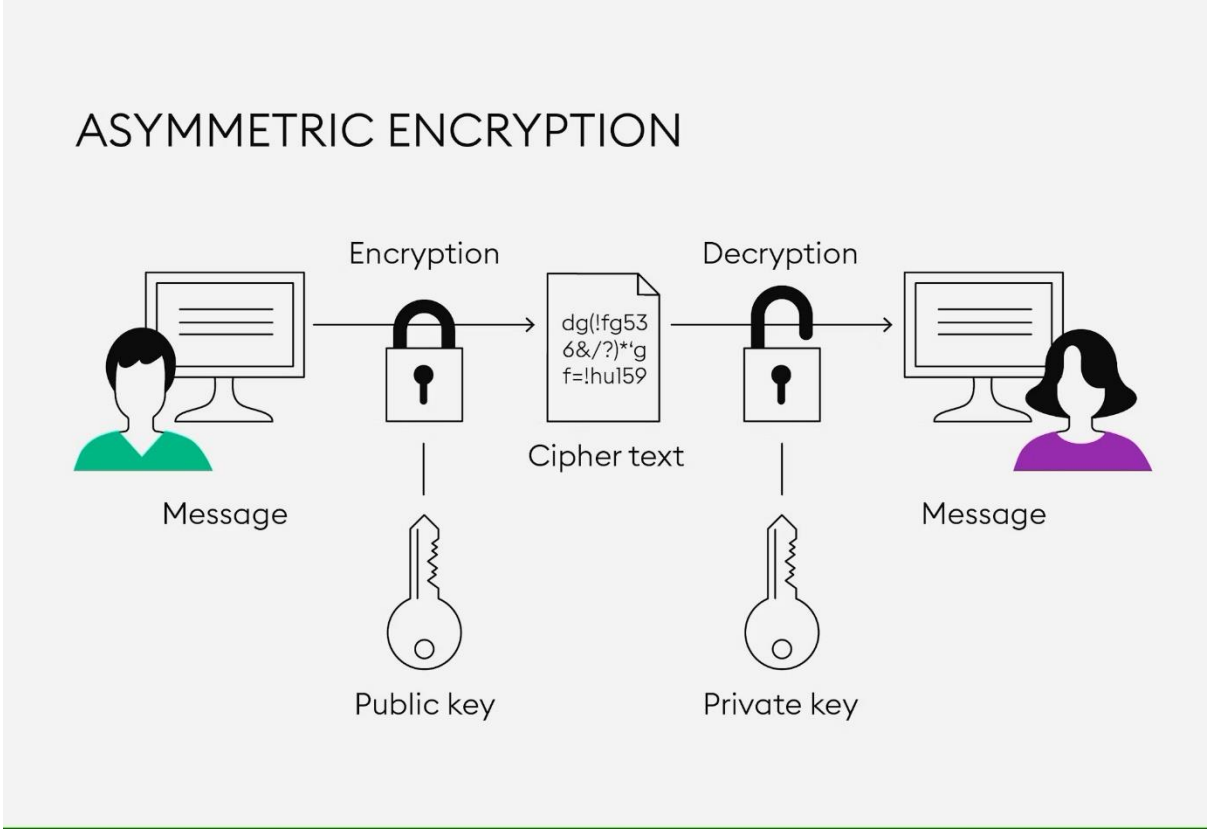
UNIT-IV

Asymmetric encryption: Introduction to public key cryptography, constructions of RSA, ElGamal, Diffie Hell man key exchange, hybrid encryption.

Asymmetric encryption:



Introduction to public key cryptography:

Asymmetric encryption, also known as public-key cryptography, is a type of encryption that uses a pair of keys to encrypt and decrypt data. The pair of keys includes a public key, which can be shared with anyone, and a private key, which is kept secret by the owner. In asymmetric encryption, the sender uses the recipient's public key to encrypt the data. The recipient then uses their private key to decrypt the data. This approach allows for secure communication between two parties without the need for both parties to have the same secret key. Asymmetric encryption has several advantages over symmetric encryption, which uses the same key for both encryption and decryption. One of the main advantages is that it eliminates the need to exchange secret keys, which can be a challenging process, especially when communicating with multiple parties. Additionally, asymmetric encryption allows for the creation of digital signatures, which can be used to verify the authenticity of data. Asymmetric

encryption is commonly used in various applications, including secure online communication, digital signatures, and secure data transfer. Examples of asymmetric encryption algorithms include RSA, Diffie-Hellman, and Elliptic Curve Cryptography (ECC).

Asymmetric encryption, commonly known as public-key cryptography, employs two distinct keys for encryption and decoding. The private key is a separate key from the public key that is kept private by the owner of the public key while the public key is made available to everyone. Anyone can encrypt a message using the public key, but only the holder of the private key can unlock it. With no chance of the communication being intercepted and read by a third party, anyone can send a secure message to the public key's owner. Asymmetric encryption is frequently used for secure internet communication, including email encryption, e-commerce, and online banking. Digital signatures, which are used to confirm the legitimacy of digital documents and messages, are another application for it.

Difference Between Encryption and Public-Key Encryption:

Basis	Encryption	Public-Key Encryption
Required for Work	• Same algorithm with the same key is used for encryption and decryption. • The sender and receiver must share the algorithm and key.	• One algorithm is used for encryption and a related algorithm decryption with pair of keys, one for encryption and other for decryption. • Receiver and Sender must each have one of the matched pair of keys (not identical) .
Required for Security	• Key must be kept secret.	• One of the two keys must be kept secret.

Basis	Encryption	Public-Key Encryption
	• If the key is secret, it is very impossible to decipher message. • Knowledge of the algorithm plus samples of ciphertext must be impractical to determine the key.	• If one of the key is kept secret, it is very impossible to decipher message. • Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be impractical to determine the other key.

Components of Public Key Encryption:

- **Plain Text:** This is the message which is readable or understandable. This message is given to the Encryption algorithm as an input.
- **Cipher Text:** The cipher text is produced as an output of Encryption algorithm. We cannot simply understand this message.
- **Encryption Algorithm:** The encryption algorithm is used to convert plain text into cipher text.
- **Decryption Algorithm:** It accepts the cipher text as input and the matching key (Private Key or Public key) and produces the original plain text
- **Public and Private Key:** One key either Private key (Secret key) or Public Key (known to everyone) is used for encryption and other is used for decryption

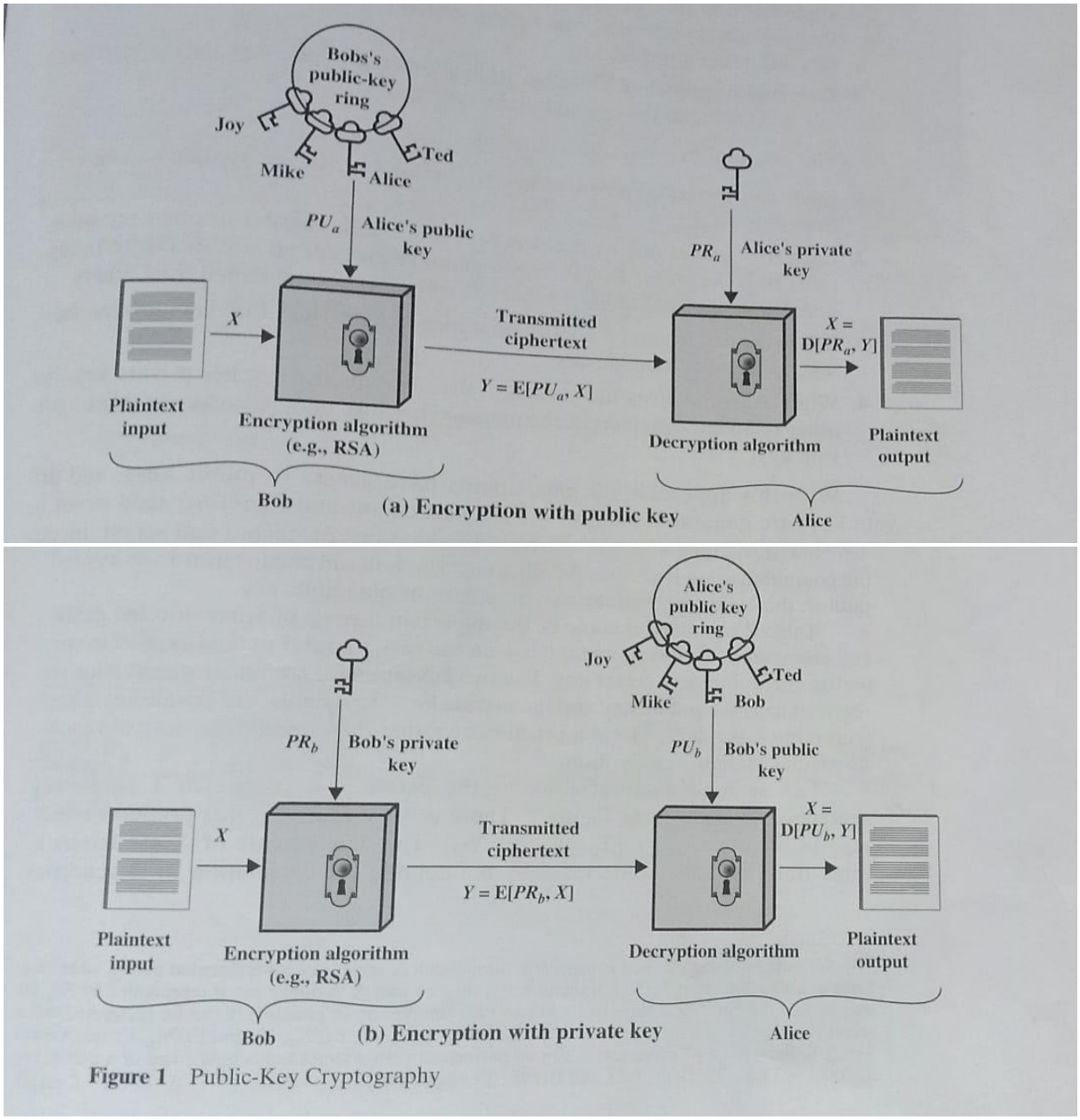


Figure 1 Public-Key Cryptography

Let us take a closer look at the essential elements of a public-key encryption scheme, using Figure 2. There is some source A that produces a message in plaintext, X = [X1, X2, …, XM] 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=[Y1, Y2,..., YN]:

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

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^. 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 PR_b^.

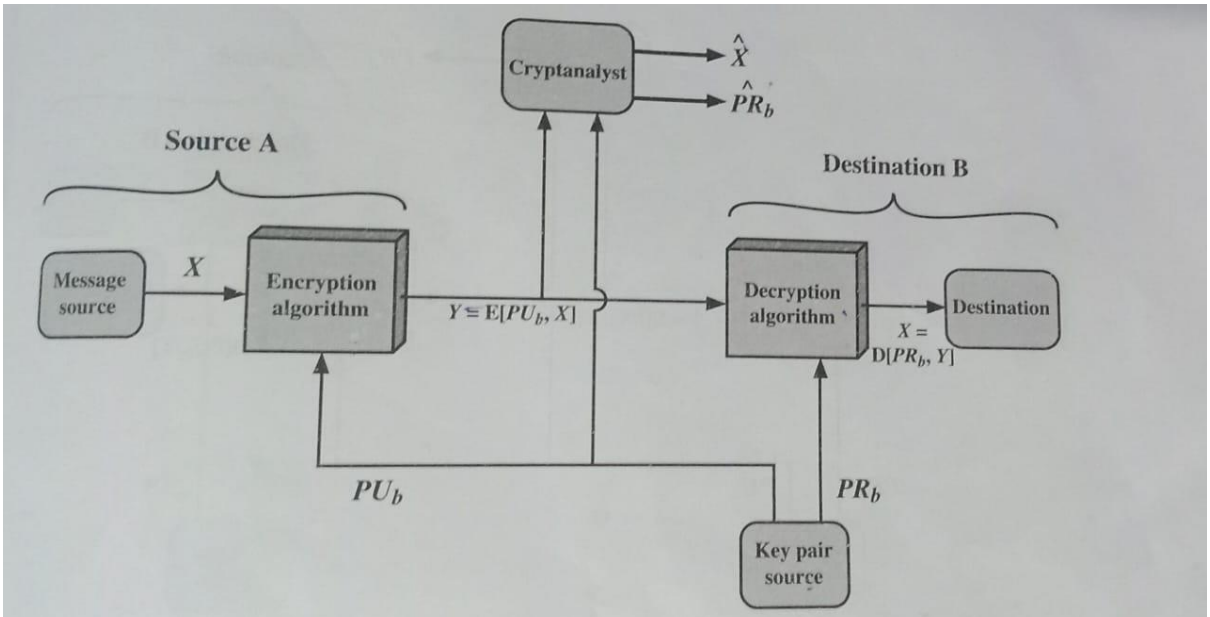


Figure 2 Public-Key Cryptosystem: Secrecy

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 Figure 2 provides confidentiality. Figures 1b and 3 show the use of public-key encryption to provide authentication:

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

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

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 1b and 3 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 3, there is no protection of confidentiality because any observer can decrypt the message by using the sender's public key.

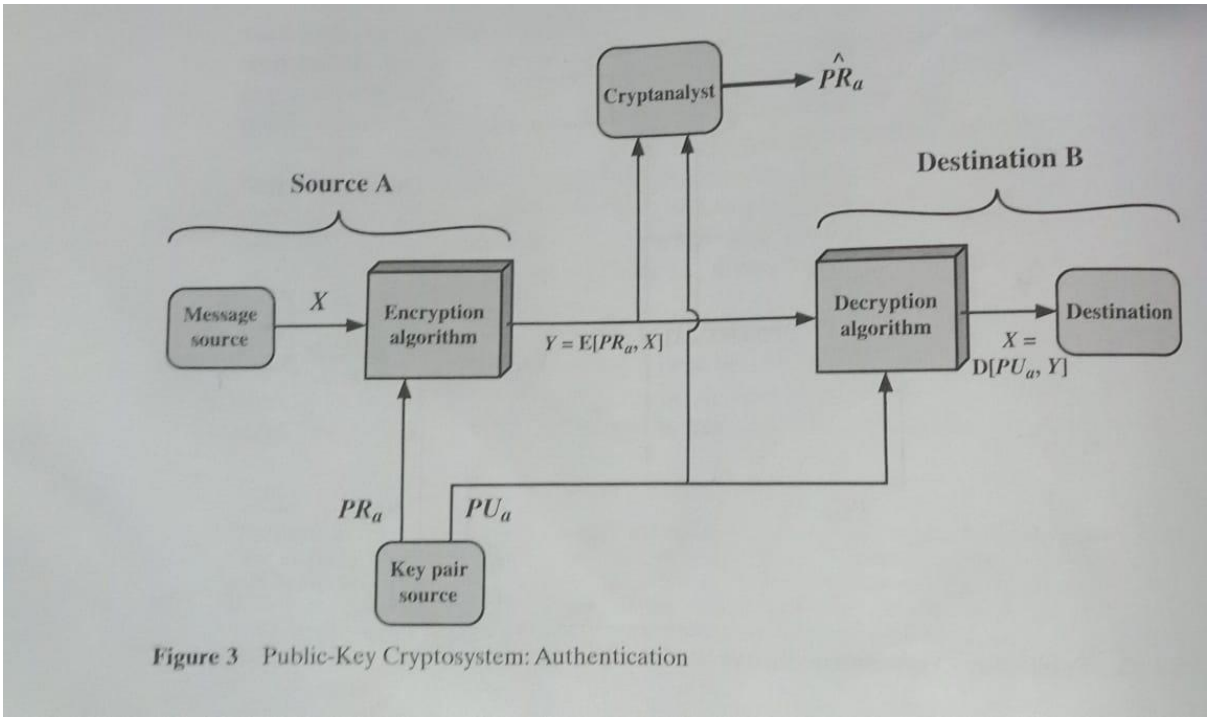


Figure 3 Public-Key Cryptosystem: Authentication

It is, however, possible to provide both the authentication function and confidentiality by a double use of the public-key scheme (Figure 4):

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

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

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.

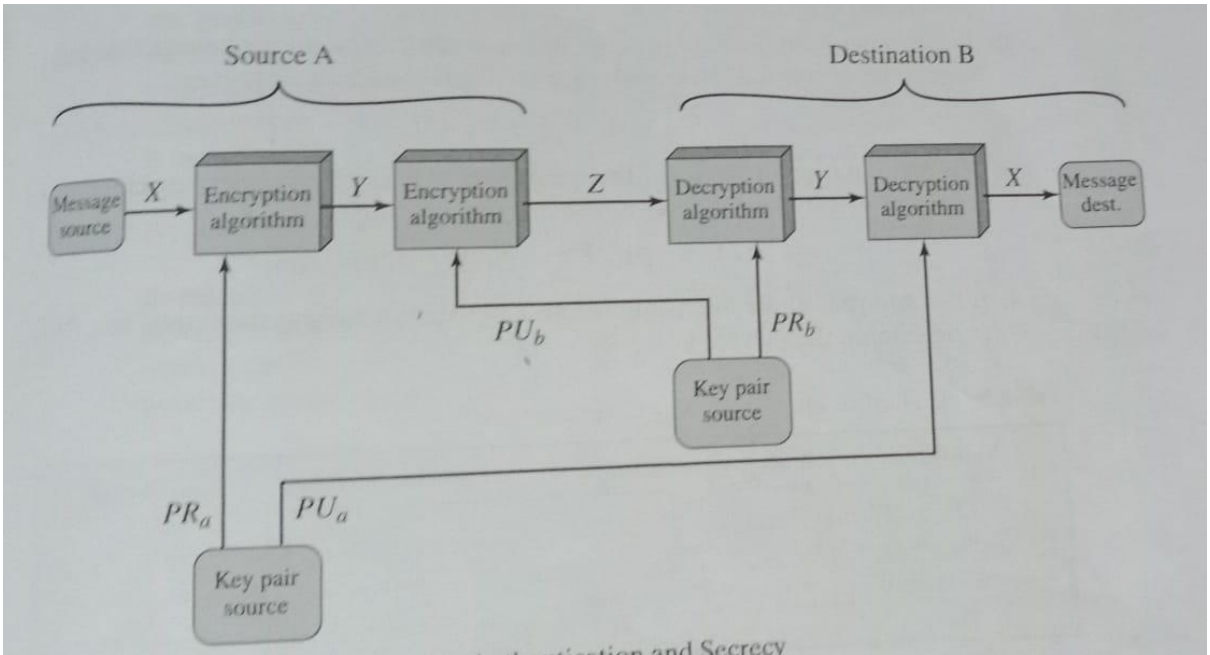


Figure 4 Public-Key Cryptosystem: Authentication and Secrecy

Advantages of asymmetric encryption:

Asymmetric encryption also known as public key cryptography is a method of cryptography that uses two different keys to encrypt and decrypt data, here are some advantages of asymmetric encryption: –

- **Enhanced Security:** Asymmetric encryption provides a higher level of security compared to symmetric encryption where only one key is used for both encryption and decryption with asymmetric encryption a different key is used for each process and the private key used for decryption is kept secret by the receiver making, it harder for an attacker to intercept and decrypt the data.

- **Authentication**: Asymmetric encryption can be used for authentication purposes which means that the receiver can verify the sender s identity. This is achieved by the sender encrypting a message with their private key which can only be decrypted with their public key if the receiver can successfully decrypt the message, it proves that it was sent by the sender who has the corresponding private key.

- **Non-repudiation:** Asymmetric encryption also provides non-repudiation which means that the sender cannot deny sending a message or altering its contents this is because the message is encrypted with the sender s private key and only their public key can decrypt it . Therefore, the receiver can be sure that the message was sent by the sender and has not been tampered with.

- **Key distribution**: Asymmetric encryption eliminates the need for a secure key distribution system that is required in symmetric encryption with symmetric encryption, the same key is used for both encryption and decryption and the key needs to be securely shared between the sender and the receiver asymmetric encryption, on the other hand, allows the public key to be shared openly and the private key is kept secret by the receiver.

- **Versatility:** Asymmetric encryption can be used for a wide range of applications including secure email communication online banking transactions and e-commerce it is also used to secure SSL/TSL connections which are commonly used to secure internet traffic.

Overall, the use of asymmetric encryption offers enhanced security authentication non-repudiation key distribution, and versatility these advantages make it a widely used and effective method for protecting sensitive data in various applications.

The main features of asymmetric encryption (also known as public-key cryptography) are:

1. **Dual keys:** Asymmetric encryption uses a pair of keys, including a public key and a private key. The public key can be freely shared with anyone, while the private key is kept secret and known only to the key owner.

2. **Encryption and decryption:** Asymmetric encryption uses the public key to encrypt data and the private key to decrypt data. This allows secure communication between two parties without the need to exchange secret keys.

3. **Digital signatures:** Asymmetric encryption enables the creation of digital signatures, which can be used to verify the authenticity of data. A digital signature is created by encrypting a hash of the data with the sender's private key.

4. **Secure key exchange:** Asymmetric encryption allows for secure key exchange, which is a critical feature in secure communication. For example, the Diffie-Hellman key exchange algorithm uses asymmetric encryption to establish a shared secret key between two parties without exchanging the key itself.

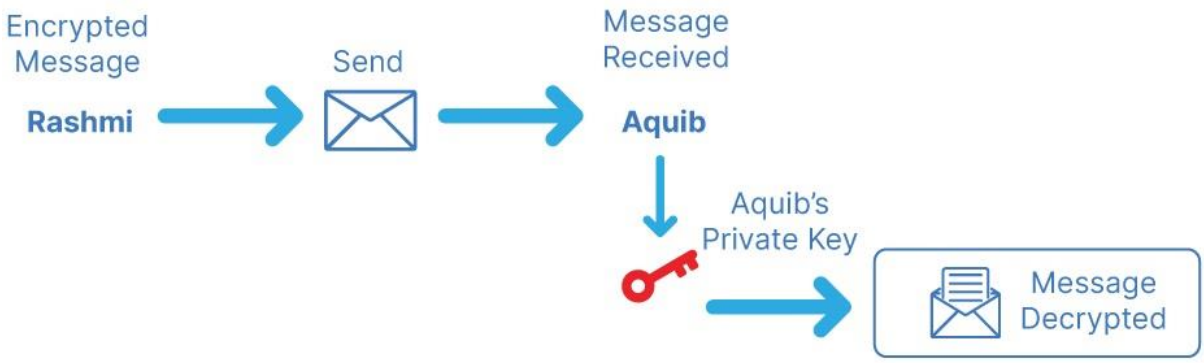
5. **Security:** Asymmetric encryption is considered more secure than symmetric encryption because it eliminates the need to exchange secret keys, which can be a security risk. Additionally, the private key is kept secret, which makes it harder for attackers to intercept or tamper with the data.

6. **Slow processing:** Asymmetric encryption is slower than symmetric encryption because it involves more complex mathematical operations. This can make it less suitable for applications that require fast data processing.

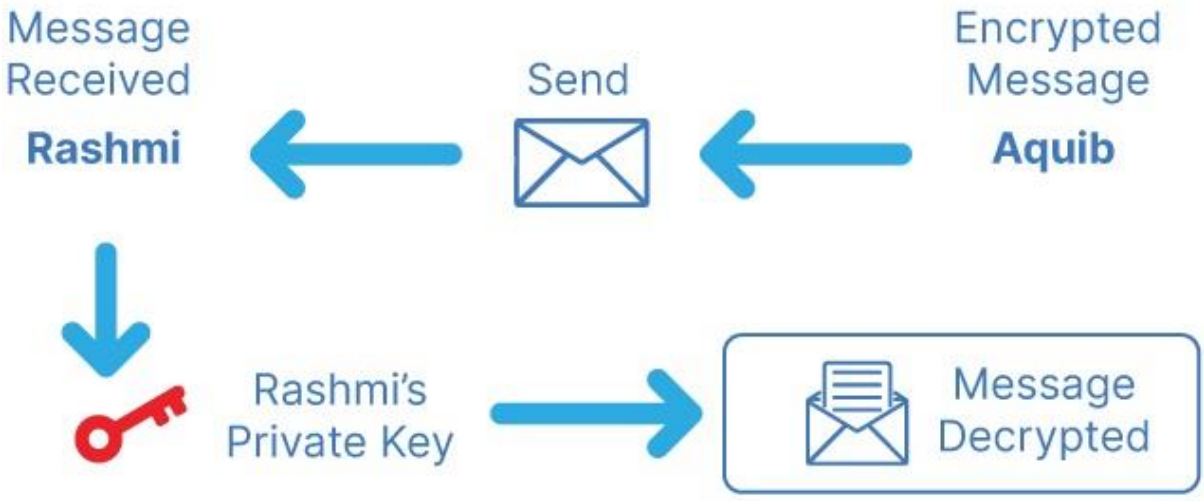
Overall, asymmetric encryption offers several unique features that make it an essential tool for secure communication and data exchange. However, it also has some limitations, such as slower processing speed, which should be considered when choosing an encryption method for a particular application.

Public key cryptography example

An HR representative (Rashmi) from Naukri.com wishes to send Aquib an encrypted email. Rashmi encrypts the message using Aquib's public key, and when Aquib receives it, he uses his private key to decrypt the message from Rashmi.



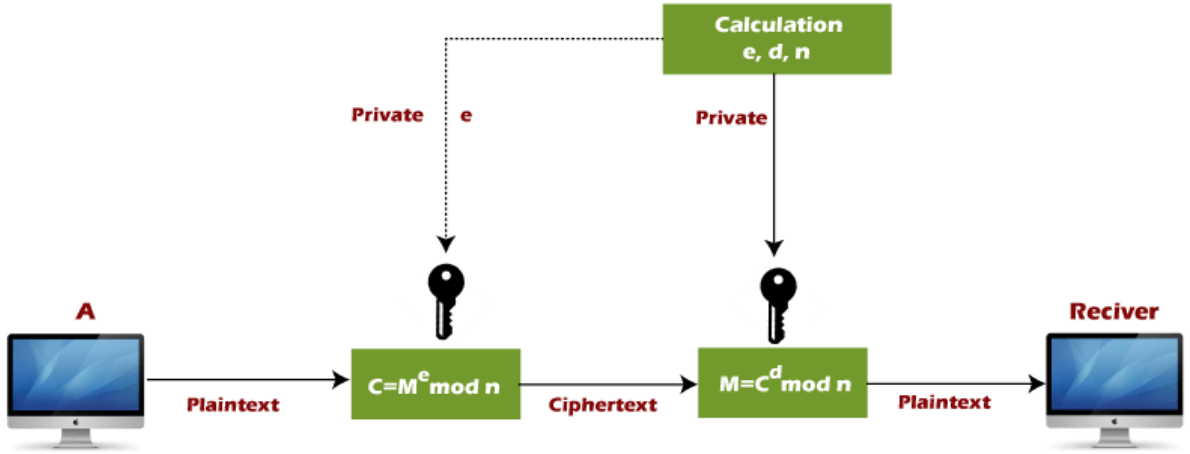
Although attackers may attempt to compromise the server in order to read the message, they will be unable to do so because they lack the private key required to decrypt the message. Because he is the only one with the private key, only Aquib will be able to decrypt the message. Aquib simply repeats the process when he wishes to respond, encrypting her message with Rasmi's public key.



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.

The idea! 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.



RSA

RSA makes use of an expression with exponentials. 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) + 1; in practice, the block size is bits, where 2 ^ i < n <= 2 ^ (i + 1) Encryption and decryption are of the following form, for some plaintext block M and ciphertext block C.

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

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

Both sender and receiver must know the value of n. The sender knows the value of e, and only the receiver knows the value of d. Thus, this is a public key encryption algorithm with a public key of PU= {e, n} and a private key of PR = {d, n}. For this algorithm to be satisfactory for public key encryption, the following requirements must be met.

1. It is possible to find values of e, d, and n such that M^ed mod n = M for all M<n.

2. It is relatively easy to calculate M^e mod n and C^d mod n for all values of M<n.

3. It is infeasible to determine d given e and n.

For now, we focus on the first requirement and consider the other questions later. We need to find a relationship of the form

$$M^{ed} \bmod n = M$$

The preceding relationship holds if e and d are multiplicative inverses modulo φ(n), where φ(n) is the Euler totient function. For p, q prime, φ(pq) = (p -1)(q - 1). The relationship between e and d can be expressed as

$$ed \bmod \varphi(n) = 1 \qquad \ldots(1)$$

This is equivalent to saying

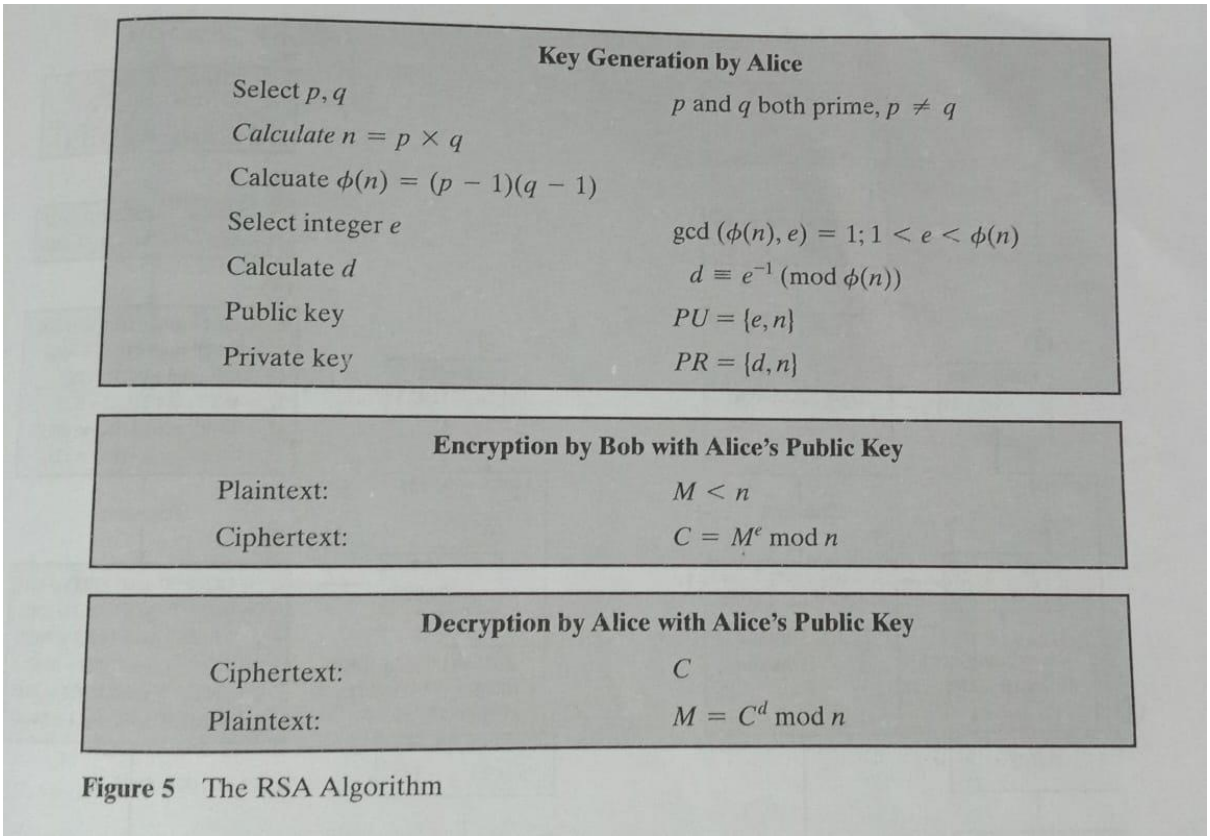
$$ed = 1 \bmod \varphi(n)$$

$$d = e^{-1} \bmod \varphi(n)$$

That is, e and d are multiplicative inverses mod φ(n). Note that, according to the rules of modular arithmetic, this is true only if d (and therefore e) is relatively prime to φ(n). Equivalently, gcd(φ(n), d) = 1. Equation (1) satisfies the requirement for RSA.

We are now ready to state the RSA scheme.

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



Key Generation by Alice

Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calcuate $\phi(n) = (p - 1)(q - 1)$	
Select integer e	$\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \pmod{\phi(n)}$
Public key	$PU = \{e, n\}$
Private key	$PR = \{d, n\}$

Encryption by Bob with Alice's Public Key

Plaintext:	$M < n$
Ciphertext:	$C = M^e \bmod n$

Decryption by Alice with Alice's Public Key

Ciphertext:	C
Plaintext:	$M = C^d \bmod n$

Figure 5 The RSA Algorithm

- 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 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 mod{(p-1)x(q-1)}=1 **Or**
  D_e mod φ (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.
  $m = c^d$ mod n

Example of RSA algorithm:

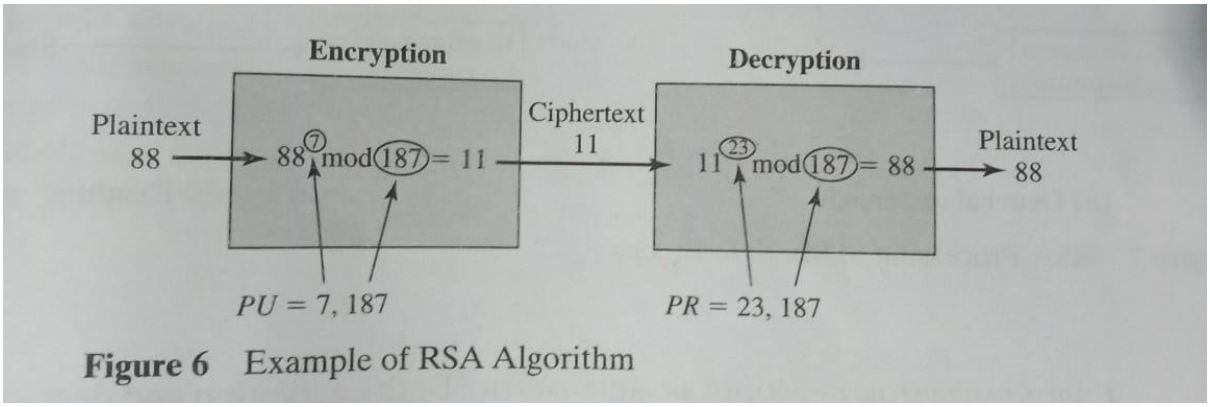


Figure 6 Example of RSA Algorithm

For this example, the keys were generated as follows

1. Select two prime numbers, p=17 and q=11.

2. Calculate n = pq=17*11=187

3. Calculate φ(n)=(p-1)(q-1)=16*10=160

4. Select e such that e is relatively prime to φ(n) = 160 and less than φ(n); we choose e=7

5. Determine d such that de= mod (160) and d < 160. The correct value is d = 23, because 23*7=161=(1*160)+1; d can be calculated using the extended Euclid's algorthim.

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 mod 187. Exploiting the properties of modular arithmetic, we can do this as follows.

88^7 mod 187=[(88^4 mod 187)x(88^2 mod 187)x(88^1 mod 187)]mod 187

88^1 mod 187=88

88^2 mod 187=7744 mod 187=77

88^4 mod 187=59969536 mod 187=132

88^7 mod 187=(88*77*132) mod 187=894,432 mod 187=11

For decryption, we calculate M=11^23 mod 187.

11^23 mod 187=[(11^1 mod 187)x(11^2 mod 187) x(11^4 mod 187) x(11^8 mod 187)x(11^ 8 mod 187)]mod 187

11^1 mod 187=11

11^2 mod 187=121

11^4 mod 187=14641 mod 187=55

11^8 mod 187=214358881 mod 187=33

11^23 mod 187=(11*121*55*33*33) mod 187=79720245 mod 187=88

Diffie Hellman key exchange:

The Diffie-Hellman algorithm is being used to establish a shared secret that can be used for secret communications while exchanging data over a public network using the elliptic curve to generate points and get the secret key using the parameters.

The Diffie Hellman algorithm depends for its effectiveness on the difficulty of computing discrete logarithms. Briefly, we can define the discrete logarithm in the following way. A primitive root of a prime number p is one whose powers modulo p generate all the integers from 1 to p-1. That is, if a is a primitive root of the prime number p, then the numbers

a mod p, a^2 mod p,….,a^p-1 mod p

are distinct and consist of the integers from 1 through p- 1 in some permutation. For any integer b and a primitive root a of prime number p, we can find a unique exponent i such that

b=a^i(mod p) where 0<=i<=(p-1)

The exponent i is referred to as the discrete logarithm of b for the base a, mod p. We express this value as dlog_a,p(b).

The Algorithm

Figure 1 summarizes the Diffie-Hellman key exchange algorithm. For this scheme, there are two publicly known numbers a prime number q and an integer a that is a primitive root of q. Suppose the users A and B wish to create a shared key.

User A selects a random integer XA<q and computes YA = a^ XA mod q. Similarly, user B independently selects a random integer XB<q and computes YB=a^XB mod q. Each side keeps the X value private and makes the Y value available publicly to the other side. Thus, XA is A's private key and YA is A's corresponding public key, and similarly for B. User A computes the key as K=(YB)^XA mod q and user B computes the key as K=(YA)^XB mod q. These two calculations produce identical results:

K=(YB)^XA mod q

= (a ^ XB mod q)^XA mod q

=(a^XB)^XA mod q

=a^XBXA mod q

=(a^XA)^XB mod q

=(a^XA mod q)^XB mod q

=(YA)^XB mod q

The result is that the two sides have exchanged a secret value. Typically, this secret value is used as shared symmetric secret key.

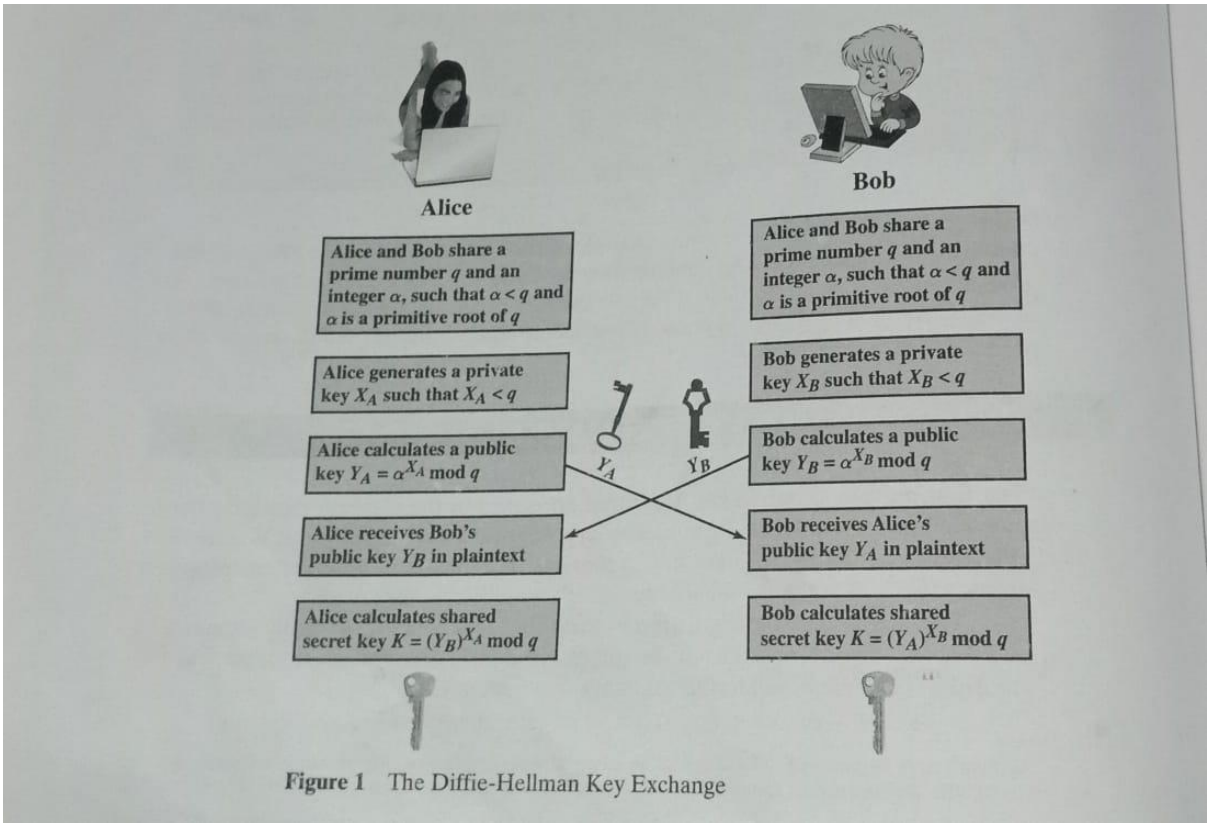


Figure 1 The Diffie-Hellman Key Exchange

Here is an example. Key exchange is based on the use of the prime number q = 353 and a primitive root of 353, in this case a=3. A and B select private keys XA=97 and XB= 233, respectively. Each computes its public key:

A computes YA =3^97 mod 353=40

B computes YB= 3^233 mod 353=248

After they exchange public keys, each can compute the common secret key:

A computes K=(YB)^XA mod 353=248^97 mod 353=160

B computes K=(YA)^XB mod 353=40^233 mod 353=160

We assume an attacker would have available the following information.

q=353; a=3; YA= 40; YB=248

In this simple example, it would be possible by brute force to determine the secret key 160. In particular, an attacker E can determine the common key by discovering a solution to the equation 3^a mod 353=40 or the equation 3^b mod 353=248. The brute-force approach is to calculate powers of 3 modulo 353, stopping when the result equals either 40 or 248. The desired answer is reached with the exponent value of 97, which provides 3^97 mod 353=40

With larger numbers, the problem becomes impractical.

Man-in-the-Middle Attack:

The protocol depicted in Figure 1 is insecure against a man-in-the-middle attack. Suppose Alice and Bob wish to exchange keys, and Darth is the adversary. The attack proceeds as follows-

1. Darth prepares for the attack by generating two random private XD1 and XD2 and then computing the corresponding public keys YD1 and YD2.

2. Alice transmits YA to Bob.

3. Darth intercepts YA and transmits YD1 to Bob. Darth also calculates K2=(YA)^XD2 mod q.

4. Bob receives YD1 and calculates K1=(YD1)^XB mod q.

5. Bob transmits YB to Alice.

6. Darth intercepts YB and transmits YD2 to Alice. Darth calculates K1=(YB)^XD1 mod q.

7. Alice receives YD2 and calculates K2=(YD2)^XA mod q.

At this point, Bob and Alice think that they share a secret key, but instead Bob and Darth share secret key K1 and Alice and Darth share secret key K2. All future communication between Bob and Alice is compromised in the following way.

1. Alice sends an encrypted message M: E(K2, M).

2. Darth intercepts the encrypted message and decrypts it to recover M.

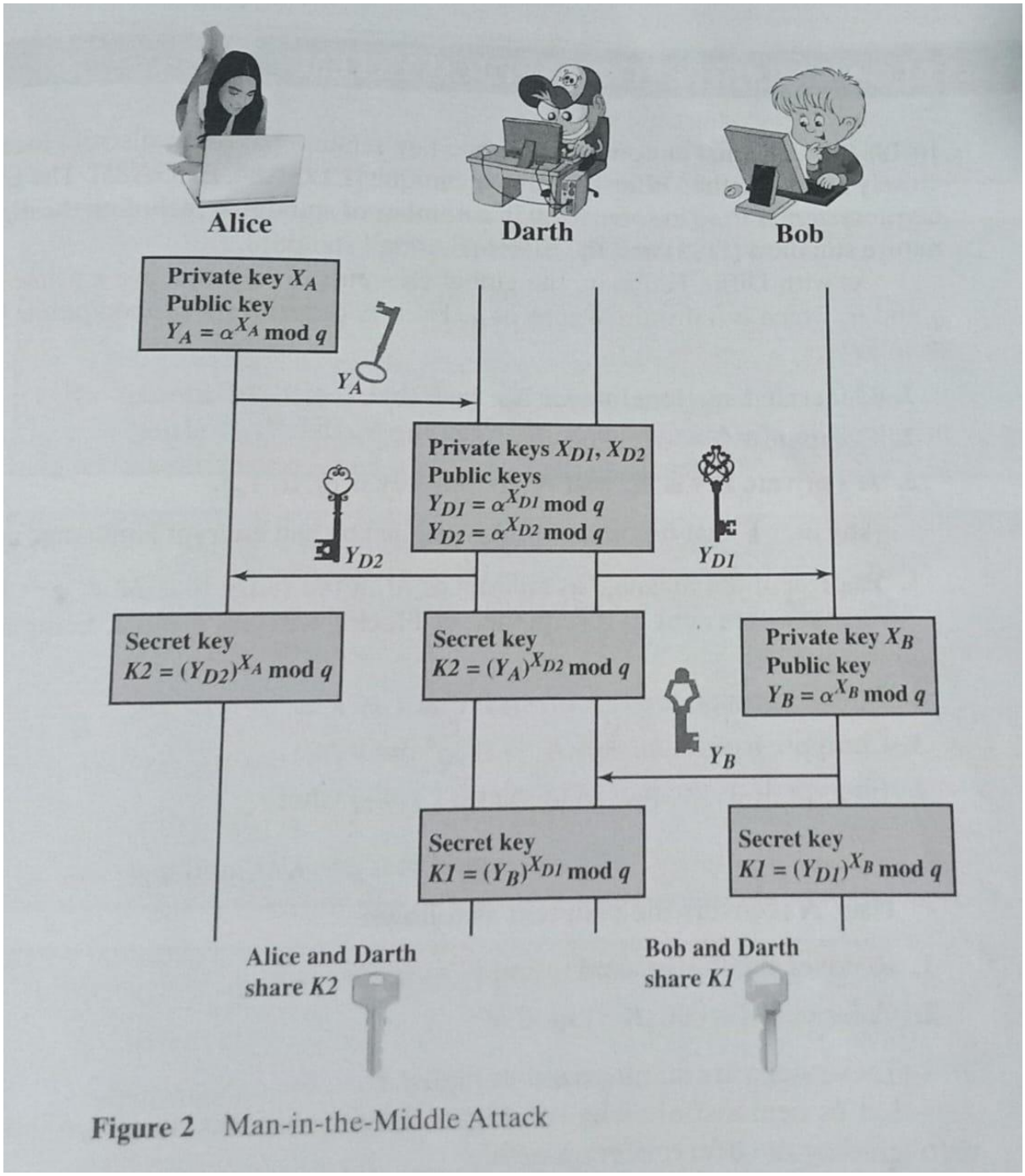


Figure 2 Man-in-the-Middle Attack

3. Darth sends Bob E(K1, M) or E(K1, M'), where M' is any message. In the first case, Darth simply wants to eavesdrop on the communication without altering it. In the second case, Darth wants to modify the message going to Bob.

The key exchange protocol is vulnerable to such an attack because it does not authenticate the participants. This vulnerability can be overcome with the use of digital signatures and public key certificates.

Elgamal cryptographic system:

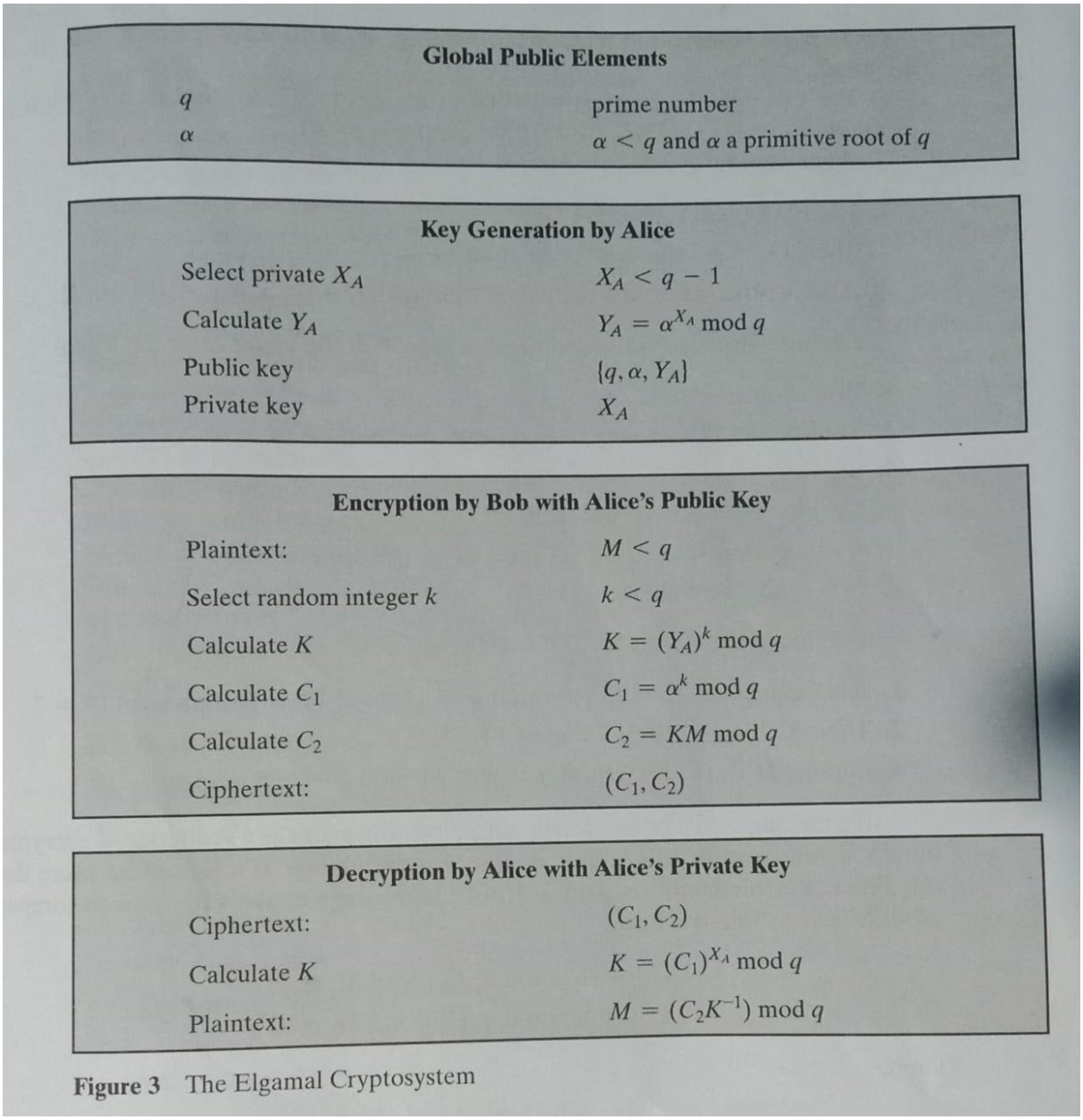
ElGamal Encryption is a public-key cryptosystem. It uses asymmetric key encryption for communicating between two parties and encrypting the message. This cryptosystem is based on the difficulty of finding **discrete logarithms** in a cyclic group that is even if we know ga and gk, it is extremely difficult to compute gak. In this article, we will get to know about the Elgamal algorithm, the components of its algorithm, its advantages & disadvantages, and the implementation of the ElGamal cryptosystem in Python.

Elgamal Cryptographic Algorithm

The ElGamal cryptographic algorithm is an asymmetric key encryption scheme based on the Diffie-Hellman key exchange

Components of the ElGamal Algorithm

1. **Key Generation**:
    - **Public Parameters**: Select a large prime number p and a generator g of the multiplicative group Z*p.
    - **Private Key**: Select a private key x such that $1 \leq x \leq p-2$.
    - **Public Key**: Compute $h=g^x \bmod p$. The public key is (p,g,h) and the private key is x.
2. **Encryption**:
    - To encrypt a message M:
        - Choose a random integer k such that $1 \leq k \leq p-2$.
        - Compute $C1 = g^k \bmod p$.
        - Compute $C2 = M \cdot h^k \bmod p$.
        - The ciphertext is (c1,c2).
3. **Decryption**:
    - To decrypt the ciphertext (c1,c2) using the private key x:
        - Compute the shared secret $s = C1^x \bmod p$.
        - Compute $s^{-1} \bmod p$ (the modular inverse of s).
        - Compute the original message $M = C2 \cdot s^{-1} \bmod p$



Global Public Elements

q	prime number
α	$\alpha < q$ and α a primitive root of q

Key Generation by Alice

Select private X_A	$X_A < q - 1$
Calculate Y_A	$Y_A = \alpha^{X_A} \bmod q$
Public key	$\{q, \alpha, Y_A\}$
Private key	X_A

Encryption by Bob with Alice's Public Key

Plaintext:	$M < q$
Select random integer k	$k < q$
Calculate K	$K = (Y_A)^k \bmod q$
Calculate C_1	$C_1 = \alpha^k \bmod q$
Calculate C_2	$C_2 = KM \bmod q$
Ciphertext:	(C_1, C_2)

Decryption by Alice with Alice's Private Key

Ciphertext:	(C_1, C_2)
Calculate K	$K = (C_1)^{X_A} \bmod q$
Plaintext:	$M = (C_2 K^{-1}) \bmod q$

Figure 3 The Elgamal Cryptosystem

Example of Elgamal encryption:

For example, let us start with the prime field GF(19); that is, q = 19. It has primitive roots [2, 3, 10, 13, 14, 15]. We choose a = 10

Alice generates a key pair as follows:

1. Alice chooses XA=5
2. Then YA=a^XA mod q=a^5 mod 19=3
3. Alice's private key is 5 and Alice's public key is {q,a,YA}={19,10,3}

Suppose Bob wants to send the message with the value M = 17. Then

1. Bob chooses k = 6

2. Then K = (YA)^k mod q = 3^6 mod 19=729 mod 19=7

3.So

C1= a^k mod q=a^6 mod 19=11

C2= KM mod q=7*17 mod 19=119 mod 19=5

4.Bob sends the ciphertext (11,5)

For decryption:

1. Alice calculates K=(C1)^XA mod q=11^5 mod 19=161051 mod 19=7

2. Then K^-1 in GF(19) is 7^-1 mod 19 = 11

3. Finally, M = (C2*K^-1) mod q=5*11 mod 19=55 mod 19=17

If a message must be broken up into blocks and sent as a sequence of encrypted blocks, a unique value of k should be used for each block. If k is used for more than one block, knowledge of one block M1 of the message enables the user to compute other blocks as follows. Let

C1,1=a^k mod q; C2,1=K*M1 mod q

C1,2=a^k mod q; C2,2=K*M2 mod q

Then

C2,1/C2,2=(KM1 mod q)/(KM2 mod q)=(M1 mod q)/(M2 mod q)

If M1 known, then M2 is easily computed as

M2=(C2,1)^-1*C2,2*M1 mod q

The security of Elgamal is based on the difficulty of computing discrete logarithms. To recover A's private key, an adversary would have to compute XA=d*log_(a, g)(YA). Alternatively, to recover the one time key K, an adversary would have to determine the random number k, and this would require computing the discrete logarithm k=d*log_(a, q)(C1) points out that these calculations are regarded as infeasible it p is at least 300 decimal digits and q-1 has at least one large prime factor.