Interested in learning
more about security?

SANS Institute
InfoSec Reading Room

This paper is from the SANS Institute Reading Room site. Reposting is not permitted without express written permission.

A Review of the Diffie-Hellman Algorithm and its Use in Secure Internet Protocols

This paper will present an overview of the Diffie-Hellman Key Exchange algorithm and review several common cryptographic techniques in use on the Internet today that incorporate Diffie-Hellman.

Copyright SANS Institute
Author Retains Full Rights

AD

A Review of the Diffie-Hellman Algorithm and its Use in Secure Internet Protocols

David A. Carts

Abstract

This paper will present an overview of the Diffie-Hellman Key Exchange algorithm and review several common cryptographic techniques in use on the Internet today that incorporate Diffie-Hellman.

Introduction

The privacy requirements normally encountered in the traditional paper document world are increasingly expected in Internet transactions today. Secure digital communications are necessary for web-based e-commerce, mandated privacy for medical information, etc. In general, secure connections between parties communicating over the Internet is now a requirement.

Whitfield Diffie and Martin Hellman discovered what is now known as the Diffie-Hellman (DH) algorithm in 1976. It is an amazing and ubiquitous algorithm found in many secure connectivity protocols on the Internet. In an era when the lifetime of "old" technology can sometimes be measured in months, this algorithm is now celebrating its 25th anniversary while it is still playing an active role in important Internet protocols.

DH is a method for securely exchanging a shared secret between two parties, in real-time, over an untrusted network. A shared secret is important between two parties who may not have ever communicated previously, so that they can encrypt their communications. As such, it is used by several protocols, including Secure Sockets Layer (SSL), Secure Shell (SSH), and Internet Protocol Security (IPSec). These protocols will be discussed in terms of the technical use of the DH algorithm and the status of the protocol standards established or still being defined.

Overview of the Diffie-Hellman Algorithm

The mathematics behind this algorithm are conceptually simple enough that a high school student should be able to understand it. The fundamental math includes the algebra of exponents and modulus arithmetic.

For this discussion we will use Alice and Bob, two of the most widely traveled Internet users in cyberspace, to demonstrate the DH key exchange. The goal of this process is for Alice and Bob to be able to agree upon a shared secret that an eavesdropper will not be able to determine. This shared secret is used by Alice and Bob to independently generate keys for symmetric encryption algorithms that will be used to encrypt the data stream between them. The "key" aspect is that neither the shared secret nor the encryption key *do not ever* travel over the network.

© SANS Institute 2001, As part of the Information Security Reading Room. Author retains full rights.

Alice and Bob agree on two numbers "**p**" and "**g**"	"**p**" is a large prime number "**g**" is called the base or generator
Alice picks a secret number "**a**"	Alice's secret number = **a**
Bob picks a secret number "**b**"	Bob's secret number = **b**
Alice computes her public number $\mathbf{x} = \mathbf{g}^{\mathbf{a}} \bmod \mathbf{p}$	Alice's public number = **x**
Bob computes his public number $\mathbf{y} = \mathbf{g}^{\mathbf{b}} \bmod \mathbf{p}$	Bob's public number = **y**
Alice and Bob exchange their public numbers	Alice knows **p, g, a, x, y** Bob knows **p, g, b, x, y**
Alice computes $\mathbf{k_a} = \mathbf{y}^{\mathbf{a}} \bmod \mathbf{p}$	$\mathbf{k_a} = (\mathbf{g}^{\mathbf{b}} \bmod \mathbf{p})^{\mathbf{a}} \bmod \mathbf{p}$ $\mathbf{k_a} = (\mathbf{g}^{\mathbf{b}})^{\mathbf{a}} \bmod \mathbf{p}$ $\mathbf{k_a} = \mathbf{g}^{\mathbf{ba}} \bmod \mathbf{p}$
Bob computes $\mathbf{k_b} = \mathbf{x}^{\mathbf{b}} \bmod \mathbf{p}$	$\mathbf{k_b} = (\mathbf{g}^{\mathbf{a}} \bmod \mathbf{p})^{\mathbf{b}} \bmod \mathbf{p}$ $\mathbf{k_b} = (\mathbf{g}^{\mathbf{a}})^{\mathbf{b}} \bmod \mathbf{p}$ $\mathbf{k_b} = \mathbf{g}^{\mathbf{ab}} \bmod \mathbf{p}$
Fortunately for Alice and Bob, by the laws of algebra, Alice's "$\mathbf{k_a}$" is the same as Bob's "$\mathbf{k_b}$", or $\mathbf{k_a} = \mathbf{k_b} = \mathbf{k}$	Alice and Bob both know the secret value "**k**"

There are requirements on the numbers picked (e.g. minimum size, ranges, primality) that can be found in the references; in particular, see RFC 2631 [1]. This algorithm has been reviewed and discussed many times in a variety of papers available online. The following references may be useful in understanding the concepts before tackling the RFC: RSA Laboratories Cryptography FAQ, section 3.6.1 [2], What is Diffie-Hellman [3], Diffie-Hellman Method for Key Agreement [4], and PKCS #3: Diffie-Hellman Key-Agreement Standard [5].

Note that some of these numbers are very large – for example, if **p** is a 512 bit binary number, the minimum allowed in the standard, that would be a number with up to 150+ digits expressed in decimal notation. Implementation details are very important as typical computer variables cannot hold numbers this big; a 512 bit (=64 byte) number will not fit into a 4 byte integer field.

Diffie-Hellman in SSL

Secure Sockets Layer (SSL) is a cryptographic protocol developed by Netscape in 1995. SSL V3.0 has since become the predominant method and de-facto standard for securing information flow between web users and web servers. This is most commonly done for business/financial traffic, e.g. credit card transactions. Specifically, "securing information" in this context, means to ensure confidentiality (prevent eavesdropping), authenticity (the sender is really who he says he is), and integrity (the message has not been changed en route). Users may not know they are using SSL, but they probably will notice the padlock

© SANS Institute 2001, As part of the Information Security Reading Room. Author retains full rights.

icon and the "https:" in the URL that indicates encryption is taking place. SSL most commonly runs on TCP using port 443.

The Internet Engineering Task Force (IETF) adopted this protocol in 1999. They renamed it Transport Layer Security (TLS) V1.0 protocol and defined it in RFC2246 [6].

SSL/TLS is composed of two layers: the lower layer called the Record Protocol rides on TCP and manages the symmetric (private) cryptography so the communication is private and reliable. The upper layer is called the Handshake Protocol and is responsible for authentication of the parties and negotiation of encryption methods and keys. It is in this layer that DH can be used. DH is considered the strongest alternative of the available options for the key exchange according to Wagner and Schneier [7].

Specifically, early in the communications process the client and server are exchanging unencrypted handshake messages. They exchange hellos, then they trade information about which encryption, key exchange, and compression options they each accept and prefer. As part of the PKI/X.509 certificate exchange they also choose a key exchange method where one of the options is DH. The key exchange process uses asymmetric (public key) cryptography to ensure to each party that the other is who they say they are. This is typically done using ephemeral key pairs newly computed for each session. After this exchange, secrets and keys are computed and the parties begin encrypting all traffic between them, using the computed keys and agreed upon methods.

In practical use today, a server is authenticated to the user, but not the other way around (because of a lack of an Internet-wide PKI system - more on this later). This one-way authentication is accomplished by encoding the certificates of well-known Certificate Authorities (CAs) into the popular browsers.

Diffie-Hellman in SSH

Secure Shell (SSH) is a both a protocol and a program used to encrypt traffic between two computers. This is most commonly done as a secure replacement for tools like telnet, ftp and the Berkeley "r" commands (rlogin, rsh, etc.). These older tools do not encrypt any of their traffic, including the authentication process, so account names and passwords are transmitted in plaintext. This is a bad thing!

SSH was authored by Tatu Ylonen in 1995 and has since spread quickly throughout the UNIX world, and has become available for other platforms. There are both commercial [8] and free implementations available [9], and a SSH Working Group [10] sponsored by the IETF that is working to formalize and standardize the protocol. Besides providing a secure interactive shell, SSH also includes other functionality, for example, tunneling X11 connections over an established SSH session.

The SSH Transport Layer protocol is defined in the IETF Draft document "SSH Transport Layer Protocol" [11]. This protocol layer most commonly runs on TCP using port 22, and supports (higher-level) protocols such as the SSH Connection Protocol, which provide for

© SANS Institute 2001, As part of the Information Security Reading Room. Author retains full rights.

remote interactive login sessions, remote execution of commands, forwarding X11 connections, etc.

The two parties to the connection (e.g., client and server) begin their conversation by negotiating parameters (e.g., preferred encryption and compression algorithms, and certain random numbers). Then a shared secret is computed using DH in a manner similar to SSL/TLS described above. A hashing function on the shared secret is used to derive an encryption key for the negotiated symmetric encryption algorithm (e.g., 3DES). From this point the two sides encrypt all traffic between them with the symmetric encryption algorithm.

Diffie-Hellman in IPSec

Internet Protocol Security (IPSec) is a protocol being developed by the IETF to incorporate secure communications into the IP network layer itself. The protocols described previously (SSL, SSH) are application specific for practical purposes and only protect only the traffic originating in their applications, while IPSec is designed to work for all IP traffic, independently of application. This approach results in the advantage that neither the applications nor the users need to know anything about the encryption, i.e., the encryption process is transparent to the users and applications.

Like the previous protocols, IPSec uses DH and asymmetric cryptography to establish identities, preferred algorithms, and a shared secret. Then, the algorithm uses a symmetric cipher to encrypt the bulk data as it is transferred.

Before IPSec can begin encrypting the data stream, some preliminary information exchange is necessary. This is accomplished with the Internet Key Exchange (IKE) protocol, defined in RFC 2409 [12]. The IKE protocol works in two phases. Phase 1 provides the mechanism for the two parties to dynamically agree on security parameters. They use DH to produce a shared secret, and they authenticate each other's identities.

Phase 2 uses the shared secret to encrypt the exchange of information that determines the encryption parameters for the actual data. In addition to the RFC, good, mid-level descriptions of the IPSec and IKE protocols can be found in "An Introduction to IPSec" by Sheila Frankel [13], and "Internet Protocol Security Revealed" by Rich Hernandez [14].

Diffie-Hellman in PKI

Public Key Infrastructure (PKI) refers to a system of protocols and services supporting *public key cryptography*, a general term for asymmetric encryption. In public key cryptography a mathematically matched pair of keys are used instead of a single key as used in symmetric cryptography. First one of the pair is used to encrypt the data, then the other key in the pair is used to decrypt the ciphertext. One key is published publicly, while the other is kept a closely guarded secret.

Two complementary uses can be made of public key cryptography. If I encrypt a message with the public key of another person, only that person can decrypt it because only that

© SANS Institute 2001, As part of the Information Security Reading Room. Author retains full rights.

person knows his private key. This provides confidentiality and is the most widely known use for public key cryptography. Alternatively, if I encrypt a message (or hash of a message) with my private key, anybody can decrypt it with my public key, but everyone knows I am the only one who could have produced that message, because I am the only one who knows my private key. This is called a digital signature, and ensures authenticity, non-repudiation, and (with the hash of the message as is most commonly done) integrity. Used in combination these two aspects of public key cryptography are very powerful.

A significant problem with public key cryptography is ensuring you have the correct public key of the other person. That is, before cryptography, you had to worry about someone intercepting or altering your message; with cryptography, you have traded that problem with worrying about someone intercepting or altering your public key. This is solved with a hierarchical system of *Certificate Authorities* (CAs). A CA will produce a digital certificate that contains a persons identification and public key. To guarantee the public key is valid, the certificate is signed with the private key of the CA. To guarantee the certificate itself is valid, the certificate also contains a digital signature of the next higher level CA in the hierarchy.

So, two parties can exchange encrypted data by getting the other's public key and certificate from the PKI. Of key importance in this process is that the public key is not altered in any way in transit, so an encrypted hash of the certificate is made, where the encryption key is derived from a DH exchange.

While various authors have differing definitions for "PKI", it is essentially a system for supporting the public keys used in public key cryptography. The base technologies, including encryption methods, certificate formats, etc. are defined in RFC 2459 [15], and Public Key Cryptography for The Financial Services Industry [16].

PKI is used in practice, or intended to be used, in several ways. SSL technology, when authenticating both parties, depends on being able to reach a PKI [6]. Sending encrypted email using S/MIME depends on the mail software being able to reach a PKI to get public keys [17]. Both the sender and recipient need this access - the sender needs to know the public key of the recipient, and vice versa.

However, as a technology for the Internet as a whole, PKI is not ready for widespread deployment because of a diversity of vendor specific "standards". Individual enterprises can effectively implement a PKI from one vendor internally, however, they can expect difficulties in exchanging certificates with PKI systems from other vendors. Standards setting bodies are working towards a common standard. Some of the organizations involved are the National Institute for Standards and Technology (NIST) [18], the IETF's PKIX and SPKI working groups [19,20] and The Open Group [21]. A useful reference for PKI and other security issues is found at the Secure Electronic Marketplace for Europe, Security and Cryptography Web site [22].

Conclusions

© SANS Institute 2001, As part of the Information Security Reading Room. Author retains full rights.

The Diffie-Hellman algorithm is an enabling technology for nearly every encryption technology in use in the Internet today, including SSL, SSH, IPSec, PKI, and everything else that depends on these protocols.

References

Diffie-Hellman References

1. Rescorla, E., Diffie-Hellman Key Agreement Method, RFC 2631, IETF Network Working Group, http://www.ietf.org/rfc/rfc2631.txt
2. RSA Laboratories, RSA Laboratories' FAQ About Today's Cryptography, Version 4.1, RSA Security Inc., 2000, http://www.rsa.com/rsalabs/faq/index.html
3. Costas Christoyannis, "What is Diffie-Hellman", http://www.hack.gr/users/dij/crypto/overview/diffie.html
4. Levy, Benjamin, "Diffie-Hellman Method for Key Agreement", http://apocalypse.org/pub/u/seven/diffie.html
5. RSA Laboratories, PKCS #3: Diffie-Hellman Key-Agreement Standard, Version 1.4. Revised November 1, 1993, http://www.rsalabs.com/pkcs/pkcs-3/index.html

SSL/TLS References

6. Dierke, T., and C. Allen, The TLS Protocol Version 1.0, RFC2246, IETF Network Working Group, http://www.ietf.org/rfc/rfc2246.txt
7. Wagner, David and Bruce Schneier, Analysis of the SSL 3.0 Protocol, PDF document available from http://www.counterpane.com/ssl.html

SSH References

8. SSH Communications Security Home Page, <http://www.ssh.com/>
9. OpenSSH Home Page, http://www.openssh.com/
10. Secure Shell IETF Working Group Home Page, http://www.ietf.org/html.charters/secsh-charter.html
11. Ylonen, T., et al., SSH Transport Layer Protocol, IETF IPSec Working Group, January 2001, http://www.ietf.org/internet-drafts/draft-ietf-secsh-transport-09.txt

IPSec References

12. Harkins, D., and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, IETF Network Working Group, November 1998, http://www.ietf.org/rfc/rfc2409.txt
13. Frankel, Sheila, An Introduction to IPSec, NIST Information Technology Bulletin, March 2001, http://www.itl.nist.gov/lab/bulletns/bltnmar01.htm
14. Hernandez, Rich, Internet Protocol Security Revealed, Dell Power Solutions Magazine, Issue 2, 2000, http://www.dell.com/us/en/esg/topics/power_ps2q00-ipsec.htm

© SANS Institute 2001, As part of the Information Security Reading Room. Author retains full rights.

PKI References

15. Housley, R., W. Ford, W. Polk, and D. Solo, Internet X.509 Public Key Infrastructure: Certificate and CRL Profile, RFC 2459, IETF Network Working Group, January 1999, http://www.ietf.org/rfc/rfc2459.txt
16. ANSI X9.42-199x, Public Key Cryptography for The Financial Services Industry: Agreement of Symmetric Algorithm Keys Using Diffie-Hellman (Working Draft), December 1997. (Fee charged for this document.)
17. Ramsdell, B., S/MIME Version 3 Certificate Handling, RFC 2632, , IETF Network Working Group, June 1999, , http://www.ietf.org/rfc/rfc2632.txt
18. NIST Public Key Infrastructure Program Home Page, http://csrc.ncsl.nist.gov/pki/.
19. Public Key Infrastructure (X.509) (PKIX) Working Group Home Page, http://www.ietf.org/html.charters/pkix-charter.html.
20. Simple Public Key Infrastructure (SPKI) Working Group Home Page, http://www.ietf.org/html.charters/spki-charter.html
21. The Open Group Public Key Infrastructure Home Page, http://www.opengroup.org/security/pki/.
22. Secure Electronic Marketplace for Europe, Security and Cryptography Home Page, http://www.semper.org/sirene/outsideworld/security.html

© SANS Institute 2001, Author retains full rights.

© SANS Institute 2001, As part of the Information Security Reading Room. Author retains full rights.

Last Updated: May 22nd, 2013

Upcoming SANS Training

Click Here for a full list of all Upcoming SANS Events by Location

SANS Malaysia @ MCMC 2013	**Cyberjaya, MY**	**Jun 03, 2013 - Jun 08, 2013**	**Live Event**
SANS Pen Test Berlin 2013	**Berlin, DE**	**Jun 03, 2013 - Jun 08, 2013**	**Live Event**
Industial Control Systems Security Training - Houston	**Houston, TXUS**	**Jun 10, 2013 - Jun 15, 2013**	**Live Event**
Security Impact of IPv6 Summit 2013	**Washington, DCUS**	**Jun 14, 2013 - Jun 16, 2013**	**Live Event**
SANSFIRE 2013	**Washington, DCUS**	**Jun 14, 2013 - Jun 22, 2013**	**Live Event**
SANS Canberra 2013	**Canberra, AU**	**Jul 01, 2013 - Jul 13, 2013**	**Live Event**
Digital Forensics & Incident Response Summit 2013	**Austin, TXUS**	**Jul 09, 2013 - Jul 16, 2013**	**Live Event**
SANS London Summer 2013	**London, GB**	**Jul 09, 2013 - Jul 16, 2013**	**Live Event**
SANS Rocky Mountain 2013	**Denver, COUS**	**Jul 14, 2013 - Jul 20, 2013**	**Live Event**
SANS Mumbai 2013	**Mumbai, IN**	**Jul 22, 2013 - Jul 27, 2013**	**Live Event**
SEC528 SANS Training Program for the CompTIA® New Advanced Security Practitioner (CASP) Certification	**Washington, DCUS**	**Jul 22, 2013 - Jul 26, 2013**	**Live Event**
SANS San Francisco 2013	**San Francisco, CAUS**	**Jul 29, 2013 - Aug 03, 2013**	**Live Event**
SANS SEC 560: Network Penetration Testing @ Bangalore 2013	**Bangalore, IN**	**Aug 05, 2013 - Aug 10, 2013**	**Live Event**
SANS Boston 2013	**Boston, MAUS**	**Aug 05, 2013 - Aug 10, 2013**	**Live Event**
Critical Security Controls Summit	**Washington, DCUS**	**Aug 12, 2013 - Aug 18, 2013**	**Live Event**
Industrial Control Systems Security Training - DC	**Washington, DCUS**	**Aug 12, 2013 - Aug 16, 2013**	**Live Event**
SANS Thailand 2013	**Bangkok, TH**	**Aug 19, 2013 - Aug 31, 2013**	**Live Event**
SANS Virginia Beach 2013	**Virginia Beach, VAUS**	**Aug 19, 2013 - Aug 30, 2013**	**Live Event**
Mobile Device Security Summit 2013	**OnlineCAUS**	**May 30, 2013 - Jun 06, 2013**	**Live Event**
SANS OnDemand	**Books & MP3s OnlyUS**	**Anytime**	**Self Paced**