Network Security

Professor Gaurav S. Kasbekar
Department of Electrical Engineering
Indian Institute of Technology, Bombay

Week - 02
Lecture - 15
Principle of Cryptography: Part 5

Hello, in the previous lecture, we discussed the public key encryption scheme, RSA. In this lecture, we will discuss another public key scheme, namely Diffie-Hellman. So, the Diffie-Hellman algorithm is a public key cryptosystem. It was invented before RSA. Unlike RSA, it cannot be used for encryption or to create digital signatures.

However, the utility of Diffie-Hellman is that it allows two individuals, say Alice and Bob, to agree on a shared secret key, even though all the messages exchanged by Alice and Bob over a network can be overheard by intruders. So, Alice and Bob are connected by a network, and any message sent by Alice or Bob can be overheard by an intruder. Despite this, the Diffie-Hellman algorithm allows Alice and Bob to agree on a shared secret key. So, the only communication that Alice and Bob can do is over the channel, and this channel can be intercepted by intruders. We'll now discuss Diffie-Hellman, which allows Alice and Bob to agree on a shared secret key, in this context.

Once Alice and Bob have agreed upon a shared key, it can be used for communication using symmetric key cryptography. So, suppose the shared key that Alice and Bob have agreed upon is K_S, then this can be used as the symmetric key in a symmetric-key cipher, such as DES or AES. So, the Diffie-Hellman algorithm operates as follows. There are numbers p and g, where p is a large prime number. For example, we might require 2048 bits to represent p. g is a number less than p. p and g can be publicly known.

They don't have to be kept secret. An example is Alice might choose p and g, and then send them over the channel to Bob. So, if this means is used to share p and g, then p and g may be overheard by intruders. After agreeing upon p and g, each of Alice and Bob then independently chooses a large number less than p at random and keeps it secret. So, Alice chooses a secret number less than p, and Bob independently chooses another secret number less than p. Let S_A denote the secret number chosen by Alice, and let S_B denote the secret number chosen by Bob.

So, we have that $S_A < p$ and $S_B < p$. S_A and S_B have to be large; otherwise, if they were small, the intruder could just try out numbers starting from 1: 1, 2, 3, 4, and so on. So, just by trying a few numbers, the intruder might be able to guess S_A or S_B. So, hence, S_A and S_B are large, but they must be less than p. So, the scheme works as follows. Alice computes $T_A = g^{S_A} \, mod \, p$, and Bob computes $T_B = g^{S_B} \, mod \, p$. Then, Alice sends T_A to Bob, and Bob sends T_B to Alice.

- Alice computes $T_A = g^{S_A} \bmod p$; Bob computes $T_B = g^{S_B} \bmod p$
- Alice sends T_A to Bob and Bob sends T_B to Alice
- Alice computes ${T_B}^{S_A} \bmod p$ and Bob computes ${T_A}^{S_B} \bmod p$
- **Theorem**: ${T_B}^{S_A} \bmod p = {T_A}^{S_B} \bmod p$
- **Proof**:
  - LHS = $(g^{S_B} \bmod p)^{S_A} \bmod p = (g^{S_B})^{S_A} \bmod p = g^{S_A S_B} \bmod p$
  - Similarly, RHS = $g^{S_A S_B} \bmod p$

Now, Alice has received T_B from Bob. Alice computes $T_B^{S_A} \, mod \, p$, and similarly, Bob computes $T_A^{S_B} \, mod \, p$ using T_A, which has been received from Alice, and S_B, which is Bob's secret value. Now, the claim is that $T_B^{S_A} \, mod \, p = T_A^{S_B} \, mod \, p$. If we prove this, then it will follow that the value computed by Alice and the value computed by Bob are the same. So, Alice and Bob have agreed upon the secret key, which is equal to $T_B^{S_A} \, mod \, p$, and it is the same as $T_A^{S_B} \, mod \, p$. This theorem is quite straightforward to prove. The LHS can be written as follows.

Just by substituting $T_B = g^{S_B} \, mod \, p$, we get that the LHS is $(g^{S_B} \, mod \, p)^{S_A} \, mod \, p$. Now, using the properties of modulo-p arithmetic, we get that this is the same as $(g^{S_B})^{S_A} \, mod \, p$, which is in turn the same as $g^{S_A S_B} \, mod \, p$. Now, notice that this is symmetric in A and B. So, similarly, the RHS is also equal to the same quantity $g^{S_A S_B} \, mod \, p$. Hence, LHS equals RHS, which proves this theorem. Thus, Alice and Bob have agreed upon the same number $g^{S_A S_B} \, mod \, p$, which is a shared key. Subsequently, Alice and Bob can use this shared key as the symmetric key in a symmetric-key cipher, such as DES or AES. So, this is Diffie-Hellman's algorithm. S_A is known as Alice's private key, and T_A is known as Alice's public key.

Similarly, S_B and T_B are known as Bob's private and public key, respectively. So, as the name suggests, S_A and S_B are secret; hence, they are known as private keys, and T_A and T_B

are sent over the channel, where they can be sniffed by intruders. T_A and T_B are hence public. Here's an example. Let $p = 23$ and $g = 5$.

So, p is a prime number. So, let $S_A = 4$ and $S_B = 3$. So, in this example, the values of p, S_A, and S_B are very small. In practice, they are obviously much larger. Now, $T_A = g^{S_A} \ mod \ p$ p by definition, so it is $5^4 \ mod \ 23 = 4$.

That turns out to be 4. Similarly, $T_B = g^{S_B} \ mod \ p$, and it turns out to be, so if you calculate $5^3 \ mod \ 23 = 10$. Now, $T_A^{S_B} \ mod \ p$, which is the value computed by Bob, so it is $4^3 \ mod \ 23 = 18$, and that comes out to be 18. Similarly, $T_B^{S_A} \ mod \ p = 10^4 \ mod \ 23 = 18$, and it comes out to be 18. So, in this example, Alice and Bob have agreed upon the shared key 18.

So, notice that Alice and Bob have both obtained the same number, 18, which is the shared symmetric key. So, now let's discuss the security of Diffie-Hellman. And recall that g and p are publicly known in general, and T_A and T_B are sent over the channel, and hence they can be sniffed by intruders. So, even though an intruder may know g, p, and $T_A = g^{S_A} \ mod \ p$, and $T_B = g^{S_B} \ mod \ p$ p, it is computationally infeasible for the intruder to calculate $g^{S_A S_B} \ mod \ p$. So, let's understand why this is true. So if g, p, and $g^{S_A} \ mod \ p$ are known, then the problem of finding S_A using these quantities is known as "Discrete Logarithm Problem".

The reason for this nomenclature is the following. Consider the equation $g^{S_A} = T_A$ So, if TA is known and g is known, then we can find S_A by just taking logarithms on both sides. $\log_g(\cdot)$; if we take $\log_g(\cdot)$ on both sides, then we get S_A from g and T_A.

So, this is the ordinary logarithm problem.

Now, in this problem, we have $g^{S_A} \ mod \ p$. So, because of the presence of this $mod \ p$, this is the discrete logarithm problem. So, the discrete logarithm problem is computationally hard to solve. So, if the intruder could compute discrete logarithms efficiently, then he or she could find the secret keys S_A, S_B, and hence once S_A and S_B are found, then it is straightforward to find $g^{S_A S_B} \ mod \ p$. But no efficient algorithm is known for finding discrete logarithms. So, the security of Diffie-Hellman rests on this fact that there is no efficient algorithm known for finding discrete logarithms.

So, recall that the security of RSA was based on the fact that there is no efficient algorithm known for factoring the number n into its prime factors p and q. In contrast, the security of Diffie-Hellman is based on the fact that there is no efficient algorithm known for finding

discrete logarithms. Now, there are some additional requirements on p and g for the protocol to be secure. The security of Diffie-Hellman is compromised unless p and g satisfy the following additional properties. One property is that $g^x \ mod \ p \neq 1$, unless x is a multiple of (p-1).

1) $g^x \bmod p$ must not equal 1, unless x is a multiple of $(p-1)$
   - Reason:
     - If $g^x \bmod p = 1$ for a small value of x, then to find S_A using $g^{S_A} \bmod p$ by brute force, an intruder only needs to try out a small number of values of S_A
     - E.g., if $g^3 \bmod p = 1$, then $g^4 \bmod p = g \bmod p$, $g^5 \bmod p = g^2 \bmod p$, $g^6 \bmod p = 1$, $g^7 \bmod p = g \bmod p$, etc.
2) $\frac{(p-1)}{2}$ must also be prime

Now, first notice that if x is a multiple of (p-1), then we get that $g^x \ mod \ p = g^{y\,(p-1)} \ mod \ p$, (let's call it y times (p-1); since x is a multiple of (p-1)). So, this can be written as $\left(g^{(p-1)} \ mod \ p\right)^y \ mod \ p$. So, by the corollary to Fermat's Little Theorem and Bezout's Identity, which we discussed in the previous class, $g^{(p-1)} \ mod \ p = 1$, so hence, this is equal to 1. So, $g^x \ mod \ p = 1$, clearly whenever x is a multiple of (p-1), but we must have the property that $g^x \ mod \ p \ \neq 1$ unless x is a multiple of (p-1). So, the reason is this: if $g^x \ mod \ p = 1$ for a small value of x, that is, for example, a value of x much smaller than (p-1); if $g^x \ mod \ p = 1$ for a value of x much smaller than (p-1), then to find S_A using $g^{S_A} \ mod \ p$ by brute force, an intruder would just need to try out a small number of values of S_A.

So, the intruder might start from $S_A = 1$, then $S_A = 2$, $S_A = 3$, and so on. So, if $g^x \ mod \ p = 1$ for a small value, then the intruder would quickly find an S_A, which satisfies $g^{S_A} \ mod \ p = T_A$. For example, suppose $g^3 \ mod \ p = 1$. So, $x = 3$ satisfies this property that $g^x \ mod \ p = 1$. Then, $g^4 \ mod \ p = g \ mod \ p$, $g^5 \ mod \ p = g^2 \ mod \ p$, $g^6 \ mod \ p = 1$, $g^7 \ mod \ p = g \ mod \ p$, and so on.

So, these values keep on repeating in a cycle, and the period of the cycle is very small. So, in this case, the intruder just has to try out a few values of S_A to get a value of S_A that satisfies the equation $g^{S_A} \ mod \ p = T_A$. Since this expression $g^{S_A} \ mod \ p$ has only a few distinct values, it is straightforward to find an S_A that satisfies $g^{S_A} \ mod \ p = T_A$, which is the target. Another property that must be satisfied is that $\frac{(p-1)}{2}$ must also be a prime.

So, apart from p, $\frac{(p-1)}{2}$ must also be a prime.

A prime that satisfies this property is called a "safe prime". For example, consider $p = 7$. $\frac{(7-1)}{2} = 3$, which is also prime. So, 7 is a safe prime.

Consider $p = 11$. $\frac{(11-1)}{2} = 5$, which is also a prime. So, 11 is also a safe prime.

Now, consider $p = 13$. $\frac{(13-1)}{2} = \frac{12}{2} = 6$, which is not a prime. So, 13 is not a safe prime.

So, we must always choose p such that it is a safe prime. The reason is that if this property is not satisfied, then it may be possible to efficiently compute discrete algorithms using an algorithm known as "Pohlig-Hellman algorithm". We omit the details, but for our purposes, we only need to know that we choose p such that p is a safe prime. Now, there is an attack on Diffie-Hellman called "man-in-the-middle" attack. The man-in-the-middle attack is when there is an intruder, say Trudy, who intercepts the communication channel between Alice and Bob and can modify messages before sending them to the recipient.

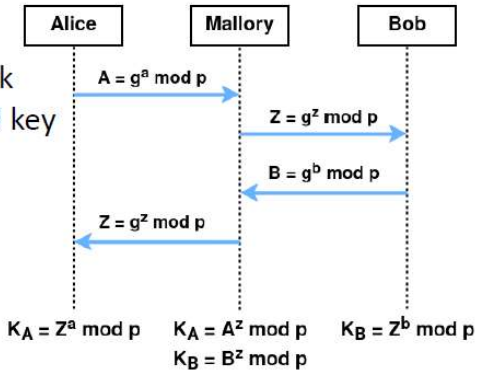
So, when Alice sends a message to Bob, Trudy can intercept that message, modify it, and then send it to Bob. Similarly, when Bob sends a message to Alice, Trudy can modify the message and then send it to Alice. Such an attacker is known as a man-in-the-middle attacker. So, the Diffie-Hellman algorithm is insecure when there can be an intruder called man-in-the-middle on the channel between Alice and Bob, who can modify messages sent from Alice to Bob or vice versa. Suppose again that g and p are publicly known.

When Alice receives T_B, there is no way for her to find out whether it was really sent by Bob or not. And even if it was sent by Bob, was it modified during transit from Bob to Alice? Trudy might just initiate communication and send a value to Alice, and Alice has no way to find out that it was sent by Trudy and not by Bob. Also, if Bob sends a value to Alice, Trudy might modify it before sending it to Alice. So, hence, it is true that when Alice receives T_B, there is no way for her to find out whether it was really sent by Bob, and whether it was modified during transit.

So, there is similar uncertainty when Bob receives T_A. So, Bob doesn't know whether it was indeed sent by Alice and whether it was modified during transit. Now, this illustrates a man-in-the-middle attack, where Mallory is the intruder who is the man-in-the-middle attacker. So, in this figure, Alice sends the public key $A = g^a \ mod \ p$ towards Bob. Now, Mallory, who is the man-in-the-middle attacker, intercepts this message, and instead of relaying it to Bob (what Mallory does is); she chooses her own private key z, and computes

the corresponding public key $Z = g^z \ mod \ p$, and then sends Z to Bob instead of sending A.

- At end of attack:
  - ❑ Alice has secret shared key $g^{az} \bmod p$ with Mallory
  - ❑ Bob has secret shared key $g^{bz} \bmod p$ with Mallory
- However, Alice and Bob think that they have secret shared key with each other



Similarly, when Bob sends his public key, $B = g^b \ mod \ p$ towards Alice, Mallory intercepts that message. Mallory does not send B to Alice. Instead, Mallory sends $Z = g^z \ mod \ p$ to Alice. Now, after performing the computations of Diffie-Hellman, Alice computes $K_A = Z^a \ mod \ p$, and Bob computes $K_B = Z^b \ mod \ p$. And Mallory computes $K_A = A^z \ mod \ p$, and $K_B = B^z \ mod \ p$. Notice that this K_A is the same as this K_A, and this K_B is the same as this K_B, by the proof that we earlier discussed.

So, what has essentially happened is that Alice and Mallory have agreed on this key K_A, and Mallory and Bob have agreed on the key K_B. So, at the end of the attack, Alice has the secret shared key $g^{az} \ mod \ p$ with Mallory, that is the same as K_A, and Bob has the secret shared key $g^{bz} \ mod \ p$. But Alice and Bob think that they have a secret shared key with each other. So, this is what this attack has done. So, it leads Alice and Bob to think that they have a secret shared key in common, whereas each one of them has a secret shared key with Mallory.

So, after the intruder has performed this man-in-the-middle attack, when Alice sends an encrypted message to Bob, the intruder Mallory can read it and modify it before forwarding it to Bob. So, the intruder decrypts the encrypted message by using the shared key between Mallory and Alice, then modifies the message, and then re-encrypts it using the shared key between Mallory and Bob. Similarly, the intruder can read and modify messages that are sent from Bob to Alice. Hence, the form of the Diffie-Hellman algorithm that we discussed above is only secure against so-called “passive attacks”, in which the intruder just watches

messages being transmitted between Alice and Bob, but doesn't modify any messages being transmitted. So, if the attacker is active, wherein the attacker can modify messages sent between Alice and Bob, then this Diffie-Hellman algorithm is not secure.

So, let's discuss ways to modify the Diffie-Hellman algorithm so that it can defend against the man-in-the-middle attack. So, as a first attempt, suppose after completing the Diffie-Hellman algorithm, Alice encrypts and transmits the established shared key to Bob to prove that she is indeed Alice. So, the shared key is g to the power SA SB mod P. Suppose after completing the Diffie-Hellman algorithm, Alice encrypts and transmits the established shared key, which is this one, to Bob to prove that she is indeed Alice. So, will Bob be able to detect the man-in-the-middle attack?

No, because the intruder can encrypt and send the shared key established between the intruder and Bob to Bob. Bob thinks that he has a shared key with Alice, but actually Bob has a shared key with Mallory, who is the intruder. So, the intruder can just break this scheme by encrypting and sending the shared key established between the intruder and Bob to Bob. So, this scheme doesn't defend against a man-in-the-middle attack. Another possible scheme for defending against a man-in-the-middle attack is the following.

Suppose p and g are public. Recall that S_B is Bob's private key and T_B is Bob's public key. In this scheme, the public key of every user is stored in a database like a telephone directory. When Alice wants to establish a secret key with Bob, Bob doesn't have to send his public key to Alice. Instead, Alice just looks up the public key T_B of Bob from the database, like the telephone directory, and computes $T_B^{S_A} \ mod \ p$. Similarly, Bob looks up Alice's public key T_A from this database and then computes $T_A^{S_B} \ mod \ p$.

- When Alice wants to establish a secret key with Bob, she just looks up T_B and computes ${T_B}^{S_A} \bmod p$; Bob looks up T_A and computes ${T_A}^{S_B} \bmod p$
- Recall that ${T_B}^{S_A} \bmod p = {T_A}^{S_B} \bmod p$; thus, Alice and Bob have agreed upon this shared secret key

Recall that $T_B^{S_A} \ mod \ p = T_A^{S_B} \ mod \ p$. So, Alice and Bob have agreed upon this shared secret key. So, does this defense work? So, it does work under the assumption that Alice can securely connect with the database and obtain Bob's public key, and similarly, Bob can securely connect to the database and obtain Alice's public key. So, under these conditions, this scheme works. This scheme works because Alice is able to securely get

Bob's public key, and Bob is able to securely get Alice's public key, and once this has been done, they agree upon a shared secret key, which is this.

The crucial question is, how do we implement the database in which the public keys are stored? So, later on, we'll study how this database, in which the public keys are stored, can be implemented. This is called "Public Key Infrastructure". So, this database needs to be such that users, Alice and Bob, can connect to the database and securely obtain the public keys of other users. So, later on, we'll discuss how such a database can be implemented.

Such a database is a part of what is known as public key infrastructure. So, we'll discuss public key infrastructure later on. This concludes our discussion of the Diffie-Hellman algorithm. With this, we have concluded our discussion of cryptography. In the next lecture, we will discuss message integrity.

Thank you.