Sharing a secret number

Description

Hereâ??s a naive method for two people to agree a secret number. The number to be shared is simply an integer, which is a key for some other encrypted communication channel. It could even be a PIN to a bank account, an access code for a door or the combination code to a shared locker at the gym. The two people donâ??t decide the number themselves. It will be generated automatically, but they both need to be told the number via a communication channel that is open for anyone else to see.

The channel we are using for our communication is entirely public. Anyone can see what passes between us â?? like the Internet. Hereâ??s the protocol.

1. We both agree some arbitrary number and exchange that across the public network. One of us may decide on that number, or it could be chosen by a third party, or we may decide to use the number of todayâ??s date. Letâ??s say that arbitrary number is 5. Algebraically Iâ??ll call that G, G=5.
2. We, or our computers, each independently make up a secret number (N). No one else will ever see or get to know that secret number except the originator, or their computer. I choose 4 as my secret number N = 4; you choose 2 as your secret number, N = 2.
3. I, or my computer, raise the arbitrary number, 5, to the power of my secret number: $G^N = 5^4 = 625$. You do the same and calculate 5 to the power of your secret number 2, i.e. $G^N = 5^2 = 25$.
4. We then transmit those numbers to each other across the public network.
5. I take your number and raise it to the power of my secret number $S = 25^4 = 390625$
6. You do same and raise the number I sent you to the power of your secret number $S = 625^2 = 390625$. So we now both have the same number ($S_{combined} = 390625$). So that value of S is going to be the key to some other encryption system, bank account PIN, door access, or gym locker.

Hereâ??s whatâ??s public and there to be exploited by a potential hacker:

- G: the shared arbitrary starting number, as a generator.
- The result of the calculation $S = G^N$ for each of us.
- This protocol (method) for generating a shared secret key.

Hereâ??s whatâ??s ***not*** shared with anyone

- N (for each of us)
- The result of the calculation G raised to the power of our secret numbers combined, which is now our shared secret key $S_{combined}$.

There are clear vulnerabilities in this encryption system. It may work as a parlour game, or in a low risk context, or where the protocol isnâ??t known.

The hacker knows G and the G^N that each of us has just exchanged. Knowing those numbers means that N can be calculated easily

- $5^N = 25$ yields $N = \log_5(25) = 2$ (your secret number)
- $5^N = 625$ yields $N = \log_5(625) = 4$ (my secret number)

Then itâ??s easy to raise G to the power of our secret numbers to reveal the shared secret key $S_{combined}$. So that method is extremely insecure. Iâ??ve derived this explanation from a very helpful Elliptic Curve Cryptography Tutorial by Will Raedy. $S = G^N$ is trivial to solve for N if you know S and G.

One way to make the codebreaking challenge harder is to introduce another factor, a divisor, p. If p = 7 and you divide that into another number, such as 25, then youâ??ll find that it divides exactly 3 times to make 21 with a remainder of 4. The remainder is called the *mod* of 25 and 7. Thatâ??s written as 25 mod 7 = 4. So it takes more computation to solve for N in the following, even if you know S, G and p. So what I transmit across the public channel is the value of S mod p, i.e. 4.

$S = G^N \bmod p$

In the example above weâ??ll agree on p = 7. Thatâ??s also public.

I transmit the result of

- $S \bmod p = 5^4 \bmod 7 = 625 \bmod 7 = 2$

You transmit the result of

- $S \bmod p = 5^2 \bmod 7 = 25 \bmod 7 = 4$

Someone intercepting the messages will have to iterate through a range of values that yield that remainder. Raedy says thatâ??s like being told that an event starts at 12.00 and ends at 1.00 but you donâ??t know the number of days in between. Is it a one hour event, or 12+1 hours, 24+1 hours, etc?

Solving for N when the other parameters are known in **$S = G^N \bmod p$** is called *the discrete log problem*, and the difficulty is compounded for very large numbers. The difficulty of the mod calculation contributes to the security of the Elliptic-curve Diffieâ??Hellman encryption method, especially when multiplication takes place on the elliptic function curve as explained in the previous post.

Reference

- Raedy, Will. 2017. Elliptic Curve Cryptography Tutorial â?? Understanding ECC through the Diffie-Hellman Key Exchange. *Fullstack Academy*, 8 August. Available online: https://www.youtube.com/watch?v=gAtBM06xwaw (accessed 9 May 2021).

Category

1. Architecture

Tags

1. elliptic
2. encryption

Date Created
June 5, 2021
Author
rcoyne99