Public Key Encryption

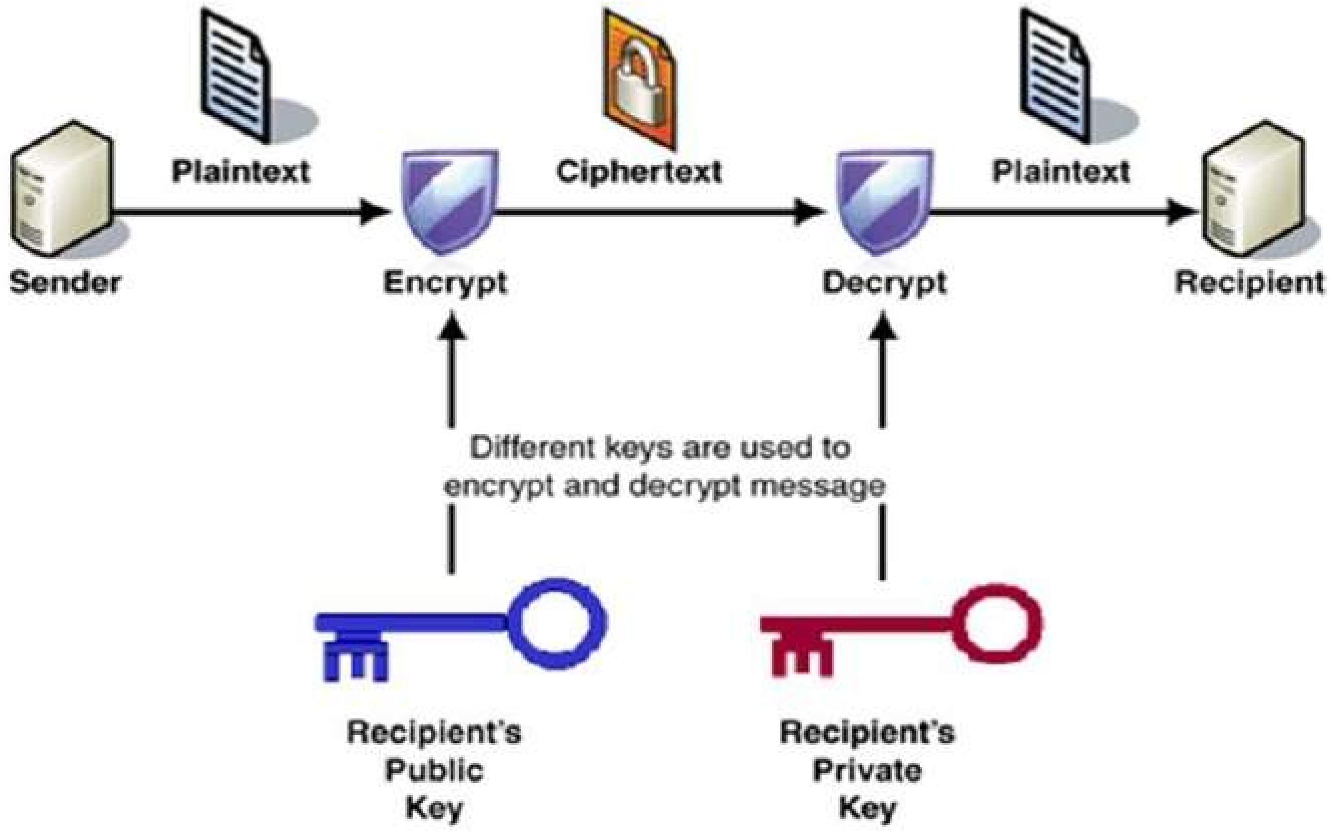
Public Key Cryptography

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 the following illustration −



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 −

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 $p-1q-1$.
    - There must be no common factor for e and $p-1q-1$ except for 1. In other words two numbers e and $p–1q–1$ 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 $\boxed{\text{p \& q}}$ 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 $p-1q-1$. This means that d is the number less than $p-1q-1$ such that when multiplied by e, it is equal to 1 modulo $p-1q-1$.
- This relationship is written mathematically as follows −

```
ed = 1 mod (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 $p-1q-1$ = 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 × 5 = 145 = 1 mod 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.

```
Plaintext = C^d mod n
```

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

```
Plaintext = 82^29 mod 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.

ElGamal Cryptosystem

Along with RSA, there are other public-key cryptosystems proposed. Many of them are based on different versions of the Discrete Logarithm Problem.

ElGamal cryptosystem, called Elliptic Curve Variant, is based on the Discrete Logarithm Problem. It derives the strength from the assumption that the discrete logarithms cannot be found in practical time frame for a given number, while the inverse operation of the power can be computed efficiently.

Let us go through a simple version of ElGamal that works with numbers modulo p. In the case of elliptic curve variants, it is based on quite different number systems.

Generation of ElGamal Key Pair

Each user of ElGamal cryptosystem generates the key pair through as follows −

- **Choosing a large prime p.** Generally a prime number of 1024 to 2048 bits length is chosen.
- **Choosing a generator element g.**

  - This number must be between 1 and p − 1, but cannot be any number.
  - It is a generator of the multiplicative group of integers modulo p. This means for every integer m co-prime to p, there is an integer k such that g^k=a mod n.

    For example, 3 is generator of group 5 (Z_5 = {1, 2, 3, 4}).

N	3^n	3^n mod 5
1	3	3
2	9	4
3	27	2
4	81	1

- **Choosing the private key.** The private key x is any number bigger than 1 and smaller than p−1.
- **Computing part of the public key.** The value y is computed from the parameters p, g and the private key x as follows −

```
y = g^x mod p
```

- **Obtaining Public key.** The ElGamal public key consists of the three parameters p, g, y.

  For example, suppose that p = 17 and that g = 6 (It can be confirmed that 6 is a generator of group Z_{17}). The private key x can be any number bigger than 1 and smaller than 71, so we choose x = 5. The value y is then computed as follows −

```
y = 6^5 mod 17 = 7
```

- Thus the private key is 62 and the public key is $17, 6, 7$.

Encryption and Decryption

The generation of an ElGamal key pair is comparatively simpler than the equivalent process for RSA. But the encryption and decryption are slightly more complex than RSA.

ElGamal Encryption

Suppose sender wishes to send a plaintext to someone whose ElGamal public key is p, g, y, then −

- Sender represents the plaintext as a series of numbers modulo p.
- To encrypt the first plaintext P, which is represented as a number modulo p. The encryption process to obtain the ciphertext C is as follows −
  - Randomly generate a number k;
  - Compute two values C1 and C2, where −

```
C1 = g^k mod p
C2 = (P*y^k) mod p
```

- Send the ciphertext C, consisting of the two separate values $C1, C2$, sent together.

- Referring to our ElGamal key generation example given above, the plaintext P = 13 is encrypted as follows −
    - Randomly generate a number, say k = 10
    - Compute the two values C1 and C2, where −

```
C1 = 6^10 mod 17
C2 = (13*7^10) mod 17 = 9
```

- Send the ciphertext C = $C1, C2 = 15, 9$.

ElGamal Decryption

- To decrypt the ciphertext $C1, C2$ using private key x, the following two steps are taken −
    - Compute the modular inverse of $C1^{x}$ modulo p, which is $C1^{-x}$, generally referred to as decryption factor.
    - Obtain the plaintext by using the following formula −

```
C2 × (C1)^-x  mod p = Plaintext
```

- In our example, to decrypt the ciphertext C = $C1, C2 = 15, 9$ using private key x = 5, the decryption factor is

```
15^-5  mod 17 = 9
```

- Extract plaintext P = 9×9 mod 17 = 13.

ElGamal Analysis

In ElGamal system, each user has a private key x. and has **three components** of public key − **prime modulus p, generator g, and public Y = g^{x} mod p**. The strength of the ElGamal is based on the difficulty of discrete logarithm problem.

The secure key size is generally > 1024 bits. Today even 2048 bits long key are used. On the processing speed front, Elgamal is quite slow, it is used mainly for key authentication protocols. Due to higher processing efficiency, Elliptic Curve variants of ElGamal are becoming increasingly popular.

Elliptic Curve Cryptography ECC

Elliptic Curve Cryptography ECC is a term used to describe a suite of cryptographic tools and protocols whose security is based on special versions of the discrete logarithm problem. It does not use numbers modulo p.

ECC is based on sets of numbers that are associated with mathematical objects called elliptic curves. There are rules for adding and computing multiples of these numbers, just as there are for numbers modulo p.

ECC includes a variants of many cryptographic schemes that were initially designed for modular numbers such as ElGamal encryption and Digital Signature Algorithm.

It is believed that the discrete logarithm problem is much harder when applied to points on an elliptic curve. This prompts switching from numbers modulo p to points on an elliptic curve. Also an equivalent security level can be obtained with shorter keys if we use elliptic curve-based variants.

The shorter keys result in two benefits −

- Ease of key management
- Efficient computation

These benefits make elliptic-curve-based variants of encryption scheme highly attractive for application where computing resources are constrained.

RSA and ElGamal Schemes – A Comparison

Let us briefly compare the RSA and ElGamal schemes on the various aspects.

RSA	ElGamal
It is more efficient for encryption.	It is more efficient for decryption.
It is less efficient for decryption.	It is more efficient for decryption.
For a particular security level, lengthy keys are required in RSA.	For the same level of security, very short keys are required.
It is widely accepted and used.	It is new and not very popular in market.