
# University of Technology
# الجامعة التكنولوجية

## Computer Science Department
## قسم علوم الحاسوب

# Public Key Cryptography
# التشفير بالمفتاح العام

**Professor: Dr. Alaa. Kadhim**

**ا.د علاء كاظم**

**Lect. Enas Tariq**

**ا.م. ايناس طارق**


**cs.uotechnology.edu.iq**

# Public key Cryptography

*University of Technology*
*Computer science branch*
*3th Class-first course*
*Assistant Professor : Dr. Alaa .Kadhim*
*Lecturer: Enas Tariq*

Computers are now fond in every layer society, and information is bei ng communicated and processed automatically on a large scale. Such as medical and financial files, automatic banking, video -phones, pay-tv, facsimiles, tele-shopping, and global computer networks. In all these cases there is a growing need for the protection of information to safeguard economic interests, to prevent fraud and to ensure privacy.

2023-2024

# Public Key Cryptography

# Lecture 1

## 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) that 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 (along with Martin Hellman, both at Stanford University at the time), reasoned that this second requirement negated the very essence of cryptography: the ability to maintain total secrecy over your own communication. As Diffie put it, "What good would it do after all to develop impenetrable cryptosystems, if their users were forced to share their keys with a KDC that could be compromised by either burglary or subpoena?"

The second problem that Diffie pondered, and one that was apparently unrelated to the first was that of "digital signatures." If the use of cryptography was to become widespread, not just in military situations but for commercial and private purposes, then electronic messages and documents would need the equivalent of signatures used in paper documents. That is, could a method be devised that would stipulate, to the satisfaction of all parties, that a digital message had been sent by a particular person? This is a somewhat broader requirement than that of authentication

- ***Diffie and Hellman*** *achieved an astounding breakthrough in 1976 by coming up with a method that addressed both problems and that was radically different from all previous approaches to cryptography, going back over four millennia*.

### *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.**

In addition, some algorithms, such as RSA, also exhibit the following characteristic:

- Either of the two related keys can be used for encryption, with the other used for decryption.

*A public-key encryption scheme has six ingredients (in the following Figure; compare with anther Figure)*

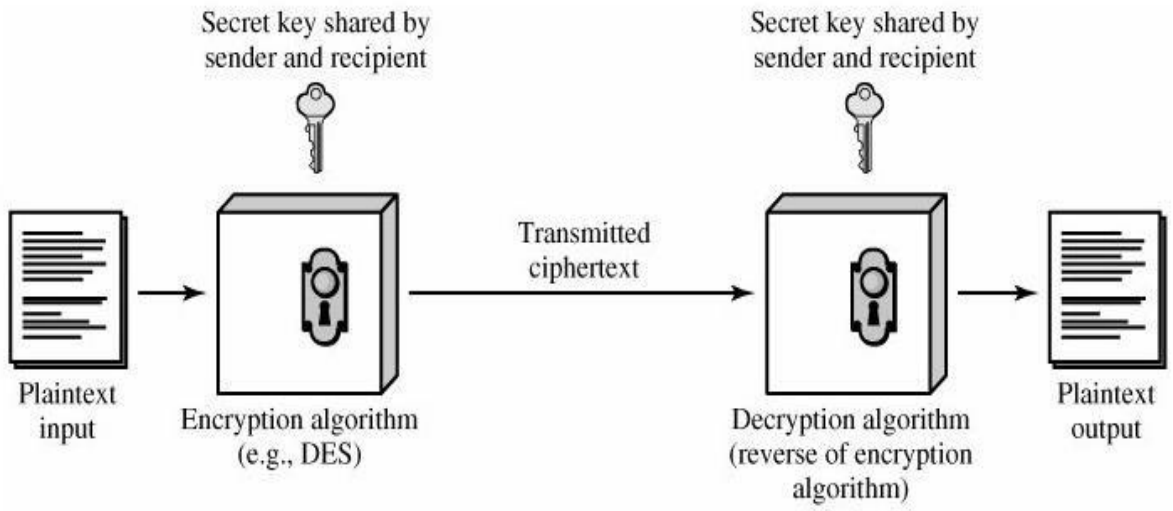



**Figure:** Simplified Model of Conventional Encryption

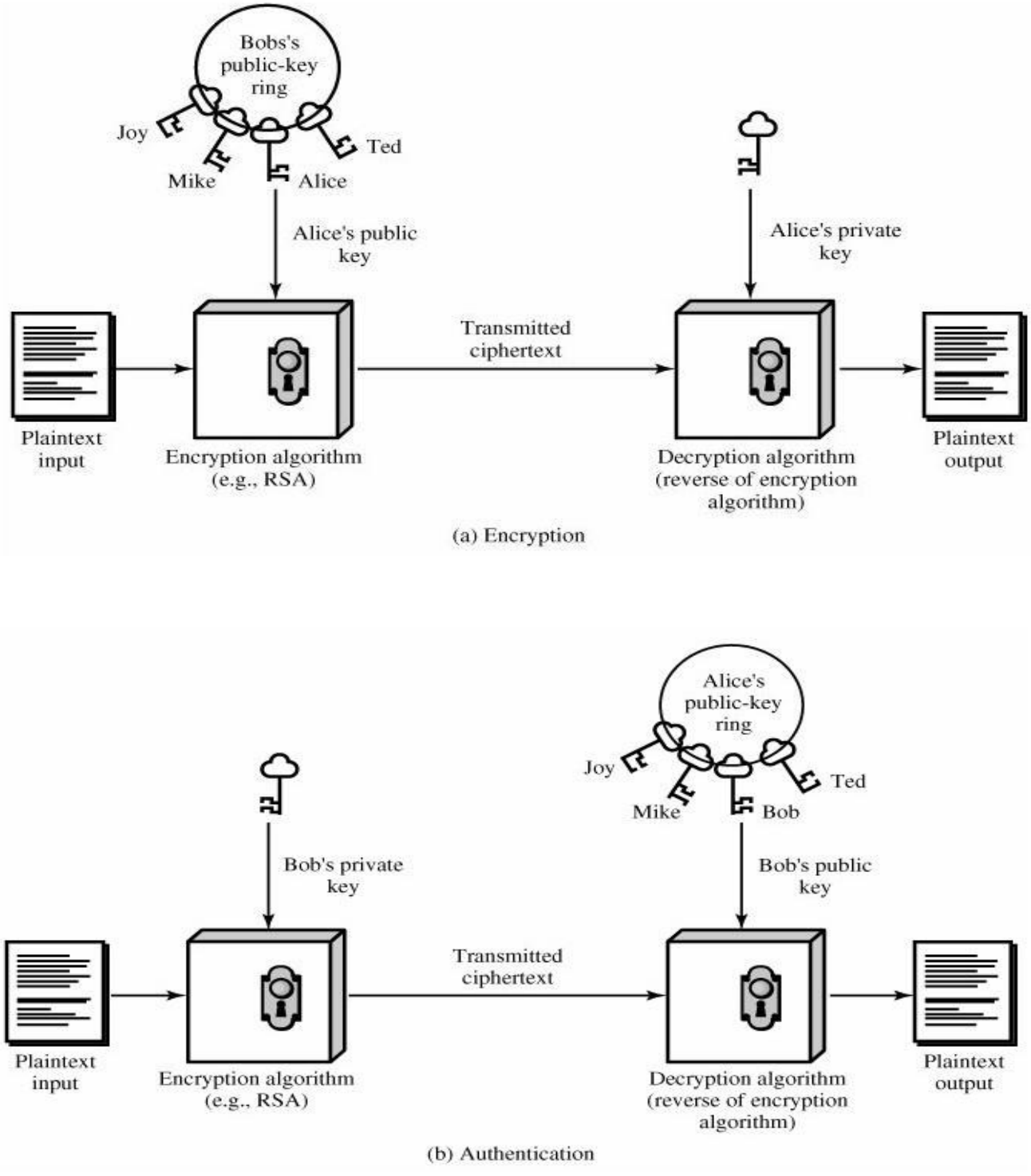



**Figure:** Public-Key Cryptography

- ***Plaintex***t: 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.

***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, 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.

*With this approach*, all participants have access to public keys, and private keys are generated locally by each participant and therefore need never be distributed. As long as a user's private key remains protected and secret, incoming communication is secure. At any time, a system can change its private key and publish the companion public key to replace its old public key.

In the following Table summarizes some of the important aspects of symmetric and public-key encryption. To discriminate between the two, we refer to the key used in symmetric encryption as a

secret key. The two keys used for asymmetric encryption are referred to as the **public key** and the **private key**. Invariably, the private key is kept secret, but it is referred to as a private key rather than a secret key to avoid confusion with symmetric encryption.

**Table**: Conventional and Public-Key Encryption

| ***Conventional Encryption*** | ***Public-Key Encryption*** |
|---|---|
| Needed to Work: | Needed to Work: |
| 1. The same algorithm with the same key is used for encryption and decryption.<br>2. The sender and receiver must share the algorithm and the key. | 1. One algorithm is used for encryption and decryption with a pair of keys, one for encryption and one for decryption.<br>2. The sender and receiver must each have one of the matched pair of keys (not the same one). |
| Needed for Security: | Needed for Security: |
| 1. The key must be kept secret.<br>2. It must be impossible or at least impractical to decipher a message if no other information is available.<br>3. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key. | 1. One of the two keys must be kept secret.<br>2. It must be impossible or at least impractical to decipher a message if no other information is available.<br>3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key. |

*Let us take a closer look at the essential elements of a public-key encryption scheme, using above Figure (Public) compare with Figure private). There is some source A that produces a message in plaintext, $X = [X_1, X_2,, X_M,]$. 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, $PU_b$. $PU_b$ is known only to B, whereas $PU_b$ is publicly available and therefore accessible by A.*

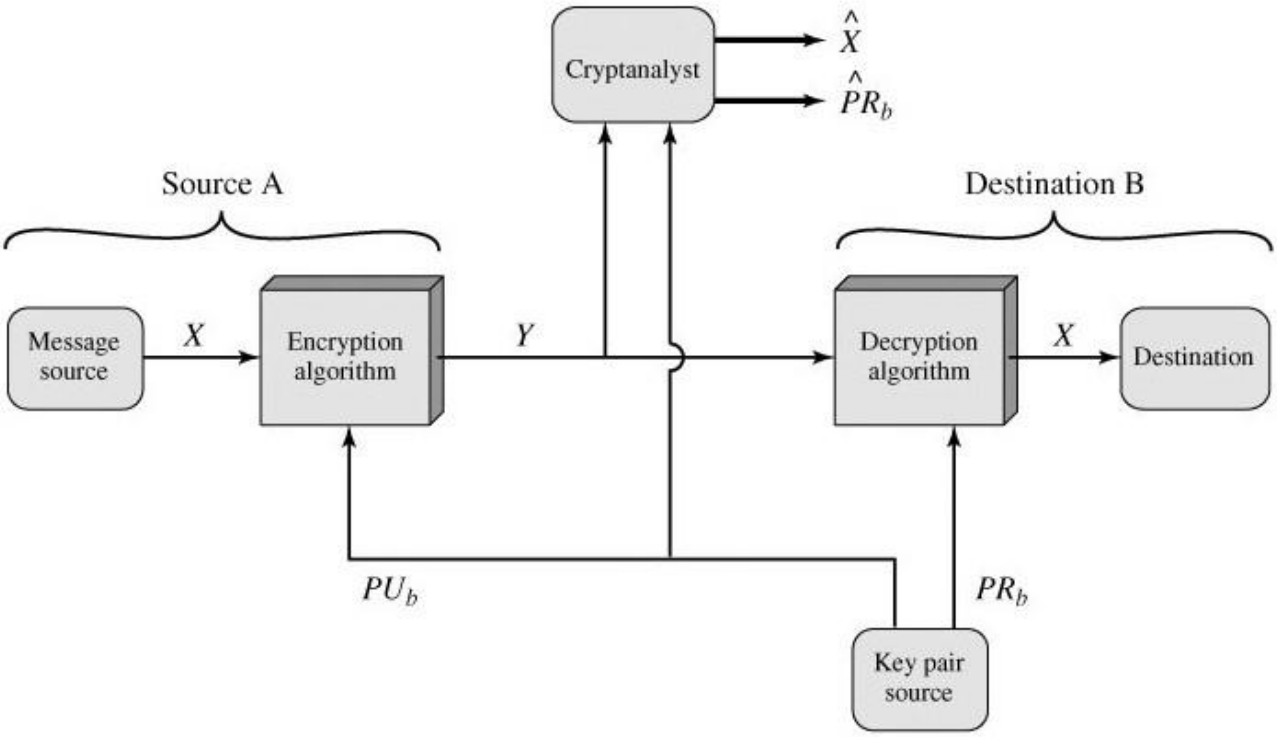



**Figure** : Public-Key Cryptosystem: Secrecy

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

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

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

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

***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 $PR_b$. It is assumed that the adversary does have knowledge of the encryption (E) and decryption (D) algorithms. If the adversary is interested only in this particular message, then the focus of effort is to recover X, by generating a plaintext estimate ($X^{\wedge}$) Often, however, the adversary is interested

in being able to read future messages as well, in which case an attempt is made to recover $PR_b$ by generating an estimate *(P^Ru)*.

We mentioned earlier that either of the two related keys can be used for encryption, with the other being used for decryption. This enables a rather different cryptographic scheme to be implemented. Whereas the scheme illustrated in above Figure provides confidentiality, Figures (Public ) and authentic show the use of public-key encryption to provide authentication:

$Y = E\ (PR_a \sim X)$

$Y = E\ (PU_a \sim Y)$

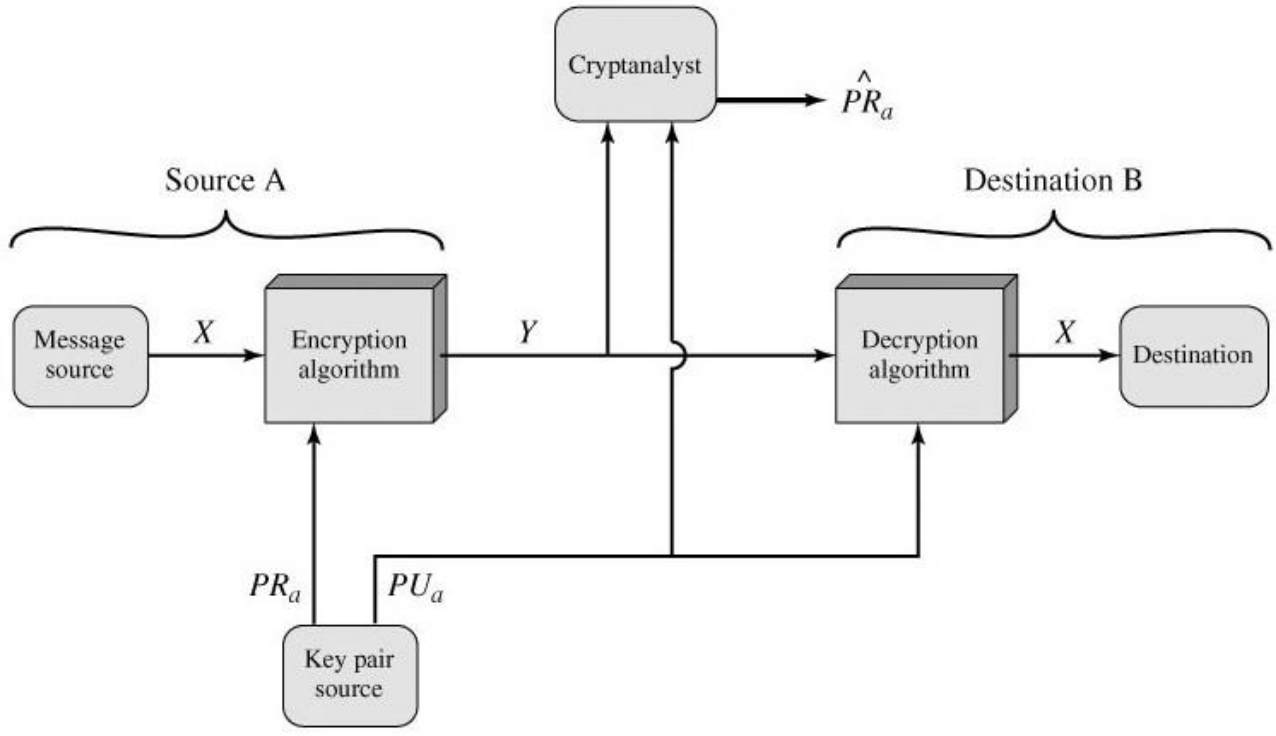



**Figure :** Public-Key Cryptosystem: Authentication

*In this case*, A prepares a message to B and encrypts it using A's private key before transmitting it. B can decrypt the message using A's public key. Because the message was encrypted using A's private key, only A could have prepared the message. Therefore, the entire encrypted message serves as a ***digital signature***. In addition, it is impossible to alter the message without access to A's private key, so the message is authenticated both in terms of source and in terms of data integrity.

In the preceding scheme, the entire message is encrypted, which, although validating both author and contents, requires a great deal of storage. Each document must be kept in plaintext to be used for practical purposes. A copy also must be stored in ciphertext so that the origin and contents can be verified in case of a dispute. A more efficient way of achieving the same results is to encrypt a small block of bits that is a function of the document. Such a block, called an authenticator, must have the property that it is infeasible to change the document without changing the authenticator. If the authenticator is encrypted with the sender's private key, it serves as a signature that verifies origin, content, and sequencing.

It is important to emphasize that the encryption process depicted in Figures (first figure b) and figure authentication does not provide confidentiality. That is, the message being sent is safe from alteration but not from eavesdropping. This is obvious in the case of a signature based on a portion of the message, because the rest of the message is transmitted in the clear. Even in the case of complete encryption, as shown in Figure (***Public-Key Cryptosystem: Authentication***), there is no protection of confidentiality because any observer can decrypt the message by using the sender's public key.

It is, however, possible to provide both the authentication function and confidentiality by a double use of the public-key scheme (*Public-Key Cryptosystem: Authentication and Secrecy*):

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

$$X = D(PU_a \sim E(PR_b \sim Z))$$

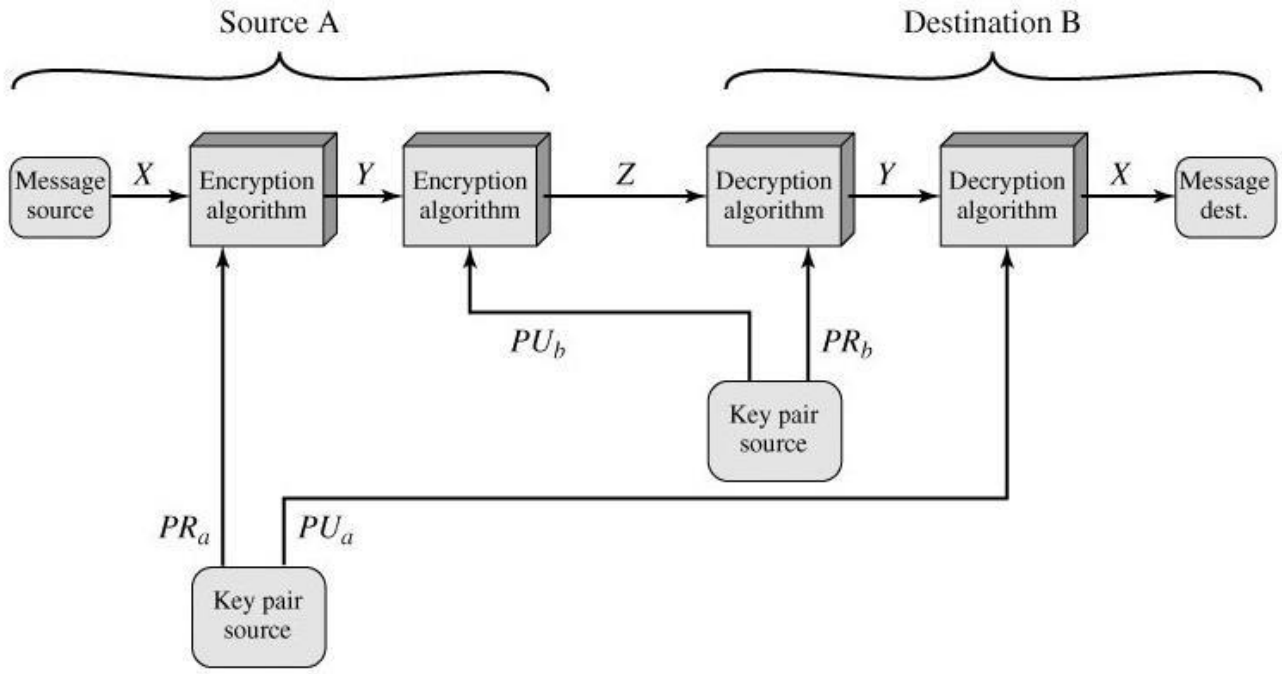



**Figure**: Public-Key Cryptosystem: Authentication and Secrecy

*In this case*, we begin as before by encrypting a message, using the sender's private key. This provides the digital signature. Next, we encrypt again, using the receiver's public key. The final ciphertext can be decrypted only by the intended receiver, who alone has the matching private key. Thus, confidentiality is provided. The disadvantage of this approach is that the public-key algorithm, which is complex, must be exercised four times rather than two in each communication.

### *Applications for Public-Key Cryptosystems*

Before proceeding, we need to clarify one aspect of public-key cryptosystems that is otherwise likely to lead to confusion. Public-key systems are characterized by the use of a cryptographic algorithm with two keys, one held private and one available publicly. Depending on the application, the sender uses either the sender's private key or the receiver's public key, or both, to perform some type of cryptographic function. In broad terms, we can classify the use of public-key cryptosystems into three categories:

- Encryption/decryption: The sender encrypts a message with the recipient's public key.

- Digital signature: The sender "signs" a message with its private key. Signing is achieved by a cryptographic algorithm applied to the message or to a small block of data that is a function of the message.
- **Key exchange:** Two sides cooperate to exchange a session key. Several different approaches are possible, involving the private key(s) of one or both parties.

Table: Applications for Public-Key Cryptosystems

| Algorithm | Encryption/Decryption | Digital Signature | Key Exchange |
|---|---|---|---|
| RSA | Yes | Yes | Yes |
| Elliptic Curve | Yes | Yes | Yes |
| Diffie-Hellman | No | No | Yes |
| DSS | No | Yes | No |

## Lecture 2:

### Asymmetric Public-key Cryptosystems

Public-key cryptography became public soon after Whitefield Diffie and Martin Hellman (1976) proposed the innovative concept of an exponential key exchange scheme. Since 1976, numerous public key algorithms have been proposed, but many of them have since been broken. Of the many algorithms that are still considered to be secure, most are impractical.

**Only a few public-key algorithms are both secure and practical**. Of these, only some are suitable for encryption. Others are only suitable for digital signatures. Among these numerous public-key cryptography algorithms, only four algorithms, **RSA (1978) and ElGamal (1985), Schnorr (1990) and ECC (1985)** are considered to be suitable for both encryption and digital signatures. Another public-key algorithm that is **designed to only be suitable for secure digital signatures is DSA (1991).** The designer should bear in mind that the security of any encryption scheme depends on the length of the key and the computational work involved in breaking a cipher.

### Diffie–Hellman Exponential Key Exchange

In 1976, Diffie and Hellman proposed a scheme using the exponentiation modulo q (a prime) as a public key exchange algorithm. Exponential key exchange takes advantage of easy computation of exponentials in a finite field GF(q) with a prime q compared with the difficulty of computing logarithms over GF(q) with q elements {1, 2,,q − 1}.Let q be a prime number and **(α) a primitive element of the prime number q**. Then the powers of α generate all the distinct integers from 1 to q − 1 in some order. For any integer Y and a primitive element α of prime number q, a unique exponent X is found such that $\mathbf{Y \equiv \alpha^X \pmod q}$, **1 <= X <= q − 1**

Then X is referred to as the discrete logarithm of Y to the base α over GF(q):

$\mathbf{X = \log \alpha^Y}$ over

GF(q), 1 <= Y <= q – 1

Calculation of Y from X is comparatively easy, using repeated squaring, but computation of X from Y is typically far more difficult. Suppose the user i chooses a random integer Xi and the user j a random integer Xj .

Then the user i picks a random number $X_i$ from the integer set {1, 2,,q − 1}.The user i keeps Xi secret, but sends $\mathbf{Y_i \equiv \alpha^{Xi} \pmod q}$ to the user j. Similarly, the user j chooses a random integer $X_j$ and sends $\mathbf{Y_j \equiv \alpha^{Xj} \pmod q}$ to the user i.

Both users i and j can now compute:

$K_{ij} \equiv \alpha^{XiXj} \pmod q$ and use $K_{ij}$ as their common key.

The user i computes $K_{ij}$ by raising $Y_j$ to the power $X_i$ :

$K_{ij} \equiv Y_j^{Xi} \pmod q$

$\equiv (\alpha^{Xj})^{Xi} \pmod q \equiv \alpha^{XjXi}$

$\equiv \alpha^{XiXj} \pmod q$

and the user j computes $K_{ij}$ in a similar fashion:

$K_{ij} \equiv Y_i^{Xj} \pmod q$

$\equiv (\alpha^{Xi})^{Xj}$

$\equiv \alpha^{XiXj} \pmod q$

Thus, both users i and j have exchanged a secret key. Since $X_i$ and $X_j$ are private, the only available factors are the public values **q, α, $Y_i$ and $Y_j$**. Therefore, the opponent is forced to compute a discrete logarithm which is considered to be unrealistic, particularly for large primes. In the following Figure illustrates the Diffie–Hellman key exchange scheme.

When utilizing finite field GF(q), where q is either a prime or $q = 2^k$, it is necessary to ensure the q − 1 factor has a large prime, otherwise it is easy to find discrete logarithms in GF(q).

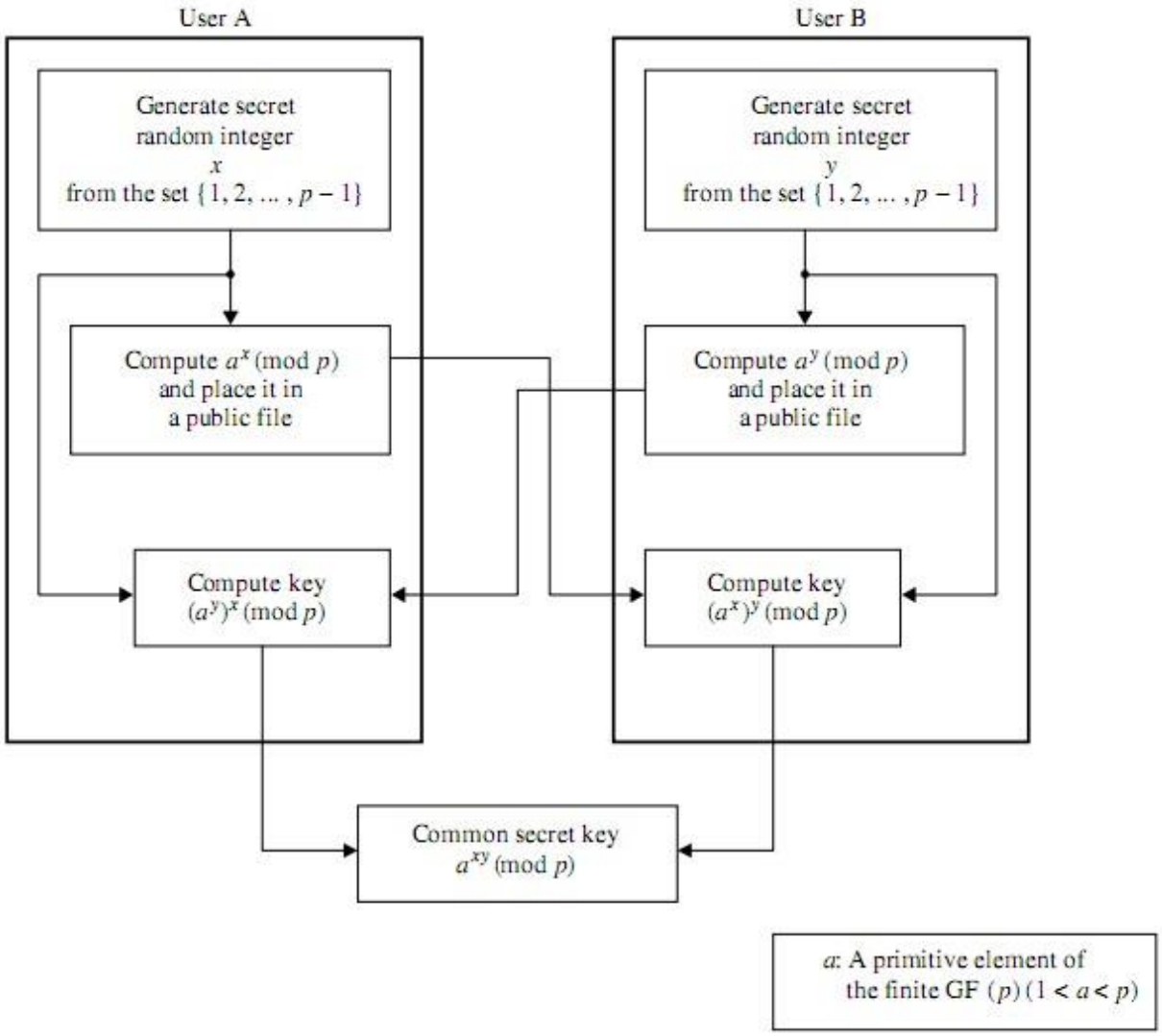



Figure: The Diffie–Hellman exponential key exchange scheme.

**Example 1:**

Consider a prime field Zq where q is a prime modulus. If **α** is a primitive root of the modulus q, then α generates the set of nonzero integer modulo q such that $\alpha, \alpha^2, \ldots, \alpha^{q-1}$. These powers of α are all distinct and are all relatively prime to q. Given α, 1 <= α <= q − 1, and **q = 11**, all the primitive elements of q are computed as shown in the following Table.

For the modulus **q = 11**, the primitive elements are **α = 2,3,4,5,6,7,8 and 9** whose order is **10**, respectively.

Table: Powers of primitive element α (over $Z_{11}$)

| $\alpha$ | $\alpha^2$ | $\alpha^3$ | $\alpha^4$ | $\alpha^5$ | $\alpha^6$ | $\alpha^7$ | $\alpha^8$ | $\alpha^9$ | $\alpha^{10}$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 4 | 8 | 5 | 10 | 9 | 7 | 3 | 6 | 1 |
| 3 | 9 | 5 | 4 | 1 | 3 | 9 | 5 | 4 | 1 |
| 4 | 5 | 9 | 3 | 1 | 4 | 5 | 9 | 3 | 1 |
| 5 | 3 | 4 | 9 | 1 | 5 | 3 | 4 | 9 | 1 |
| 6 | 3 | 7 | 9 | 10 | 5 | 8 | 4 | 2 | 1 |
| 7 | 5 | 2 | 3 | 10 | 4 | 6 | 9 | 8 | 1 |
| 8 | 9 | 6 | 4 | 10 | 3 | 2 | 5 | 7 | 1 |
| 9 | 4 | 3 | 5 | 1 | 9 | 4 | 3 | 5 | 1 |
| 10 | 1 | 10 | 1 | 10 | 1 | 10 | 1 | 10 | 1 |

**Example 2:** Consider a finite field GF(q) of a prime q. Choose a primitive element **α = 2** of the modulus **q = 11**.

Sol:

Compute:

$2^{\lambda}$ ($1 <= \lambda <= 10$): $2^1$ $2^2$ $2^3$ $2^4$ $2^5$ $2^6$ $2^7$ $2^8$ $2^9$ $2^{10}$

$2^{\lambda}$ (mod 11): 2 4 8 5 10 9 7 3 6 1

To initiate communication, the user **i** chooses $\mathbf{X_i = 5}$ randomly from the integer set $2^{\lambda}$ (mod 11) = {1, 2,..., 10} and keep it secret.

The user i sends

$Y_i \equiv \boldsymbol{\alpha}^{X_i} \pmod q$

$\equiv 2^5 \pmod{11} \equiv 10$

to the user j. Similarly,

the user j chooses a random number $X_j = 7$ and sends

$Yj \equiv \alpha^{Xj} \pmod q$

$\equiv 2^7 \pmod{11} \equiv 7$ to

the user i.

Finally, compute their common key $K_{ij}$ as follows:

$K_{ij} \equiv Y_j^{Xi} \pmod q$

$\equiv 7^5 \pmod{11}$

$\equiv \mathbf{10}$

**AND**

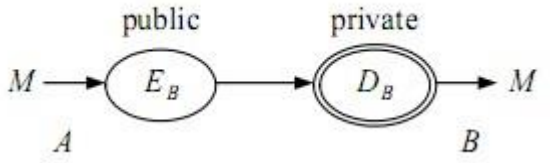
$K_{ji} \equiv Y_i^{Xj} \pmod q$

$\equiv 10^7 \pmod{11}$

$\equiv \mathbf{10}$

Thus, each user computes the common key.

## Lecture 3:



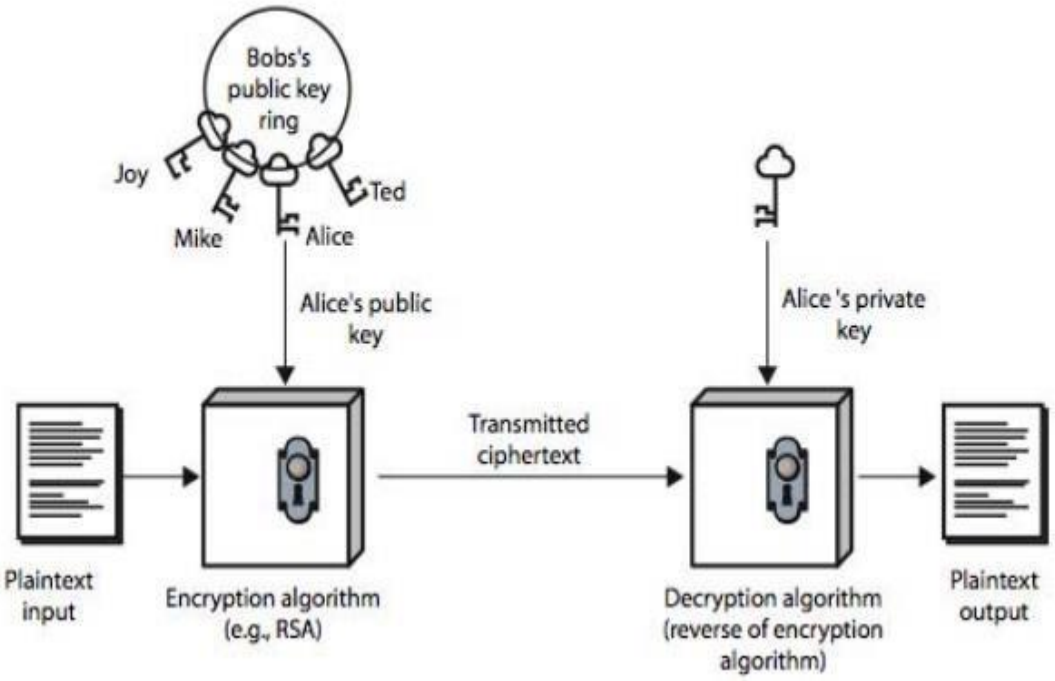

# RSA

The RSA cryptosystem, named after its inventors R. Rivest, A. Shamir, and L. Adleman, is the most widely used public-key cryptosystem. It may be used to provide both secrecy and digital signatures and its security is based on the intractability of the fact that factorizing very large numbers is a `hard' problem. This problem is the Integer Factorization Problem discussed in lecture 1.

It was described to the public in August 1977, and was the invention of Ronald Rivest, Adi Shamir, and Leonard Adleman. However, it was recently revealed that this method was originally invented in 1973 within GCHQ by Clifford Cocks.

The following Algorithms describe the setup process required by each user upon initialization of the cryptosystem and the algorithms for encryption and decryption. In a group of users, each user must have access to each other user's public key, while retaining their own private key.

When each person in the set of users has completed the initialization in Algorithm.




**Algorithm :(*RSA Initialization*)**

***INPUT**: A way to generate or select large random prime numbers.*

***OUTPUT**: A public key, (n, e), and a private key, d.*

***1.** Select or generate two large random prime numbers, p and q.*

***2.** Compute $n = p*q$ and $Ø = (p - 1) *(q -1)$.*

***3.** Select a random integer e, $1 < e < Ø$, such that $\gcd(e, Ø) = 1$.*

***4.** Using the extended Euclidean Algorithm, find the unique integer d; $1 < d < Ø$, such that $e*d \equiv 1 (mod\ Ø)$.*

***5.** Publish the public key, (n, e), and keep the private key, d, secret.*

**Algorithm: (*RSA Encryption*).**

***INPUT:*** *The plaintext to encrypt, and the receiving user's public key* (*n, e*).
***OUTPUT:*** *The encrypted ciphertext.*
*User **A** sends the message to user **B**.*

***1.*** *Using an agreed hash function, convert the plaintext into a unique integer **m** in the interval* [0, ***n*** **-** 1]
***2.*** *compute* $c = m^e \pmod n$ *and send **c** to user **B.***

**Algorithm: (*RSA Decryption*).**

***INPUT:*** *The received encrypted ciphertext and the receiver's private key **d.***
***OUTPUT:*** *The original plaintext. User **B** receives the message from user **A**.*

***1.*** *Use the private key to compute* $m = c^d \bmod n$
***2.*** *Recover the plaintext by applying the inverse of the hash function from above Algorithm, returning the integer in the interval* [0*;* ***n*** **-** 1] *to the unique message it represents.*

***Note****: encryption proof , however, using some simple algebra, we see that*

$c^d \equiv (m^e)^d \equiv m^{ed} \equiv m^{1+k\emptyset} \equiv m \pmod n;$

*returning back to the original message m.*

***Note:*** *must gcd(e,Øn)=1.*

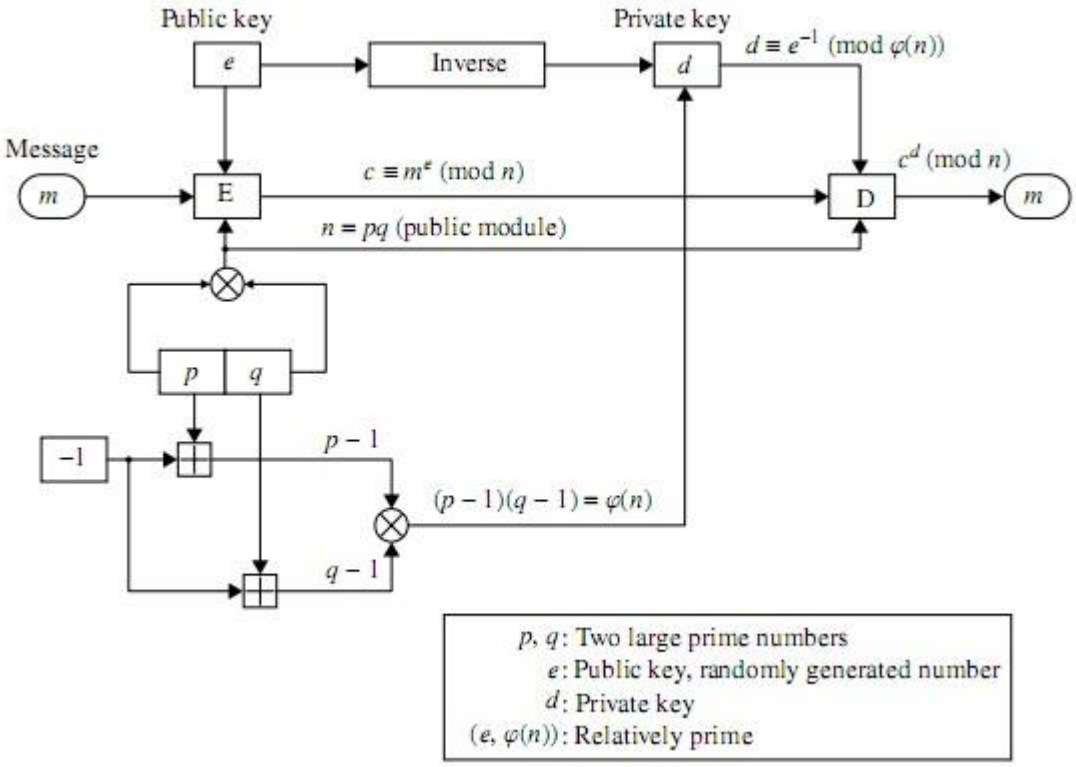



Figure: RSA public-key cryptosystem for encryption/decryption.

**Example 1:** (RSA encryption with artificially small parameters)

- **Key generation**.

Entity Alice chooses the primes p = 2357,

q = 2551, output n =p*q =

6012707

and

Ø = (p-1) *(q-1) = 6007800.

A chooses e = 3674911 (public key) and,

finds d = 422191 such that e*d ≡ 1 (modØ).

Alice public key is the pair (n = 6012707, e = 3674911), while A's private key is d = 422191.

- **Encryption**. To encrypt a message m = 5234673, Bob uses an algorithm for modular exponentiation to compute: $c = m^{e} \bmod n = 5234673^{3674911} \bmod 6012707 = 3650502$;

and sends this to Alice.

- **Decryption**. To decrypt c, Alice computes

$$M = c^{d} \bmod n = 3650502^{422191} \bmod 6012707 = 5234673.$$

**Example 2:**

- **Key generation**.

Entity Alice chooses the primes p =

47, q = 71, output n =p*q =

3337

and

Ø = (p-1) *(q-1) = 46 * 70 = 3220.

A chooses e = 79 (public key) and,

finds d =1019 ,such that e*d ≡ 1 (mod Ø).

Alice public key is the pair (n =3337, e = 79), while Alice private key is d = 1019.

**Encryption**. To encrypt a message m = 6882326879666683, Bob uses an algorithm for modular exponentiation to compute: first break it into small blocks. Three-digit blocks work nicely in this case. The message is split into six blocks, $m_i$, in which

$m_1 = 688$ $m_2$

$= 232$ $m_3 =$

687 $m_4 =$

966 $m_5 =$

668 $m_6$ =

003

The first block is encrypted as

$688^{79} \bmod 3337 = 1570 = c_1$

Performing the same operation on the subsequent blocks generates an encrypted message:

$c$ = 1570 2756 2091 2276 2423 158

- **Decryption**. To decrypt c, Alice computes

Decrypting the message requires performing the same exponentiation using the decryption key of 1019, so

$1570^{1019} \bmod 3337 = 688 = m_1$

The rest of the message can be recovered in this manner.

**RSA Signature Scheme**

The RSA public-key cryptosystem can be used for both encryption and signatures. Each user has three integers e, d and n, n = pq with p and q large primes. For the key pair (e, d), $ed \equiv 1 \pmod{\varphi(n)}$ must be satisfied. If sender **A** wants to send signed message c corresponding to message m to receiver **B**, A signs it using A's private key, computing $c \equiv m^{d}_{A} \pmod{n_A}$.

First A computes

$\phi(n_A) \equiv lcm\ (p_A - 1, q_A - 1)$ where **lcm** stands for the least common multiple. The sender **A** selects his own key pair $(e_A, d_A)$ such that $e_A \cdot d_A \equiv 1 \pmod{\phi(n_A)}$

The modulus $n_A$ and the public key $e_A$ are published. The following Figure illustrates the RSA signature scheme.

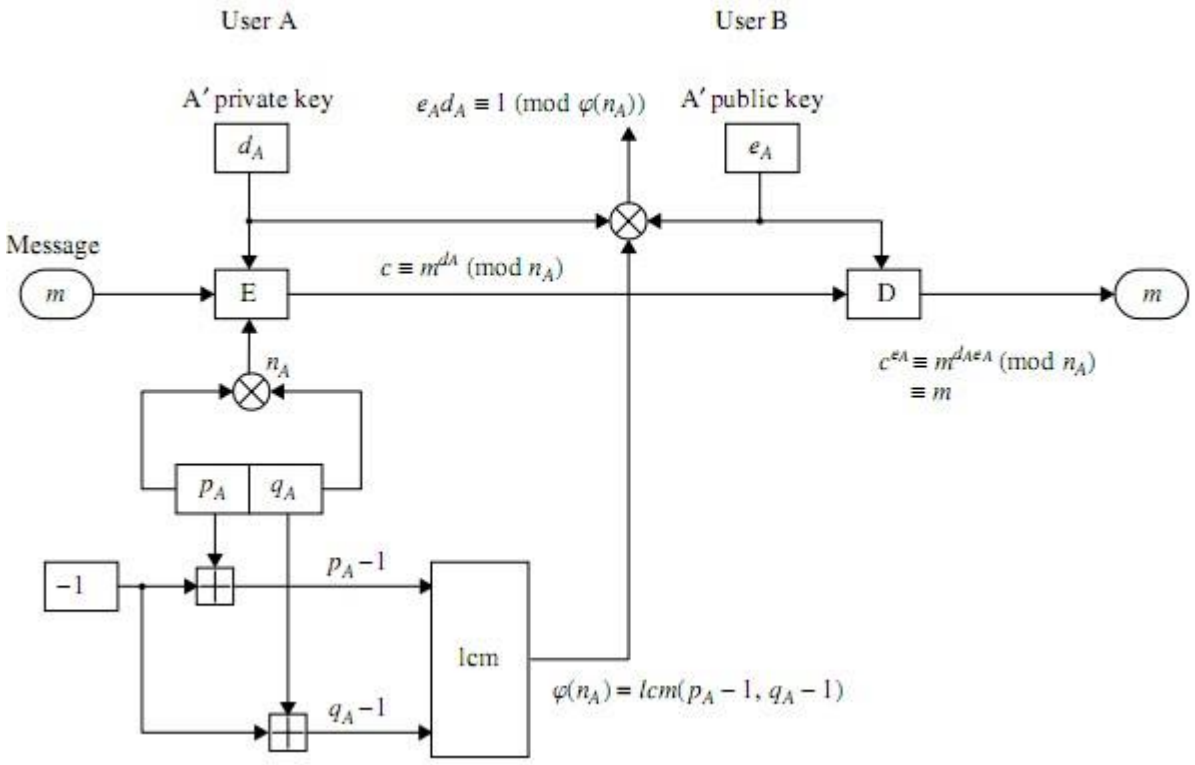



Figure: The RSA signature scheme**.**

**Example** 1:

Choose p = 11 and q = 17. Then n = p*q = 187.

Compute ϕ(n) = 1cm (p − 1, q − 1)

= 1cm (10, 16) = 80

Select $e_A = 27$. Then $e_A d_A \equiv 1 \pmod{\Phi(nA)}$

$27d_A \equiv 1 \pmod{80}$

$d_A = 3$

++6+++++++6+++6

Suppose **m = 55**. Then the signed message is c

$\equiv m^{dA} \pmod{187}$

$\equiv 55^3 \pmod{187} \equiv 132$ The

message will be recreated as:

$m \equiv c^{eA} \pmod{n}$

$\equiv 132^{27} \pmod{187} \equiv$ **55**

Thus, the message m is accepted as authentic.

**Security of RSA**

This subsection discusses various security issues related to RSA encryption. Various attacks that have been studied in the literature are presented, as well as appropriate measures to counteract these threats.

- **Common Modulus Attack on RSA**

A possible RSA implementation gives everyone the same n, but different values for the exponents e and d. Unfortunately, this doesn't work. The most obvious problem is that if the same message is ever encrypted with two different exponents (both having the same modulus), and those two exponents are relatively prime (which they generally would be), then the plaintext can be recovered without either of the decryption exponents.

Let m be the plaintext message. The two encryption keys are $e_1$ and $e_2$. The common modulus is n. The two ciphertext messages are:

$$\mathbf{c_1 = m^{e1} \bmod n}$$
$$\mathbf{c_2 = m^{e2} \bmod n}$$

The cryptanalyst knows n, $e_1$, $e_2$, $c_1$, and $c_2$. Here's how he recovers m.

Since $e_1$ and $e_2$ are relatively prime, the extended **Euclidean algorithm** can find r and s, such that

$$\mathbf{re_1 + se_2 = 1}$$

Assuming r is negative (either r or s has to be, so just call the negative one r), then the extended Euclidean algorithm can be used again to calculate $c_1^{-1}$. Then

$$\mathbf{(c_1^{-1})^{-r} * C_2^{s} = m \bmod n}$$

There are two others, more subtle, attacks against this type of system. One attack uses a probabilistic method for factoring n. The other uses a deterministic algorithm for calculating someone's secret key without factoring the modulus.

- **Multiplicative properties**

Let *m1* and *m2* be two plaintext messages, and let *c1* and *c2* be their respective *RSA* encryptions. Observe that:

$$(m1\ m2)^e \equiv m1^e\ m2^e \equiv c1\ c2 \pmod n$$

In other words, the ciphertext corresponding to the plaintext $m = m1\ m2$ mod n is $c = c1\ c2$ mod n; this is sometimes referred to as the *homomorphic property* of RSA. This observation leads to the following *adaptive chosen-ciphertext attack* on RSA encryption.

Suppose that an active adversary wishes to decrypt a particular ciphertext $c = m^e \pmod n$ intended for *A*. Suppose also that *A* will decrypt arbitrary ciphertext for the adversary, other than *c* itself. The adversary can conceal *c* by selecting a random integer $x \in z_n^*$ and computing $c` = c\ x^e$ mod n. Upon presentation of *c`*, *A* will compute for the adversary: $m` = (c`)^d \mod n$ .

Since

$$m` \equiv (c`)^d \equiv c^d (x^e)^d \equiv m\ x \pmod n$$

The adversary can then compute $m = m`\ x^{-1} \mod n$.

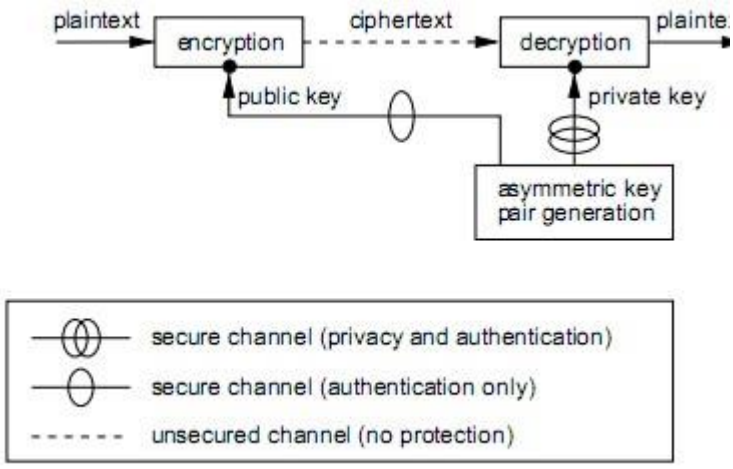
This adaptive *chosen-ciphertext* attack should be circumvented in practice by imposing some structural constraints on plaintext messages. If a ciphertext *c* is decrypted to a message not possessing this structure, then *c* is rejected by the decryptor as being fraudulent. Now, if a plaintext message m has this (carefully chosen) structure, then with high probability *mx mod n* will not for $x \in z_n^*$ . Thus, the adaptive *chosen ciphertext* attack described in the previous paragraph will fail because *A* will not decrypt *c* for the adversary.

## Lecture 5:

# ElGamal Public-Key Encryption

ElGamal proposed a public-key cryptosystem in 1985. The ElGamal algorithm can be used for both encryption and digital signatures. The security of the ElGamal scheme relies on the difficulty of computing discrete logarithms over GF(p)where p is a large prime. Prime factorisation and discrete logarithms are required to implement the RSA and ElGamal cryptosystems.

The ElGamal system is a public-key cryptosystem based on the ***discrete logarithm problem***(DLP). It consists of both encryption and signature algorithms. The encryption algorithm is similar in nature to the *Diffie-Hellman* key agreement protocol.




**Algorithm:** ***(ELGamal Key generation)***

SUMMARY: each entity creates a public key and a corresponding private key.

Each entity *A* should do the following:

1. *Generate a large random prime(p) and a generator (**α**) of the multiplicative group* $z_p^*$ *of the integers modulo p .*
2. *Select a random integer* $a$, $1 \leq a \leq p-2$, *and compute* $\alpha^a \bmod p$ .
3. *A's public key is* $(p, \alpha, \alpha^a)$; *A's private key is* $a$.(Alice)

**Algorithm:** ***(ELGamal Encryption and Decryption )***

SUMMARY: B encrypts a message m for *A*, which *A* decrypts. (Alice and Bob)

1. **Encryption**. B should do the following:

   (a) *Obtain A's authentic public key* $(p, \alpha, a^a)$;

   (b) *Represent the message as an integer m in the range [0,1,..., p -1].*

   (c) *Select a random integer* $k$, $1 \leq k \leq p-2$.

   (d) *Compute* $\gamma = \alpha^k \bmod p$ *and* $\delta = m \cdot (\alpha^a)^k \bmod p$.

   (e) *Send the ciphertext* $c = (\gamma, \delta)$ *to A.*

2. **Decryption**. To recover plaintext m from c, A should do the following:

   (a) *Use the private key a to compute* $\gamma^{p-1-a} \bmod p$ (note: $\gamma^{p-1-a} = \gamma^{-a} = \alpha^{-ak}$)

   (b) *Recover m by computing* $(\gamma^{-a}) \cdot \delta \bmod p$.

Proof that decryption works. The decryption of Algorithm3.5 allows recovery of original plaintext because

$$\gamma^{-a} \cdot \delta \equiv \alpha^{-ak}\, m \cdot \alpha^{a\,k} \equiv m \pmod p$$

**Example 1**: (*ElGamal encryption with artificially small parameters*)

- **Key generation:**

Entity A selects the

prime **p = 2357**

and a generator

**α = 2 of $z_{2357}$**

*A* chooses the private key a **a = 1751** and

computes

$\alpha^{a} \bmod p = 2^{1751} \bmod 2357 = 1185$

**A's public key is (p = 2357; α = 2; $\alpha^{a} = 1185$ ).**

- **Encryption:** To encrypt a message *m = 2035*

*B* selects a random integer **k = 1520** and
Computes:

$\gamma = 2^{1520} \bmod 2357 = 1430$ *and*

$\delta = 2035 \cdot 1185^{1520} \bmod 2357 = 697$

B sends **γ= 1430** and **δ = 697** to A.

- **Decryption***: To decrypt, A computes*

$\gamma^{p-1-a} = 1430^{605} \bmod 2357 = 872$

*and recovers m by computing*

**m = 872. 697 mod 2357 = 2035.**

**Security of ElGamal Encryption**

**(i)** The problem of breaking the ElGamal encryption scheme, i.e., recovering *m* given *(p, α, $a^a$)*,and *(γ)* is equivalent to solving the *Diffie-Hellman*. In fact, the *ElGamal encryption scheme* can be viewed as simply comprising a Diffie-Hellman key exchange to determine a session key $\alpha^{ak}$, and then encrypting the message by multiplication with that session key. For this reason, the security of the *ElGamal encryption scheme* is said to be based on the *discrete logarithm problem* in $z_p^*$ ,although such an equivalence has not been proven.

**(ii)** It is critical that different random integers k be used to encrypt different messages.
Suppose the same *k* is used to encrypt two messages *m1* and *m2* and the resulting cipher text pairs are *( δ1 , γ1)* and *( δ2 , γ2).*Then *δ1/δ2= m1/m2* ,and *m2* could be easily computed if *m1* were known.

- **ElGamal Signatures**

The ElGamal scheme can be used for both digital signatures and encryption; it gets its security from the difficulty of calculating discrete logarithms in a finite field To generate a key pair, first choose a prime, *p*, and two random numbers, *g* and *x*, such that both *g* and *x* are less than *p*. Then calculate $\boldsymbol{y = g^x \bmod p}$

The public key is ***y, g,*** and ***p.*** Both ***g*** and ***p*** can be shared among a group of users. The private key is ***x.***

**Working**: To sign a message, *M*, first choose a random number, ***k,*** such that ***k*** is relatively prime to ***p* - 1**. Then compute $\boldsymbol{a = g^k \bmod p}$ **and** use the extended Euclidean algorithm to solve for ***b*** in the following equation:

$$\boldsymbol{M = (xa + kb) \bmod (p - 1)}$$

The signature is the pair: ***a*** and ***b.*** The random value, ***k,*** must be kept secret.

To verify a signature, confirm that $y^a a^b$

$$\bmod p = g^M \bmod p$$

Each ElGamal signature or encryption requires a new value of *k*, and that value must be chosen randomly. If Eve ever recovers a ***k*** that Alice used, she can recover Alice's private key, ***x***. If Eve ever gets two messages signed or encrypted using the same *k*, even if she doesn't know what it is, she can recover ***x.***

***Public Key:***

p prime (can be shared among a group of users)

g <p (can be shared among a group of users) y

$= g^x \bmod p$

Private Key:

x <p

Signing:

k choose at random, relatively prime to p – 1 i.e. (k,p-1) a

(signature) $= g^k \bmod p$

b (signature) such that $M = (xa + kb) \bmod (p - 1)$

Verifying:

Accept as valid if $y^a a^b \bmod p = g^m \bmod p$ i.e. $y^a = g^{xa}$ and $a^b = g^{kb}$ then m=(xa+kb)

$=g^m$

**Example 1:** choose $p = 11$ and $g = 2$. Choose private key $x = 8$. Calculate $y$

$$= g^x \bmod p = 2^8 \bmod 11 = 3$$

The public key is $y = 3$, $g = 2$, and $p = 11$, private key **x**.

To authenticate $M = 5$, first choose a random number $k = 9$. Confirm that gcd(9, 10) = 1.

Compute $a = g^k \bmod p = 2^9 \bmod 11 = 6$ and use the

extended **Euclidean** algorithm to solve for ***b:***

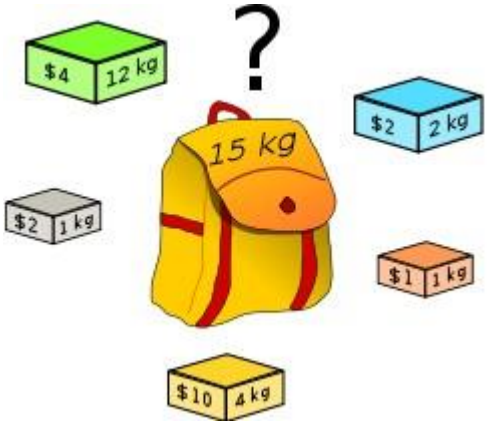
$$M = (ax + kb) \bmod (p - 1)$$
$$5 = (8 * 6 + 9 * b) \bmod 10$$

The solution is $b = 3$, and the **signature is the pair:** $a = 6$ **and** $b = 3$.

To verify a signature, confirm that:

$$y^a a^b \bmod p = g^M \bmod p$$
$$3^6\ 6^3 \bmod 11 = 2^5 \bmod 11$$

## Lecture 6




# Knapsack public-key encryption

The knapsack problem has been studied for more than a century, with early works dating as far back as 1897. It is not known how the name "knapsack problem" originated, though the problem was referred to as such in the early works of mathematician Tobias Dantzig (1884–1956)), suggesting that the name could have existed in folklore before a mathematical problem had been fully defined.The quadratic knapsack problem was first introduced by Gallo, Hammer, and Simeone in 1960.A 1998 study of the Stony Brook University algorithms repository showed that, out of 75 algorithmic problems, the knapsack problem was the 18th most popular and the 4th most needed after kd-trees, suffix trees, and the bin packing problem.

Knapsack public-key encryption schemes are based on the subset *sum* problem, which is NPcomplete. The basic idea is to select an instance of the subset sum problem that is easy to solve, and then to disguise it as an instance of the general subset sum problem which is hopefully difficult to solve. The original knapsack set can serve as the private key, while the transformed knapsack set serves as the public key.

*The **Merkle-Hellman knapsack*** encryption scheme is important for historical reasons, as it was the first concrete realization of a public-key encryption scheme. Many variations have subsequently been proposed but most, including the original, have been demonstrated to be insecure, a notable exception being the *Chor-Rivest knapsack scheme.*

**Definition 3.1:** A superincreasing sequence is a sequence $(b_1, b_2, ..., b_n)$ of positive integers with the

property that $b_i > \sum_{j=1}^{i-1} bj$ *>for each I ,2≤ I ≤ n.*

*Algorithm 3.8 efficiently solves the subset sum problem for super increasing sequences.*

**Algorithm:** Solving a super increasing subset sum problem

**INPUT**: a super increasing sequence ($b_1$, $b_2$,..., $b_n$) and an integer **S** which is the sum of a subset of the $b_i$ .

**OUTPUT:** ($x_1$ , $x_2$,..., $x_n$) where $x_i \in \{0, 1\}$, such that $\sum_{i=1}^{n} xibi = s$

1. i = n.
2. While i ≥ 1 do the following:
   *2.1 If s ≥ bi then xi = 1 and s = s - bi. Otherwise xi= 0.*
   *2.2 i= i - 1.*
3. Return ($x_1$ , $x_2$,..., $x_n$).

**Algorithm: Key generation for basic Merkle-Hellman knapsack encryption** SUMMARY: each entity creates a public key and a corresponding private key.

**1.** An integer **n** is fixed as a common system parameter.
**2.** Each entity A should perform steps 3 – 7.
**3.** Choose a super increasing sequence ($b_1$ , $b_2$,..., $b_n$) and modulus **M** such that M > $b_1$ + $b_2$ + ... + $b_n$.
**4.** Select a random integer W, **1 ≤ W ≤ M - 1**, such that gcd(W, M) = 1.
**5.** Select a random permutation π of the integers {1, 2,..., n}
**6.** Compute ai= $\mathbf{Wb_{\pi(i)}}$ mod M for i = 1, 2,... , n.
**7.** A's public key is ($a_1$ , $a_2$,..., $a_n$) A's private key is (π, M, W, ( $b_1$ , $b_2$,..., $b_n$)

**Algorithm: Basic Merkle-Hellman knapsack public-key encryption** SUMMARY:

*B* encrypts a message m for *A,* which *A* decrypts.

**1.** Encryption. B should do the following:

*(a) Obtain* A's *authentic public key* ($a_1, a_2, ..., a_n$).

*(b) Represent the message* m *as a binary string of length* n, **m** = $m_1, m_2, ..., m_n$.

*(c) Compute the integer* c = $m_1 a_1 + m_2 a_2 + ..... + m_n a_n$ *(d) Send the ciphertext* c *to* A.

**2.** Decryption. To recover plaintext m from, A should do the following:

*(a) Compute* $d = w^{-1}c \bmod M$.

*(b) By solving a super increasing subset sum problem (Algorithm super increasing), find integers* $r_1, r_2, ....., r_n$, $r_i \in \{0,1\}$, *such that* $d = r_1 b_1 + r_2 b_2 + ..... + r_n b_n$.

*(c) The message bits are* $m_i = r_{\pi(i)} = 1, 2, ....,n$.

**Example 1:**

S= [1,2,4,9]

Choose

- a multiplier **w**, and
- modulus **n** , n should be larger than the largest integer in your knapsack

**Hint**: Choose modulus (n) to be a prime number. Generate the Hard knapsack by

**$h_i = w * s_i \bmod n$**

H= [$h_1, h_2, h_3, .. H_m$]

S= [1,2,4,9]

$h_i = w * s_i \bmod n$

Let $w$=15 Let $n$=17

1*15 = 15 mod 17 = 15

2*15 = 30 mod 17 = 13

4*15 = 60 mod 17 = 9

9*15 = 135 mod 17= 16

**H**=[15,13,9,16] - public key

**Encryption**

P = 0100101110100101

P = 0100 1011 1010 0101

[0,1,0,0]*[15,13,9,16]=13

[1,0,1,1]*[15,13,9,16]=40

[1,0,1,0]*[15,13,9,16]=24

[0,1,0,1]*[15,13,9,16]=29

Encrypted message **C** is 13, 40, 24, 29 with **H** public key

**Decryption**

To decipher multiply $C_i$ by $W^{-1}$ using your secret knapsack

H= [15,13,9,16] S=[1,2,4,9] C=[13,40,24,29]

W=15 $15^{-1} \bmod 17 = 8$

13*8 = 104 mod 17 = 2 = [0100]

40*8 = 320 mod 17 = 14 = [1011]

24*8 = 192 mod 17 = 5 = [1010]

29*8 = 232 mod 17 = 11 = [0101]

**Example 2:**

Let, a super increasing sequence **w** be created **w = {2, 7, 11, 21, 42, 89, 180, 354}** This is the basis for a private key.

From this, calculate the sum. Then, choose a number **q** that is greater than the sum.

**q = 881** Also, choose a number **r** that is in the range [1,q] and is **coprime** to q.

**r = 588**

The private key consists of **q, w and r. To calculate a public key,**

**generate** the sequence **β** by multiplying each element in **w by r mod**

**q β = {295, 592, 301, 14, 28, 353, 120, 236}** because

2 * 588 mod 881 = 295

7 * 588 mod 881 = 592

11 * 588 mod 881 = 301

21 * 588 mod 881 = 14

42 * 588 mod 881 = 28

89 * 588 mod 881 = 353

180 * 588 mod 881 = 120

354 * 588 mod 881 = 236

The sequence β makes up the public key.

Encryption

Say Alice wishes to encrypt "**a**". First, she must translate "a" to binary

"a"=01100001

She multiplies each respective bit by the corresponding number in β a

= 01100001

0 * 295 + 1 * 592 + 1 * 301 + 0 * 14 + 0 * 28 + 0 * 353 + 0 * 120 + 1 * 236 = 1129 She sends this to the recipient.

***To decrypt***,

Bob multiplies **1129** by $\mathbf{r^{-1} \bmod q}$

1129 * 442 mod 881 = 372

Now Bob decomposes 372 by selecting the largest element in **w** which is less than or equal to 372. Then selecting the next largest element less than or equal to the difference, until the difference is 0: 372 - 354 = 18

18 - 11 = 7

7 - 7 = 0 The elements we selected from our private key correspond to the 1 bits in the message **01100001** When translated back from binary, this "a" is the final decrypted message.

## Lecture 7:

# McEliece public-key encryption

In cryptography, the McEliece cryptosystem is an asymmetric encryption algorithm developed in 1978 by Robert McEliece. It was the first such scheme to use randomization in the encryption process. The algorithm has never gained much acceptance in the cryptographic community but is a candidate for 'post-quantum cryptography' as it is immune to attacks using Shor's algorithm and more generally measuring coset states using Fourier sampling. A recent improvement of an information-set decoding algorithm for quantum computing, however, requires key sizes to be increased by a factor of four

The **McEliece** public-key encryption scheme is based on **error-correcting codes**. The idea behind this scheme is to first select a particular code for which an efficient decoding algorithm is known, and then to disguise the code as a general linear code.

Since the problem of decoding an arbitrary linear code is ***NP-hard***, a description of the original code can serve as the private key, while a description of the transformed code serves as the public key.

The **McEliece** encryption scheme (*when used with Goppa codes*) has resisted crypt-analysis to date. It is also notable as being the first public-key encryption scheme to use randomization in the encryption process. Although very efficient, the McEliece encryption scheme has received little attention in practice because of the very large public keys.

- **Algorithm: key generator**

1- Integers **K, N,** and **T** are fixed as common system parameters.

2- Each entity A should

2.1 Choose a **K*N** generator (**G**) for a binary (**N, K**) liner code that can correct (**T**) errors and for which an efficient decoding algorithm is known.

2.2 Select a random **K*K** binary non-singular matrix **S**

2.3 Select a random **N*N** permutation matrix **P**

3- Compute **K*N** matrix **Ğ=SGP**

4- Alice public key is (**Ğ, T**) and private key (**S, G, P**)

- **Algorithm: Encryption**

Encryption **Bob** should do the following

1- Obtain Bob authentic public key (**Ğ**, **T**)
2- Representation the message as binary string **M** of length **K**.
3- Choose a random binary vector (**Z**) of the length (**N**) have at most (**T**)
4- Compute the binary vector C=**M Ğ +Z**
5- Send the cipher text **C** to Alice

**Algorithm: Decryption**

To recover the plain text (**M**) from **C**, Alice should do the following:

1- Compute $\check{C} = CP^{-1}$ , where $P^{-1}$ is the inverse of the matrix P
2- Compute $M^{\wedge} = \check{C}\ [\check{C}_1, \check{C}_2, \ldots \check{C}_n]$
   If $\check{C}_i > 0$ then $M_i^{\wedge} = 1$ else $M_i^{\wedge} = 0$
3- Compute $M = M^{\wedge} S^{-1}$

**Example 1:**

- **Key Generator** assume K=2, N=2

G=1 0
0 1
S=0 1
1 0
P=2 3
1 5

**Ğ**=SGP = 0 1 1 0 2 3
* *
1 0 0 1 1 5 = 1 5
2 3

Then: Public key **Ğ** Private key (**S,G,P**)

- **Encryption**

  Let Plain text **M**=011101 and $M_1$=01 ,$M_2$=11,$M_3$=01

  The length of $M_i$= K=2

  Let Z=1 0

  C=M Ğ+Z [ 0 1 ]* 1 5 +[ 1 0 ]
  2 3

  C=[ 2 3]+[ 1 0]

  C=[3 3]

- **Decryption**

  **Č =CP$^{-1}$**

  **P**=2 3
  1 5

**P$^{-1}$**=adj(P) div /P/

Let /P/= (2*5)-(3*1)

/p/=7

Adj(P)=P$^t$

/P/=5 -3
-1 2

And

**P$^{-1}$** =5/7 -3/7
-1/7 2/7

**Č =CP$^{-1}$ = [3 3] *** 5/7 -3/7
-1/7 2/7 = [12/7 -3/7]

**M**^= Č [$Č_1$, $Č_2$,…. $Č_n$] = [12/7 >0 then 1 and -3/7 <0 =0]

M^=[1 0]

S= 0 1 S$^{-1}$=adj(S) div /S/
1 0

/S/=(0*0)-(1*1)=-1 Adj(S)=

0 -1

-1 0

Adj(S) div /S/=0/-1 -1/-1

-1/-1 0/-1= 0 1

1 0

$S^{-1}$=0 1

1 0

M= M^ $S^{-1}$

M=[1 0] * 0 1

1 0 =[0 1] the plain text

**ملاحظات**

معكوس المصفوفة هو تدوير المصفوفة مقسمة على المحدد

التدوير هو تبديل القطر الرئيسي مع بعضه ووضع أشارة سالب للثانوي

المحدد هو طرح حاصل ضرب القطر الرئيسي من حاصل ضرب الثانو ي