7

Public Key Cryptography

Great Ideas in Theoretical Computer Science
Saarland University, Summer 2014

Cryptography studies techniques for secure communication in the presence of third parties. A typical cryptography consists of two schemes: the *encryption scheme* and the *decryption* scheme. The sender uses encryption scheme to encode the plaintext, and the encrypted message, called *ciphertext*, looks nonsense to a third person. Upon receiving a ciphertext, the receiver applies the decryption scheme to decrypt the ciphertext, and receive the plaintext. Cryptography has a long history, and design of modern cryptographic protocols are based on computational complexity theory.

We first look at some simple encryption/decrpytion schemes. Assume that Alice wants to send the plaintext $s \in \Sigma^n$ to Bob, where $\Sigma = \{A, B, \ldots, Z\}$. Instead of sending the string s to Bob directly, Alice could create a random permutation $\pi : \Sigma \mapsto \Sigma$, and send $\pi(s_1), \pi(s_2), \ldots, \pi(s_n)$ to Bob. After Bob receives the ciphertext, he can use the same permutation to recover the plain text. This is one of the cryptography systems widely used before the computer era. The drawback of this system is the security: Although any person without knowing the permutation π couldn't get the plaintext directly, the encryption scheme does not change the frequency of the letters in Σ, and letters in a meaningful text follow certain frequency rules. Hence a third person could recover partial (or even complete) information of the plaintext after receiving enough encrypted text.

To overcome this, we can use another way for the encryption. Instead of creating a random permutation over Σ, Alice uses a dictionary of length the same as the plaintext to encrypt the plaintext: take module operation between the ith letter of the plaintext and the ith letter of the dictionary to obtain the ciphertext. The receiver uses the same dictionary to take the inverse operation and get the plaintext from the ciphertext. Through this way, the same letter in Σ maps to different letters, and it becomes more difficult for an adversary to decrypt the ciphertext. However, certain problems remain: the dictionary, the key of the cryptography system, has the same size as the plaintext, and sharing the key in a secure way may be as difficult as sharing the plaintext in a secure way.

In these two examples above, Alice and Bob use the same key to encrypt and decrypt the text. This kind of systems is the oldest cryptography techniques, and called the *Symmetric Encryption*. One general problem of Symmetric Encryption is the secure exchanging of the

key over the Internet or large networks is difficult: Anyone who get the key can decrypt the ciphertext.

In 1976, Whitfield Diffe and Martin Hellman created public key cryptography. Instead of a single shared, secrete key, they propose to used two separate key in a cryptography system. One key is called the *private key*, and is held by only on party. The second key is called the *public key*, and is not a secret and can be shared widely. These two keys form a pair, and can be used together in encryption and decryption operations. Moreover, a public key and its corresponding private key are paired together and are related to no other keys.

This pairing is possible because of a special mathematical relationship between the algorithms for the public keys and private keys. The key pairs are mathematically related to one another such that using the key pair together achieves the same result as using a symmetrical key twice. The keys must be used together: each individual key cannot be used to undo its own operation. This means that the operation of each individual key is a one-way operation: a key cannot be used to reverse its operation. In addition, the algorithms used by both keys are designed so that a key cannot be used to determine the opposite key in the pair. Thus, the private key cannot be determined from the public key.

7.1 Background Knowledge

7.1.1 Modular Arithmetic

Definition 7.1. *The number a is equivalent (congruent) to the number b modulo n, expressed by $a \equiv b \pmod{n}$, if a differs from b by an exact multiple of n.*

Lemma 7.2. *The following statements hold:*

- *If $a \equiv b \pmod{n}$ and $c \equiv d \pmod{n}$, then $a + c \equiv b + d \pmod{n}$.*
- *If $a \equiv b \pmod{n}$ and $c \equiv d \pmod{n}$, then $ac \equiv bd \pmod{n}$.*

Example 7.3. $321 \times 741 \equiv 1 \times 1 \equiv 1 \pmod{5}$.

Example 7.4. $715^{10000} \equiv 1 \pmod{7}$.

Example 7.5. $321^3 \equiv 6^3 \pmod{7} = 36 \times 6 \pmod{7} \equiv 6 \pmod{7}$.

Example 7.6. $320^{984} \equiv 1 \pmod{7}$

Proof. We first write down the binary expression of 984, i.e.

$$\begin{aligned} 984 &= 512 + 256 + 128 + 64 + 16 + 8 \\ &= 2^9 + 2^8 + 2^7 + 2^6 + 2^4 + 2^3. \end{aligned}$$

Note that $320^{984} \equiv 5^{984} \pmod{7}$. Moreover, we have the following:

- $5^2 = 25 \equiv 4 \pmod{7}$
- $5^4 = 4 \times 4 \pmod{7} \equiv 2 \pmod{7}$

- $5^8 = 2 \times 2 \pmod 7 \equiv 4 \pmod 7$
- $5^{16} = 4 \times 4 \pmod 7 = 2 \pmod 7$
- $5^{32} \equiv 4 \pmod 7$
- $5^{64} \equiv 2 \pmod 7$
- $5^{128} \equiv 4 \pmod 7$
- $5^{256} \equiv 2 \pmod 7$
- $5^{512} \equiv 4 \pmod 7$

Hence

$$\begin{aligned} 5^{984} &= 5^{512+256+128+64+16+8} \pmod 7 \\ &\equiv 4 \times 2 \times 4 \times 2 \times 2 \times 2 \times 4 \pmod 7 \\ &\equiv 2 \pmod 7 \end{aligned}$$

□

7.1.2 Fermat's Little Theorem

We call that n is divisible by m if $n = km$.

Theorem 7.7 (Fermat's Little Theorem)**.** *If p is a prime number, then $a^p \equiv a \pmod p$ for all a.*

Theorem 7.8 (Fermat's Little Theorem, Alternative Form)**.** *If p is a prime number and a is any integer not divisible by p, then $a^{p-1} \equiv 1 \pmod p$.*

7.2 The Euclidean Algorithm

Given two integers r_0 and r_1, the Euclidean Algorithm finds the greatest common divisor of r_0 and r_1, denoted by $\gcd(r_0, r_1)$.

Before present the algorithm, we first look at the following lemma.

Lemma 7.9. $\gcd(r_0, r_1) = \gcd(r_1, r_0 \bmod r_1)$

Proof. Let $x = \gcd(r_0, r_1)$. Then we can write $r_0 = cx$ and $r_1 = dx$, where $c, d \in \mathbb{Z}$. Without loss of generality we assume that $r_0 \geq r_1$, otherwise the statement holds trivially. Then we can write $r_0 = pr_1 + q$, where $q \in \{0, 1, \ldots, r_1 - 1\}$. Hence,

$$\begin{aligned} \gcd(r_0, r_1) &= \gcd(cx, dx) = \gcd(pdx + q, dx) = \gcd(q, dx) \\ &= \gcd(r_0 \bmod r_1, r_1). \end{aligned}$$

□

Algorithm 7.1 Euclidean Algorithm

1: write a as $a = bq_1 + r_1$, for $r_1 \in \{1, 2, \ldots, b-1\}$;
2: write b as $b = r_1q_2 + r_2$, for $r_2 \in \{0, \ldots, r_1 - 1\}$
3: Let $j = 0$
4: **while** $r_{j+2} \neq 0$ **do**
5: $\quad j \leftarrow j + 1$;
6: $\quad$ write r_j as $r_j = r_{j+1}q_{j+2} + r_{j+2}$, where $r_{j+1} \in \{0, \ldots, r_{j+1} - 1\}$;
7: **return** r_{j+1} $\triangleright\ r_{j+1} = \gcd(a, b)$

Now we show that the Euclidean Algorithm can be used to compute a multiplicative inverse.

Definition 7.10. *If $ab \equiv 1 \pmod{p}$, then b is called the multiplicative inverse of a module p.*

Theorem 7.11 (Multiplicative Inverse Algorithm)**.** *Given two integers $0 < b < a$, consider the Euclidean Algorithm equations which yield $\gcd(a, b) = r_j$. Rewrite all of these equations except the last one, by solving for the remainders:*

$$\begin{aligned} r_1 &= a - bq_1 \\ r_2 &= b - r_1q_2 \\ r_3 &= r_1 - r_2q_3 \\ &\cdots \\ r_{j-1} &= r_{j-3} - r_{j-2}q_{j-1} \\ r_j &= r_{j-2} - r_{j-1}q_j \end{aligned}$$

Then, in the last of these equations, $r_j = r_{j-2} - r_{j-1}q_j$, replace r_{j-1} with its expression in terms of r_{j-3} and r_{j-2} from the equation immediately above it. Continue this process successively, replacing $r_{j-2}, r_{j-3}, \cdots$, until we obtain the final equation

$$r_j = ax + by,$$

where x and y are integers. In the special case that $\gcd(a, b) = 1$, the integer equation reads

$$1 = ax + by.$$

Therefore we deduce

$$1 \equiv by \pmod{a}$$

so that the residue of y is the multiplicative inverse of b, mod a.

7.3 The RSA Algorithm

In the initialization step, we choose two prime numbers p, q, and let $n = p \cdot q$. We further pick a positive integer r that has no common factor with $(p-1) \cdot (q-1)$, and find a multiplicative inverse of r modulo $(p-1)\cdot(q-1)$, i.e. we find a number s such that $rs \equiv 1 \pmod{(p-1)\cdot(q-1)}$.

Encryption. The pair of values n, r are called the **public encryption key**, and these two numbers are publicly available. Given the private key, any plaintext $x \leq n$ is encrypted by

$$y \triangleq x^r \pmod{n}.$$

Decryption. The pair of values n, s are called the **private decryption key**. With these two numbers, we can compute

$$z \triangleq y^s \pmod{n}.$$

That is, you need to know s to decrypt. Now s is the multiplicative inverse of r modulo $(p-1)(q-1)$. The outsiders know r, and if they knew $(p-1)(q-1)$, then it would be easy (with the Euclidean Algorithm) to compute s. But they do not know $(p-1)(q-1)$. They know n, which is equal to pq, but they do not have n factored into p and q. To find $(p-1)(q-1)$, they need to know the prime factors p and q of n, and factoring large numbers is difficult.

Theorem 7.12. *The decrypted message $z = x$.*

Proof. By definition, we have that

$$z = y^s \pmod{n} = x^{rs} \pmod{n} = x^{rs} \pmod{pq}.$$

Since $rs \equiv 1 \pmod{(p-1)\cdot(q-1)}$, we have that $rs = c \cdot (p-1)(q-1) + 1$ for an integer c, and

$$z = x^{c(p-1)(q-1)+1} \pmod{pq}.$$

It suffices to show that $x^{c(p-1)(q-1)+1} \equiv 1 \pmod{p}$ and $x^{c(p-1)(q-1)+1} \equiv 1 \pmod{q}$. We only look at the first case, and the second case can be proven in the same way.

(1) If x is divisible by p, then $x^{c(p-1)(q-1)+1} \equiv 0 \pmod{p} \equiv x \pmod{p}$ holds trivially. (2) If x is not divisible by p, then by Fermat's Little Theorem we know that $z = x^{c(p-1)(q-1)} \cdot x \equiv x \pmod{p}$ and $z = x^{c(p-1)(q-1)} \cdot x \equiv x \pmod{q}$.

Therefore, we have that $z = x^{c(p-1)(q-1)} \cdot x \equiv x \pmod{pq}$. □