Lecture 8: Cryptography

Rajat Mittal*

IIT Kanpur

Through out the history of mankind, transmitting messages securely has been a difficult and challenging task. There are two main reasons: the message can accumulate error over transmission or it can be intercepted by an opponent or adversary. Both directions, making and breaking such protocols to transmit messages, have been an important topic of study, specially in the context of war. The computing great, Alan Turing, became famous because he was instrumental in breaking such a machine, called *Enigma*, from German side in second world war.

In *cryptography* we look at the latter case, where the message needs to be transmitted in the presence of an adversary. The challenge is to design a protocol so that the message is not revealed even if the adversary has an access to the transmission line. The case of noise/errors is handled by error correcting codes and will be studied later in this course.

Both these fields have borrowed heavily from number theory. We will use the concepts learned so far to make secure message transmission protocol.

1 Basic cryptography

The process of converting a message into a secret code/text such that the adversary can't recover the message from the secret code (or text) is called *encryption*. As mentioned before, this has been an important problem historically. Our goal is to devise protocols to transfer these secret messages and also come up with some sort of guarantee (or evidence) that this protocols are secure and do not reveal the message.

The standard (at least traditionally) way of encryption is known as *private key encryption*. Here, the two parties, Alice and Bob, want to securely communicate and hide their messages from an adversary/ eavesdropper (we call her Eve). For this purpose, they meet beforehand and agree on a secret key. This secret key is used to encrypt the message at the time of transmission. We assume that this secret key is not known to Eve.

Exercise 1. Why don't they exchange the message when they meet to agree on the secret key?

We will assume that a message M is an element of Σ^n, where Σ is the alphabet and n is the length of the message. For example, Σ could be the letters of English alphabet or simple the binary set $\{0, 1\}$.

A private key encryption protocol consists of two algorithms, encryption algorithm ($\mathcal{E}$) and decryption algorithm ($\mathcal{D}$). The encryption algorithm $\mathcal{E}$ will convert the message M into a ciphertext C using the secret key S. Alice will send C on the transmission line to Bob. Bob will use the secret key S and ciphertext C with the decryption algorithm $\mathcal{D}$ to recover M.

These algorithms, $\mathcal{E}$ and $\mathcal{D}$ should be *easy* to perform if the secret key is known. We assume that the description of the algorithms are known to Eve, but the secret key is hidden. Hence, Eve can't apply $\mathcal{D}$ (even if she intercepts C) to recover M.

Let us start with some very simple private key encryption schemes.

Substitution cipher: This is one of the most basic encryption technique. Suppose the message M is a string of length n over the alphabet Σ. The substitution cipher is specified by a permutation (secret key) $\pi : \Sigma \to \Sigma$. Alice replace every letter of the message with the permuted letter.

Exercise 2. To what set does the ciphertext belong?

* Thanks to Nitin Saxena for his notes from the previous iteration of the course.

If the message M is $a_1 \cdots a_n$; Alice sends $\pi(a_1) \cdots \pi(a_n)$ to Bob (this will be the encryption algorithm). Bob knows π and hence π^{-1}. He can easily apply π^{-1} (decryption algorithm) to obtain the original message.

To take an example, let Σ be the English alphabet and let π be the permutation which takes every letter to next letter (A to B, B to C ... Z to A). The message CRYPTO will be encrypted as DSZQUP.

Exercise 3. If Bob receives HPPE KPC, what was the message sent by Alice.

Substitution cipher is easy to break if Eve sees a lot of messages encrypted using the same permutation π. It is known that some letters occur much more than other letters in English language. So a simple frequency analysis will reveal the secret key.

One time pad: Another well known example of a private key encryption scheme is known as *one time pad*. Suppose the message M is an element of $\{0,1\}^n$. For a one time pad, the secret key S is another string in $\{0,1\}^n$.

Alice simply send the XOR of M and S to Bob. For example, if M is 0110 and the secret key decided is 1010, Alice will send the ciphertext 1100 to Bob. Bob will also XOR the secret key S to ciphertext to obtain the message.

The secret key is not very secure if it is used multiple times. For example, suppose Alice encrypts two messages M and M' using the same secret key S. Eve can then XOR C and C' (corresponding ciphertexts) to get the XOR of M and M'.

Exercise 4. Why is the XOR of C and C' same as the XOR of M and M'?

The XOR of M and M' does not tell us M and M' but it still gives lot of information about these messages. It tell us at what points are the message same and at what points they are distinct.

Block cipher: A block cipher is not really a private key encryption system but a primitive (a subroutine) used to design a private key encryption scheme. A block cipher is specified by a function $F : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$.

The function should be interpreted as $C = F(S, M)$, taking secret key and message as input and outputting a ciphertext. Notice that block ciphers are specified on a fixed length. One way to use block ciphers is to divide your input into blocks of required length and apply the block cipher to each of the blocks to obtain the ciphertext. Block ciphers generally use the tricks like permutation and substitution to improve the security.

The old standard for block ciphers is known as DES and the new one is known as AES. They are widely used, many banking application and softwares rely on them and their variants. Readers are encouraged to read the descriptions of DES and AES from other sources.

2 Public key cryptography

In the previous section, we looked at private key cryptography. In this section, we will look at *public key cryptography*. We will look at the problem of public key cryptography and setting in this section and solution in the next section.

One of the issue with private key cryptography is the necessity of key exchange. Alice and Bob need to meet and decide upon the key in advance to make our protocols work. Is it possible for Alice to send a message to Bob without having a key exchange?

At the first glance, the problem seems impossible to solve. Since there is no key exchange, it looks like Bob and Eve should have equal power to decipher the message.

Exercise 5. Can you think of a solution?

The idea is to use two keys instead of one. To start, Bob fixes a key pair, S - a secret key and P - a public key. He announces P to everyone (even to Eve) and keeps S as the secret key.

If Alice wants to send a message M, she uses the encryption algorithm $E(M, P)$ to get the ciphertext C. When Bob receives the ciphertext C, he decrypts it by applying the decryption algorithm $D(S, C)$. The security requirement is same as before, Eve should not be able to get message M from the ciphertext C.

Notice that the encryption of message can be done by anyone in the data transmission network. If Eve can send messages on the network (and not just intercept them), then she can potentially send confusing/erroneous messages to Bob (she can pretend to be Alice).

In the next section, we will see how to generate such a key pair, encryption and decryption algorithm.

2.1 RSA

Diffie and Hellman were the first researchers to propose public key cryptography using the existence of *trapdoor functions* in 1976. In 1977, Rivest, Shamir and Adelman constructed a trapdoor function called *RSA*. Combined, that gave us a public key cryptosystem. Arguably, RSA is the most famous example of public key cryptography.

Let us take a look at the RSA protocol first, we will discuss the efficiency and security later. Remember, a public key cryptosystem consists of three steps: key generation, encryption and decryption algorithm.

Key generation: To generate the key pair, Bob picks two large prime integers p and q and sets $n := pq$. He also chooses an integer e, such that, $\gcd(e, \phi(n)) = 1$. Here, $\phi(n)$ is the Euler's totient function of n. He also computes the inverse of e modulo $\phi(n)$ using extended Euclidean algorithm, call it d. The key consists of two parts,

- public key P: (n, e),
- secret key S: d, $p, q, \phi(n)$ also need to be kept secret so that d is not disclosed.

Encryption: Bob announces the public key (n, e) on the entire network (it is known to everyone). To encrypt message M (think of it as a number between 0 and $n-1$), Alice calculates the ciphertext $C = M^e \mod n$. This ciphertext is sent to Bob on the transmission line (Eve can access C).

Decryption: To decrypt the ciphertext, Bob uses the secret key and calculates $C^d \mod n$. Using Euler's theorem, $C^d = M^{ed} = M.M^{k.\phi(n)} \mod n = M$, where k is some integer.

Clearly, the protocol works, Alice can encrypt the message M and Bob will also be able to decrypt the message from ciphertext C because of Euler's theorem. What about the efficiency and security of the protocol? In other words, are all the three steps computationally efficient and can Eve figure out M from C?

Efficiency For the question of efficiency, we need to accomplish every step in time polynomial of logarithm of n (also called polylog(n)). Encryption and decryption require us to compute exponent (say b) of a number (say a) modulo a number (say m) fast. This is called *modular exponentiation* and is accomplished by *repeated squaring*.

We calculate $a, a^2 \mod m, a^4 \mod m, \cdots, a^{2^{\lfloor b \rfloor}} \mod m$ by squaring repeatedly. These can be combined to get any exponent.

Exercise 6. How do we obtain $a^b \mod m$?

Use binary expansion of b.

For the key generation, we pick two big numbers and check if they are prime. We have a polynomial time primality testing algorithm now. There are faster randomized algorithms too to check primality.

Exercise 7. What is $\phi(n)$?

$(p-1)(q-1)$

A number e relatively prime to $\phi(n)$ can be picked randomly. The inverse of e modulo $\phi(n)$ is computed using extended Euclidean algorithm we saw before.

Hence, all steps of RSA protocol are efficient.

Security As you might have heard before, RSA is based on hardness of factoring. Hardness of factoring means, it is computationally hard (according to present knowledge) to factor a sufficiently large number into prime factors.

Notice that if we can factor large numbers, specifically n, then $\phi(n)$ and d can be calculated revealing the message M.

To decrypt C, we need to find the solution of equation $x^e = C \mod n$. As mentioned above, factorization of n is one way to find x. There might be other options, but at this point, factorization seems to be the best bet.

Since we believe that RSA is hard and finding x from equation $C = x^e \mod n$ requires factoring n, RSA seems to be secure.

Note 1. The textbook RSA described in the notes is not very secure in practice. Implementers use various techniques like padding the message, choosing p and q carefully to make it more secure.

For particular choices of key, there are specialized attacks which can break RSA. Let us take a look at few of them.

- e is small: If M^e is smaller than n (actual value and not the reduced one), we can use usual e-th root finding.

  Exercise 8. Given $a^e < n$, how can you find a in time polylog(n).

  Use binary search over $[n]$

- p and q are close: Notice that if $n = pq$ then $n = ((p+q)/2)^2 - ((p-q)/2)^2$. If p and q are close, $p - q$ is small. An attacker can try all small values of b and check if $n + b^2$ is a perfect square.

2.2 Use of Chinese remaindering in RSA

To make the computations faster, e is chosen to be not big in RSA (though very small e could create problem as mentioned in the previous section). In this case, d turns out to be big. So, calculating $C^d \mod n$ could be really slow for Bob. Bob might receive multiple messages and such big modular exponentiation can be slow in practice.

We can break this computation into smaller parts using Chinese remainder theorem. The idea is to do calculations modulo p and q separately and then combine them to get the result.

So, Bob needs to compute two quantities, $C_1 = C^d \mod p$ and $C_2 = C^d \mod q$. He can combine these two using Chinese remaindering to get $C^d \mod n$. Notice that p and q will have only half the digits of n and hence calculations are easier to perform under these moduli.

The calculations of C_1 and C_2 can be further simplified. Since d is part of the private key, Bob can compute $d_1 = d \mod (p-1)$ and $d_2 = d \mod (q-1)$ in advance.

Exercise 9. Why do we compute $d \mod (p-1)$ and not $d \mod p$?

p is in the exponent, we use Fermat's theorem

Now, Bob can calculate $C_1 = C^{d_1} \mod p$ and $C_2 = C^{d_2} \mod q$. Let p^{-1} be the inverse of p modulo q which can again be precomputed. Using CRT,

$$M = C^d \mod n = C_1 + p(p^{-1}(C_2 - C_1) \mod q).$$

Exercise 10. Show that M is C_1 modulo p and C_2 modulo q.

Notice that d_1, d_2, p^{-1} can be precomputed using the knowledge of the private key. If Bob needs to decrypt many messages, this will make the computation faster.

References

1. K. H. Rosen. Discrete Mathematics and Its Applications. *McGraw-Hill*, 1999.
2. N. L. Biggs. Discrete Mathematics. *Oxford University Press*, 2003.