " and " $C^d \bmod n$ " for  $M < n$
3. It is infeasible to determine "d" given 'e' & 'n'. The " $M^{ed} \bmod n = M$ " relationship holds if 'e' & 'd' are multiplicative inverses modulo  $\phi(n)$ .

$\phi(n) \rightarrow$  Euler Totient function

For p,q primes where  $p \neq q$  and  $p \cdot q$ .

$$\phi(n) = \phi(pq) = (p-1)(q-1)$$

Then the relation between 'e' & 'd' can be expressed as “  $ed \bmod \phi(n)=1$  ” 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: according to the rules of modular arithmetic, this is true only if 'd' (and 'e') is relatively prime to  $\phi(n)$ .

Equivalently  $\gcd(\phi(n), d)=1$ .

### Steps of RSA algorithm:

Step 1→Select 2 prime numbers p & q

Step 2→Calculate  $n=pq$

Step 3→Calculate  $\phi(n)=(p-1)(q-1)$

Step 4→ Select or find integer e (public key) which is relatively prime to  $\phi(n)$ .

ie., e with  $\gcd(\phi(n), e)=1$  where  $1 < e < \phi(n)$ .

Step 5→ Calculate “d” (private key) by using following condition  $d < \phi(n)$ .

$$ed \equiv 1 \bmod \phi(n)$$

Step 6→ Perform encryption by using

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

Step 7→ perform Decryption by using

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

### Example:

1. Select two prime numbers,  $p = 17$  and  $q = 11$ .
2. Calculate  $n = pq = 17 \times 11 = 187$ .
3. Calculate  $\phi(n) = (p - 1)(q - 1) = 16 \times 10 = 160$ .
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  $23 * 7 = 161 = (1 \times 160) + 1$ ;  $d$  can be calculated using the extended Euclid's algorithm.

The resulting keys are public key  $PU = \{7, 187\}$  and private key  $PR = \{23, 187\}$ .

The example shows the use of these keys for a plaintext input of  $M = 88$ . For encryption, we need to calculate  $C = 88^7 \bmod 187$ . Exploiting the properties of modular arithmetic, we can do this as follows.

$$88^7 \bmod 187 = [(88^4 \bmod 187) \times (88^2 \bmod 187) \times (88^1 \bmod 187)] \bmod 187$$

$$88^1 \bmod 187 = 88$$

$$88^2 \bmod 187 = 7744 \bmod 187 = 77$$

$$88^4 \bmod 187 = 59,969,536 \bmod 187 = 132$$

$$88^7 \bmod 187 = (88 \times 77 \times 132) \bmod 187 = 894,432 \bmod 187 = 11$$

For decryption, we calculate  $M = 11^{23} \bmod 187$ :

$$11^{23} \bmod 187 = [(11^1 \bmod 187) \times (11^2 \bmod 187) \times (11^4 \bmod 187) \times (11^8 \bmod 187) \times (11^8 \bmod 187)] \bmod 187$$

$$11^1 \bmod 187 = 11$$

$$11^2 \bmod 187 = 121$$

$$11^4 \bmod 187 = 14,641 \bmod 187 = 55$$

$$11^8 \bmod 187 = 214,358,881 \bmod 187 = 33$$

$$11^{23} \bmod 187 = (11 \times 121 \times 55 \times 33 \times 33) \bmod 187 = 79,720,245 \bmod 187 = 88$$

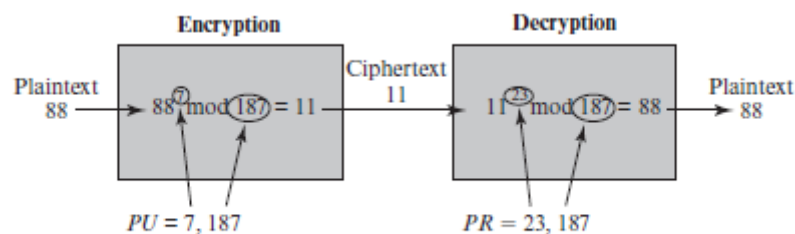


Figure 9.6 Example of RSA Algorithm

## The Security of RSA

Four possible approaches to attacking the RSA algorithm are as follows:

- **Brute force:** This involves trying all possible private keys.
- **Mathematical attacks:** There are several approaches, all equivalent in effort to factoring the product of two primes.
- **Timing attacks:** These depend on the running time of the decryption algorithm.
- **Chosen ciphertext attacks:** This type of attack exploits properties of the RSA algorithm.

### **i)Brute force:**

The defense against the brute-force approach is the same for RSA as for other cryptosystems, namely, **use a large key space**. Thus, the larger the number of bits in  $d$ , the better.

However, because the calculations involved, both in key generation and in encryption/decryption, are complex, the larger the size of the key, the slower the system will run.

**ii)The Factoring Problem:**

We can identify three approaches to attacking RSA mathematically:

- Factor  $n$  into its two prime factors. This enables calculation of  $\phi(n)=(p-1)*(q-1)$ , which, in turn, enables determination of  $d = e^{-1} \bmod \phi(n)$ .
- Determine  $\phi(n)$  directly, without first determining  $p$  and  $q$ . Again, this enables determination of  $d = e^{-1} \bmod \phi(n)$ .
- Determine  $d$  directly, without first determining  $\phi(n)$ .

Determining  $\phi(n)$  given  $n$  is equivalent to factoring  $n$ . With presently known algorithms, determining  $d$  given  $e$  and  $n$  appears to be at least as time-consuming as the factoring problem. Hence, we can use factoring performance as a benchmark against which to evaluate the security of RSA.

To avoid values of  $n$  that may be factored more easily, the algorithm's inventors suggest the following constraints on  $p$  and  $q$ :

1.  $p$  and  $q$  should differ in length by only a few digits. Thus, for a 1024-bit key (309 decimal digits), both  $p$  and  $q$  should be on order of 1075 to 10100.
2. Both  $(p-1)$  and  $(q-1)$  should contain a large prime factor.
3.  $\gcd(p-1, q-1)$  should be small.

**iii)Timing Attacks:**

- Timing Attacks demonstrated that a snooper can determine a private key by keeping track of how long a computer takes to decipher messages.
- Timing Attacks are based on observing how long it takes to compute the cryptographic operations. Timing attacks are applicable not just to RSA, but to other public-key cryptography systems.
- This attack is alarming for two reasons: It comes from a completely unexpected direction and it is a ciphertext-only attack.

The timing attack is a serious threat; there are simple countermeasures that can be used, including using constant exponentiation time algorithms, adding random delays, or using blind values in calculations.

- **Constant exponentiation time:** Ensure that all exponentiations take the same amount of time before returning a result. This is a simple fix but does degrade performance.
- **Random delay:** Better performance could be achieved by adding a random delay to the

exponentiation algorithm to confuse the timing attack. Kocher points out that if defenders don't add enough noise, attackers could still succeed by collecting additional measurements to compensate for the random delays.

- **Blinding:** Multiply the ciphertext by a random number before performing exponentiation. This process prevents the attacker from knowing what ciphertext bits are being processed inside the computer and therefore prevents the bit-by-bit analysis essential to the timing attack.

#### iv) Chosen Ciphertext Attacks:

The RSA algorithm is vulnerable to a chosen ciphertext attack (CCA).

- CCA is defined as an attack in which adversary chooses a number of ciphertexts and is then given the corresponding plaintexts, decrypted with the target's private key.
- The adversary exploits properties of RSA and selects blocks of data that, when processed using the target's private key, yield information needed for cryptanalysis.
- More sophisticated variants need to modify the plaintext using a procedure known as **optimal asymmetric encryption padding (OAEP)**.

To counter such attacks RSA Security, a leading RSA vendor and former holder of the RSA patent, recommends modifying the plaintext using a procedure known as optimal asymmetric encryption padding (OAEP).

As a first step the message  $M$  to be encrypted is padded. A set of optional parameters  $P$  is passed through a hash function  $H$ .

The output is then padded with zeros to get the desired length in the overall data block (DB). Next, a random seed is generated and passed through another hash function, called the mask generating function (MGF).

- The resulting hash value is bit-by-bit XORed with DB to produce a masked DB.
- The masked DB is in turn passed through the MGF to form a hash that is XORed with the seed to produce the masked seed.
- The concatenation of the masked seed and the masked DB forms the encoded message EM.
- Note that the EM includes the padded message, masked by the seed, and the seed, masked by the masked DB. The EM is then encrypted using RSA.