Diffie-Hellman Key Exchange

Cryptographic explanation

The simplest and the original implementation, later formalized as **Finite Field Diffie–Hellman** of the protocol uses the multiplicative group of integers modulo p, where p is prime, and g is a primitive root modulo p. These two values are chosen in this way to ensure that the resulting shared secret can take on any value from 1 to p–1. Here is an example of the protocol, with non-secret values in blue, and secret values in **red**.

1. Alice and Bob publicly agree to use a modulus $p = 23$ and base $g = 5$ (which is a primitive root modulo 23).
2. Alice chooses a secret integer $\boldsymbol{a} = 4$, then sends Bob $A = g^a \bmod p$
   - $A = 5^4 \bmod 23 = 4$ (in this example both A and a have the same value 4, but this is usually not the case)
3. Bob chooses a secret integer $\boldsymbol{b} = 3$, then sends Alice $B = g^b \bmod p$
   - $B = 5^3 \bmod 23 = 10$
4. Alice computes $\boldsymbol{s} = B^a \bmod p$
   - $\boldsymbol{s} = 10^4 \bmod 23 = 18$
5. Bob computes $\boldsymbol{s} = A^b \bmod p$
   - $\boldsymbol{s} = 4^3 \bmod 23 = 18$
6. Alice and Bob now share a secret (the number 18).

Both Alice and Bob have arrived at the same values because under mod p,

$$A^b \bmod p = g^{ab} \bmod p = g^{ba} \bmod p = B^a \bmod p$$

More specifically,

$$(g^a \bmod p)^b \bmod p = (g^b \bmod p)^a \bmod p$$

Only a and b are kept secret. All the other values – p, g, $g^a \bmod p$, and $g^b \bmod p$ – are sent in the clear. The strength of the scheme comes from the fact that $g^{ab} \bmod p = g^{ba} \bmod p$ take extremely long times to compute by any known algorithm just from the knowledge of p, g, $g^a \bmod p$, and $g^b \bmod p$. Such a function that is easy to compute but hard to invert is called a one-way function. Once Alice and Bob compute the shared secret they can use it as an encryption key, known only to them, for sending messages across the same open communications channel.

Of course, much larger values of a, b, and p would be needed to make this example secure, since there are only 23 possible results of n mod 23. However, if p is a prime of at least 600 digits, then even the fastest modern computers using the fastest known algorithm cannot find a given only g, p and g^a mod p. Such a problem is called the discrete logarithm problem. The computation of g^a mod p is known as modular exponentiation and can be done efficiently even for large numbers. Note that g need not be large at all, and in practice is usually a small integer (like 2, 3, ...).

Secrecy chart

The chart below depicts who knows what, again with non-secret values in blue, and secret values in **red**. Here Eve is an eavesdropper – she watches what is sent between Alice and Bob, but she does not alter the contents of their communications.

- g, public (primitive root) base, known to Alice, Bob, and Eve. $g = 5$
- p, public (prime) modulus, known to Alice, Bob, and Eve. $p = 23$
- $\mathbf{a}$, Alice's private key, known only to Alice. $\mathbf{a = 6}$
- $\mathbf{b}$, Bob's private key known only to Bob. $\mathbf{b = 15}$
- A, Alice's public key, known to Alice, Bob, and Eve. $A = g^a \bmod p = 8$
- B, Bob's public key, known to Alice, Bob, and Eve. $B = g^b \bmod p = 19$

Alice		Bob		Eve	
Known	**Unknc**	**Known**	**Unknc**	**Known**	**Unknown**
$p = 23$		$p = 23$		$p = 23$	
$g = 5$		$g = 5$		$g = 5$	
$\mathbf{a = 6}$	$\mathbf{b}$	$\mathbf{b = 15}$	$\mathbf{a}$		$\mathbf{a}$, $\mathbf{b}$
$A = 5^a \bmod 23$		$B = 5^b \bmod 23$			
$A = 5^6 \bmod 23 = 8$		$B = 5^{15} \bmod 23 = 19$			
$\mathbf{B = 19}$		$\mathbf{A = 8}$		$A = 8$, $B = 19$	
$s = B^a \bmod 23$		$s = A^b \bmod 23$			
$s = 19^6 \bmod 23 = \mathbf{2}$		$s = 8^{15} \bmod 23 = \mathbf{2}$			$\mathbf{s}$

Now ***s*** is the shared secret key and it is known to both Alice and Bob, but *not* to Eve. Note that it is not helpful for Eve to compute *AB*, which equals g^{a+b} mod p.

Note: It should be difficult for Alice to solve for Bob's private key or for Bob to solve for Alice's private key. If it is not difficult for Alice to solve for Bob's private key (or vice versa), then an eavesdropper, Eve, may simply substitute her own private / public key pair, plug Bob's public key into her private key, produce a fake shared secret key, and solve for Bob's private key (and use that to solve for the shared secret key). Eve may attempt to choose a public / private key pair that will make it easy for her to solve for Bob's private key.

Generalization to finite cyclic groups

Here is a more general description of the protocol:

1. Alice and Bob agree on a natural number n and a generating element g in the finite cyclic group G of order n. (This is usually done long before the rest of the protocol; g and n are assumed to be known by all attackers.) The group G is written multiplicatively.
2. Alice picks a random natural number a with $1 < a < n$, and sends the element g^a of G to Bob.
3. Bob picks a random natural number b with $1 < b < n$, and sends the element g^b of G to Alice.
4. Alice computes the element $(g^b)^a = g^{ba}$ of G.
5. Bob computes the element $(g^a)^b = g^{ab}$ of G.

Both Alice and Bob are now in possession of the group element $g^{ab} = g^{ba}$, which can serve as the shared secret key. The group G satisfies the requisite condition for secure communication as long as there is no efficient algorithm for determining g^{ab} given g, g^a, and g^b.

For example, the elliptic curve Diffie–Hellman protocol is a variant that represents an element of G as a point on an elliptic curve instead of as an integer modulo n. Variants using hyperelliptic curves have also been proposed. The supersingular isogeny key exchange is a Diffie–Hellman variant that was designed to be secure against quantum computers, but it was broken in July 2022.

Ephemeral and/or static keys

The used keys can either be ephemeral or static (long term) key, but could even be mixed, so called semi-static DH. These variants have different properties and hence different use cases. An overview over many variants and some also discussions can for example be found in NIST SP 800-56A. A basic list:

1. ephemeral, ephemeral: Usually used for key agreement. Provides forward secrecy, but no authenticity.
2. static, static: Would generate a long term shared secret. Does not provide forward secrecy, but implicit authenticity. Since the keys are static it would for example not protect against replay-attacks.
3. ephemeral, static: For example, used in ElGamal encryption or Integrated Encryption Scheme (IES). If used in key agreement it could provide implicit one-sided authenticity (the ephemeral side could verify the authenticity of the static side). No forward secrecy is provided.

It is possible to use ephemeral and static keys in one key agreement to provide more security as for example shown in NIST SP 800-56A, but it is also possible to combine those in a single DH key exchange, which is then called triple DH (3-DH).

Triple Diffie–Hellman (3-DH)

In 1997 a kind of triple DH was proposed by Simon Blake-Wilson, Don Johnson, Alfred Menezes in 1997, which was improved by C. Kudla and K. G. Paterson in 2005 and shown to be secure.

The long term secret keys of Alice and Bob are denoted by *a* and *b* respectively, with public keys *A* and *B*, as well as the ephemeral key pairs *x*, *X* and *y*, *Y*. Then protocol is:

Triple Diffie–Hellman (3-DH) protocol

Alice ($A = g^a$)		Bob ($B = g^b$)
$X = g^x$	$X \rightarrow$	
	$\leftarrow Y$	$Y = g^y$
$K = \mathrm{KDF}(Y^x, B^x, Y^a, X, Y, A, B)$		$K = \mathrm{KDF}(X^y, X^b, A^y, X, Y, A, B)$

The long term public keys need to be transferred somehow. That can be done beforehand in a separate, trusted channel, or the public keys can be encrypted using some partial key agreement to preserve anonymity. For more of such details as well as other improvements like side channel protection or explicit key confirmation, as well as early messages and additional password authentication, see e.g. US patent "Advanced modular handshake for key agreement and optional authentication".

Extended Triple Diffie–Hellman (X3DH)

X3DH was initially proposed as part of the Double Ratchet Algorithm used in the Signal Protocol. The protocol offers forward secrecy and cryptographic deniability. It operates on an elliptic curve.

The protocol uses five public keys. Alice has an identity key IK_A and an ephemeral key EK_A. Bob has an identity key IK_B, a signed prekey SPK_B, and a one-time prekey OPK_B. Bob first publishes his three keys to a server, which Alice downloads and verifies the signature on. Alice then initiates the exchange to Bob. The OPK is optional

Problem-01:

Suppose that two parties A and B wish to set up a common secret key (D-H key) between themselves using the Diffie Hellman key exchange technique. They agree on 7 as the modulus and 3 as the primitive root. Party A chooses 2 and party B chooses 5 as their respective secrets. Their D-H key is-

1. 3 2. 4 3. 5 4. 6

Solution-

Given-

- n = 7
- a = 3
- Private key of A = 2
- Private key of B = 5

Step-01:

Both the parties calculate the value of their public key and exchange with each other.

Public key of A

$= 3^{\text{private key of A}} \bmod 7$

$= 3^2 \bmod 7$

$= 2$

Public key of B

$= 3^{\text{private key of B}} \bmod 7$

$= 3^5 \bmod 7$

$= 5$

Step-02:

Both the parties calculate the value of secret key at their respective side.

Secret key obtained by A

$= 5^{\text{private key of A}} \bmod 7$

$= 5^2 \bmod 7$

$= 4$

Secret key obtained by B

$= 2^{\text{private key of B}} \bmod 7$

$= 2^5 \bmod 7$

$= 4$

Finally, both the parties obtain the same value of secret key.

The value of common secret key = 4.

Thus, Option (B) is correct.