CRYPTOGRAPHY AND NUMBER THEORY

XINYU SHI

Abstract. In this paper, we will discuss a few examples of cryptographic systems, categorized into two different types: symmetric and asymmetric cryptography. We will mainly discuss RSA and Diffie-Hellman key exchange.

Contents

1. Introduction to Cryptography

Cryptography is a study of methods to communicate securely over an insecure line of communication. The main idea is to "encipher" the message into a form that only the intended recipient can understand. Crypt, the root of the word, cryptography, comes from the Greek word "kryptos", meaning hidden or secret. It has played an important role throughout history. In ancient time, people used some simple forms of cryptography to protect their messages and knowledge. Some of Chinese silk traders studied how to protect the secret of manufacturing silk, and the Germans wanted to protect their military secrets by applying their famous Enigma machine [1]. In recent times, cryptography is being widely used in computer industries, such as storing the user's information securely online, and protecting governmental secrets and so on.

Before getting to know the actual cryptosystems, we will start with some basic number theory that will be helpful to understand the cryptographic algorithms in section 2. There are roughly two categories of cryptography. One is symmetric, and the other is asymmetric, which will show up in the following section 3 and section 4 respectively. Symmetric cryptography is that people use the same key to communicate the message while asymmetric cryptography uses different keys for both sender and receiver. Each of them has its advantages and disadvantages. Symmetric cryptography requires both parties to find a way to a priori share some secure

Date: AUGUST 28, 2018.

knowledge, which could be achieved by meeting physically, while asymmetric cryptography does not. However, asymmetric cryptography typically takes much longer to actually communicate messages than symmetric cryptography. We will introduce examples of both types of cryptosystems in the following sections, namely RSA (asymmetric cryptography) and Diffie-Hellman (symmetric cryptography). Last but not least, we will study the discrete logarithm to understand a possible attack on both RSA and Diffie-Hellman.

2. Some Number Theory

Before we start studying cryptography, we need a few basic concepts in elementary number theory to explain the algorithms involved.

First we will discuss the Euclidean algorithm, which also provides us with a useful property of the greatest common divisor, which is denoted as "gcd".

Theorem 2.1. *If gcd$(a, b) = d$, then there exist x,y such that $ax + by = d$.*

Proof. The theorem will follow from the following description of the Euclidean algorithm. Euclidean algorithm proceeds by dividing the larger number (say a) by the smaller number (say b) to get quotient q_1 and remainder r_1(less than b), that is,

$$a = q_1 b + r_1$$

Then repeat the procedure with b and r_1 to get

$$b = q_2 r_1 + r_2$$

Continue in this way, and obtain

$$r_{n-2} = q_n r_{n-1} + r_n$$

Once we gain $r_n \mid r_{n-1}$, we are done in finding the greatest common divisor. The reason is shown as following:

Once we obtain the above equation $r_{n-2} = q_n r_{n-1} + r_n$, we do one more step and get

$$r_{n-1} = q_{n+1} r_n + 0$$

This means $r_n \mid r_{n-1}$. By looking for the opposite direction, we check $r_n \mid r_{n-2}$ from

$$r_{n-2} = q_n r_{n-1} + r_n$$

Continue in this way, we will finally check $r_n \mid b$, and $r_n \mid a$. Therefore, we check r_n is a common divisor of a and b.

Next, we express r_n as a linear combination of a and b. From the previous step, we gain $r_{n-2} = q_n r_{n-1} + r_n$, which is the same as $r_n = r_{n-2} - q_n r_{n-1}$ and then substitute the remainder r_{n-1} with $r_{n-3} - q_{n-1} r_{n-2}$ and obtain $r_n = r_{n-2} - q_n(r_{n-3} - q_{n-1} r_{n-2})$. Continuing in this way, at the end, we gain a linear combination $r_n = ax + by$.

Now, suppose d is the gcd of a and b. So we get $d \mid a$ and $d \mid b$. Hence, we get r_n is the gcd of a and b. □

Corollary 2.2. *If gcd$(a, b) = 1$, then a has an inverse mod b.*

Proof. By the theorem, we know that $ax + by = 1$. Reducing both sides modulo b, hence, we have $ax + 0b = 1 \pmod b$, which is $ax = 1 \pmod b$. Therefore, we have shown that a has an inverse mod b. □

Example 2.3. We will apply the algorithm to the pair of numbers 3198 and 299.

$$3198 = 10 \cdot 299 + 208$$

$$299 = 1 \cdot 208 + 91$$

$$208 = 2 \cdot 91 + 26$$

$$91 = 3 \cdot 26 + 13$$

Since $13 \mid 26$, we get the gcd(3198,299)$= 13$.

Once we arrive at the gcd(3198,299) through Euclidean algorithm, we are able to express 13 as a linear combination of 3198 and 299, we can compute:

$$\begin{aligned} 13 &= 91 - 3(26) \\ &= 91 - 3(208 - 2 \cdot 91) \\ &= 7(91) - 3(208) \\ &= 7(299 - 1 \cdot 208) - 3(208) \\ &= 7(299) - 10(208) \\ &= 7(299) - 10(3198 - 10 \cdot 299) \\ &= -10(3198) + 107(299) \end{aligned}$$

Theorem 2.4 (Fermat's Little Theorem)**.** *Let p be a prime. Any integer a satisfies $a^p \equiv a$ mod p, and any integer a not divisible by p satisfies $a^{p-1} \equiv 1$ mod p.*

Proof. This theorem can be proved by multiplying each non-zero equivalence class by a. The collection of resulting classes is just a rearrangement of the original collection. The proof details are on page 20 Proposition I.3.2 [4] . □

Example 2.5. Let $a = 3$, and $p = 11$. $3^{11} = 177147 \equiv 3 \pmod{11}$, and gcd(3,11)=1, we obtain $3^{11-1} = 59049 \equiv 1 \pmod{11}$.

Definition 2.6. Let n be a positive integer. The Euler phi-function $\varphi(n)$ is defined to be the number of positive integers b less than n which are coprime to n: $\varphi(n) = \{0 < b < n \mid gcd(b, n) = 1\}$.

Theorem 2.7 (Euler's Theorem)**.** *If $gcd(a, m) = 1$, then $a^{\varphi(m)} \equiv 1$ mod m.*

Remark 2.8. If $n = p_1^{\alpha_1} \dots p_r^{\alpha_r}$, then $\varphi(n) = p_1^{\alpha_1}(1-\frac{1}{p_1})p_2^{\alpha_2}(1-\frac{1}{p_2}) \dots p_r^{\alpha_r}(1-\frac{1}{p_r}) = n\prod_{p|n}(1-\frac{1}{p})$

This theorem is a generalization of Fermat's Little Theorem and it can be proved by induction. The proof details are on page 22 Proposition I.3.5 [4] .

Example 2.9. We know gcd(5,23)=1. Show that $5^{\varphi(23)} \equiv 1$ mod 23. Use the fact to calculate 5^{10000} (mod 23).

First, by the formula, $\varphi(23) = 23 - 1 = 22$, $5^{\varphi(23)} = 5^{22} = (5^2)^{11} \equiv 2^{11} \pmod{23} = 2048$. Then, we check $2048 \equiv 1 \pmod{23}$. Next, to calculate $5^{10000} \pmod{23}$, we can apply the fact that $5^{22} \equiv 1 \pmod{23}$. $5^{10000} = (5^{22})^{454} \cdot 5^{12} = 1$

$$\begin{aligned} 5^{10000} &= (5^{22})^{454} \cdot 5^{12} \\ &= (1)^{454} \cdot (5^2)^6 \\ &= 1 \cdot (2)^6 \\ &= 64 \\ &\equiv 18(mod\ 23) \end{aligned}$$

Therefore, we can see the power of Euler's theorem in raising residues to large exponents.

Corollary 2.10. *If $ed= 1$ mod $\varphi(n)$, then $a^{ed} = a$ (mod n).*

Proof. Suppose p and q are two primes and $n = pq$. We get $\varphi(n) = (p-1)(q-1)$ from remarks 2.8. So from Euler's Theorem, we gain $a^{\varphi(n)} \equiv 1 \bmod n$ when gcd(a, n)=1. We replace $\varphi(n)$ with $(p-1)(q-1)$ and get $a^{\varphi(n)} = a^{(p-1)(q-1)} \equiv 1 \pmod n$. Next, we multiply $(p-1)(q-1)$ with some integer k, then we get $a^{k(p-1)(q-1)} = [a^{(p-1)(q-1)}]^k \equiv 1^k = 1 \pmod n$. Now we use this result and multiply with an a, and we get $a \cdot a^{k(p-1)(q-1)} = a^{k(p-1)(q-1)+1} \equiv a \cdot 1 = a \bmod n$.

Therefore, if $ed \equiv 1 \pmod{\varphi(n)}$, which means $ed = k\varphi(n) + 1 = k(p-1)(q-1) + 1$, then $a^{ed} \equiv a \pmod n$. We arrived at our goal, if ed= 1 mod $\varphi(n)$, then $a^{ed} = a$ (mod n).

□

3. Some simple cryptosystems

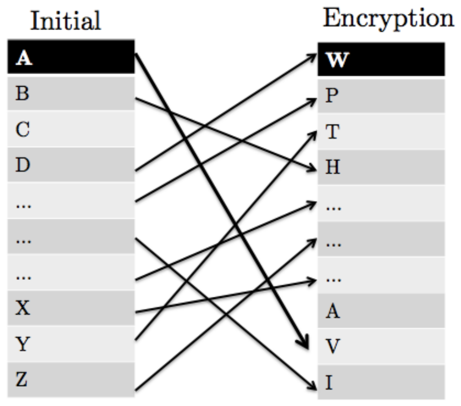
In this section, we will focus on symmetric cryptography. First, we introduce some basic terminology in cryptography. We denote by P the plaintext message that the sender wants to give to the intended recipient and denote by C the ciphertext, which is the disguised message that is being sent through the public unsecured channel. The process of converting plaintext to ciphertext is called enciphering or encryption, and the reverse process is called deciphering or decryption. We associate the process of encryption with a one-to-one function f from plaintext message to ciphertext message, and the reverse decryption process function f^{-1} is the map from ciphering text to plain text.

$$P \underset{\rightarrow}{f} C \underset{\longrightarrow}{f^{-1}} P$$

We establish some ways of converting messages to numbers, for example by associating the alphabet with the residues mod 26 [4]. So from now on, we will consider our plaintexts and ciphertexts to be integers, or perhaps residues mod some large N.

Now we focus on symmetric cryptography. Symmetric cryptography uses the same information to encrypt the plaintext message and decrypt the cipher message. The following is one of the simplest forms of symmetric key cryptography, called the Caesar cipher. Both the sender and recipient have the "symmetric" key b. We encrypt the message P by $C = f(P) \equiv P + b \bmod N$, we can compute $P = f^{-1}(C) \equiv C - b \bmod N$. For instance, say we take $P = 18$ to $C = 26$, in modular $N = 32$, so we substitute P and C in the forms of $C \equiv P + b \bmod N$ and get $26 \equiv 18 + b \pmod{32}$. By solving for key of b, and b=8 and both sender and recipient know this key. Now, when the recipient only gets the ciphertext C from the sender, he can eventually decipher the message by $P = C - b = 26 - 8 \equiv 18 (\text{mod } 32)$. This is correct.

Instead of just "shifting" the letters, we can also assign them arbitrarily to other letters. This is known as a substitution cipher and modeled in the following diagram.



4. Public key

As time went by, people realized the limitations of symmetric cryptography since they always require that both parties keep the information secret and safe [2]. Hence, in 1976, Whitfield Diffie and Martin Hellman invented a new idea of cryptography, which is called public key cryptography [2]. Each user has his own public key K_e (n, e) used for encryption and private key K_d (n, d) used for decryption. This is an asymmetric cryptography since the sender and recipient have different pairs of keys.

The RSA cryptosystem, which was invented by Rivest, Sharmir and Adleman in 1978, is one form of public key cryptography. RSA works based on tremendous prime numbers. The steps are shown in the following:

1. Choose large prime numbers p and q subject to extra conditions, and calculate $n = pq$.
2. Recall $\varphi(n) =(p-1)(q-1)$ when $n = pq$ because p, q are primes.
3. Randomly pick e_A satisfying the property of $1 < e_A < \varphi(n)$ with gcd $(e_A, \varphi(n))$=1.
4. Find d_A such that $d_A e_A \equiv 1(\text{mod } \varphi(\text{n}))$ through the Euclidean algorithm. The reason refers to Corollary 2.10.
5. At the end of the process, the public enciphering key is $k_E = (n, e)$ and deciphering key is $k_D = (n, d)$.

Example 4.1. RSA

1. Say we pick two prime numbers $p = 3863$, $q = 83$, so $n = pq = 320629$.

2. Calculate $\varphi(n) = (p-1)(q-1) = 316684$

3. Pick $e_A = 997$, which satisfies gcd $(e_A, \varphi(n))$=1.

4. We need to find d_A such that $d_A e_A \equiv 1 \pmod{\varphi(n)}$. By applying the extended Euclidean algorithm, we obtain $d_A = 263321$.

5. So the public enciphering key is $k_{E,A} = (320629, 997)$ and the private deciphering key is $k_{D,A} = (320629, 263321)$.

So now we can use these keys to encipher and decipher the messages to check. Say person A wants to send the plaintext message P=5 to the person B. First, A encrypts the message with B's public key $k_{E,A} = (320629, 997)$ and gets the ciphertext C=5^{997} = 145463 (mod 320629), and only sends the ciphertext C=145463 to B. Once B receives the ciphertext C from A. He applies his own private key $k_{D,A} = (320629, 263321)$ to the ciphertext C=145463 and gets 145463^{263321} (mod 320629). After calculation, at the end, he gets the plaintext message $P = 5$, which is the correct plaintext.

Remark 4.2. Given n, p and q should be hard to find. Otherwise, an attacker can find $\varphi(n)$ easily and solve for the private deciphering key d_A since e_A is public.

RSA uses knowledge of public key and private key. In addition, it uses authentication to ensure the security of the sending the message is not compromised.

Authentication When people send the message, we often need to verify the identity of the sender. We often call this the signature. We can construct a digital signature similar to a physical signature while communicating the message. Suppose Alice wants to send message P to Bob. She sends her signature S_A along with her message. Similarly to section 3 3, Let f_A be the enciphering transformation for Alice and f_B be the same for Bob and make them public. First, Alice composes Bob's f_B with her own private key f_A^{-1} to obtain $f_B f_A^{-1}(S_A)$. When Bob gets the message, he first applies his own private key f_B^{-1} to obtain $f_A^{-1}(S_A)$. Then he applies Alice's public enciphering transformation f_A, and obtains S_A. Since he can decipher the signature, he concludes this message was indeed sent by Alice.

Hash functions Hash function, in short, is a one-way function. This is a map that sends plain text to "hash codes", and is very easy to compute, but the inverse will be extremely hard to compute. Hash functions are usually used to log in to any computer or any private account with a password. This has tangential use especially in Section 6: Discrete logarithm.

5. Diffie-Helman key exchange system

RSA is a very important and useful cryptosystem, however, one disadvantage is it requires both public key and private key, which complicates and slows the process of communicating with a large amount of data. In this situation, key exchange is a process to generate symmetric keys from asymmetric keys so that both sender and recipient can use the new key to communicate. Diffie-Hellman key exchange is one of such method.

Definition 5.1 (Primitive root)**.** Let p be prime. We say g is a primitive root of p if the list g, g^2, g^3, …, g^{p-1} (mod p) includes every non-zero residue 1,2,3,…, $p-1$ of p, that is, if g is a multiplicative generator.

Example 5.2. We compute primitive roots modulo 5.

The non-zero residues modulo 5 are 1,2,3,4. We could test each residue at a time: Since 1 to any power of 1 is always 1, we know it is not a primitive root mod 5. In fact, 1 is never a primitive root.

$$2^1 \equiv 2, 2^2 \equiv 4, 2^3 \equiv 3, 2^4 \equiv 1 \dots$$
$$3^1 \equiv 3, 3^2 \equiv 4, 3^3 \equiv 2, 3^4 \equiv 1 \dots$$
$$4^1 \equiv 4, 4^2 \equiv 1, 4^3 \equiv 4, 4^4 \equiv 1 \dots$$

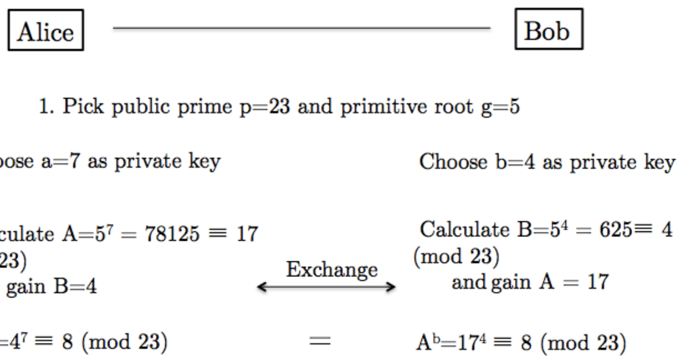
As we notice, 2 and 3 are the primitive roots mod 5, but 4 is not because it does not include residual 2 and 3.

The Diffie-Hellman key exchange works in the following way:

1. Pick a prime number p, and let g be one of the primitive roots of p, and make it public.
2. Person A chooses $a \in$ (2,3,..., $p-2$) as his private key, and person B chooses $b \in$ (2,3,..., $p-2$) as his private key.
3. Person A calculates $A = g^a \bmod p$ and person B calculates $B = g^b \bmod p$ and they exchange their result publicly.
4. Then person A raises result B to his own private key, and obtains $k = B^a = (g^b)^a \bmod p$; while person B does the opposite and gains $k = A^b = (g^a)^b \bmod p$. Even though it seems person A and B are doing different encryption process, but it is indeed the same algorithm.
5. Now they can use k as their private symmetric key to send the message.

Remark 5.3. To make the algorithm hard to break, we use the primitive root g mod the prime number p. In this way, attackers need to search for $p-1$ powers of g.

Example 5.4.



1. Pick public prime p=23 and primitive root g=5

2. Choose a=7 as private key | Choose b=4 as private key

3. Calculate A=5^7 = 78125 ≡ 17 (mod 23) so gain B=4 | Calculate B=5^4 = 625≡ 4 (mod 23) and gain A = 17

4. B^a=4^7 ≡ 8 (mod 23) = A^b=17^4 ≡ 8 (mod 23)

6. Discrete logarithm

Some of the obvious attacks on the algorithms mentioned so far can be described in terms of the "discrete logarithm" problem. In particular if we think of encryption methods as hash functions, their inverses should be hard to compute. Discrete log is roughly an inverse of the functions we have been discussing.

Definition 6.1. If p is a prime, a and b are residuals mod p, then the discrete logarithm of b to the base a is any integer x such that $a^x = b$.

Solving discrete logs is not an easy problem. If we are attempting to use the "brute-force" method of substituting each possible answer at a time, it will be very time consuming.

One way of solving the discrete logarithm problem quicker is the baby step-giant step method. Say we want to compute the discrete log of x in $a^x = b \pmod p$. First we select an integer k such that $k \geqslant \sqrt{p}$. Next, calculate all the a^x from $1 \leqslant x \leqslant k$ and record all values (This is called the baby step). Third, we apply $ba^{(-k)x}$ where $-k$ is the inverse of k until we arrive at a value that was already recorded in the baby step (This is called giant step). More specifically, we terminate when we find m and n such that $a^m \equiv l \pmod p$, and $ba^{-kn} \equiv l \pmod p$. Then we obtain $a^m \equiv ba^{-kn} \pmod p$, which is $a^m a^{kn} \equiv b \pmod p$. Therefore, we arrive at our result $x = m + nk$.

Example 6.2. Solve for x when $7^x \equiv 210 \pmod{359}$. First, we solve $\sqrt{p}$ to find k, $\sqrt{p}$= $\sqrt{359}$ ≈18.95. So we take $k = 19$.

x	1	2	...	...	13	14	...	...	19
7^x	7	49	...	...	309	9	...	...	124
$210(7)^{-19x}$	297	179	...	...	9	...	...	...	...

From the table above, we notice that $7^{14} \equiv 9 \pmod{359}$, and $210(7)^{13(-19)} \equiv 9 \pmod{359}$ also. Therefore, we gain $7^{14} \equiv 210(7)^{13(-19)} \pmod{359}$, which we can rewrite as $7^{14}7^{13(19)} \equiv 210 \pmod{359}$. Hence, $7^{216} \equiv 210 (\text{mod } 359)$. The discrete logarithm is 216.

The discrete logarithm is not being used during the process of encryption or decrytion, but the way of computing discrete logarithms would provide a viable attack on cryptographic protocols. An attacker can use discrete log to directly find private keys so we can get the message easily from Diffie-Helman key exchange. Similarly for RSA, knowing how to compute the discrete logarithm allows attackers to find d_A and then obtain the plaintext message through $(x^{e_A})^{d_A} \pmod{N_A}$, given x^{e_A} and N_A.

Acknowledgments

It is a pleasure to thank my mentor, Ronno Das, for guiding me to this interesting topic. I thank the graduate student: Boming Jia, for showing me a different proof on Fermat's Little Theorem. I also appreciate Huygens Ravelomanana for comments, and the participants Cecelia Higgins and Tejasi Bhatnagar for helping me to understand the binomial coefficient forms and give me comments.

References

[1] History of Encryption. Version 2. SANS Institute. 2001. Retrieve from: https://www.sans.org/reading-room/whitepapers/vpns/history-encryption-730.
[2] L. Maurits. Public Key Cryptography Using Discrete Logarithms in Finite Fields: Algorithms, Efficient Implementation and Attacks. The University of Adelaide.
[3] Evan Dummit. Cryptography (part 3): Discrete Logarithms in Cryptography. v. 1.01. 2016.
[4] Neil Koblitz. A Course in Number Theory and Cryptography 2nd Edition. Springer-Verlag. New York, Inc. 1994, 1987