



Global Knowledge™

Expert Reference Series of White Papers

Diffie-Hellman Key
Exchange – A
Non-Mathematician's
Explanation

Diffie-Hellman Key Exchange – A Non-Mathematician’s Explanation

Keith Palmgren, Global Knowledge Instructor, CISSP, Security+, TICSA



Opening Discussion

A colleague once asked if I could help him understand the Diffie-Hellman key exchange protocol . . . without digging through the math. My answer was, “Yes, I can, 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 should 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 should 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 happening. DH is commonly used when you encrypt data on the Web using either Secure Socket Layer (SSL) or Transport Layer Security (TLS). 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. (The overall IPsec key management framework is Internet Security Association and Key Management Protocol or ISAKMP from RFC 2408. Within that framework is the Internet Key Exchange, IKE, protocol in RFC 2401. IKE relies on yet another protocol known as OAKLEY, and it uses Diffie-Hellman as described in RFC 2412. It is an admittedly long trail to follow, but the result is that DH is, indeed, a part of the IPsec standard.)

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 believe them. 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.

Diffie-Hellman’s Background

The DH 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 enormous issue. At one time, the National Security Agency maintained an entire fleet of trucks and planes

manned by armed couriers to shuttle around 15 tons of paper-based symmetric key used by the U.S. government every year. Businesses simply do not want to mess with that sort of burden. Asymmetric key systems alleviate that issue because they use 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 speed—they are extremely slow for any sort of bulk encryption. Today, it is typical practice to use a symmetric system to encrypt the data and an asymmetric system to encrypt the symmetric keys for distribution. That is precisely what Diffie-Hellman is capable of doing—and does do—when used for key exchange as described here.

The Actual Process

Diffie-Hellman is not an encryption mechanism as we normally think of them, in that we do not typically use it to encrypt data. Instead, it is a method to securely exchange 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 Key (DEK). 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, next page). 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).

Noting that the public key is a derivative of the private key is important because 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 failed bolsters my confidence a great deal.

I should also explain the box labeled “Optional: CA Certifies Public Key.” It is not common, but the ability does exist with the Diffie-Hellman protocol to have a Certificate Authority certify that the public key is, indeed, coming from the source you think it is. The purpose of this certification is to prevent Man-In-the-Middle (MIM) attacks. The attack consists of someone intercepting both public keys and forwarding bogus public keys of their own. The “man in the middle” potentially intercepts encrypted traffic, decrypts it, copies or modifies it, re-encrypts it with the bogus key, and forwards it on to its destination. If successful, the parties on each end would have no idea that there is an 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.

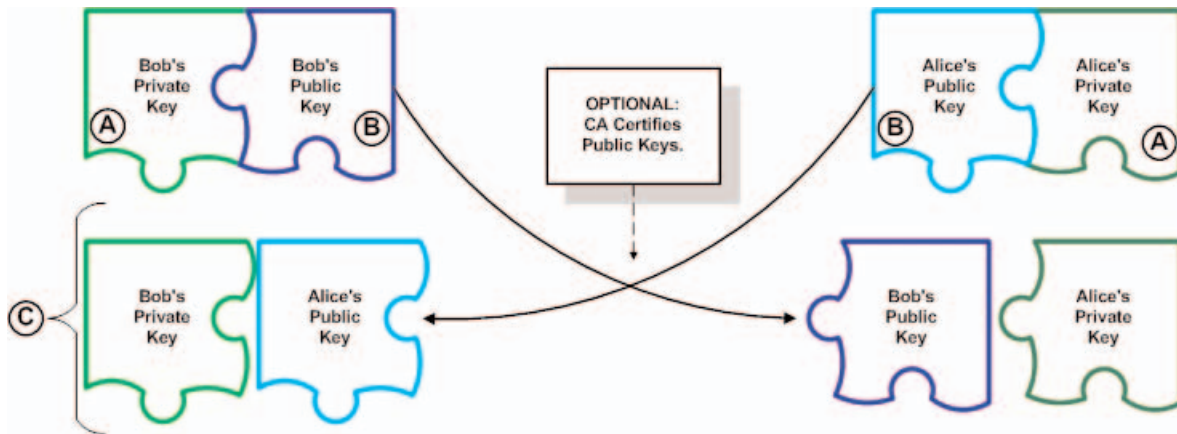


Figure 1*: Key Generation and Exchange

Once the key exchange is complete, the process continues. The DH protocol generates “shared secrets”—identical cryptographic keys shared by each side of the communication. Figure 2 depicts this operation with the “DH Math” box (trust me, the actual mathematical equation is a good deal longer and more complex; a simple example appears elsewhere 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 their own private key, they also generate a value. The important point is that the two values generated are identical. They are the “Shared Secret” that can encrypt information between systems.

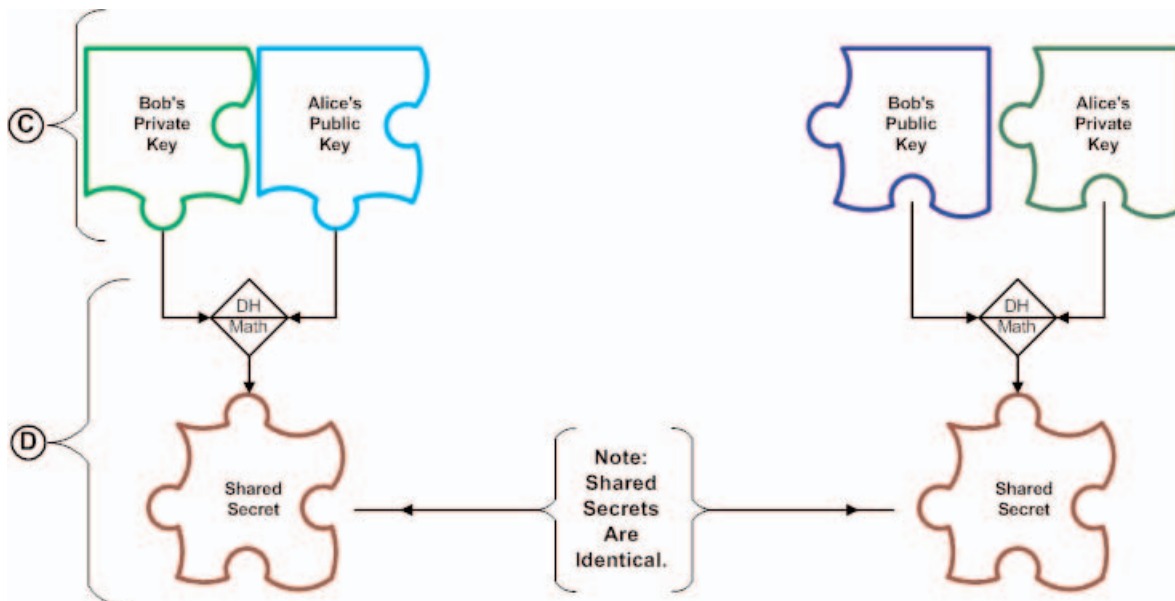


Figure 2: Shared Secret Creation

At this point, the DH operation could be considered complete. The shared secret is, after all, a cryptographic key that could encrypt traffic. That is very rare, however, because the shared secret is, by its mathematical nature, an asymmetric key. As with all asymmetric key systems, it is inherently slow. If the two sides are passing very little traffic, the shared secret may encrypt actual data. Any attempt at bulk traffic encryption requires

a symmetric key system such as DES, Triple DES, or Advanced Encryption Standard (AES), etc. In most real applications of the Diffie-Hellman protocol (SSL, TLS, SSH, and IPsec, in particular), the shared secret encrypts a symmetric key for one of the symmetric algorithms, 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 actually generates and transmits the symmetric key varies. However, it is most 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, mode of the algorithms (e.g., Cipher Block Chaining, etc.), hash functions (MD5, SHA1, 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 an obviously important task since both sides must support the same schemes for encryption to function. This also points out why key management planning is so important—and why poor key management so often leads to failure of systems.

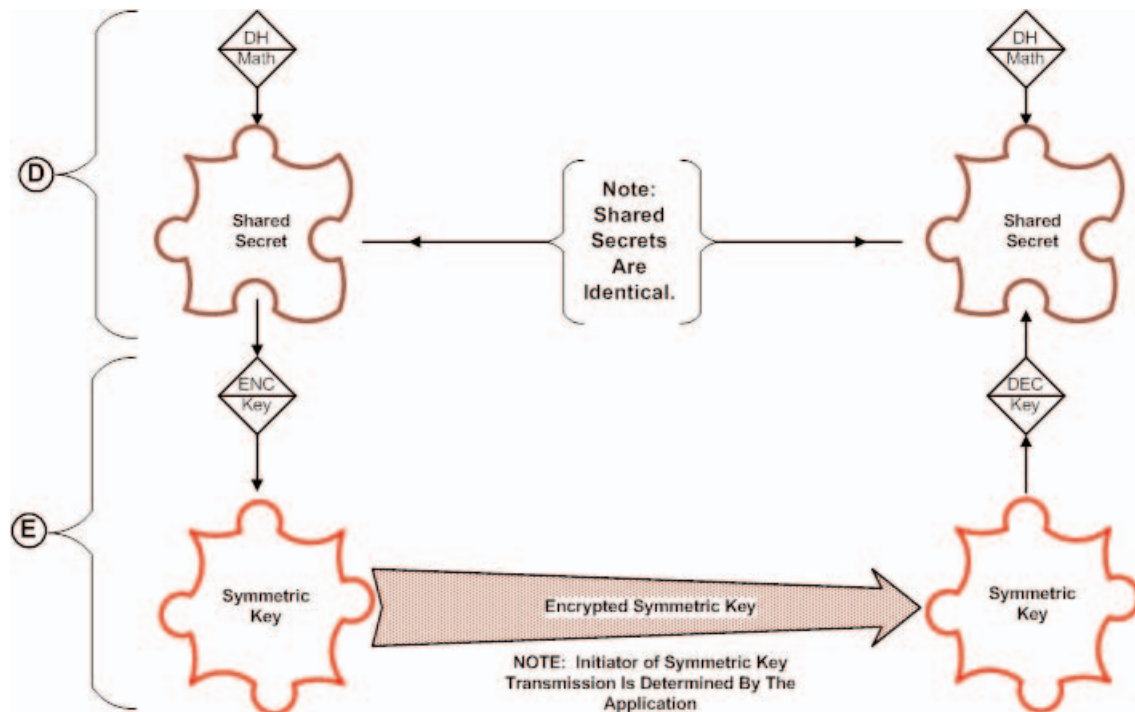


Figure 3: Encrypting, Passing, and Decrypting the Symmetric Key

Once secure exchange of the symmetric key is complete (and note that passing that key is the whole point of the Diffie-Hellman operation), data encryption and secure communication can occur. Figure 4 depicts data encrypted and decrypted on each end of the communication by the symmetric key. Note that changing the symmetric key for increased security is simple at this point. The longer 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 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.

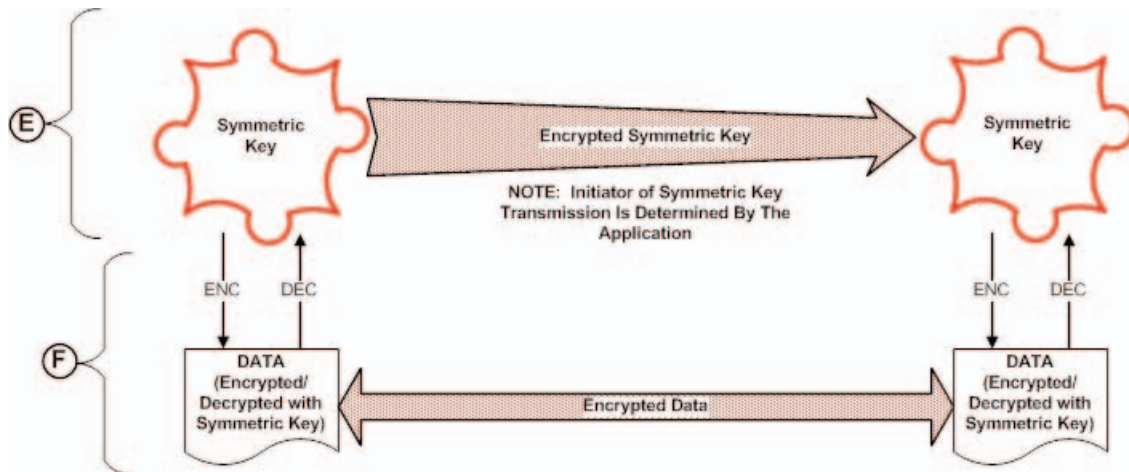


Figure 4: Encrypted Data Transmission

The Amateur Mathematician's Explanation

I am often asked for a better, though still simplified, explanation of exactly what happens in the "DH Math" portion of the diagram above. The best such explanation I can find is contained in the "Computer Desktop Encyclopedia" (CDE) published by The Computer Language Co., Inc. (see <http://www.computerlanguage.com/>). The example below is printed here with their permission. Remember that this example uses small numbers such as 2, 16, 5, and 32. In reality, these would be very large numbers.

===== From the CDE =====
 "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.

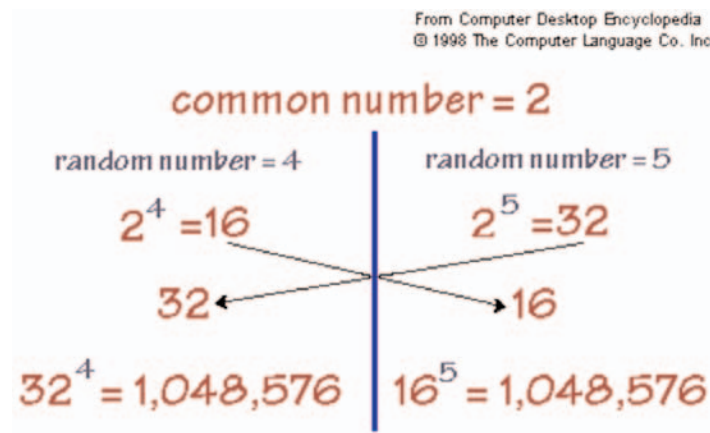


Figure 5: Simplified "DH Math"

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."

===== End CDE Entry =====

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 it works. The good news is that it almost always does work. Understanding the underlying protocol only becomes necessary in the rare case that it doesn't, and you need to troubleshoot the problem.

The complete Diffie-Hellman Key Exchange Diagram is shown on the next page:

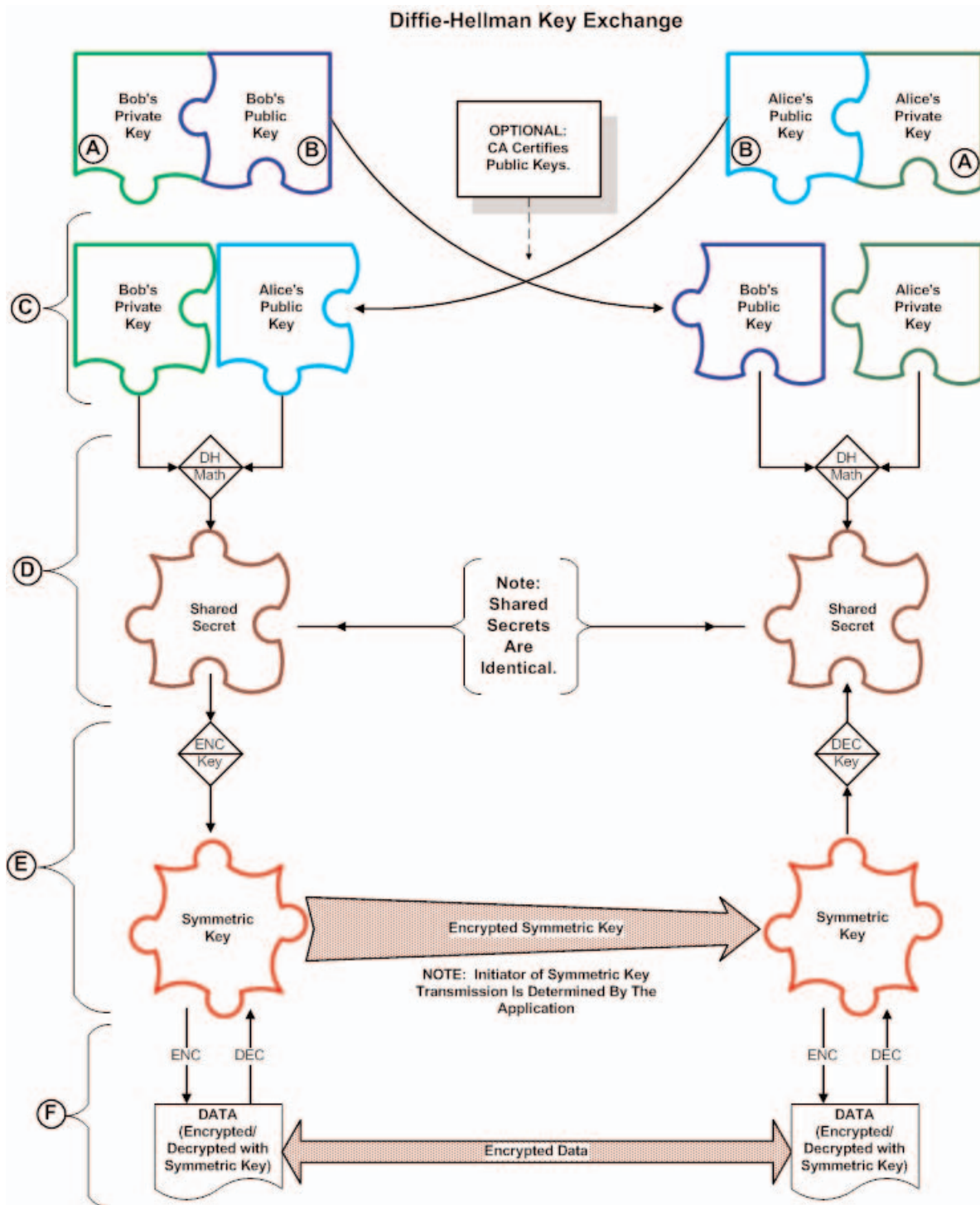


Figure 6: Complete Diffie-Hellman Key Exchange Process

Learn More

Learn more about how you can improve productivity, enhance efficiency, and sharpen your competitive edge. Check out the following Global Knowledge courses:

[Essentials of Network Security](#)

[CISSP Prep Course](#)

[Check Point NGX CCSA/CCSE](#)

[Certified Ethical Hacker \(CEH\)](#)

[Computer Hacking Forensic Investigator \(CHFI\)](#)

For more information or to register, visit www.globalknowledge.com or call 1-800-COURSES to speak with a sales representative.

Our courses and enhanced, hands-on labs offer practical skills and tips that you can immediately put to use. Our expert instructors draw upon their experiences to help you understand key concepts and how to apply them to your specific work situation. Choose from our more than 700 courses, delivered through Classrooms, e-Learning, and On-site sessions, to meet your IT and management training needs.

About the Author

Keith Palmgren has over 20 years of experience as a security professional and has held the CISSP certification since 1998. 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 teaching to share his knowledge with other IT professionals. Keith teaches a variety of courses in the Global Knowledge Security curriculum.

Special Thanks

The diagrams in figures 1, 2, 3, 4, and 6 were created with input and collaboration from Bernie Dixon, CISSP of Protected Networks, Inc. Bernie is also a Global Knowledge security instructor, and it is only fair that I give him credit for his input.