Diffie–Hellman key exchange

Diffie–Hellman key exchange (**D–H**)[nb 1] is a specific method of exchanging cryptographic keys. It is one of the earliest practical examples of key exchange implemented within the field of cryptography. The Diffie–Hellman key exchange method allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communications channel. This key can then be used to encrypt subsequent communications using a symmetric key cipher.

The scheme was first published by Whitfield Diffie and Martin Hellman in 1976, although it had been separately invented a few years earlier within GCHQ, the British signals intelligence agency, by James H. Ellis, Clifford Cocks and Malcolm J. Williamson but was kept classified.[1]

Although Diffie–Hellman key agreement itself is an *anonymous* (non-*authenticated*) key-agreement protocol, it provides the basis for a variety of authenticated protocols, and is used to provide perfect forward secrecy in Transport Layer Security's ephemeral modes (referred to as EDH or DHE depending on the cipher suite).

The method was followed shortly afterwards by RSA, an implementation of public key cryptography using asymmetric algorithms.

In 2002, Hellman suggested the algorithm be called **Diffie–Hellman–Merkle key exchange** in recognition of Ralph Merkle's contribution to the invention of public-key cryptography (Hellman, 2002), writing:

> The system...has since become known as Diffie–Hellman key exchange. While that system was first described in a paper by Diffie and me, it is a public key distribution system, a concept developed by Merkle, and hence should be called 'Diffie–Hellman–Merkle key exchange' if names are to be associated with it. I hope this small pulpit might help in that endeavor to recognize Merkle's equal contribution to the invention of public key cryptography.[2]

U.S. Patent 4,200,770, from 1977 is now expired and describes the algorithm. It credits Hellman, Diffie, and Merkle as inventors.

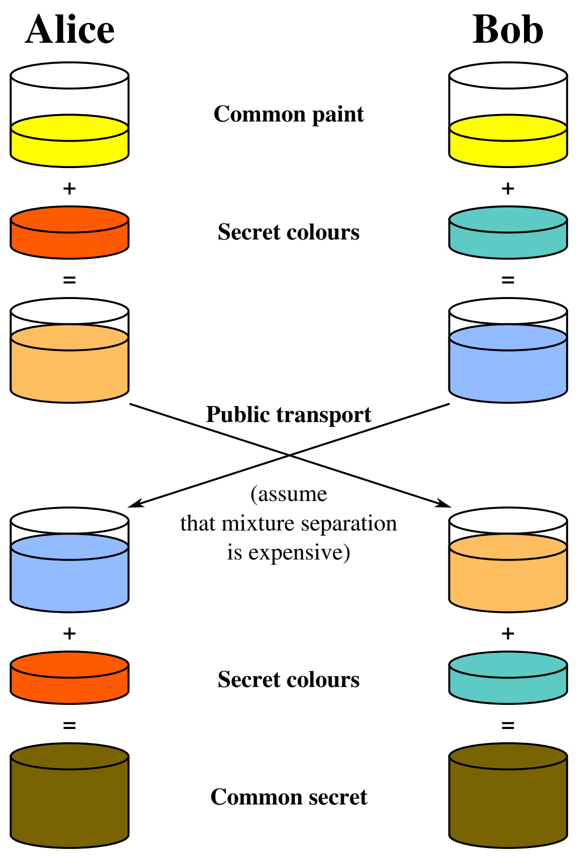

Illustration of the Diffie–Hellman Key Exchange

1 Description

Diffie–Hellman establishes a shared secret that can be used for secret communications while exchanging data over a public network. The following diagram illustrates the general idea of the key exchange by using colors instead of a very large number. The crucial part of the process is that Alice and Bob exchange their secret colors in a mix only. Finally this generates an identical key that is computationally difficult (impossible for modern supercomputers to do in a reasonable amount of time) to reverse for another party that might have been listening in on them. Alice and Bob now use this common secret to encrypt and decrypt their sent and received data. Note that the starting color (yellow) is arbitrary, but is agreed on in advance by Alice and Bob. The starting color is assumed to be known to any eavesdropping opponent. It may even be public.

1.1 Cryptographic explanation

The simplest and the original implementation of the protocol uses the multiplicative group of integers modulo p, where p is prime, and a primitive root modulo p. Here is an example of the protocol, with non-secret values in blue, and secret values in **red**.

1. Alice and Bob agree to use a prime number $p = 23$ and base $g = 5$ (which is a primitive root modulo 23).
2. Alice chooses a secret integer $\boldsymbol{a} = \mathbf{6}$, then sends Bob $A = g^{\boldsymbol{a}} \bmod p$
   - $A = 5^{\mathbf{6}} \bmod 23 = 8$
3. Bob chooses a secret integer $\boldsymbol{b} = \mathbf{15}$, then sends Alice $B = g^{\boldsymbol{b}} \bmod p$
   - $B = 5^{\mathbf{15}} \bmod 23 = 19$
4. Alice computes $\boldsymbol{s} = B^{\boldsymbol{a}} \bmod p$
   - $\boldsymbol{s} = 19^{\mathbf{6}} \bmod 23 = \mathbf{2}$
5. Bob computes $\boldsymbol{s} = A^{\boldsymbol{b}} \bmod p$
   - $\boldsymbol{s} = 8^{\mathbf{15}} \bmod 23 = \mathbf{2}$
6. Alice and Bob now share a secret (the number **2**).

Both Alice and Bob have arrived at the same value, because $(g^a)^b$ (for Bob, $8^{15} \bmod 23 = (g^a \bmod p)^b \bmod p = (g^a)^b \bmod p$) and $(g^b)^a$ are equal mod p. Note that only a, b, and $(g^{ab} \bmod p = g^{ba} \bmod p)$ are kept secret. All the other values – p, g, $g^a \bmod p$, and $g^b \bmod p$ – are sent in the clear. 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 $n \bmod 23$. However, if p is a prime of at least 300 digits, and a and b are at least 100 digits long, then even the fastest modern computers cannot find a given only g, p, $g^b \bmod p$ and $g^a \bmod p$. The problem such a computer needs to solve is called the discrete logarithm problem. The computation of $g^a \bmod p$ 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 prime (like 2, 3, 5...) because primitive roots usually are quite numerous.

1.2 Generalization to finite cyclic groups

Here's a more general description of the protocol,

1. Alice and Bob agree on a finite cyclic group G and a generating element g in G. (This is usually done long before the rest of the protocol; g is assumed to be known by all attackers.) We will write the group G multiplicatively.
2. Alice picks a random natural number a and sends g^a to Bob.
3. Bob picks a random natural number b and sends g^b to Alice.
4. Alice computes $(g^b)^a$.
5. Bob computes $(g^a)^b$.

Both Alice and Bob are now in possession of the group element g^{ab}, which can serve as the shared secret key. The values of $(g^b)^a$ and $(g^a)^b$ are the same because groups are power associative. (See also exponentiation.)

If m is a message, and an element of the group, then we can encrypt $e = mg^{ab}$. Then, to decrypt e we must first compute $(g^{ab})^{-1}$, as follows:

Bob knows $|G|$, b, and g^a. A corollary of Lagrange's theorem states that $h^{|G|} = 1$, the group identity, for all $h \in G$.

Bob then calculates $(g^a)^{|G|-b} = g^{a(|G|-b)} = g^{a|G|-ab} = g^{a|G|}g^{-ab} = (g^{|G|})^a g^{-ab} = 1^a g^{-ab} = g^{-ab} = (g^{ab})^{-1}$.

When Alice sends Bob the encrypted message, $e = mg^{ab}$, Bob computes $(g^{ab})^{-1}e = mg^{ab}(g^{ab})^{-1} = m(1) = m$.

When the group is too large for a multiplication table, then a multiplication algorithm is needed, and Exponentiation for finite cyclic groups should be used.

How could the key be extracted, in general, from the shared group element g^{ab}? One could extract a set of AES keys from it. This would require a mapping from G to the set of n-bit binary strings, and the mapping would depend on G. For example, if G consisted of the powers of a fixed nonsingular matrix A defined over a finite field, and if $g^{ab} = \{g_{i,j}\}$, then the element g_{11} would be an element of the field and would have a binary representation that could be divided up into AES keys.

1.3 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 (prime) base, known to Alice, Bob, and Eve. $g = 5$
- p = public (prime) number, known to Alice, Bob, and Eve. $p = 23$
- $\boldsymbol{a}$ = Alice's private key, known only to Alice. $\boldsymbol{a} = \mathbf{6}$
- $\boldsymbol{b}$ = Bob's private key known only to Bob. $\boldsymbol{b} = \mathbf{15}$

- A = Alice's public key, known to Alice, Bob, and Eve. $A = g^{\boldsymbol{a}} \bmod p = 8$

- B = Bob's public key, known to Alice, Bob, and Eve. $B = g^{\boldsymbol{b}} \bmod p = 19$

- Now $\boldsymbol{s}$ = the shared secret key and it is known to both Alice and Bob, but *not* to Eve. $\boldsymbol{s} = \mathbf{2}$

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), 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). Another demonstration of Diffie-Hellman (also using numbers too small for practical use) is given here [3]

2 Operation with more than two parties

Diffie–Hellman key agreement is not limited to negotiating a key shared by only two participants. Any number of users can take part in an agreement by performing iterations of the agreement protocol and exchanging intermediate data (which does not itself need to be kept secret). For example, Alice, Bob, and Carol could participate in a Diffie–Hellman agreement as follows, with all operations taken to be modulo p :

1. The parties agree on the algorithm parameters p and g .
2. The parties generate their private keys, named a , b , and c .
3. Alice computes g^a and sends it to Bob.
4. Bob computes $(g^a)^b = g^{ab}$ and sends it to Carol.
5. Carol computes $(g^{ab})^c = g^{abc}$ and uses it as her secret.
6. Bob computes g^b and sends it to Carol.
7. Carol computes $(g^b)^c = g^{bc}$ and sends it to Alice.
8. Alice computes $(g^{bc})^a = g^{bca} = g^{abc}$ and uses it as her secret.
9. Carol computes g^c and sends it to Alice.
10. Alice computes $(g^c)^a = g^{ca}$ and sends it to Bob.
11. Bob computes $(g^{ca})^b = g^{cab} = g^{abc}$ and uses it as his secret.

An eavesdropper has been able to see g^a , g^b , g^c , g^{ab} , g^{ac} , and g^{bc} , but cannot use any combination of these to reproduce g^{abc} .

To extend this mechanism to larger groups, two basic principles must be followed:

- Starting with an "empty" key consisting only of g , the secret is made by raising the current value to every participant's private exponent once, in any order (the first such exponentiation yields the participant's own public key).

- Any intermediate value (having up to $N-1$ exponents applied, where N is the number of participants in the group) may be revealed publicly, but the final value (having had all N exponents applied) constitutes the shared secret and hence must never be revealed publicly. Thus, each user must obtain their copy of the secret by applying their own private key last (otherwise there would be no way for the last contributor to communicate the final key to its recipient, as that last contributor would have turned the key into the very secret the group wished to protect).

These principles leave open various options for choosing in which order participants contribute to keys. The simplest and most obvious solution is to arrange the N participants in a circle and have N keys rotate around the circle, until eventually every key has been contributed to by all N participants (ending with its owner) and each participant has contributed to N keys (ending with their own). However, this requires that every participant perform N modular exponentiations.

By choosing a more optimal order, and relying on the fact that keys can be duplicated, it is possible to reduce the number of modular exponentiations performed by each participant to $\log_2(N)+1$ using a divide-and-conquer-style approach, given here for eight participants:

1. Participants A, B, C, and D each perform one exponentiation, yielding g^{abcd} ; this value is sent to E, F, G, and H. In return, participants A, B, C, and D receive g^{efgh} .

2. Participants A and B each perform one exponentiation, yielding g^{efghab} , which they send to C and D, while C and D do the same, yielding g^{efghcd} , which they send to A and B.

3. Participant A performs an exponentiation, yielding $g^{efghcda}$, which it sends to B; similarly, B sends $g^{efghcdb}$ to A. C and D do similarly.

4. Participant A performs one final exponentiation, yielding the secret $g^{efghcdba} = g^{abcdefgh}$, while B does the same to get $g^{efghcdab} = g^{abcdefgh}$; again, C and D do similarly.

5. Participants E through H simultaneously perform the same operations using g^{abcd} as their starting point.

Once this operation has been completed all participants will possess the secret $g^{abcdefgh}$, but each participant will have performed only four modular exponentiations, rather than the eight implied by a simple circular arrangement.

3 Security

The protocol is considered secure against eavesdroppers if *G* and *g* are chosen properly. The eavesdropper ("Eve") would have to solve the Diffie–Hellman problem to obtain g^{ab}. This is currently considered difficult. An efficient algorithm to solve the discrete logarithm problem would make it easy to compute *a* or *b* and solve the Diffie–Hellman problem, making this and many other public key cryptosystems insecure. Fields of small characteristic may be less secure.[4]

The order of *G* should have a large prime factor to prevent use of the Pohlig–Hellman algorithm to obtain *a* or *b*. For this reason, a Sophie Germain prime *q* is sometimes used to calculate $p = 2q + 1$, called a safe prime, since the order of *G* is then only divisible by 2 and *q*. *g* is then sometimes chosen to generate the order *q* subgroup of *G*, rather than *G*, so that the Legendre symbol of g^a never reveals the low order bit of *a*.

If Alice and Bob use random number generators whose outputs are not completely random and can be predicted to some extent, then Eve's task is much easier.

The secret integers *a* and *b* are discarded at the end of the session. Therefore, Diffie–Hellman key exchange by itself trivially achieves perfect forward secrecy because no long-term private keying material exists to be disclosed.

In the original description, the Diffie–Hellman exchange by itself does not provide authentication of the communicating parties and is thus vulnerable to a man-in-the-middle attack. Mallory may establish two distinct key exchanges, one with Alice and the other with Bob, effectively masquerading as Alice to Bob, and vice versa, allowing her to decrypt, then re-encrypt, the messages passed between them. Note that Mallory must continue to be in the middle, transferring messages every time Alice and Bob communicate. If she is ever absent, her previous presence is then revealed to Alice and Bob. They will know that all of their private conversations had been intercepted and decoded by someone in the channel.

A method to authenticate the communicating parties to each other is generally needed to prevent this type of attack. Variants of Diffie–Hellman, such as STS protocol, may be used instead to avoid these types of attacks.

4 Other uses

4.1 Password-authenticated key agreement

When Alice and Bob share a password, they may use a password-authenticated key agreement (PAKE) form of Diffie–Hellman to prevent man-in-the-middle attacks. One simple scheme is to compare the hash of **s** concatenated with the password calculated independently on both ends of channel. A feature of these schemes is that an attacker can only test one specific password on each iteration with the other party, and so the system provides good security with relatively weak passwords. This approach is described in ITU-T Recommendation X.1035, which is used by the G.hn home networking standard.

4.2 Public key

It is also possible to use Diffie–Hellman as part of a public key infrastructure. Alice's public key is simply $(g^a \bmod p, g, p)$. To send her a message, Bob chooses a random *b* and then sends Alice $g^b \bmod p$ (un-encrypted) together with the message encrypted with symmetric key $(g^a)^b \bmod p$. Only Alice can decrypt the message because only she has *a* (the private key). A preshared public key also prevents man-in-the-middle attacks.

In practice, Diffie–Hellman is not used in this way, with RSA being the dominant public key algorithm. This is largely for historical and commercial reasons, namely that RSA Security created a certificate authority for key signing that became Verisign. Diffie–Hellman cannot be used to sign certificates. However, the ElGamal and DSA signature algorithms are mathematically related to it, as well as MQV, STS and the IKE component of the IPsec protocol suite for securing Internet Protocol communications.

4.3 Cryptocurrency

The sender can produce only the public part of the key, whereas only the receiver can compute the private part. Because of that, the receiver is the only one who can release the funds after the transaction is committed. They need to perform a single-formula check on each transactions to establish if it belongs to them. This process involves their private key, therefore no third party can perform this check and discover the link between the one-time key generated by the sender and the receiver's unique public address.

5 See also

- Key exchange
- Cryptography portal

- Modular arithmetic
- Elliptic curve Diffie–Hellman
- Public-key cryptography
- ElGamal encryption
- Diffie–Hellman problem
- MQV
- Password-authenticated key agreement
- Secure Remote Password Protocol

6 Notes

[1] Synonyms of Diffie–Hellman key exchange include:

- Diffie–Hellman key agreement
- Diffie–Hellman key establishment
- Diffie–Hellman key negotiation
- Exponential key exchange
- Diffie–Hellman protocol
- Diffie–Hellman handshake

7 References

[1] "GCHQ trio recognised for key to secure shopping online". *BBC*. 5 October 2010. Retrieved 5 August 2014.

[2] IEEE Communications Magazine Homepage | IEEE Communications Society. Comsoc.org. Retrieved on 2013-10-29.

[3] http://buchananweb.co.uk/security02.aspx

[4] A Heuristic Quasi-Polynomial Algorithm for Discrete Logarithm in Finite Fields of Small Characteristic," Razvan Barbulescu, Pierrick Gaudry, Antoine Joux, Emmanuel Thomé, Advances in Cryptology – EUROCRYPT 2014, Lecture Notes in Computer Science, Volume 8441, 2014, pp 1-16.

- Dieter Gollmann (2006). *Computer Security Second Edition* West Sussex, England: John Wiley & Sons, Ltd.
- The possibility of Non-Secret digital encryption J. H. Ellis, January 1970.
- Non-Secret Encryption Using a Finite Field MJ Williamson, January 21, 1974.
- Thoughts on Cheaper Non-Secret Encryption MJ Williamson, August 10, 1976.
- Diffie, W.; Hellman, M. (1976). "New directions in cryptography". *IEEE Transactions on Information Theory* **22** (6): 644–654. doi:10.1109/TIT.1976.1055638.
- Cryptographic apparatus and method Martin E. Hellman, Bailey W. Diffie, and Ralph C. Merkle, U.S. Patent #4,200,770, 29 April 1980
- The History of Non-Secret Encryption JH Ellis 1987 (28K PDF file) (HTML version)
- The First Ten Years of Public-Key Cryptography Whitfield Diffie, Proceedings of the IEEE, vol. 76, no. 5, May 1988, pp: 560–577 (1.9MB PDF file)
- Menezes, Alfred; van Oorschot, Paul; Vanstone, Scott (1997). *Handbook of Applied Cryptography* Boca Raton, Florida: CRC Press. ISBN 0-8493-8523-7. (Available online)
- Singh, Simon (1999) *The Code Book: the evolution of secrecy from Mary Queen of Scots to quantum cryptography* New York: Doubleday ISBN 0-385-49531-5
- An Overview of Public Key Cryptography Martin E. Hellman, IEEE Communications Magazine, May 2002, pp:42–49. (123kB PDF file)

8 External links

- Public Key Cryptography: Diffie-Hellman Key Exchange - A 5 minute YouTube video by Khan Academy faculty member Brit Cruise
- Oral history interview with Martin Hellman, Charles Babbage Institute, University of Minnesota. Leading cryptography scholar Martin Hellman discusses the circumstances and fundamental insights of his invention of public key cryptography with collaborators Whitfield Diffie and Ralph Merkle at Stanford University in the mid-1970s.
- RFC 2631 – *Diffie–Hellman Key Agreement Method* E. Rescorla June 1999.
- Summary of ANSI X9.42: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography (64K PDF file) (Description of ANSI 9 Standards)
- Diffie–Hellman Key Exchange – A Non-Mathematician's Explanation by Keith Palmgren
- Crypt::DH Perl module from CPAN
- Hands-on Diffie–Hellman demonstration
- C implementation using GNU Multiple Precision Arithmetic Library
- Diffie Hellman in 2 lines of Perl (using dc)
- Smart Account Management (SAcct) (using DH key exchange to derive session key)
- Talk by Martin Hellman in 2007, Google video (broken link)

9 Text and image sources, contributors, and licenses