PUBLIC KEY INFRASTRUCTURE

Up to now we have seen a model in which Alice and Bob share the same secret $K_{A,B}$ and it must be secretly generated and exchanged *prior* to using the unsecure channel. Then, in 1976, *Diffie* and *Hellman* came up with a rivolutionary idea in which it is possible to achieve confidentiality and authentication without exchanging secrets in advance. So the *Public Key Infrastructure* was born. It is a model that uses asymmetric encryption, where each party has two keys: K_E and K_D. These keys are asymmetric in the sense that once encrypted with one of the keys (no matter which) then the cipher can be decrypted only with the other key. What the public key model states then is: choose one of the keys, and make it public, keep the other one secret. Now, in order to achieve confidentiality, the keys have to be used in the following way: suppose Bob and Alice have their pair of keys, Bob has $K_{B,S}$ and $K_{B,P}$ and Alice has $K_{A,S}$ and $K_{A,P}$ where S stands for *secret* and P stands for *public*. If Alice has to send a *confidential* message to Bob, she encrypts the message with $K_{B,P}$ in such a way that the message can be decrypted only by $K_{B,S}$ which only Bob knows.

The peculiarity of Public Key Infrastructure is that the keys can also be used in the reversed order, achieving authentication. Imagine Alice encrypts a message with her secret key $K_{A,S}$, this means that Bob will be able to decrypt it with Alice's public key $K_{A,P}$, do we have confidentiality here? No confidentiality, everyone can decrypt the message. But in this way Bob is sure that it was Alice for sure to encrypt that message, because only Alice could do that by means of her *private key*, and therefore with this pattern we achieve authentication. This is the concept of non repudiation (Bob cannot deny the authenticity of the message, it is clear that the original source was Alice).

Another practical advantage that comes with this infrastructure is that, imagine there is a group of people that need to talk secretly, with the previous approach every pair of parties had to share a shared secret, arriving at a point where we have a quadratic number of keys so n^2, with this approach there are just 2 keys for each party so at the end we have a linear number of keys, 2n.

The main problem is implementing such a cryptosystem where keys are asymmetric.

One Way Function

One way functions are kinds of functions for which given the input it is easy to compute the output, but given a possible output of the function, it is hard (computationally hard) to retrieve the input that generated that output. One example of such a function is the multiplication: given two numbers *p* and *q*, compute *N=pq*. We understand that it is pretty easy to compute the product, while having N it is very hard to factorize it in order to get *p* and *q*.

The mathematical definition of One Way Function is the following:

A function $f : \{0,1\}^* \rightarrow \{0,1\}^*$ is one-way if f can be computed in poly-time, but for every poly-time algorithm A and for every polynomial p(n) with n sufficiently large:

$$Pr\left[f\left(A(f(x))\right) = f(x)\right] < \frac{1}{p(n)}$$

It means that, given as input to A the outcome of the one way function on the input x (i.e. f(x)), the probability that A finds the correct x s.t. the one way function gave that outcome given that x as input is very small.

The existence of One Way Functions has not been prooved yet, if a One Way Function exists then P≠NP.

*A very typical theoretical question also I may ask you in the exam: can we consider **SHA-2** as a one way function? Answer is **Yes**. Since now we are introducing the concept that one way functions they seem good to compute encryptions, next question is ok if SHA-2 is a good candidate for one way function, can we use it to make encryption? No, because it's not a one-to-one, not injective! If you don't know the key the ciphertext is hard to be inverted to find the plaintext, theoretically speaking you can **brute force**, very hard.*

Discrete Log

Let G be a finite cyclic group with n elements and g a generator of G. Let y be any element in G, then it can be written as $y = g^x$, for some integer x. Let $y = g^x$ and x the minimal non negative integer satisfying the equation. Then x is called ***discrete log*** of y to base g.

In the Z_p group, calculating $y = g^x \ mod \ p$ is easy and requires $O(log\ x)$ steps, but retrieve the discrete log, that is x, given g, y and p is believed to be a ONE WAY FUNCTION. Therefore Discrete Log is a good candidate as a One Way Function.

Diffie-Hellman Key Exchange

The goal is to have the possibility that two parties, A and B, who do not share any secret information i.e. did not exchange any secret in advance, could perform a protocol in order to derive the same shared key, without exchanging in clear the key. We require that *Eve*, the attacker, can not obtain the shared secret key having limited computational resources. The protocol came up thanks to Diffie and Hellman again. A common practice is to use such a protocol when two parties need a *session key*, i.e. a shared secret key that has to be used for a confidential conversation just for a certain period (minutes, hours, few days) and then the key will be dropped. Here is the descriptio of *Diffie-Hellman protocol*:

Public parameters: a prime p, and a number g (possibly a generator of Z_p^*. Note: it could be better if p was a *safe prime*, i.e. $p = 2q + 1$ where q is also prime.

Alice chooses a at random where $a \in [1, p-1]$, computes $A = g^a \ mod \ p$ and sends to Bob (g,p,A)

Bob chooses b at random where $b \in [1, p-1]$, computes $B = g^b \ mod \ p$ and sends to Alice B.

Alice chooses as key $K_A = B^a mod\ p = (g^b)^a \ mod \ p = g^{ba} \ mod \ p$

Bob chooses as key $K_B = A^b \ mod \ p = (g^a)^b \ mod \ p = g^{ab} \ mod \ p$

At the end we can understand that they choose the same key!

Of course the Diffie-Hellman protocol can be performed even with multiple parties, what changes is that the number of exchanged messages increases and the sequence of exchanges is slightly different.

For what concerns the security of this protocol, an attacker should retrieve a from (g,p,A) and b from (g,p,B). This challenge is as hard as finding a Discrete Log in Z_p^*.

Another important property of Diffie-Hellman Key Exchange is the perfect forward secrecy, that is the property for which if the attacker discovers secret informations about a previous session of message exchanging, he cannot be able to retrieve any information for the next session of message exchanging. Diffie-Hellman has this property since once the session is concluded, a and b are dropped so the key will never be used again.

However…

Man-in-the-middle against D-H

Diffie-Hellman suffers for the Man-in-the-middle attack, morover such an attack is LETHAL in this protocol, the attacker can control the whole conversation. It works in the following way: when Alice sends (g,p,A) to Bob, Eve intercepts it and chooses a number *z*, performing $Z = g^z \bmod p$, sending to Bob (g,p,Z). Bob will reply with $B = Z^b$, but the attacker intercepts it and sends to Alice Z. What has happened? It happened that the attacker chooses a pair of keys, one for Alice and one for Bob, so Alice thinks to talk with Bob and Bob with Alice, but instead they talk to Eve. In particular, Alice uses the key $k_A = Z^a mod\ p$ which can be computed by the attacker, and Bob uses the key $k_B = Z^b mod\ p$ which can also be computed by the attacker.

At the end, in order to have a secure D-H protocol, an *authentication* ingredient must be added, otherwise the whole scheme suffers from the Man-in-the-middle attack. When Alice sends (g,p,A), this message has to be signed, and the same has to be done by Bob when sending B, then once established a secure shared key there is no longer need to sign further messages. So D-H with authentication can be considered a secure approach.