

# Diffie-Hellman Key Exchange: A Non-mathematician's explanation

By Keith Palmgren

**You care about Diffie-Hellman because it is one of the most common protocols used in networking today. This is true, even though the vast majority of the time the user has no idea it is operating.**

A colleague once asked if I could help him understand the Diffie-Hellman key exchange protocol without digging through the math. I told him yes, I could, but not easily. Doing so requires a few diagrams because, in this particular case, a picture is worth several complex equations!

First things first. What is Diffie-Hellman (DH), and why do you care? DH is a mathematical algorithm that allows two computers to generate an identical shared secret on both systems, even though those systems may never have communicated with each other before. That shared secret can then be used to securely exchange a cryptographic encryption key. That key then encrypts traffic between the two systems.

You care about Diffie-Hellman because it is one of the most common protocols used in networking today. This is true, even though the vast majority of the time the user has no idea it is operating. DH is commonly used when you encrypt data on the Web using either SSL (Secure Socket Layer) or TLS (Transport Layer Security). The Secure Shell (SSH) protocol also utilizes DH. Of course, because DH is part of the key exchange mechanism for IPsec, any VPN based on that technology utilizes DH as well<sup>1</sup>.

It is true that a VPN or SSL system could be in use for years without the system administrator knowing anything about Diffie-Hellman. However, I find that an understanding of underlying protocols and processes helps a great deal when trouble-shooting a system. That does not mean we have to completely understand the math behind the protocol. In fact, I have worked with encryption systems for years even though any mathematical operation more difficult than

balancing a checkbook baffles me completely. There are bona fide experts in the field of encryption mathematics. When they say a system is mathematically sound, I take them at their word. This frees me to concentrate on other issues, such as understanding how the background processes need to function in order to keep the system operational.

## Some detail

The Diffie-Hellman algorithm, introduced by Whitfield Diffie and Martin Hellman in 1976, was the first system to utilize “public-key” or “asymmetric” cryptographic keys. These systems overcome the difficulties of “private-key” or “symmetric” key systems because asymmetric key management is much easier. In a symmetric key system, both sides of the communication must have identical keys. Securely exchanging those keys has always been an issue of enormous concern. At one time the National Security Agency maintained an entire fleet of trucks and planes manned by armed couriers to shuttle around fifteen tons of paper-based symmetric key used by the US Government every year.

Businesses simply do not want to deal with that sort of burden. Asymmetric key systems alleviate this issue by using two keys: one called the “private key” that the user keeps secret, and one called the “public key” that can be shared with the world and therefore easily distributed.

Unfortunately, the advantages of asymmetric key systems are overshadowed by the need for speed – they are extremely slow for any sort of bulk encryption. Today, it is common practice to use a symmetric system to encrypt data and an asymmetric system to encrypt the symmetric keys for distribution. That is precisely what DH is capable of doing, and what it does do when used for key exchange as described here.

<sup>1</sup> The overall IPsec key management framework is Internet Security Association and Key Management Protocol (ISAKMP) from RFC 2408. Within that framework lies the Internet Key Exchange (IKE) protocol in RFC 2401. IKE relies on yet another protocol known as OAKLEY, and OAKLEY uses Diffie-Hellman as described in RFC 2412. This is an admittedly long trail to follow, but the conclusion is that DH is indeed a part of the IPsec standard.

## The process

Diffie-Hellman is not an encryption mechanism as we normally think of them, in that we do not typically use DH to encrypt data. Instead, it is a method for secure exchange of the keys that encrypt data. DH accomplishes this secure exchange by creating a “shared secret” (sometimes called a “Key Encryption Key” or KEK) between two devices. The shared secret then encrypts the symmetric key for secure transmittal. The symmetric key is sometimes called a “Traffic Encryption Key” (TEK) or “Data Encryption Dey” (DEK).

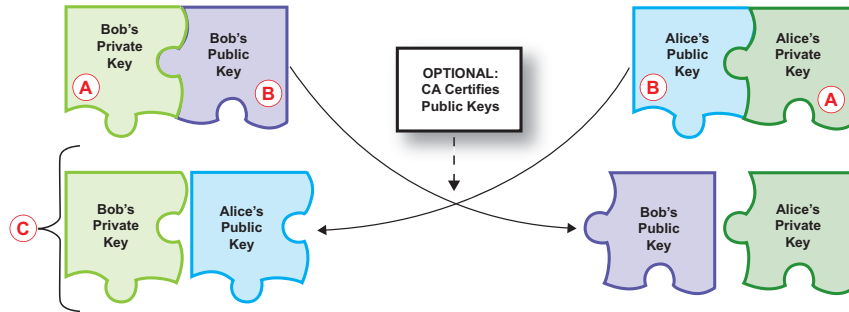


Figure 1 – Key Generation and Exchange

Therefore, the KEK provides for secure delivery of the TEK, while the TEK provides for secure delivery of the data itself.

The process begins when each side of the communication generates a private key (depicted by the letter A in Figure 1). Each side then generates a public key (letter B), which is a derivative of the private key. The two systems then exchange their public keys. Each side of the communication now has its own private key and the other system's public key (see the area labeled letter C in the diagrams).

It is important to note that the public key is a derivative of the private key – the two keys are mathematically linked. However, in order to trust this system, you must accept that you cannot discern the private key from the public key. Because the public key is indeed public and ends up on other systems, the ability to figure out the private key from it would render the system useless. This is one area requiring trust in the mathematical experts. The fact that the very best in the world have tried for years to defeat this and have failed, bolsters my confidence a great deal.

I should also explain the box labeled “OPTIONAL: CA Certifies Public Keys.” It is not common, but the capability does exist within the Diffie-Hellman protocol to have a Certificate Authority certify that the public key is indeed coming from the source in which you

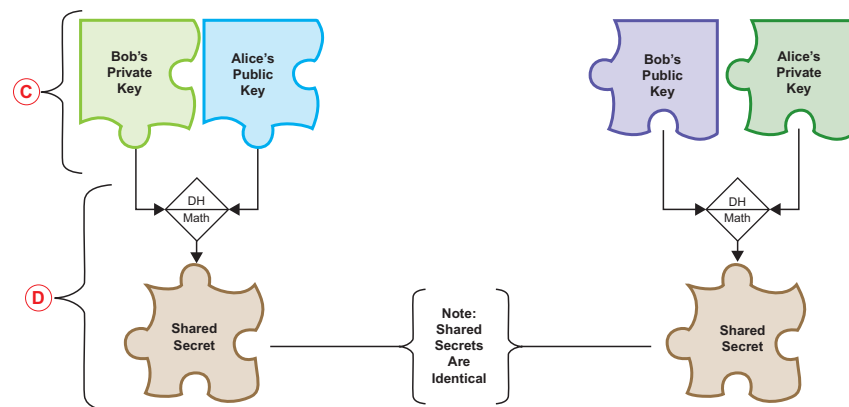


Figure 2 – Shared Secret Creation

believe. The purpose of this certification is to prevent man-in-the-middle (MITM) attacks. These attacks involve intercepting both public keys and then forwarding to both recipients the attacker's bogus public keys. The “man in the middle” can potentially intercept encrypted traffic, decrypt it, copy or modify it, re-encrypt it with the bogus key, and forward it on to its destination. If this is successful, the parties on each end have no idea of the unauthorized intermediary. It is an extremely difficult attack to pull off outside the laboratory, but it is certainly possible. Properly implemented Certificate Authority systems have the potential to disable the attack.

Once the key exchange is complete, the process continues. The Diffie-Hellman protocol generates a “shared secret” – an identical cryptographic key shared by each side of the communication. Figure 2 depicts this operation in the “DH Math” box. (Trust me, the actual mathematical equation is a good deal longer and more complex; a simple example appears later in this article.) By running the mathematical operation against your own private key and the other side's public key, you generate a value. When the distant end runs the same operation against your public key and its own private key, that end also generates a value. The important point is that the two values generated are identical. They are the “shared secret” that can encrypt information between systems.

At this point, the Diffie-Hellman operation could be considered complete. The shared secret is, after all, a cryptographic key that could encrypt traffic. But completion at this point is very rare, because the shared secret is an asymmetric key by its mathematical nature, and all asymmetric key systems are inherently slow. If the two sides are passing very little traffic, the shared secret may encrypt actual data.

But any attempt at bulk traffic encryption requires a symmetric key system, such as DES, Triple DES or Advanced Encryption Standard (AES). In most real applications of the DH protocol (SSL, TLS, SSH, and IPSec in particular), the shared secret encrypts a symmetric key for one of the symmetric algorithms, then transmits it securely, and the distant end decrypts it with the shared secret. (Figure 3 depicts this operation.) Because the symmetric key is a relatively short value (256 bits for example) as compared to bulk data, the shared secret can encrypt and decrypt it very quickly. Speed is less of an issue with short values.

Which side of the communication generates and transmits the symmetric key varies. However, it is more common for the initiator of the communication to be the one that transmits the key. I should also point out that some sort of negotiation typically occurs to decide on the symmetric algorithm, the mode of the algorithms (e.g., cipher block chaining, or CBC), hash functions (MD5, SHA-1, etc.), key lengths, refresh rates, and so on. That negotiation is handled by the application, and is not a part of Diffie-Hellman, but it is obviously an important task, since both sides must support the same schemes for encryption for it to function. This also points to why key-management planning is so important – and why poor key management so often leads to failure of systems.

Once secure exchange of the symmetric key is complete, data encryption and secure communication can occur (note that passing the

symmetric key is the whole point of the Diffie-Hellman operation). Figure 4 depicts data encrypted and decrypted on each end of the communication by the symmetric key.

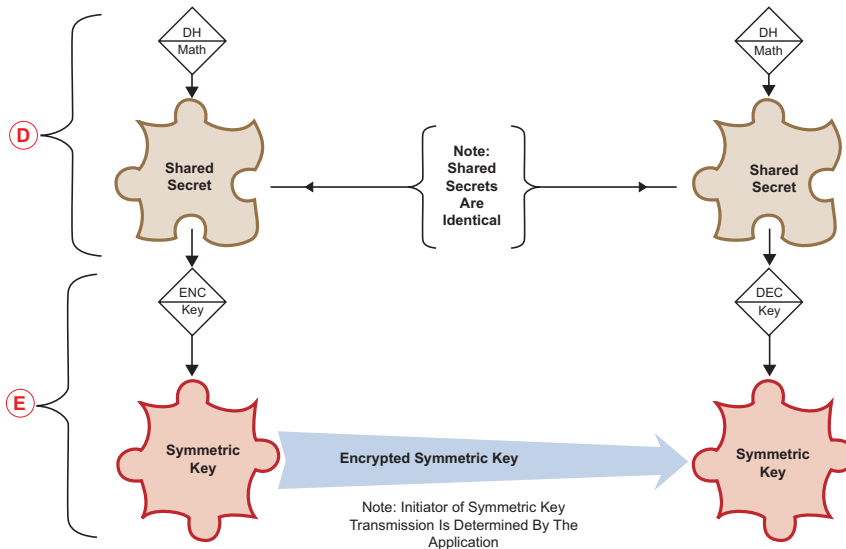


Figure 3 – Encrypting, Passing, and Decrypting the Symmetric Key

Changing the symmetric key for increased security is simple at this point. The longer the time a symmetric key is in use, the easier it is to perform a successful cryptanalytic attack against it. Therefore, changing keys frequently is important. Both sides of the communication still have the shared secret, and it can be used to encrypt future keys at any time and with any frequency desired. In some IPSec implementations, for example, it is not uncommon for a new symmetric data encryption key to be generated and shared every 60 seconds.

### A slightly more mathematical explanation

I am often asked for a better, though still simplified, explanation of exactly what happens in the “DH Math” portion of the diagram

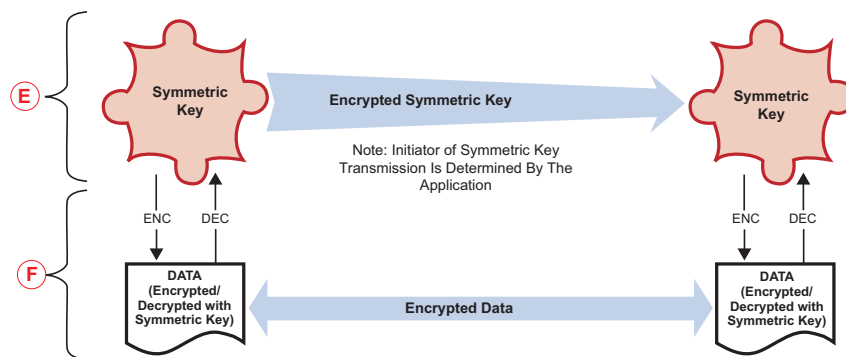


Figure 4 – Encrypted Data Transmission

above. The best such explanation I can find is contained in the “Computer Desktop Encyclopedia” (CDE) published by The Computer Language Co., Inc. (see www.computerlanguage.com). The following example is printed with their permission. Remember that this example uses only small numbers, such as 2, 16, 5 and 32. In reality, these would be very large numbers.

”Using a common number, both sides use a different random number as a power to raise the common number. The results are then sent to each other. The receiving party raises the received number to the same random power they used before, and the results are the same on both sides.

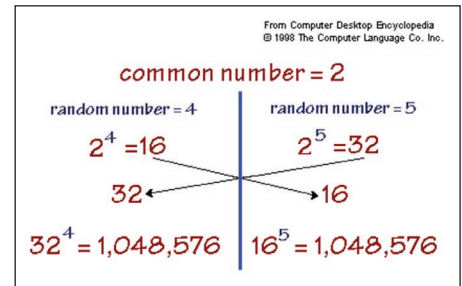


Figure 5 – Simplified “DH Math”

”It’s Very Clever. There is more computation in actual practice, but this example, which uses tiny numbers to illustrate the concept, shows a very clever mathematical approach. Each party raises the common number, which is 2 in this example (this has nothing to do with binary – it is just the number “2”) to a random power and sends the result to the other. The received number is raised to the same random power. Note that both parties come up with the same secret key, which was never transmitted intact.”

To relate the Computer Desktop Encyclopedia example here to the diagrams above or below: The number 1,048,576 is our shared secret at letter D. The numbers “16” and “32” are the public keys at letter B. The random numbers here (4 and 5) are the private keys lettered A. The common number (2 in this example) is not shown in the other diagrams, but would normally be passed system to system as part of the application’s negotiation.

### Summary

The use of Diffie-Hellman greatly reduces the headache of using symmetric key systems. These systems have astounding speed benefits, but managing their keys has always been difficult, to say the very least. Because the steps we have just gone through happen in a matter of a second or two, and are completely transparent to the user, ease of use could not be better – provided that DH works. The good news is, it almost always does work. Understanding the underlying protocol only becomes necessary in the rare case that it doesn’t work, and you need to troubleshoot the problem.

The complete Diffie-Hellman Key Exchange Diagram follows on the next page.

### About the Author

Keith Palmgren, CISSP, has over 20 years of experience as a security professional. He spent the majority of his career as a security consultant for various firms, including starting and running Sprint’s first International Security Consulting Practice. Currently, Keith is the president of NetIP, Inc. – A Knowledge Transfer Company. As such, he does freelance writing and teaches a variety of courses in the Global Knowledge Security curriculum.

## Diffie-Hellman Key Exchange

