



IPsec Simplified

Peter J. Welcher

Introduction

First, a couple of personal and company news items:

- A couple of weeks ago I had the pleasure of attending a Train The Trainer session for the Cisco MPLS Essentials (MLSTE) course, and am now certified to teach it. The course is a 3-day overview of MPLS Technology, including coverage and labs for QoS, MPLS Traffic Engineering, and MPLS-based VPN's. It is being supplemented with other MPLS courses and also with matching BGP courses. I hope we'll have the MLSTE course in our schedule by the time you read this.
- The Mentor Technologies course ECP1, based on the CCIE prep book by Bruce Caslow (editor Val Pavlichenko), is quite popular. If our web pages won't let you register for it, it's due to the long lead times to get a seat. We're glad the course is popular, we regret the lead times, we're adding capacity. You might want to plan ahead and book a seat in the course early, especially if you're booking a CCIE lab slot and want to take the ECP1 course first!

Our technical topic this month will be a brief introduction to the IP Security standard, IPsec. Since there's enough to say that it won't fit into one column (I tried!), this will be a series of at least two articles.

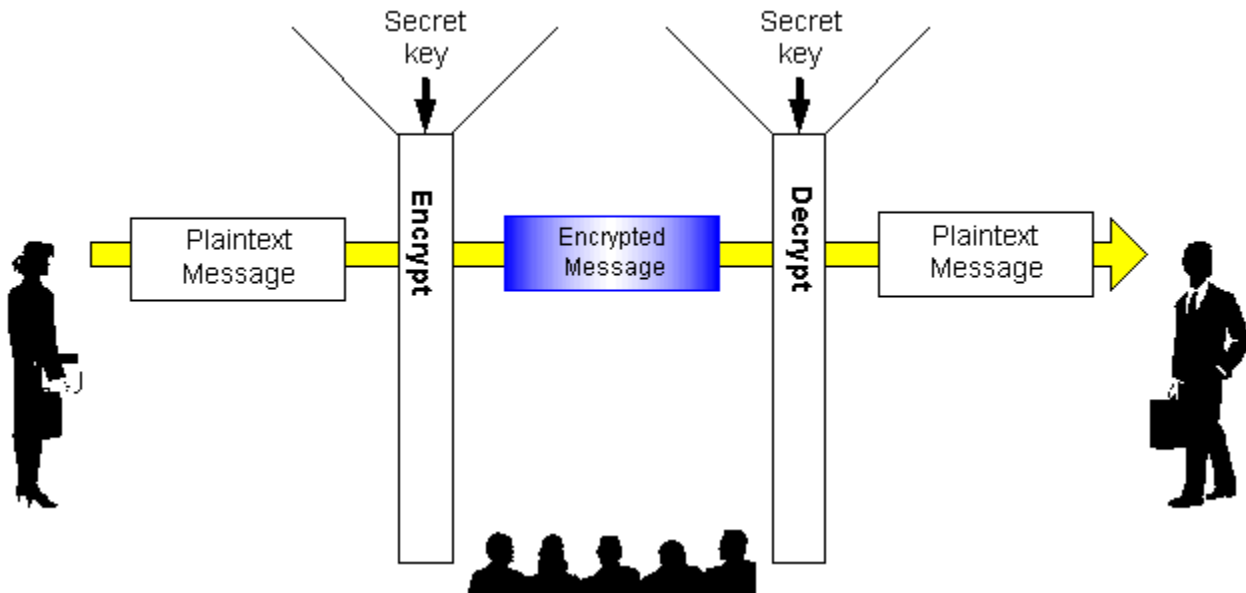
I confess, I personally resisted looking at IPsec for a while. I was put off by the apparent complexity of IPsec, and I suspect you might have too! So this article represents my attempt to render IPsec in simple terms, and give you some idea how it works and how to configure it. The thought here is that if you understand the Big Picture, how it all fits together, going back to the manual and filling in the details is a whole lot easier. And please note, I claim no particular expertise on IPsec. This will be just my version of how I think it all fits together.

Just a Very Wee Bit of Cryptology

IPsec phobia is caused by confusion. To cure that, we need some background information and terminology. Believe me, with a little orientation, this stuff makes a whole lot more sense! By the way, there's lots of mathematics theory behind all this, but you won't see it here. After all, you don't need to

know any of that to use IPsec. If you are interested in mathematical details, see the Stallings book or link below. We're trying to stick with just what you really need to know here.

If you've ever taken any kind of look at cryptography, you know there is generally an **encryption key**. A **cleartext** message is processed somehow using the key and turned into an encrypted message, which looks rather random and hard to read. Using the key, at least with historical encryption techniques, you can also decrypt the message, turning it back into cleartext. The cleartext message is generally supposed to make sense after all this.



The current (expiring) U.S. government standard is DES, Data Encryption Standard. DES encrypts 64 bit blocks using a 56 bit key. Single DES encryption is thought to be vulnerable to brute force attack. So we do it three times, using a different key each time. Three-key triple DES uses three 56 bit keys, with effective key length of $3 \times 56 = 168$ bits. This is a compromise improving resistance to brute force attack, but preserving investment in existing software and hardware.

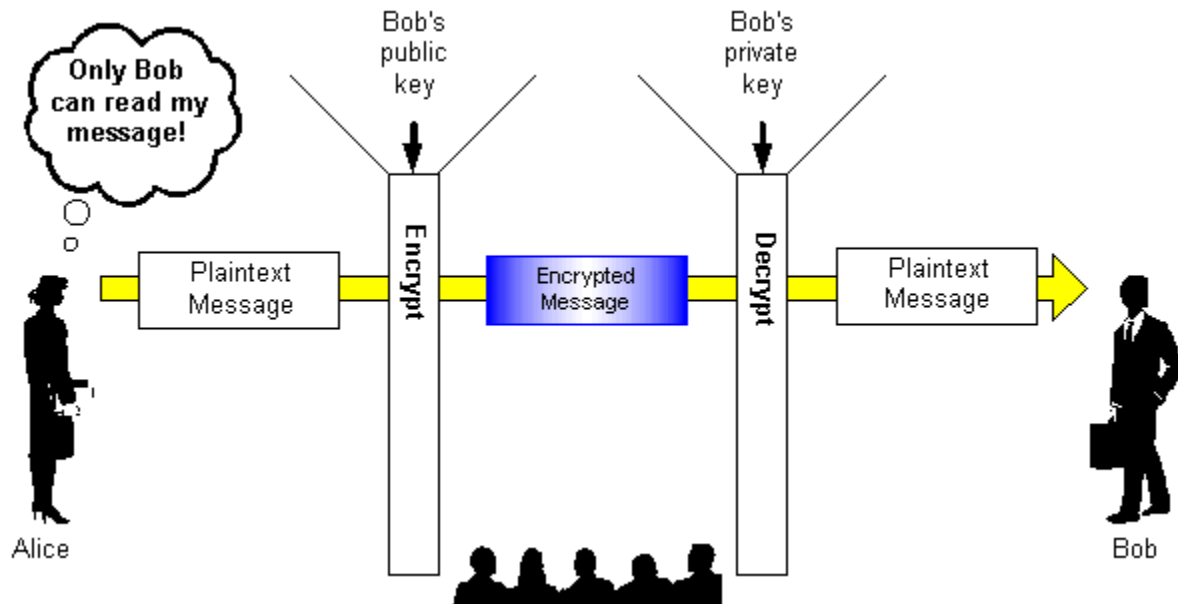
For what it's worth (and as a minor digression?), there is work on a new U.S. encryption standard, the Advanced Encryption Standard. Recently, the Rijndael (pronounced "Rhine Doll") encryption algorithm was chosen for future use. One imagines that as this technology matures, IPsec will make use of it. See also the following links:

- <http://csrc.nist.gov/encryption/aes/>
- http://www.nist.gov/public_affairs/releases/g00-176.htm

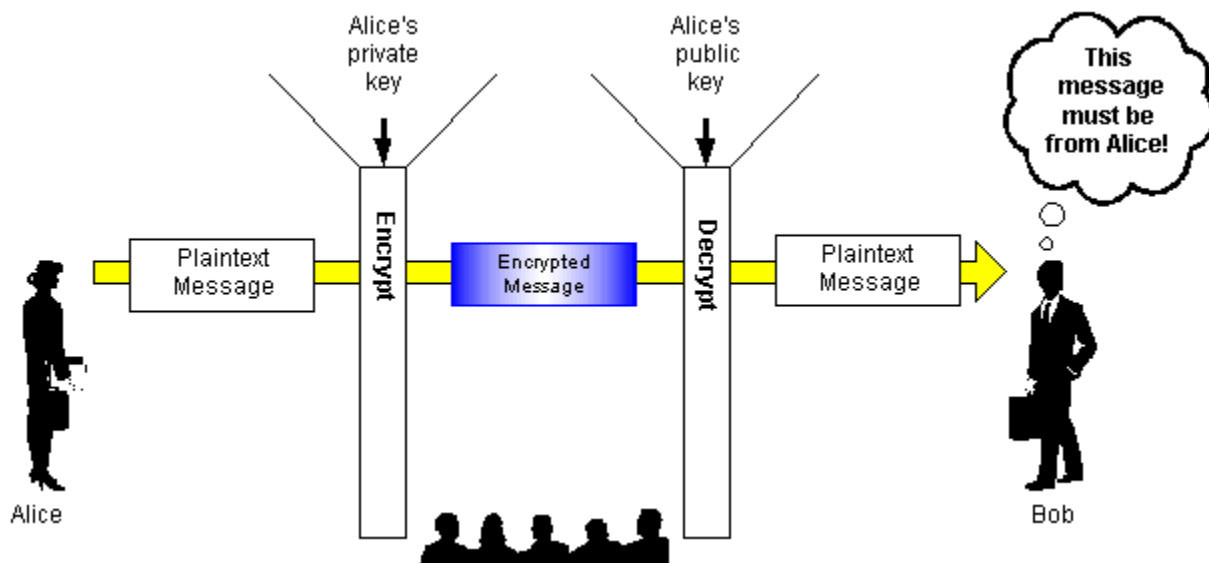
Public key encryption was invented conceptually by Diffie and Hellman (and others, per Stallings). More about Diffie and Hellman in a moment. The first algorithm meeting their requirements was developed in 1977 by Rivest, Shamir, and Adelman, and is based on the (apparent) computational complexity of factoring large numbers. The patent on this just expired.

Public key cryptology uses two keys, a **public key** and a **private key**. In the current schemes, you encrypt a message with either key, and use the *other* key to decrypt it. The general idea is that you (mathematically?) generate two keys, publish one as public key, and protect the other one, the private key, carefully.

Suppose that Alice encrypts a message with Bob's public key. The only way to decrypt that message is by using Bob's private key. So using someone else's public key gives you a way to send them a message no-one else can read.



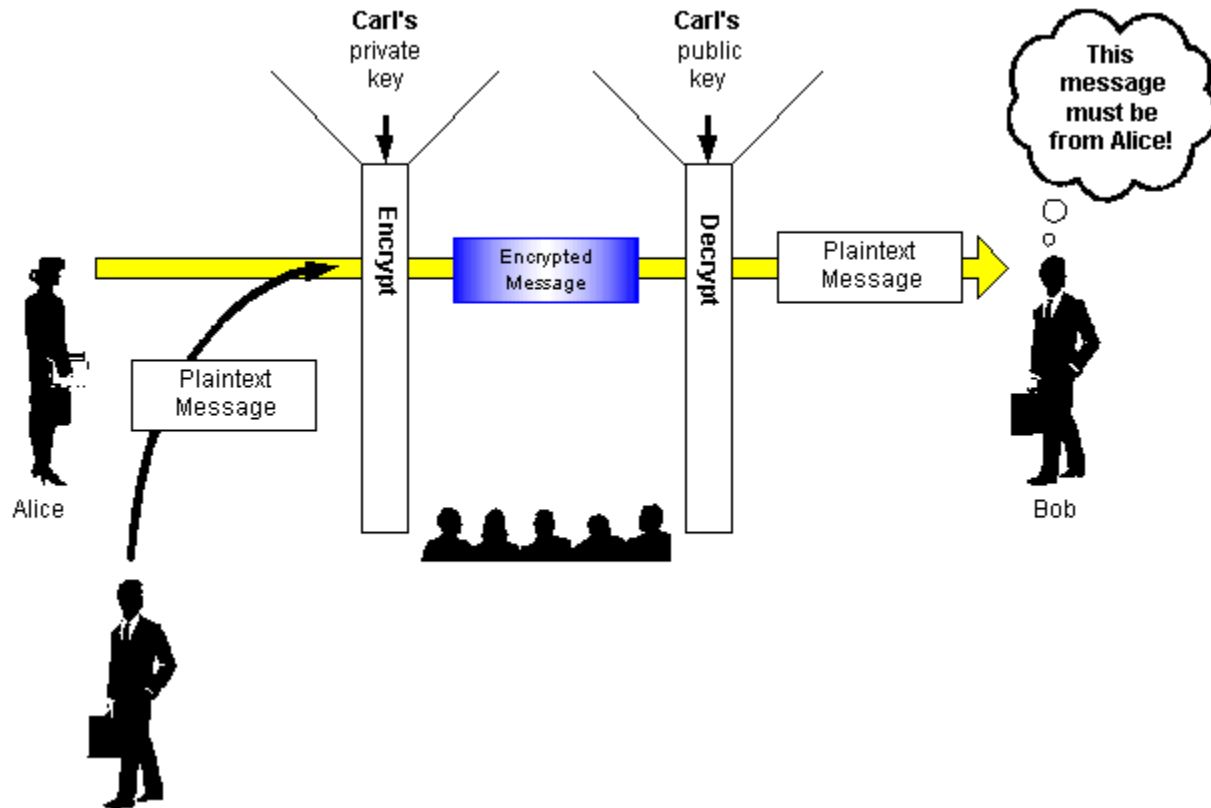
Furthermore, suppose Alice encrypts a message with her private key. Then upon receiving that message, Bob can obtain Alice's public key, and decrypt the message. This gives Bob some confidence that Alice really did send the message. After all, whoever sent it had the private key that goes with Alice's apparent public key. This would be a form of a **digital signature**. (There are other possible uses for this, like making sure the message doesn't get changed by either party after signing.)



If you followed all that, let's try one more mind-bender: if Alice encrypts a message with Bob's public key, then encrypts the result with her private key, then the resulting message must have come from her, and it can only be read by Bob. This gives the two a secure authenticated way to communicate.

These concepts assume the private keys were not compromised, and that the public keys have not been

spoofed or substituted for. If bad guy Carl can substitute his public key in place of Alice's (in a directory), then Bob might be tricked into communicating with Carl thinking he was communicating with Alice.



Now that you've got a beginning of an idea about public key cryptography, let's go back to the Big Picture. Determining suitable public and private keys, and encrypting messages with them, is too slow and computationally difficult to do very often, at least at present. Another point of view: public key cryptography at speeds like T1 and above may require offloading to specialized hardware, hence costs real money.) So instead, we use public key cryptography for authentication, and also to exchange keys for more conventional (DES and 3DES) encryption.

Terminology and Background

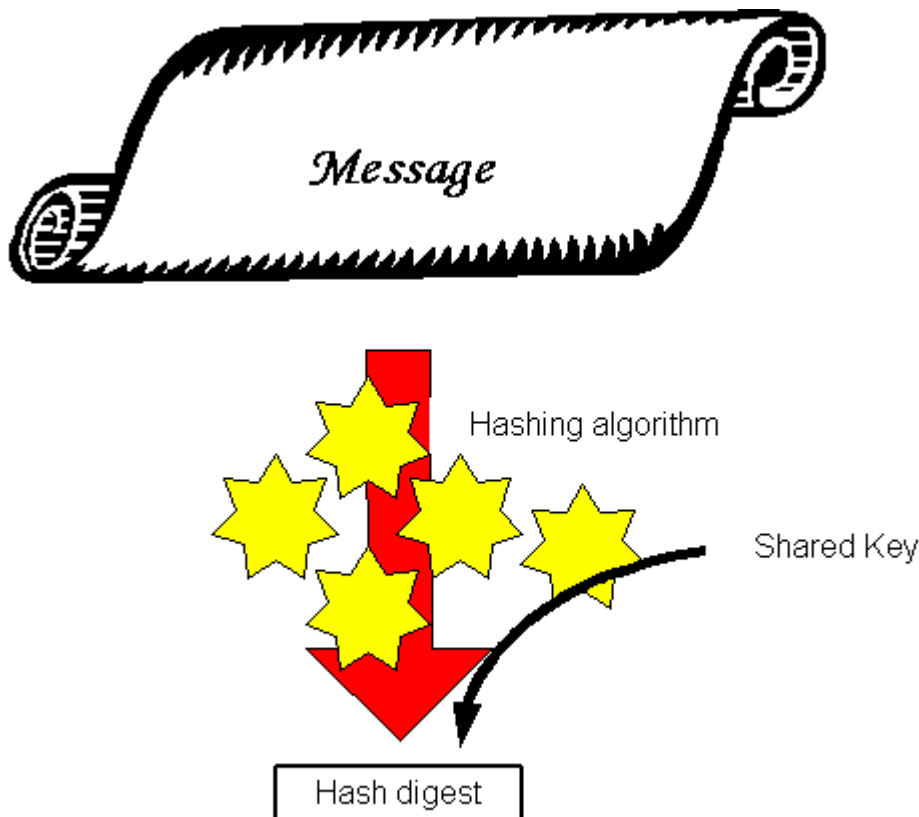
Diffie-Hellman came up with an accepted algorithm for two systems to use public key encryption to publicly but securely exchange DES or 3DES (or other) encryption keys. (Exercise for the reader: what does this have to do with what we talked about a couple of paragraphs back?) **OAKLEY** is an improvement on this. (Historical humor: Diffie is a Sun Microsystems fellow; supposedly Oakley is named after the sunglasses company, as in "Sun protection".) In IPsec, OAKLEY/Diffie-Hellman techniques are used by two devices to come up with a common shared key, to be used for DES or 3DES encryption. **ISAKMP** (Internet Security Association and Key Management Protocol) is the packet format and protocol used by IPsec to manage keys. The initial version of ISAKMP uses OAKLEY for key exchange. The process of exchanging keys for IPsec is referred to as **IKE** (Internet Key Exchange).

One other basic concept and we'll be ready to talk about how IPsec works, and configuration choices on Cisco IPsec devices.

We need to briefly talk about **hashing algorithms**. The idea of hashing is to do bit operations on blocks of a message, producing far fewer bits as output. The output is called the **hash code**. All this is done in such a way that changes to the original message are extremely likely to change the hash code. You can think of the hash algorithm output as a *tamper-proof seal*. Should someone intercept and attempt to alter a message, then the hash code will be different. I hope you're wondering, "yes, but how does the receiver of the message know what the hash value is *supposed* to be". Downloading software with hashing, you'd just check the original download site for what the hash code is supposed to be. But that sort of thing doesn't work for messages.

Obviously, if you just send a hash code along with a message, then the hacker can change the message, compute the new hash code, and replace the hash code with the correct one for the new message. What we need is a way to prevent changing the tamper seal! So instead, we use an shared secret key in performing the hashing. The hashing mangles bits of the secret key in with bits of the message. This makes it highly unlikely that a hacker can change the message unless they know the key. They'd have to see quite a few hashes before being able to guess the shared secret key. So using keyed hashing, we can safely send our hash code along with the message, knowing that even if the hacker changes the message, they still won't be able to fix up the attached hash code to match up with the message.

The name for the keyed hash algorithm used to produce hash codes in IPsec is **HMAC** (Hash Message Authentication Code), defined in RFC 2104. HMAC is also used in SSL (Secure Sockets Layer) for e-commerce. Two of the hash functions used in IPsec HMAC are **MD5** (Message Digest 5) and **SHA** (Secure Hash Algorithm). These are just different ways of computing a hash code. Cisco supports both.



A **Security Association (SA)** is an agreement between two devices (parties) as to how to go about doing encryption. This includes such items as shared key, key lifetime, hashing algorithm, etc. More on

this later.

Basic IPsec Operation in Cisco Routers

We now know enough to be dangerous. Let's look at the sequence of steps for IPsec operation in Cisco routers.

- 1) Interesting traffic initiates IPsec
- 2) IKE Phase 1: set up IKE SA
- 3) IKE Phase 2: set up IPsec SA
- 4) Data transfer
- 5) IPsec terminates

Step 1: Interesting traffic initiates IPsec. What this means in practical terms: it takes some interesting traffic to get the router to try to do IPsec. This is good, since you don't want idle routers maintaining a Security Association (SA) -- that takes work!

Step 2: IKE Phase 1: set up IKE SA. For routers to get started doing IPsec, they first need to negotiate and agree on how to do IKE. There are several choices, and they have to agree on **something** or Step 2 fails. This roughly corresponds to agreeing as to how securely the devices are going to be, about how they exchange keys. Part of IKE is mutual authentication, and there are several choices for this: pre-shared key, encrypted RSA nonces, RSA or DSS signatures, Certificate Authority (CA). For now, please content yourself with recognizing that these names all represent authentication techniques, in order of increasing security. We may look at them a little more in a subsequent article.

Step 3: IKE Phase 2: set up IPsec SA. Once the IPsec devices form the IKE SA, they negotiate an IPsec SA. As we'll see (below), there are several IPsec choices the devices need to agree upon. And while they're at it, they also need to come up with a shared DES or 3DES key.

Step 4: Data transfer. Once all this work is done, data can flow. Interesting traffic matching the access list (Step 1) gets encrypted. By the way, the access list also tells the router what traffic to decrypt. Practical tip: if your access list is sloppy, or if it encompasses too many packets, you may find that traffic is getting encrypted unnecessarily, or as part of an SA different than you expected. If a router decrypts such traffic using a different SA (and probably DES or 3DES key), then the resulting IP datagram may well be garbage. The router discards garbage. The best thing to do is to be precise about which hosts or subnets are on each end (senders and receivers). If you don't know which subnets are where, this is rather hopeless. If you have a well-thought out addressing scheme and network design, particularly one using summarizable blocks of subnets for routing, then you'll find the access list much, much easier to write! Another tip: since encryption should be thought of as costly, **not** encrypting traffic that doesn't require encryption is a Best Practice.

Step 5: IPsec terminates. IPsec terminates because of SA lifetime timeout, or because the SA lifetime packet byte counter was exceeded. The idea here is that if your TCP connection is done, there's no point to maintaining IPsec state. Lifetime and packet byte count matter because all codes can be cracked, the key question (pun intended) is how long it takes to crack them. Expiring the IPsec SA after an amount of time forces the formation of a new IPsec SA, hence a new key. Expiring early

renders the encryption less likely to be cracked, but also means the IPsec device will need to re-key more often: more work. Spy novel fans also know the general cipher principle, the more coded text you have, the faster you can crack the key. So rekeying after a certain number of bytes have been sent is desirable. The Cisco IOS defaults are considered "reasonable", and if you don't like them you can configure your own settings, consistently across devices (since time and byte lifetimes are part of the SA that has to be negotiated). Note that a new SA is negotiated before the old one expires, to make sure it is available when needed.

References

Books:

- [Applied Cryptography \[2nd Edition\], Bruce Schneier, John Wiley and Sons](#)
- [Cryptography and Network Security, William Stallings, Prentice Hall](#)

Presentation:

- Advanced Security Technology Concepts, Networkers 2000 Australia:
http://www.cisco.com/networkers/presentations/networkers/security/advanced_security_vpns-timr

RFCs:

The series starting at RFC 1825 was updated with the 2401 series listed below.

- RFC 1828: **IP Authentication using Keyed MD5**, <http://www.ietf.org/rfc/rfc1828.txt>
- RFC 1829: **The ESP DES-CBC Transform**, <http://www.ietf.org/rfc/rfc1829.txt>
- RFC 2085: **HMAC-MD5 IP Authentication with Replay Prevention** ,
<http://www.ietf.org/rfc/rfc2085.txt>
- RFC 2104: **HMAC: Keyed-Hashing for Message Authentication**,
<http://www.ietf.org/rfc/rfc2104.txt>
- RFC 2401: **Security Architecture for the Internet Protocol**,
<http://www.ietf.org/rfc/rfc2401.txt>
- RFC 2402: **IP Authentication Header**, <http://www.ietf.org/rfc/rfc2402.txt>
- RFC 2403: **The Use of HMAC-MD5-96 within ESP and AH**,
<http://www.ietf.org/rfc/rfc2403.txt>
- RFC 2404: **The Use of HMAC-SHA-1-96 within ESP and AH**,
<http://www.ietf.org/rfc/rfc2404.txt>
- RFC 2405: **The ESP DES-CBC Cipher Algorithm With Explicit IV** ,
<http://www.ietf.org/rfc/rfc2405.txt>
- RFC 2406: **IP Encapsulating Security Payload (ESP)**, <http://www.ietf.org/rfc/rfc2406.txt>
- RFC 2407: **The Internet IP Security Domain of Interpretation for ISAKMP**,
<http://www.ietf.org/rfc/rfc2407.txt>
- RFC 2408: **Internet Security Association and Key Management Protocol (ISAKMP)**,
<http://www.ietf.org/rfc/rfc2408.txt>
- RFC 2409: **The Internet Key Exchange (IKE)**, <http://www.ietf.org/rfc/rfc2409.txt>
- RFC 2410: **The NULL Encryption Algorithm and Its Use With IPsec** ,
<http://www.ietf.org/rfc/rfc2410.txt>
- RFC 2411: **IP Security Document Roadmap**, <http://www.ietf.org/rfc/rfc2411.txt>

- RFC 2412: **The OAKLEY Key Determination Protocol**, <http://www.ietf.org/rfc/rfc2412.txt>
- RFC 2451: **The ESP CBC-Mode Cipher Algorithms**, <http://www.ietf.org/rfc/rfc2451.txt>
- RFC 2857: **The Use of HMAC-RIPEND-160-96 within ESP and AH**,
<http://www.ietf.org/rfc/rfc2857.txt>

IETF

- IETF Security Area info: http://www.ietf.org/html.charters/wg-dir.html#Security_Area
- IETF Security Area dedicated web page: <http://web.mit.edu/network/ietf/sa/>
- IETF IPsec Working Group web page: <http://www.ietf.org/html.charters/ipsec-charter.html>

In Conclusion

In the next article, we'll look at basic IPsec configuration. Then, once we're gone through the bare-bones basics, we'll talk about some of the alternatives we have in configuring IPsec. And we'll of course take a look at a sample IPsec configuration.

Your comments, preferences and ideas and suggestions for topics are always more than welcome! I enjoy hearing from you!

Dr. Peter J. Welcher (CCIE #1773, CCSI #94014) is a Senior Consultant with Chesapeake NetCraftsmen. NetCraftsmen is a high-end consulting firm and Cisco Premier Partner dedicated to quality consulting and knowledge transfer. NetCraftsmen has nine CCIE's, with expertise including large network high-availability routing/switching and design, VoIP, QoS, MPLS, network management, security, IP multicast, and other areas. See <http://www.netcraftsmen.net> for more information about NetCraftsmen. Pete's links start at <http://www.netcraftsmen.net/welcher> . New articles will be posted under the Articles link. Questions, suggestions for articles, etc. can be sent to pjw@netcraftsmen.net .

1/8/2001

Copyright (C) 2001, Peter J. Welcher