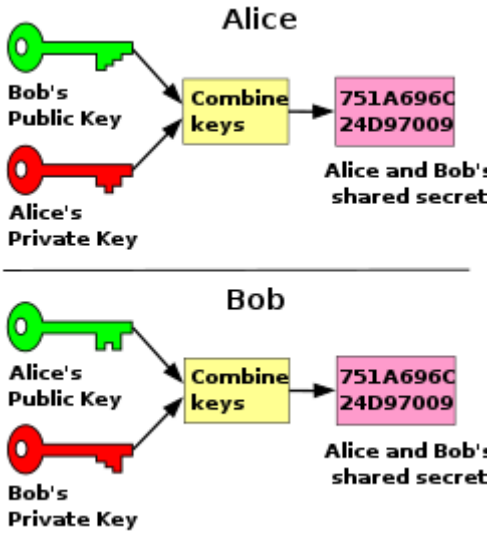
Diffie–Hellman key exchange

Diffie–Hellman key exchange[nb 1] is a mathematical method of securely exchanging cryptographic keys over a public channel and was one of the first public-key protocols as conceived by Ralph Merkle and named after Whitfield Diffie and Martin Hellman.[1][2] DH is one of the earliest practical examples of public key exchange implemented within the field of cryptography. Published in 1976 by Diffie and Hellman, this is the earliest publicly known work that proposed the idea of a private key and a corresponding public key.

Traditionally, secure encrypted communication between two parties required that they first exchange keys by some secure physical means, such as paper key lists transported by a trusted courier. 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 channel. This key can then be used to encrypt subsequent communications using a symmetric-key cipher.

Diffie–Hellman is used to secure a variety of Internet services. However, research published in October 2015 suggests that the parameters in use for many DH Internet applications at that time are not strong enough to prevent compromise by very well-funded attackers, such as the security services of some countries.[3]



In the Diffie–Hellman key exchange scheme, each party generates a public/private key pair and distributes the public key. After obtaining an authentic copy of each other's public keys, Alice and Bob can compute a shared secret offline. The shared secret can be used, for instance, as the key for a symmetric cipher.

The scheme was published by Whitfield Diffie and Martin Hellman in 1976,[2] but in 1997 it was revealed that James H. Ellis,[4] Clifford Cocks, and Malcolm J. Williamson of GCHQ, the British signals intelligence agency, had previously shown in 1969[5] how public-key cryptography could be achieved.[6]

Although Diffie–Hellman key agreement itself is a non-authenticated key-agreement protocol, it provides the basis for a variety of authenticated protocols, and is used to provide 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.

Expired US patent 4,200,770[7] from 1977 describes the now public-domain algorithm. It credits Hellman, Diffie, and Merkle as inventors.

Name

In 2006, 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, 2006), 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.[8]

Description

General overview

Diffie–Hellman key exchange establishes a shared secret between two parties that can be used for secret communication for exchanging data over a public network. An analogy illustrates the concept of public key exchange by using colors instead of very large numbers:

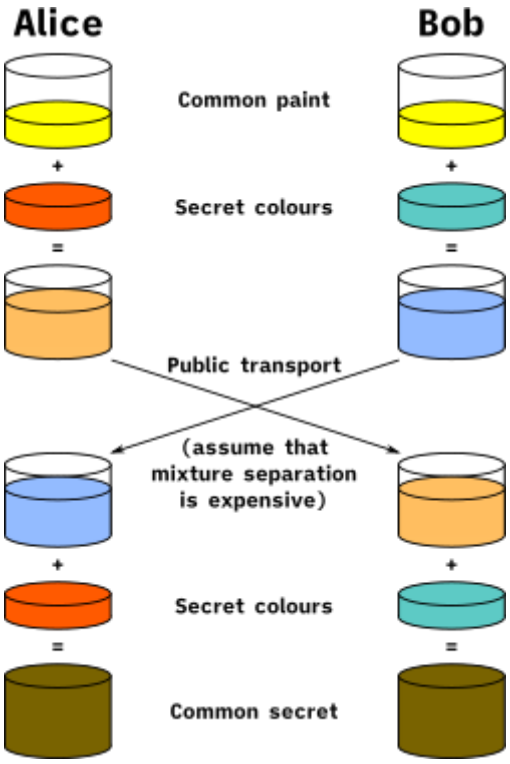


Illustration of the concept behind Diffie–Hellman key exchange

The process begins by having the two parties, Alice and Bob, publicly agree on an arbitrary starting color that does not need to be kept secret. In this example, the color is yellow. Each person also selects a secret color that they keep to themselves – in this case, red and cyan. The crucial part of the process is that Alice and Bob each mix their own secret color together with their mutually shared color, resulting in orange-tan and light-blue mixtures respectively, and then publicly exchange the two mixed colors. Finally, each of them mixes the color they received from the partner with their own private color. The result is a final color mixture (yellow-brown in this case) that is identical to their partner's final color mixture.

If a third party listened to the exchange, they would only know the common color (yellow) and the first mixed colors (orange-tan and light-blue), but it would be very hard for them to find out the final secret color (yellow-brown). Bringing the analogy back to a real-life exchange using large numbers rather than colors, this determination is computationally expensive. It is impossible to compute in a practical amount of time even for modern supercomputers.

Cryptographic explanation

The simplest and the original implementation,[2] later formalized as **Finite Field Diffie-Hellman** in *RFC 7919*,[9] 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 $\mathbf{a} = 4$, then sends Bob $A = g^{\mathbf{a}} \bmod p$

- $A = 5^4 \bmod 23 = 4$ (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 **b** = 3, then sends Alice $B = g^b \bmod p$
   - $B = 5^3 \bmod 23 = 10$
4. Alice computes $s = B^a \bmod p$
   - $s = 10^4 \bmod 23 = 18$
5. Bob computes $s = A^b \bmod p$
   - $s = 4^3 \bmod 23 = 18$
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 $g^b \bmod p$. 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 600 digits, then even the fastest modern computers using the fastest known algorithm cannot find a given only g, p and $g^a \bmod p$. Such a problem is called the discrete logarithm problem.[3] 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 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. $p = 23$
- **a**, Alice's private key, known only to Alice. **$a = 6$**
- **b**, Bob's private key known only to Bob. **$b = 15$**
- 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	**Unknown**	**Known**	**Unknown**	**Known**	**Unknown**
$p = 23$		$p = 23$		$p = 23$	
$g = 5$		$g = 5$		$g = 5$	
$\mathbf{a = 6}$	$\mathbf{b}$	$\mathbf{b = 15}$	$\mathbf{a}$		$\mathbf{a}$, $\mathbf{b}$
$A = 5^{\mathbf{a}} \bmod 23$		$B = 5^{\mathbf{b}} \bmod 23$			
$A = 5^{\mathbf{6}} \bmod 23 = 8$		$B = 5^{\mathbf{15}} \bmod 23 = 19$			
$\mathbf{B = 19}$		$\mathbf{A = 8}$		$A = 8$, $B = 19$	
$\mathbf{s} = B^{\mathbf{a}} \bmod 23$		$\mathbf{s} = A^{\mathbf{b}} \bmod 23$			
$\mathbf{s} = 19^{\mathbf{6}} \bmod 23 = \mathbf{2}$		$\mathbf{s} = 8^{\mathbf{15}} \bmod 23 = \mathbf{2}$			$\mathbf{s}$

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 $g^{a+b} \bmod p$.

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:[10]

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 is 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.[11]

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 "Key Agreement Protocols and their Security Analysis (1997)",[12] which was improved by C. Kudla and K. G. Paterson in “Modular Security Proofs for Key Agreement Protocols (2005)”[13] and shown to be secure. It's also used or mentioned in other variants. For example:

- Extended Triple Diffie-Hellman
- sci.crypt news group (from 18.08.2002)[14]
- Double Ratchet Algorithm
- Signal Protocol

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

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, one could e.g. have a look at "Advanced modular handshake for key agreement and optional authentication"[15]

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 $\boldsymbol{g^a \bmod p}$ and sends it to Bob.
4. Bob computes $\boldsymbol{(g^a)^b \bmod p = g^{ab} \bmod p}$ and sends it to Carol.
5. Carol computes $\boldsymbol{(g^{ab})^c \bmod p = g^{abc} \bmod p}$ and uses it as her secret.
6. Bob computes $\boldsymbol{g^b \bmod p}$ and sends it to Carol.
7. Carol computes $\boldsymbol{(g^b)^c \bmod p = g^{bc} \bmod p}$ and sends it to Alice.
8. Alice computes $\boldsymbol{(g^{bc})^a \bmod p = g^{bca} \bmod p = g^{abc} \bmod p}$ and uses it as her secret.
9. Carol computes $\boldsymbol{g^c \bmod p}$ and sends it to Alice.
10. Alice computes $\boldsymbol{(g^c)^a \bmod p = g^{ca} \bmod p}$ and sends it to Bob.
11. Bob computes $\boldsymbol{(g^{ca})^b \bmod p = g^{cab} \bmod p = g^{abc} \bmod p}$ and uses it as his secret.

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

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 desirable 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.

Security

The protocol is considered secure against eavesdroppers if G and g are chosen properly. In particular, the order of the group G must be large, particularly if the same group is used for large amounts of traffic. The eavesdropper has to solve the Diffie–Hellman problem to obtain g^{ab}. This is currently considered difficult for groups whose order is large enough. 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.[16]

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. A protocol using such a choice is for example IKEv2.[17]

g is often a small integer such as 2. Because of the random self-reducibility of the discrete logarithm problem a small g is equally secure as any other generator of the same group.

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

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 (an active attacker executing the man-in-the-middle attack) 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, actively decrypting and re-encrypting 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. In most cases it will not help them get Mallory's private key, even if she used the same key for both exchanges.

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.

Practical attacks on Internet traffic

The number field sieve algorithm, which is generally the most effective in solving the discrete logarithm problem, consists of four computational steps. The first three steps only depend on the order of the group G, not on the specific number whose finite log is desired.[18] It turns out that much Internet traffic uses one of a handful of groups that are of order 1024 bits or less.[3] By precomputing the first three steps of the

number field sieve for the most common groups, an attacker need only carry out the last step, which is much less computationally expensive than the first three steps, to obtain a specific logarithm. The Logjam attack used this vulnerability to compromise a variety of Internet services that allowed the use of groups whose order was a 512-bit prime number, so called export grade. The authors needed several thousand CPU cores for a week to precompute data for a single 512-bit prime. Once that was done, individual logarithms could be solved in about a minute using two 18-core Intel Xeon CPUs.[3]

As estimated by the authors behind the Logjam attack, the much more difficult precomputation needed to solve the discrete log problem for a 1024-bit prime would cost on the order of $100 million, well within the budget of a large national intelligence agency such as the U.S. National Security Agency (NSA). The Logjam authors speculate that precomputation against widely reused 1024-bit DH primes is behind claims in leaked NSA documents that NSA is able to break much of current cryptography.[3]

To avoid these vulnerabilities, the Logjam authors recommend use of elliptic curve cryptography, for which no similar attack is known. Failing that, they recommend that the order, p, of the Diffie–Hellman group should be at least 2048 bits. They estimate that the pre-computation required for a 2048-bit prime is 10^9 times more difficult than for 1024-bit primes.[3]

Other uses

Encryption

Public key encryption schemes based on the Diffie–Hellman key exchange have been proposed. The first such scheme is the ElGamal encryption. A more modern variant is the Integrated Encryption Scheme.

Forward secrecy

Protocols that achieve forward secrecy generate new key pairs for each session and discard them at the end of the session. The Diffie–Hellman key exchange is a frequent choice for such protocols, because of its fast key generation.

Password-authenticated key agreement

When Alice and Bob share a password, they may use a password-authenticated key agreement (PK) 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.

An example of such a protocol is the Secure Remote Password protocol.

Public key

It is also possible to use Diffie–Hellman as part of a public key infrastructure, allowing Bob to encrypt a message so that only Alice will be able to decrypt it, with no prior communication between them other than Bob having trusted knowledge of Alice's public key. Alice's public key is $(g^a \bmod p, g, p)$. To send her a message, Bob chooses a random b and then sends Alice $g^b \bmod p$ (unencrypted) together with the

message encrypted with symmetric key $(g^a)^b \bmod p$. Only Alice can determine the symmetric key and hence decrypt the message because only she has *a* (the private key). A pre-shared 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, as elaborated above, cannot directly 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.

See also

- Elliptic-curve Diffie–Hellman key exchange
- Supersingular isogeny key exchange
- Forward secrecy

Notes

1. Synonyms of Diffie–Hellman key exchange include:
   - Diffie–Hellman–Merkle key exchange
   - Diffie–Hellman key agreement
   - Diffie–Hellman key establishment
   - Diffie–Hellman key negotiation
   - Exponential key exchange
   - Diffie–Hellman protocol
   - Diffie–Hellman handshake

References

1. Merkle, Ralph C. (April 1978). "Secure Communications Over Insecure Channels". *Communications of the ACM*. **21** (4): 294–299. CiteSeerX 10.1.1.364.5157 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.364.5157). doi:10.1145/359460.359473 (https://doi.org/10.1145%2F359460.359473). S2CID 6967714 (https://api.semanticscholar.org/CorpusID:6967714). "Received August, 1975; revised September 1977"
2. Diffie, Whitfield; Hellman, Martin E. (November 1976). "New Directions in Cryptography" (http://ee.stanford.edu/%7Ehellman/publications/24.pdf) (PDF). *IEEE Transactions on Information Theory*. **22** (6): 644–654. CiteSeerX 10.1.1.37.9720 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.37.9720). doi:10.1109/TIT.1976.1055638 (https://doi.org/10.1109%2FTIT.1976.1055638). Archived (https://web.archive.org/web/20141129035850/https://ee.stanford.edu/%7Ehellman/publications/24.pdf) (PDF) from the original on 2014-11-29.
3. Adrian, David; et al. (October 2015). "Imperfect Forward Secrecy: How Diffie–Hellman Fails in Practice" (https://weakdh.org/imperfect-forward-secrecy-ccs15.pdf) (PDF). Archived (https://web.archive.org/web/20150906213656/https://weakdh.org/imperfect-forward-secrecy-ccs15.pdf) (PDF) from the original on 2015-09-06.

4. Ellis, J. H. (January 1970). "The possibility of Non-Secret digital encryption" (https://web.archive.org/web/20141030210530/https://cryptocellar.web.cern.ch/cryptocellar/cesg/possnse.pdf) (PDF). *CESG Research Report*. Archived from the original (http://cryptocellar.web.cern.ch/cryptocellar/cesg/possnse.pdf) (PDF) on 2014-10-30. Retrieved 2015-08-28.
5. "The Possibility of Secure Secret Digital Encryption" (https://www.gchq.gov.uk/sites/default/files/document_files/CESG_Research_Report_No_3006_0.pdf) (PDF). Archived (https://web.archive.org/web/20170216051636/https://www.gchq.gov.uk/sites/default/files/document_files/CESG_Research_Report_No_3006_0.pdf) (PDF) from the original on 2017-02-16. Retrieved 2017-07-08.
6. "GCHQ trio recognised for key to secure shopping online" (https://www.bbc.co.uk/news/uk-england-gloucestershire-11475101). *BBC News*. 5 October 2010. Archived (https://web.archive.org/web/20140810044800/http://www.bbc.co.uk/news/uk-england-gloucestershire-11475101) from the original on 10 August 2014. Retrieved 5 August 2014.
7. US patent 4200770 (https://worldwide.espacenet.com/textdoc?DB=EPODOC&IDX=US4200770)
8. Hellman, Martin E. (May 2002), "An overview of public key cryptography" (http://www-ee.stanford.edu/~hellman/publications/31.pdf) (PDF), *IEEE Communications Magazine*, **40** (5): 42–49, CiteSeerX 10.1.1.127.2652 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.127.2652), doi:10.1109/MCOM.2002.1006971 (https://doi.org/10.1109%2FMCOM.2002.1006971), S2CID 9504647 (https://api.semanticscholar.org/CorpusID:9504647), archived (https://web.archive.org/web/20160402093741/http://www-ee.stanford.edu/%7Ehellman/publications/31.pdf) (PDF) from the original on 2016-04-02
9. Wong, David (2021). "Key exchange standards" (https://archive.today/20200921005545/https://freecontent.manning.com/key-exchange-standards/). *Real World Cryptography* (https://books.google.com/books?id=Qd5CEAAAQBAJ). Manning. ISBN 9781617296710 – via Google Books.
10. Buchmann, Johannes A. (2013). *Introduction to Cryptography* (https://books.google.com/books?id=BuQlBQAAQBAJ&pg=PA190) (Second ed.). Springer Science+Business Media. pp. 190–191. ISBN 978-1-4419-9003-7.
11. "An efficient key recovery attack on SIDH" (https://eprint.iacr.org/2022/975.pdf) (PDF).
12. Blake-Wilson, Simon; Johnson, Don; Menezes, Alfred (1997), *Key Agreement Protocols and their Security Analysis*, CiteSeerX 10.1.1.25.387 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.25.387)
13. Kudla, Caroline; Paterson, Kenneth G. (2005). "Modular Security Proofs for Key Agreement Protocols". In Roy, Bimal (ed.). *Advances in Cryptology - ASIACRYPT 2005*. Lecture Notes in Computer Science. Vol. 3788. Berlin, Heidelberg: Springer. pp. 549–565. doi:10.1007/11593447_30 (https://doi.org/10.1007%2F11593447_30). ISBN 978-3-540-32267-2.
14. "Triple Diffie-Hellman (ECC compatible). Any attacks against it?" (https://groups.google.com/g/sci.crypt/c/ZhR98p3fpXk). Retrieved 2021-11-25.
15. US11025421B2 (https://patents.google.com/patent/US11025421B2/en?oq=11025421), Fay, Bjorn, "Advanced modular handshake for key agreement and optional authentication", issued 2021-06-01

16. Barbulescu, Razvan; Gaudry, Pierrick; Joux, Antoine; Thomé, Emmanuel (2014). "A Heuristic Quasi-Polynomial Algorithm for Discrete Logarithm in Finite Fields of Small Characteristic" (http://hal.inria.fr/docs/00/90/90/87/PDF/article.pdf) (PDF). *Advances in Cryptology – EUROCRYPT 2014*. Proceedings 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques. *Lecture Notes in Computer Science*. Vol. 8441. Copenhagen, Denmark. pp. 1–16. doi:10.1007/978-3-642-55220-5_1 (https://doi.org/10.1007%2F978-3-642-55220-5_1). ISBN 978-3-642-55220-5. Archived (https://web.archive.org/web/20200322030320/https://hal.inria.fr/docs/00/90/90/87/PDF/article.pdf) (PDF) from the original on 2020-03-22.
17. "RFC 4306 Internet Key Exchange (IKEv2) Protocol". Internet Engineeringrg/web/20150107073645/http://www.ietf.org/rfc/rfc4306.txt.
18. Whitfield Diffie, Paul C. Van Oorschot, and Michael J. Wiener "Authentication and Authenticated Key Exchanges", in Designs, Codes and Cryptography, 2, 107–125 (1992), Section 5.2, available as Appendix B to U.S. Patent 5,724,425 (https://patents.google.com/patent/US5724425)

General references

- Gollman, Dieter (2011). *Computer Security* (2nd ed.). West Sussex, England: John Wiley & Sons, Ltd. ISBN 978-0470741153.
- Williamson, Malcolm J. (January 21, 1974). *Non-secret encryption using a finite field* (https://www.gchq.gov.uk/sites/default/files/document_files/nonsecret_encryption_finite_field_0.pdf) (PDF) (Technical report). Communications Electronics Security Group. Archived (https://web.archive.org/web/20170323052715/https://www.gchq.gov.uk/sites/default/files/document_files/nonsecret_encryption_finite_field_0.pdf) (PDF) from the original on 2017-03-23. Retrieved 2017-03-22.
- Williamson, Malcolm J. (August 10, 1976). *Thoughts on Cheaper Non-Secret Encryption* (http://www.fi.muni.cz/usr/matyas/lecture/paper3.pdf) (PDF) (Technical report). Communications Electronics Security Group. Archived (https://web.archive.org/web/20040719085349/http://www.fi.muni.cz/usr/matyas/lecture/paper3.pdf) (PDF) from the original on 2004-07-19. Retrieved 2015-08-25.
- The History of Non-Secret Encryption (https://web.archive.org/web/20130404174201/https://cryptocellar.web.cern.ch/cryptocellar/cesg/ellis.pdf) JH Ellis 1987 (28K PDF file) (HTML version (https://web.archive.org/web/20040808040209/http://jya.com/ellisdoc.htm))
- The First Ten Years of Public-Key Cryptography (http://cr.yp.to/bib/1988/diffie.pdf) 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 (http://www.cacr.math.uwaterloo.ca/hac/))
- 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 (https://dx.doi.org/10.1109/MCOM.2002.1006971) Martin E. Hellman, IEEE Communications Magazine, May 2002, pp. 42–49. (123kB PDF file)

External links

- Oral history interview with Martin Hellman (https://conservancy.umn.edu/handle/11299/107353), 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 (https://datatracker.ietf.org/doc/html/rfc2631) – *Diffie–Hellman Key Agreement Method*. E. Rescorla. June 1999.
- RFC 3526 (https://datatracker.ietf.org/doc/html/rfc3526) – *More Modular Exponential (MODP) Diffie–Hellman groups for Internet Key Exchange (IKE)*. T. Kivinen, M. Kojo, SSH Communications Security. May 2003.
- Summary of ANSI X9.42: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography (https://web.archive.org/web/20040903080553/http://csrc.nist.gov/CryptoToolkit/kms/summary-x9-42.pdf) (64K PDF file) (Description of ANSI 9 Standards (https://web.archive.org/web/20040816210145/http://www.rsasecurity.com/rsalabs/node.asp?id=2306))
- Talk by Martin Hellman in 2007, YouTube video (https://www.youtube.com/watch?v=zTGqP0nxX08)
- Crypto dream team Diffie & Hellman wins $1M 2015 Turing Award (a.k.a. "Nobel Prize of Computing") (https://www.networkworld.com/article/3039820/security/crypto-dream-team-diffie-hellman-win-nobel-prize-of-computing.html)
- A Diffie–Hellman demo written in Python3 (http://neilrieck.net/dh_demo.html) – This demo properly supports very-large key data and enforces the use of prime numbers where required.
