Key exchange

Public key cryptography resulted from the need to be able to exchange keys between people around the world who had never met; it was impractical to have trusted couriers deliver secret keys to senders and receivers. Although in their 1976 paper Diffie and Hellman did not propose a public key cryptosystem, they did propose a scheme to do key exchange. The security of their method, called Diffie-Helllman key exchange, is based upon the discrete logarithm problem.

Discrete logarithms

Recall that a logarithm is an exponent. Another way of expressing that 2 raised to the power 5 is 32 is that the logarithm of 32 to the base 2 is 5: $\log_2 32 = 5$..

It is relatively easy to raise numbers to powers; e.g., $3^{20} = 3486784401$, but it is harder to find a logarithm; e.g., what is the base-2 logarithm of 8589934592? . $2^{?} = 8589934592$.

For real numbers, we can solve $2^{?} = 8589934592$ by playing the "high-low" game. For example, $2^{10} = 1024$, which is too small. We need a larger exponent. $2^{20} = 1048576$; which is still too small. $2^{30} = 1073741824$ -- too small. $2^{40} = 1099511627776$ -- too large. $2^{35} = 34359735368$ -- still too large. Finally, we find that $2^{33} = 8589934592$. So, the base-2 logarithm of 8589934592 is 33. This scheme works because $y = \log_2 x$ increases monotonically with *x*.

The discrete logarithm problem (DLP) refers to the problem of finding logarithms modulo some integer. Recall that when we mod out by an integer *n*, we are left with only finitely many integers – a discrete set – usually represented as 0, 1, 2, … *n*-1.

The discrete logarithm problem asks for a solution of something like this: $2^{?} = 9 \bmod 11$.

Raising an integer to powers modulo *n* scrambles the order of the results. For example, here are the powers of 2 modulo 11:

$$2^1 = 2 \bmod 11$$
$$2^2 = 4 \bmod 11$$
$$2^3 = 8 \bmod 11$$
$$2^4 = 5 \bmod 11$$
$$2^5 = 10 \bmod 11$$
$$2^6 = 9 \bmod 11$$
$$2^7 = 7 \bmod 11$$

$$2^8 = 3 \bmod 11$$
$$2^9 = 6 \bmod 11$$
$$2^{10} = 1 \bmod 11$$

So, the answer to our discrete logarithm problem is 6: $2^6 = 9 \bmod 11$.

Because the powers of 2 modulo 11 do not increase monotonically with the exponent, we cannot play the "high-low" game. There is no known efficient algorithm for computing discrete logarithms.

It is easy to discrete exponentiation, but it is hard to determine discrete logarithms. Diffie-Hellman key exchange is based upon the discrete logarithm problem.

The discrete logarithm problem (DLP)

The discrete logarithm problem is:

Given a, n, and $g = a^x \bmod n$ determine x.

There is no known efficient, non-quantum algorithm to solve the discrete logarithm problem. There is a quantum algorithm.

The discrete logarithm problem is the basis for the following cryptographic algorithms: Diffie-Hellman key exchange, ElGamal encryption, and the Digital Signature Algorithm.

The Diffie-Hellman Problem (DHP)

Notice that breaking Diffie-Hellman key exchange might not be equivalent to solving the discrete logarithm problem. To break Diffie-Hellman key exchange, what must be solved is:

Given a, n, a^x, and a^y, determine a^{xy}.

Clearly a solution to the DLP would imply a solution to the DHP, but it is not known that they are equivalent problems.

The Diffie-Hellman assumption

The Diffie-Hellman assumption is that the DHP is hard.

Primitive roots

Notice that in one of the examples above, the modulus 11 is a prime and the ten powers of the base 2 are representatives of the ten nonzero equivalence classes modulo 11: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10. We describe this property by saying that 2 is a primitive root of (the prime) 11. A prime always has at least one primitive root. Because all ten of the nonzero integers modulo 11 occur as powers of 2, it could have taken 10 trials to discover that $2^6 = 9 \bmod 11$.

If a primitive root is not used for the base of the exponentiation, not all of the remainders modulo n will occur as powers of the base. For example, 3 is not a primitive root of 11. Let us try to solve $3^? = 5 \bmod 11$.

Computing the powers of 3 modulo 11 results in:

$$3^1 = 3 \bmod 11$$
$$3^2 = 9 \bmod 11$$
$$3^3 = 5 \bmod 11$$
$$3^4 = 4 \bmod 11$$
$$3^5 = 1 \bmod 11$$
$$3^6 = 3 \bmod 11$$
$$3^7 = 9 \bmod 11$$
$$3^8 = 5 \bmod 11$$
$$3^9 = 4 \bmod 11$$
$$3^{10} = 1 \bmod 11$$

Notice that only five of the representatives modulo 11 occur: 3, 9, 5, 4, 1. When we try to solve $3^? = 5 \bmod 11$, we are twice as likely to find a logarithm because there are only half as many possible powers. Either 3 or 8 can be used as the logarithm of 5 for base 3 and modulo 11. It is not absolutely essential that we pick the base of the exponentiation to be a primitive root of the modulus; however, if there is too much repetition, the number of trials needed to determine a logarithm would be reduced considerably.

There are tables of primitive roots for various moduli.

Diffie-Hellman key exchange

Say that Alex has a long message that he wants to send to Nick. Because public-key cryptosystems are slow, Alex and Nick decide to set up a channel using a symmetric-key cryptosystem, but they need to exchange a key. They select a large prime *p* and a primitive root *a*. Alex selects an exponent k_A so that $1 \le k_A \le p-2$ (we are going to calculate a^{k_A} mod *p* so if $k_A = p-1$ then $a^{k_A} \bmod p = 1$ by Fermat's Theorem). Nick also selects an exponent k_N so that $1 \le k_N \le p-2$. Alex calculates $a^{k_A} \bmod p$ and sends the result to Nick; in some sense, Alex is sending Nick the exponent k_A, but neither Nick nor anyone who might intercept the message (e.g., Beth) can determine the exponent even though they know *p* and *a* because determining the exponent would mean solving the discrete logarithm problem. Similarly Nick calculates $a^{k_N} \bmod p$ and sends the result to Alex. So, in some sense, Alex and Nick have "exchanged exponents."

$$\begin{tikzcd}\text{Alex} \arrow[r, shift left, "a^{k_A}"] & \text{Nick} \arrow[l, shift left, "a^{k_N}"]\end{tikzcd}$$

Now, of course, Alex knows his exponent k_A and $a^{k_N} \bmod p$, and Nick knows his exponent k_N and $a^{k_A} \bmod p$; so, Alex can calculate $\left(a^{k_N}\right)^{k_A} \bmod p$, and Nick can calculate $\left(a^{k_A}\right)^{k_N} \bmod p$. Each then has $a^{k_A k_N} \bmod p$, and this or some portion of it can be used as their common key.

Man-in-the-middle attack

The Diffie-Hellman key exchange algorithm is vulnerable to the man-in-the-middle (intruder-in-the middle or person-in-middle) attack. For example, if Beth is able to insert herself into the channel and receive and stop transmission between Alex and Nick, then she can do this.

$$\begin{tikzcd}\text{Alex} \arrow[r, shift left, "a^{k_A}"] & \text{Beth} \arrow[l, shift left, "a^{k_E}"] \arrow[r, shift left, "a^{k_E}"] & \text{Nick} \arrow[l, shift left, "a^{k_N}"]\end{tikzcd}$$

Beth inserts herself in the channel between Alex and Nick. Beth "exchanges exponents" with Alex, and Beth "exchanges exponents" with Nick. When, for example, Alex thinks that he is sending a message to Nick he (unknowingly) uses the key that is common for him and Beth and is, in fact, sending his message to Beth (who can decrypt it). Beth then can, for example, not send the message on to Nick or read the message and send it on to Nick or modify the message and send the new message on to Nick using the key that is common for her and Nick. If Nick receives a message, he decrypts it thinking that it came from Alex because Nick believes that the secret key that he has is shared with Alex (when, in fact, it is shared with Beth).

Exchanging a key with more than two people

There is an obvious way to extend the Diffie-Hellman key exchange to exchange a key among more than two people. For example, for three people:

Alex, Nick, and Dave want to exchange a key for a symmetric-key cryptosystem. They agree on a prime p and a primitive root a.

Round one

$$\begin{tikzcd}
\text{Nick} \arrow[rr, "a^{k_N} \bmod p"] & & \text{Alex} \arrow[dl, "a^{k_A} \bmod p"] \\
& \text{Dave} \arrow[ul, "a^{k_D} \bmod p"] &
\end{tikzcd}$$

At this point, Nick knows k_N and $a^{k_D} \bmod p$, Alex knows k_A and $a^{k_N} \bmod p$, and Dave knows k_D and $a^{k_A} \bmod p$.

Round two

$$\begin{tikzcd}
\text{Nick} \arrow[rr, "\left(a^{k_D}\right)^{k_N} \bmod p"] & & \text{Alex} \arrow[dl, "\left(a^{k_N}\right)^{k_A} \bmod p"] \\
& \text{Dave} \arrow[ul, "\left(a^{k_A}\right)^{k_D} \bmod p"] &
\end{tikzcd}$$

At this point, Nick knows k_N and $a^{k_D} \bmod p$ and $a^{k_A k_D} \bmod p$, Alex knows k_A and $a^{k_N} \bmod p$ and $a^{k_D k_N} \bmod p$, and Dave knows k_D and $a^{k_A} \bmod p$ and $a^{k_N k_A} \bmod p$.

Nick, Alex, and Dave can all agree on the key $a^{k_N k_A k_D} = \left(a^{k_A k_D}\right)^{k_N} = \left(a^{k_D k_N}\right)^{k_A} = \left(a^{k_N k_A}\right)^{k_D} \bmod p$.

There are a couple of problems with this scheme: this procedure can obviously be extended to a group of n people, but the number of rounds will increase to $n - 1$; and, if a new person is added to the group or if a person leaves the group, the entire key exchange must be repeated.