A Study On Diffie-Hellman Key Exchange Protocol

Narendra Internship Project Report

Saripella Vivekananda varma, VIT, Vellore.

Mentor:- Prof. R.C. Hansdah.

Dept. of CSA, IISc Bangalore.

Problem statement:-

Establish a 128-bit shared secret key between two processes in different machines with the help of Authenticated Diffie-Hellman key establishment protocol. That is, the processes already have a shared secret key, and they want to establish a new shared secret key ensuring perfect forward secrecy. For authentication, you can use any of the MAC function (CMAC or HMAC). The program should use the following group to establish the shared secret key.

- Multiplicative residue group Zp*, where p is a 512-bit prime number.

For this case you need to get a prime subgroup of Zp*, where Diffie-Hellman key establishment protocol can be run. The program should display the common key established in a separate window for each process, and the time required to compute the key (excluding communication delay) in each of the group. Each process must be executed in a separate terminal and these two processes should communicate with each other using TCP or UDP protocol.

Introduction:-

The first published public-key algorithm appeared in the seminal paper by Diffie and Hellman that defined public-key cryptography and is generally referred to as Diffie-Hellman key exchange protocol. A number of commercial products employ this key exchange. The purpose of the algorithm is to enable two users to securely exchange a key that can be used for subsequent symmetric encryption of messages. The algorithm itself is limited to the exchange of secret values.

The Diffie-Hellman algorithm depends for its effectiveness on the difficulty of computing discrete logarithms. A primitive root of a prime number p is defined as one whose powers modulo p generate all the integers from 1 to p-1. That is, if a is a primitive root of the prime number p, then the numbers

$$a \bmod p, a^2 \bmod p, \ldots, a^{p-1} \bmod p$$

are distinct and consist of the integers from 1 through p-1 in some permutation. For any integer b and a primitive root a of prime number p, there exists an unique exponent i such that

b ≡ a^i (mod p) where 0≤i≤(p-1)

The exponent ‘i’ is referred to as the discrete logarithm.

Problem in key exchange:-

The difficulty lies in finding a primitive root of the prime number p and verifying it. Hence instead, we consider a generator of subgroup. we choose a prime order subgroup so that any element in that group will be a generator. When considering a big prime p, the group of invertible integers modulo p are all integers from 1 to p-1. There are p-1 of them. The order of an integer g modulo p is the smallest integer k>0 such that g^k = 1(modp).

Group theory states that the powers of an element g, i.e. 1, g, g^2… are collectively a subgroup of the group of invertible integers modulo p, and the order of g is really the size of that subgroup (called "the subgroup generated by g"). The cardinal of a subgroup is always a divisor of the cardinal of the group that contains it(By lagrange’s theorm). Thus, k divides p-1.

Recall that a primitive element g is one such that the subgroup it generates really is all of the invertible integers modulo p, not just some of them. Therefore, to verify whether an integer g is primitive or not, all you have to do is check that its order is p-1 but not one of the other possible subgroup orders. That is, we check that g^k' not equal to 1(mod p) for all k'< p−1 such that k' divides p-1.

Now comes the actual difficulty:

You must somehow know the list of prime factors of p-1. Hence we choose a prime order subgroup so that any element in that group will be a generator. This is achieved by following the below process :

- Let p be a random large-sized prime.
- Keep on generating p until a medium-sized prime q is obtained such that it satisfies p = q*z+1 for some integer z.
- To make an element g of order exactly q, a random integer h modulo p is generated, and we set g = (h^(p-1/q))mod p . It is easily shown that g^q = 1(mod p), which means that the order of g is a divisor of q. Since q has been chosen to be prime, the order of g can be either 1 or q. If g not equal to 1, then its order cannot be 1. Therefore, g will have order exactly q as long as it is not equal to 1.

Problem Understanding:-

As we find out the prime number and its generator we are ready to perform diffie-hellman key exchange. The key exchange can be volatile to man-in-the-middle attack which can overcome by authentication. So, before the shared secret key is generated between the two processes authentication has to be done with HMAC or other authentication method. As 128-bit shared secret key is already there we can go with HMAC authentication. Once authenticated, the shared secret key is generated between the two processes. The perfect forward secrecy can be ensured by generating different private keys for each session so that the compromise of a single session key will not affect data of past sessions i.e, will not compromise the past data.

Concepts Learned in this tenure:-

1) Group theory concepts
2) Important Theorms and their proofs which are used.
3) Cryptographic algorithms

Algorithms Implemented :-

Algorithm to find n-bit prime number:-

The most efficient miller-Rabin algorithm is used to find the n-bit large prime efficiently.

Algorithm to find multiplicative inverse of a number:-

The prime is generated using the miller-rabin algorithm and extended Euclidean algorithm can be used to find the multiplicative inverse of a given number with respect to a prime number.

Algorithm to find generator:-

1) Generate a random 512 bit prime number p using primality tests such as Miller-Rabin algorithm.

2) Get the group of invertible integers modulo p which are all integers from 1 to p-1.

3) Find a prime 'q' using the following mechanism such that q divides p-1 ie a random integer z is obtained that satisfies p = q*z + 1.

- In order to get 'z', multiply all the numbers between 2 and 10 that divide p-1.
- Now substitute 'z' in the equation q = (p-1) /z to get 'q'.
- Test the primality of 'q', if not then generate a 512-bit prime succeeding the present prime and repeat the above process else proceed.

4) Once you get prime number 'q' then select a random integer 'h' from the group of invertible integers modulo p.

5: Now, compute g = (h^((p−1)/q)) mod p and check if its value is greater than 1.

6: If greater, then the value calculated is the generator otherwise repeat step 4 with some other valid value for 'h'.

7: So, once you have p, q and secret key generated by both parties then Diffie-Hellman key exchange can be performed.

Results:-

Perfomance Metrics :

The below analysis was obtained in 100 runs of the algorithm to find generator:

- The algorithm took 8.89 seconds at it's best run to find a generator where as it's worst case was 232 seconds.
- The average time taken was 65.32 seconds.

Conclusion:-

For Diffie-Hellman protocol any value of 'g' would suffice i.e, Both the parties would end up with the same key. However for security we need the order of g to be a multiple of a large enough prime value. As primitive root finding is tough we choose a prime order subgroup so that any element in that group will be a generator. The generator can be generated within sufficient time for such large prime numbers efficiently. Thus, Shared secret key through Authenticated diffie-Hellman key exchange protocol is established perfectly ensuring the perfect forward secrecy.