Information Systems Security
Series 7 : Authentication and Key Establishment Protocols

December 8th, 2021

The Diffie-Hellman Protocol

The Diffie-Hellman Protocol is a pre-distribution Key Agreement Protocol. It allows two users to use a public, non-confidential channel to build a shared secret key :

- **Initialisation** : We choose a big prime number p, and a generator $\alpha \in \mathbb{Z}_{p-1}$ of the multiplicative group $\mathbb{Z}_p^*$ (This two quantities can be chosen by a trusted third party, or by Alice herself as she starts the protocol). Both numbers are public.
- **Key Generation** :
  1. Alice chooses a number $x \in \mathbb{Z}_{p-1}$. x is kept secret. She sends $\alpha^x \mod p$ to Bob (and eventually p and α is Alice generated these numbers).
  2. Similarly, Bob chooses $y \in \mathbb{Z}_{p-1}$. y is kept secret. Bob sends $\alpha^y \mod p$ to Alice.
  3. Alice and Bob can then compute $(\alpha^y)^x \mod p$ (Alice) and $(\alpha^x)^y \mod p$ (Bob), which gives both of them the same shared key $K = \alpha^{xy} \mod p$.

Exercise 1 : Trying a Basic Authentication Scheme

Let's consider the following mutual authentication scheme between A and B, with K^a_{priv} (resp. K^b_{priv}) the private key of A (resp. of B), and K^a_{pub} (resp. K^b_{pub}) the corresponding public key. We consider that A and B have a authenticated copy of the other's public key :

If A starts the protocol, then :

- A sends a random challenge r_1 to B.
- B chooses a random challenge r_2, and sends $(r_2, K^b_{priv}(r_1))$ to A (i.e. B sends the new challenge, and the first challenge encrypted with his private key).
- A verifies $K^b_{priv}(r_1)$ with B's public key : A accepts B's identity if and only if she finds r_1. If it's the case, then A sends $K^a_{priv}(r_2)$ to B (A sends the second challenge ciphered).
- Similarly, B verifies $K^a_{priv}(r_2)$ with A's public key, and accepts A's identity if and only if he finds r_2.

We consider this exchanges are done in a secure channel, and cannot be intercepted. Show this protocol is unsafe, as C can usurp A's identity to authenticate to B.

Exercise 2 : Improvement of the Authentication Scheme

Let's consider A and B, with the same keys and in the same situation as in exercise 1, but with this updated protocol :

If A starts the protocol, then :

- A sends a random challenge r_1 to B.
- B chooses a random challenge r_2, and sends $(r_2, K^b_{priv}(r_1 \parallel r_2))$ to A (this time, B ciphers the concatenation of both challenges).
- A verifies $K^b_{priv}(r_1 \parallel r_2)$ with B's public key : A accepts B's identity if and only if she finds $r_1 \parallel r_2$. If it's the case, then A sends $K^a_{priv}(r_1 \parallel r_2)$ to B (A sends the ciphered concatenation).
- B verifies $K^a_{priv}(r_1 \parallel r_2)$ with A' public key, and accepts her identity if ans only if he founds $r_1 \parallel r_2$.

Once again, we consider that messages can't be intercepted. Show that this protocol is still unsafe, and that if A starts this protocol with C, then C can start an exchange with B and usurp A's identity.

Exercise 3 : Diffie-Hellman

- Alice and Bob are using Diffie-Hellman to generate a key. We have $p = 17$, $\alpha = 3$. A chooses $x = 7$, and B chooses $y = 11$. Finish the protocol, i.e. describe with quantities will be computed and exchanged by Alice and Bob, and what's the key obtained.
- With the same values, add Charlie as the Man-In-The-Middle. Choose his x' and y', and show how he can create two keys, one with Alice and one with Bob.

Exercise 4 : Simple Key Establishment Protocol Analysis

We have the following protocol, in a public, non-confidential channel, used to create session keys :

- (Initialisation) A and B share a long-term symmetric key, S.
- A generates a random number r_a, and sends this number to B.
- B generates a random number r_b, and sends this number to A.

Then, A and B both compute the session key $K = E_S(r_a \oplus r_b)$.

We'll analyse if this protocol respects a few properties :

- Implicit key authentication : Nobody except for A and B may have the generated key.
  Do we have implicit key authentication here ?
- key confirmation : A and B are sure that the other has the key.
  Do we have key confirmation here ?
- Perfect forward secrecy : If the long term key S is compromised by an attacker, the previous session keys are still safe.
  Do we have perfect forward secrecy here ?
- Future secrecy : If the long term key S is compromised, the future session keys are safe from a passive attacker.
  Do we have future secrecy here ?