Diffie-Hellman for multiple parties

HW4 - CNS Sapienza

Pietro Spadaccino 1706250

23 November 2018

1 Diffie-Hellman key exchange

Diffie-Hellman key exchange is a protocol that makes possible exchanging a secret on a public or insecure channel between two parties. The protocol starts by agreeing a prime number p and considering the generator g of its multiplicative group Z_p^*. Both parties Alice and Bob chose a secret a and b in the range $[1, p-1]$. Now Alice calculates $A = g^a \text{mod } p$, Bob calculates $B = g^b \text{mod } p$ and exchange their result with Alice, who does the viceversa. At the end Alice can get $K = B^a \text{mod } p$ and Bob can get $K = A^b \text{mod } p$. Since:

$$K = (g^a \bmod p)^b \bmod p = (g^b \bmod p)^a \bmod p = g^{ab} \bmod p \tag{1}$$

then Alice and Bob exchanged the same secret K in a secure way.

This protocol is resistant to *passive* man in the middle attack, because all the messages that go over the communication channel don't give any useful information to an attacker, if he doesn't know the secrets a and b which are kept private. However, if the attacker performs an *active* man in the middle then he could act as Bob from the point of view of Alice and act as Alice from the point of view of Bob, exchanging two different keys between the parties and being able to arbitrarily disrupt the communication.

2 Diffie-Hellman for 3 parties

There are different ways to adapt the protocol for three parties, which we will call Alice, Bob and Charlie. We will discuss about how to do this on a ring topology, where each one of the parties has two different neighbors, receiving message from one and sending messages to the other.

For now we assume for now that p and g are known to everyone. The three parties start by choosing their secret a, b, c and calculate respectively: $K_a = g^a \bmod p$, $K_b = g^b \bmod p$, $K_c = g^c \bmod p$. Alice is the first of the ring and starts by sending K_a to Bob. He then calculates $K_{ab} = (K_a)^b \bmod p$ and sends to Charlie K_{ab} and K_b. Charlie is the first getting the final shared secret $K_{abc} = (K_{ab})^c \bmod P$. Charlie also calculates K_{bc} and sends it to Alice alongside K_c. Now also she can get the secret $K_{abc} = (K_{bc})^a \bmod p$. To make also Bob to get the secret, Alice calculates $K_{ac} = (K_c)^a \bmod p$ and sends back to Bob. Finally he gets $K_{abc} = (K_{ac})^b \bmod p$.

Following this protocol, an attacker is only able to see K_a, K_b, K_c, K_{ab}, K_{bc} and K_{ca}, but since he doesn't know a, b, c he can't calculate the shared secret K_{abc}. Otherwise, if an attacker Trudy is able to control the input and output communications of one of the three parties, let's say Alice, then Trudy can act as Alice, exchanging a key between her, Bob and Charlie and simultaneously a different one between her and Alice.

3 Diffie-Hellman for N parties

Now let's consider the key exchange applied to N different parties $P = \{P_0, P_1, ..., P_{N-1}\}$. We will use the same ring topology described in the previous section, where each P_i receives messages from P_{i-1} and sends responses to the next P_{i+1}. For simplicity, let us denote $P_i = P_{(i \bmod N)}$ so that $P_N = P_0$. Also, we denote $K_{\{P_i\}}$ the result of the exponentiation of g modulo p by the secret of party P_i, $K_{\{P_i, P_j\}}$ the result of the exponentiation by parties P_i and P_j and so on. More in general, let be $I \subseteq P$: we can denote K_I as the result of the exponentiations made by parties I. Our final shared secret will be K_P.

Let be M a message exchanged from different parties. M will contain up to $N - 1$ exponentiations made by the previous parties and it can be seen as a token. Each party P_i takes the message M sent to him by P_{i-1}: for every exponentiation K_I in M, the party P_i substitutes it into M with its exponentiation of them $K_{I \cup P_i}$. Finally P_i appends its K_{P_i} to M and forwards M to P_{i+1}. each exponentiation in K_I in M will have $|I|$ grown by 1 at each party P_i, until M arrives to the last party P_{N-1}: it will be the first party to decode the final secret K_P. Once this happens, obviously P_{N-1} can't just sent the secret to the next party P_0, so he will have to remove the secret before forwarding M. After we complete two loops of the ring, we are sure that every party has decoded the shared secret. More precisely,

Table 1: Example of execution with 5 parties P_0, P_1, P_2, P_3, P_4. The first column is the sender and the receiver of the message, the second column is the content of the message. We can see that party P_4 is the first decoding the final secret K_P, by receiving $K_{\{P_0,P_1,P_2,P_3\}}$ from party P_3 and performing its exponentiation. Obviously the obtained secret is not appended into M, and also he can safely remove $K_{\{P_0,P_1,P_2,P_3\}}$ from M because it is not needed by any other party. We note that after $2|P| - 2 = 10 - 2 = 8$ exchanged messages every party has the final secret.

P_i to P_j	Content of M
$0 \to 1$	$K_{\{P_0\}}$
$1 \to 2$	$K_{\{P_0,P_1\}}, K_{\{P_1\}}$
$2 \to 3$	$K_{\{P_0,P_1,P_2\}}, K_{\{P_1,P_2\}}, K_{\{P_2\}}$
$3 \to 4$	$K_{\{P_0,P_1,P_2,P_3\}}, K_{\{P_1,P_2,P_3\}}, K_{\{P_2,P_3\}}, K_{\{P_3\}}$
$4 \to 0$	$K_{\{P_1,P_2,P_3,P_4\}}, K_{\{P_2,P_3,P_4\}}, K_{\{P_3,P_4\}}, K_{\{P_4\}}$
$0 \to 1$	$K_{\{P_2,P_3,P_4,P_0\}}, K_{\{P_3,P_4,P_0\}}, K_{\{P_4,P_0\}}$
$1 \to 2$	$K_{\{P_3,P_4,P_0,P_1\}}, K_{\{P_4,P_0,P_1\}}$
$2 \to 3$	$K_{\{P_4,P_0,P_1,P_2\}}$

$2|P| - 2$ exchanged messages are needed. An execution example is shown in Table 1.

The messages sent over the communication channel includes all the exponentiations K_I with I being all possible subsets of P made by adjacent parties except for $I = P$ and $I = \emptyset$. Without having access to any single secret of the parties P_i, a passive attacker can't find the shared secret. In the next section we will describe what an attacker can achieve by performing an active man in the middle.

3.1 Man in the middle

We describe an active man in the middle attack in two cases: where an attacker Trudy controls only the communications of one party, and where she controls the communications of all the parties. In both cases we assume that there must be a check on the number of parties participating on the key exchange, otherwise Trudy can join as a separate party and get the shared secret just by following the protocol.

Let's consider the case where T is able to control only one party P_j. In this situation, T can perform the exponentiations by means of P_j, substituting it and communicating with the other parties $P_{i \neq j}$ following the protocol. By doing so the final shared secret between them will be $H_T =$

$K_{\{P_0,P_1,\ldots,P_{j-1},P_T,P_{j+1},\ldots,P_{N-1}\}}$ instead of $K_{P=\{P_0,P_1,\ldots,P_{N-1}\}}$. The message M that Trudy forwards to P_j can contain all random numbers except for the exponentiation $K_{P\setminus\{P_j\}}$: when P_j receives this key, he expects to perform the exponentiation with his private secret thus obtaining the final secret K_P. Since P_j is not able to check which party performed the exponentiation given only the value of $K_{P\setminus\{P_j\}}$, Trudy can give to him the key $K_{\{T\}}$. By doing so P_j will perform the exponentiation obtaining the key $P_{\{T,P_j\}}$ thinking that this is the final shared secret, but this secret is only shared between him and Trudy. Resuming, Trudy is able to read all communications of the parties $P_{i\neq j}$ using the key H_T and to read and to modify the communications of party P_j using the key $K_{\{T,P_j\}}$.

Now let's move to the case where Trudy can control all the communication channel between the parties. This situation is a generalization of the previous one: Trudy can perform the same steps that did before when exchanging the key between her and party P_j. By doing so she can read and modify all the communications of all the parties.