Diffie - Hellman key exchange for multiple parties

HW4 - CNS Sapienza

Luigi Russo 1699981

23/11/2018

1 Overview

Diffie – Hellman (**DH**) key exchange is a method of exchanging keys over a public channel in a secure way. It allows two parties, say Alice (A) and Bob (B) to agree a common secret key to be used later, after sharing over the public channel some informations. The method is basically this:

1. Alice generates a number a (secret) and so does Bob with b
2. Alice also chooses a prime number[1] p and a group g. Then she computes $K_A := g^a \, mod \, p$ and sends to Bob the triple K_A, g, p
3. Bob computes $K_B := g^b \, mod \, p$ and $K := K_A^b \, mod \, p$, sending back to Alice the number B
4. Alice can now compute $K := K_B^a \, mod \, p$

Alice and Bob now share the common and secret key K. What happens if also Carol (C) wants to join the process?

2 DH - Three parties

Now also Carol wants to share the secret key K. She decides a secret value c, as Alice and Bob did in the first case.

1. Alice picks a prime number p and decides the group g, sending them to Bob and Carol.

[1]Here I describe the canonical DH schema. However, variants of the protocol built over elliptic curves (**ECDH**) and hyperelliptic curves are also very popular

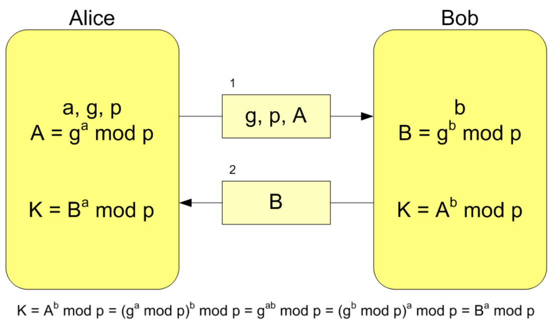
Alice
Bob
a, g, p
A = g^a mod p
K = B^a mod p
1
g, p, A
2
B
b
B = g^b mod p
K = A^b mod p
K = A^b mod p = (g^a mod p)^b mod p = g^ab mod p = (g^b mod p)^a mod p = B^a mod p

Figure 1: *Diffie – Hellman key exchange (from Wikimedia Commons)*

2. Alice computes g^a and sends it to Bob.
3. Bob can now compute g^{ab} and sends it to Carol.
4. Carol computes g^{abc} and is the first one of the group to have the final secret key K

What about Bob and Alice? They do not have enough informations to derive the key. Carol, instead, who has the final key, cannot send it back to Bob and Alice, since the channel is insecure. A simple workaround is this:

1. Bob computes g^b and sends it to Carol, who forwards to Alice g^{bc}
2. Alice can now compute $g^{bca} = K$
3. In order to provide also to Bob the key, Carol sends to Alice g^c, who forwards to Bob g^{ca}
4. Finally Bob computes $g^{cab} = K$ and the process can terminate.

So the idea here to handle the 3 parties protocol is to share partial computations among parties. This process can be generalized for n parties, with n greater than 3.

2.1 Security and MITM

The original DH method does not provide any form of authentication: this results in a critical vulnerability and exposes the parties to Man in the middle (**MITM**) attacks. In particular, if Alice and Bob want to agree a common key K, the attacker Trudy can agree a key K1 with Alice and a key K2 with Bob, pretending to be Alice when communicating with Bob and vice-versa with Alice. Every single message from Alice to Bob can be decrypted by Trudy, who then can forward whatever she wants to Bob. This security issue can be easily overcome if we use variants of DH that provide authentication.

Note: In the next sections I will always consider the authenticated DH, i.e. DH + a secure authentication method.

2.1.1 MITM on a single link

Say the attacker can sniff packets over the link from Alice to Bob. He is then able to see g^a and g^{ac}. We should also assume that the attacker can forge packets; however, he cannot derive in any way the third exponent b, that is the secret number picked by Bob; this kind of attack is unsuccessful as long as b is not compromised.

2.1.2 MITM on a single user

What happens if the attacker can sniff all packets from and to Alice? He is able to collect g^a, g^{bc}, g^{ac}. Again, the attacker still lacks some crucial informations to derive the secret key K: in this case the protocol is secure if both a and b remain secret (not only b as in the previous case).

2.2 Performance

DH is known to provide a very efficient way to share a key between two parties, since both Alice and Bob have to compute 2 exponentiations and exchange only 2 messages. With 3 parties the process is not that simple: in fact, we have that each party has to compute 3 exponentiations for a total of $3^2 = 9$ exponentiations (quadratic, as it will be pointed out later). Moreover, also the number of exchanged messages increases, since for each computation the partial value is forwarded to at least another party.

2.3 A spurious alternative

A simple alternative, despite a bit *spurious*, could be using DH to let Alice and Bob agree on a key *K1* and also Alice and Carol on the key *K2*. Now Alice generates autonomously a key K and sends it to Bob encrypted with *K1* and so does for Carol, encrypting it with *K2*. This is not a pure DH approach, since the final shared key has been decided only by Alice: it requires 8 exponentiations (4 by Alice, 2 by Bob, 2 by Carol) instead of 9. The number of exchanged messages in this case is 2 (Alice and Bob) + 2 (Alice and Carol) + 1 (Alice to Bob) + 1 (Alice to Carol) = 6

3 DH - N parties

In this section we see how the DH method can be generalized to allow n parties to share a common key K. As seen before, every user i has to choose a secret number (exponent) e_i; the secret key K is equal to g^m, where $m := \prod_{i=1}^{n} e_i$. Since partial computations can be sent over the insecure channel, because even with them the attacker cannot derive m (nor K of course), a simple strategy to reach the goal is this:

1. Arrange the users in a circle. Now every user i sends to his "next" (i+1 mod N) the partial g^{e_i}.
2. At this point everyone forwards the key received from his "previous", after exponentiating it to e_i.
3. The process terminates after N rounds.

The idea here [1] is that each user can add his contribute to partial keys, applying a transformation (exponentiation to e_i) that cannot be performed by anyone else.

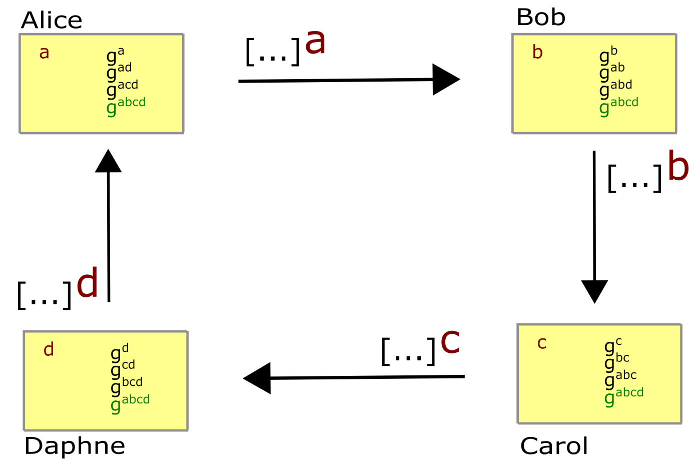

Figure 2: *Diffie – Hellman key exchange for 4 parties*

3.1 Security and MITM

Again we focus on the authenticated DH schema.

3.1.1 MITM on a single link

The attacker can only sniff packets from party $user_i$ to $user_{i+1}$. But the final key is never shared between two adjacent users, and all the partial computations shared through the channel, despite quite numerous, are not enough to derive the key. If the exponent e_{i+1} is compromised, however, the attacker can derive the key since in the final round $user_i$ sends to $user_{i+1}$ a partial key $\widetilde{K}$ that exponentiated to e_{i+1} produces K.

3.1.2 MITM on a single user

Again this attack is unsuccessful as long as e_i and e_{i+1} remain secret. This is due to the fact that the attacker can sniff both $\widetilde{K_i}$ and $\widetilde{K_{i+1}}$, where $K = \widetilde{K_i}^{e_i}$

3.2 Performance

Each user performs N exponentiations, so the total number of exponentiations is quadratic (cfr. section 2.2). It is possible, however, to reduce the number of exponentiations with a *Divide and Conquer* approach [1], thus resulting in a logarithmic number of operations per user.

References

[1] *Wikipedia Cryptography Portal*
https://en.wikipedia.org/wiki/Portal:Cryptography