PUBLIC (ASYMMETRIC) KEY CRYPTOGRAPHY

STRUCTURE

6.0 Public Key Cryptography

Unlike symmetric key cryptography, we do not find historical use of public-key cryptography. It is a relatively new concept.

Symmetric cryptography was well suited for organizations such as governments, military, and big financial corporations were involved in the classified communication.

With the spread of more unsecure computer networks in last few decades, a genuine need was felt to use cryptography at larger scale. The symmetric key was found to be non-practical due to challenges it faced for key management. This gave rise to the public key cryptosystems.

The process of encryption and decryption is depicted in Fig. 6.1 the following illustration

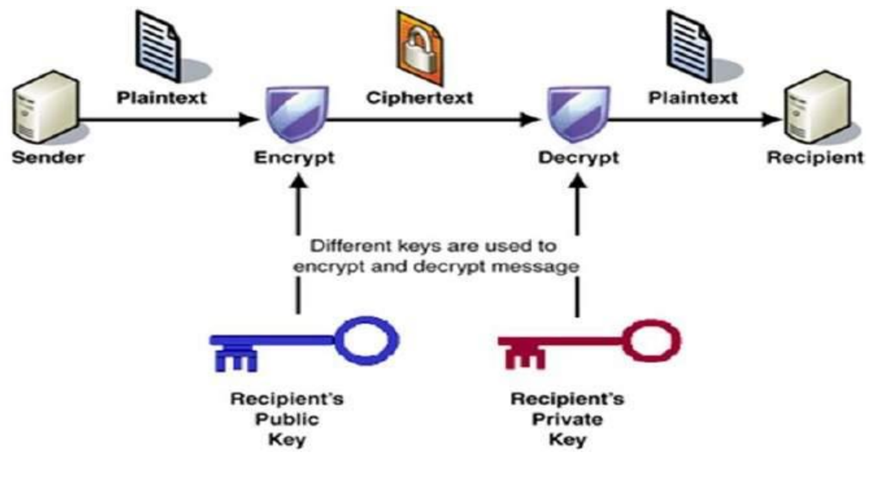


Fig. 6.1 Process of encryption and decryption

The most important properties of public key encryption scheme are:

- Different keys are used for encryption and decryption. This is a property which set this scheme different than symmetric encryption scheme.
- Each receiver possesses a unique decryption key, generally referred to as his private key.
- Receiver needs to publish an encryption key, referred to as his public key.
- Some assurance of the authenticity of a public key is needed in this scheme to avoid spoofing by adversary as the receiver. Generally, this type of cryptosystem involves trusted third party which certifies that a particular public key belongs to a specific person or entity only.
- Encryption algorithm is complex enough to prohibit attacker from deducing the plaintext from the ciphertext and the encryption (public) key.
- Though private and public keys are related mathematically, it is not be feasible to calculate the private key from the public key. In fact, intelligent part of any public-key cryptosystem is in designing a relationship between two keys.

There are three types of Public Key Encryption schemes. We discuss them in following sections.

6.1 RSA Cryptosystem

This cryptosystem is one the initial system. It remains most employed cryptosystem even today. The system was invented by three scholars **Ron Rivest, Adi Shamir,** and **Len Adleman** and hence, it is termed as RSA cryptosystem.

We will see two aspects of the RSA cryptosystem, firstly generation of key pair and secondly encryption-decryption algorithms.

Generation of RSA Key Pair

Each person or a party who desires to participate in communication using encryption needs to generate a pair of keys, namely public key and private key. The process followed in the generation of keys is described below

- **Generate the RSA modulus (n)**

  Select two large primes, p and q.

Calculate n=p*q. For strong unbreakable encryption, let n be a large number, typically a minimum of 512 bits.

- **Find Derived Number (e)**

  Number **e** must be greater than 1 and less than (p − 1)(q − 1).

  There must be no common factor for e and (p − 1)(q − 1) except for 1. In other words two numbers e and (p – 1)(q – 1) are coprime.

- **Form the public key**

  The pair of numbers (n, e) form the RSA public key and is made public.

  Interestingly, though n is part of the public key, difficulty in factorizing a large prime number ensures that attacker cannot find in finite time the two primes (p & q) used to obtain n. This is strength of RSA.

- **Generate the private key**

  Private Key d is calculated from p, q, and e. For given n and e, there is unique number d.

  Number d is the inverse of e modulo (p - 1)(q – 1). This means that d is the number less than (p - 1)(q - 1) such that when multiplied by e, it is equal to 1 modulo (p - 1)(q - 1).

This relationship is written mathematically as follows

$$ed = 1 \bmod (p - 1)(q - 1)$$

The Extended Euclidean Algorithm takes p, q, and e as input and gives d as output.

Example

An example of generating RSA Key pair is given below. (For ease of understanding, the primes p & q taken here are small values. Practically, these values are very high).

- Let two primes be p = 7 and q = 13. Thus, modulus n = pq = 7 x 13 = 91.
- Select e = 5, which is a valid choice since there is no number that is common factor of 5 and (p − 1)(q − 1) = 6 × 12 = 72, except for 1.
- The pair of numbers (n, e) = (91, 5) forms the public key and can be made available to anyone whom we wish to be able to send us encrypted messages.

- Input p = 7, q = 13, and e = 5 to the Extended Euclidean Algorithm. The output will be d = 29.
- Check that the d calculated is correct by computing −

$$de = 29 \times 5 = 145 = 1 \bmod 72$$

- Hence, public key is (91, 5) and private keys is (91, 29).

Encryption and Decryption

Once the key pair has been generated, the process of encryption and decryption are relatively straightforward and computationally easy.

Interestingly, RSA does not directly operate on strings of bits as in case of symmetric key encryption. It operates on numbers modulo n. Hence, it is necessary to represent the plaintext as a series of numbers less than n.

RSA Encryption

- Suppose the sender wish to send some text message to someone whose public key is (n, e).
- The sender then represents the plaintext as a series of numbers less than n.
- To encrypt the first plaintext P, which is a number modulo n. The encryption process is simple mathematical step as −

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

- In other words, the ciphertext C is equal to the plaintext P multiplied by itself e times and then reduced modulo n. This means that C is also a number less than n.
- Returning to our Key Generation example with plaintext P = 10, we get ciphertext C −

$$C = 10^5 \bmod 91$$

RSA Decryption

- The decryption process for RSA is also very straightforward. Suppose that the receiver of public-key pair (n, e) has received a ciphertext C.
- Receiver raises C to the power of his private key d. The result modulo n will be the plaintext P.

$$\text{Plaintext} = C^d \bmod n$$

- Returning again to our numerical example, the ciphertext C = 82 would get decrypted to number 10 using private key 29 −

$$\text{Plaintext} = 82^{29} \bmod 91 = 10$$

RSA Analysis

The security of RSA depends on the strengths of two separate functions. The RSA cryptosystem is most popular public-key cryptosystem strength of which is based on the practical difficulty of factoring the very large numbers.

- **Encryption Function** − It is considered as a one-way function of converting plaintext into ciphertext and it can be reversed only with the knowledge of private key d.
- **Key Generation** − The difficulty of determining a private key from an RSA public key is equivalent to factoring the modulus n. An attacker thus cannot use knowledge of an RSA public key to determine an RSA private key unless he can factor n. It is also a one way function, going from p & q values to modulus n is easy but reverse is not possible.

If either of these two functions are proved non-one-way, then RSA will be broken. In fact, if a technique for factoring efficiently is developed then RSA will no longer be safe.

The strength of RSA encryption drastically goes down against attacks if the number p and q are not large primes and/ or chosen public key e is a small number.

6.2 Elliptic Curve Cryptography (ECC)

Elliptic Curve Cryptography (ECC) is an approach to public-key cryptography, based on the algebraic structure of elliptic curves over finite fields. ECC requires a smaller key as compared to non-ECC cryptography to provide equivalent security (a 256-bit ECC security has equivalent security attained by 3072-bit RSA cryptography).

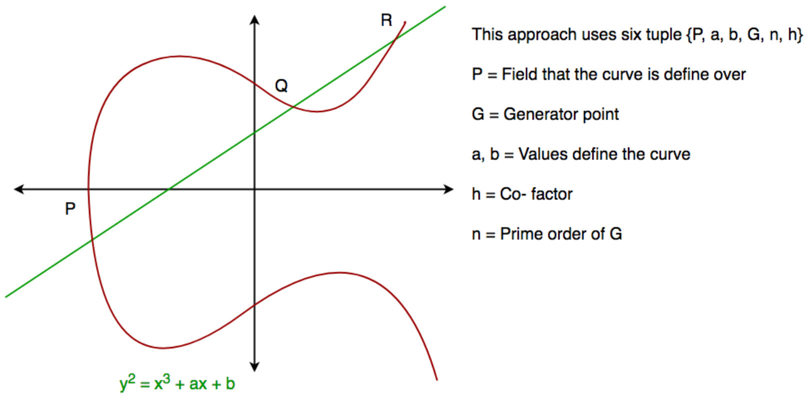
For a better understanding of Elliptic Curve Cryptography, it is very important to understand the basics of the Elliptic Curve. An elliptic curve is a planar algebraic curve defined by an equation of the form

$$y^2 = x^3+ax+b$$

Where ‘a’ is the co-efficient of x and ‘b’ is the constant of the equation

The curve is non-singular; that is, its graph has no cusps or self-intersections (when the characteristic of the Coefficient field is equal to 2 or 3).

In general, an elliptic curve looks like as shown below. Elliptic curves can intersect almost 3 points when a straight line is drawn intersecting the curve. As we can see, the elliptic curve is symmetric about the x-axis. This property plays a key role in the algorithm.



This approach uses six tuple {P, a, b, G, n, h}

P = Field that the curve is define over

G = Generator point

a, b = Values define the curve

h = Co- factor

n = Prime order of G

6.3 Key Exchange (Diffie-Hellman algorithm)

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.

- For the sake of simplicity and practical implementation of the algorithm, we will consider only 4 variables, one prime P and G (a primitive root of P) and two private values a and b.
- P and G are both publicly available numbers. Users (say Alice and Bob) pick private values a and b and they generate a key and exchange it publicly. The opposite person receives the key and that generates a secret key, after which they have the same secret key to encrypt.

Step-by-Step explanation is as follows:

Alice	**Bob**
Public Keys available = P, G	Public Keys available = P, G
Private Key Selected = a	Private Key Selected = b
Key generated: $x = G^a \bmod P$	Key generated: $y = G^b \bmod P$
Exchange of generated keys takes place	
Key received = y	key received = x
Generated Secret Key: $K_a = y^a \bmod P$	Generated Secret Key: $K_b = x^b \bmod P$
Algebraically, it can be shown that: $k_a = k_b$	
Users now have a symmetric secret key to encrypt	

Example:

Step 1: Alice and Bob get public numbers P = 23, G = 9

Step 2: Alice selected a private key a = 4 and

Bob selected a private key b = 3

Step 3: Alice and Bob compute public values

Alice: x =(9^4 mod 23) = (6561 mod 23) = 6

Bob: y = (9^3 mod 23) = (729 mod 23) = 16

Step 4: Alice and Bob exchange public numbers

Step 5: Alice receives public key y =16 and

Bob receives public key x = 6

Step 6: Alice and Bob compute symmetric keys

Alice: ka = y^a mod p = 65536 mod 23 = 9

Bob: kb = x^b mod p = 216 mod 23 = 9

Step 7: 9 is the shared secret.

6.4 Java Cryptography Extension (JCE)

The Java Cryptography Extension (JCE) is an application program interface (API) that provides a uniform framework for the implementation of security features in Java. It was originally developed to supplement the Java 2 Software Developer's Kit (SDK), Standard Edition, versions 1.2.x and 1.3.x, but has since been integrated into the Java 2 SDK, version 1.4.

The JCE architecture is provider-based, and new applications can be added seamlessly. JCE supports several applications in digital security, such as the following:

Symmetric ciphers
Asymmetric ciphers
Stream ciphers
Block ciphers
Key generation
Key storage
Key retrieval
Secure streams
Sealed objects
Digital signatures
Message Authentication Code (MAC) algorithms

6.5 Attacks

Public key cryptography, also known as asymmetric cryptography, uses a pair of mathematically related keys: a public key and a private key. The public key is openly shared and used for encryption, while the private key is kept secret and used for decryption. While public key cryptography offers several advantages, it is not without its vulnerabilities. Here are some common attacks on public key cryptography:

1. **Brute-Force Attack:** Brute-forcing the private key by trying all possible combinations until the correct one is found. The security of public key cryptography relies on the computational infeasibility of brute-force attacks due to large key sizes.
2. **Chosen Plaintext Attack (CPA):** In a CPA, the attacker can obtain the

encryption of chosen plaintexts using the public key and analyze the corresponding ciphertexts to gain information about the private key.

3. **Chosen Ciphertext Attack (CCA):** In a CCA, the attacker can obtain the decryption of chosen ciphertexts using the public key and analyze the decrypted plaintexts to gain information about the private key.
4. **Timing Attacks:** Timing attacks analyze the time taken by cryptographic operations to deduce information about the private key. Variations in execution time can leak information about the private key.
5. **Side-Channel Attacks:** Side-channel attacks exploit information leaked during the execution of cryptographic operations, such as power consumption, electromagnetic radiation, or sound, to deduce information about the private key.
6. **Key Management Attacks:** These attacks exploit weaknesses in key generation, storage, or distribution, making it easier for attackers to obtain the private key.
7. **Man-in-the-Middle (MITM) Attack:** An attacker intercepts communication between parties, impersonates each party to the other, and can use their own public keys to communicate secretly with each party.
8. **Padding Oracle Attack:** This attack targets vulnerabilities in padding schemes used in some public key encryption protocols to recover the plaintext from ciphertext.

To counter these attacks and ensure the security of public key cryptography, it is crucial to:

Use strong cryptographic algorithms with appropriate key lengths.

Implement secure key management practices, including key generation, storage, and distribution.

Protect private keys from unauthorized access and tampering.

Employ countermeasures against side-channel attacks, such as constant-time algorithms and secure hardware design.

Regularly update cryptographic algorithms and libraries to address known vulnerabilities.

Use digital signatures and certificates to verify the authenticity of public keys during key exchange.

Deploy secure protocols to protect against man-in-the-middle attacks.

While public key cryptography is generally more secure than symmetric key cryptography in certain use cases, it requires careful implementation and management to protect against potential vulnerabilities and attacks.