SANS Institute
InfoSec Reading Room

IPSec Tunnel Creation

There are many types of VPNs (Virtual Private Networks) available for use in today's networks. One of these types involves use of the IPSec standard. Within IPSec, there are further options on ways to define your VPN. The actual building or construction of the IPSec VPN is very involved. The purpose of this paper is to detail, explain, and illustrate the specific processes that occur in creating an IPSec VPN tunnel. Some of the concepts and theory will also be explained.

SANS GSEC Practical version 1.4b option 1
Chris Gutridge
March 1, 2003

IPSec Tunnel Creation

Abstract

There are many types of VPNs (Virtual Private Networks) available for use in today's networks. One of these types involves use of the IPSec standard. Within IPSec, there are further options on ways to define your VPN. The actual building or construction of the IPSec VPN is very involved. The purpose of this paper is to detail, explain, and illustrate the specific processes that occur in creating an IPSec VPN tunnel. Some of the concepts and theory will also be explained.

Like other VPNs, an IPSec tunnel is secure. It is encrypted using cryptographic techniques. I will be using the example of what one vendor refers to as a site-to-site VPN (How 2). These are not the same processes that occur with a host-to-host or host-to-gateway VPN, although they are similar. There are many attacks that come with IPSec and many that are avoided. I will cover a few of them, when relevant, but not all.

Site-to-Site VPN Overview

In this type of VPN, there are two separate "true" private networks that need to communicate with each other over a public link, the Internet. Every host in each private network needs to communicate with every host in the other private network. To do this, a tunnel is created between the two gateways that border the private networks' Internet connections. These two gateways become the two endpoints that make up the VPN, and include equipment such as firewalls or routers.

The data conversation is actually taking place between any host behind one gateway and any host(s) behind the other gateway. The gateways are only transiting traffic. None of the traffic is actually destined for the gateway. It is important to note that when the data first leaves the sender and reaches the first gateway, it is always in clear text, or unencrypted. The gateway then encrypts it for Internet travel and then the second gateway decrypts it and sends it in clear text to the destination host. This is important because if an adversary can get inside the company's physical building, they can capture data in clear text and circumvent your VPN protection. See Figure 1 for a visual example.

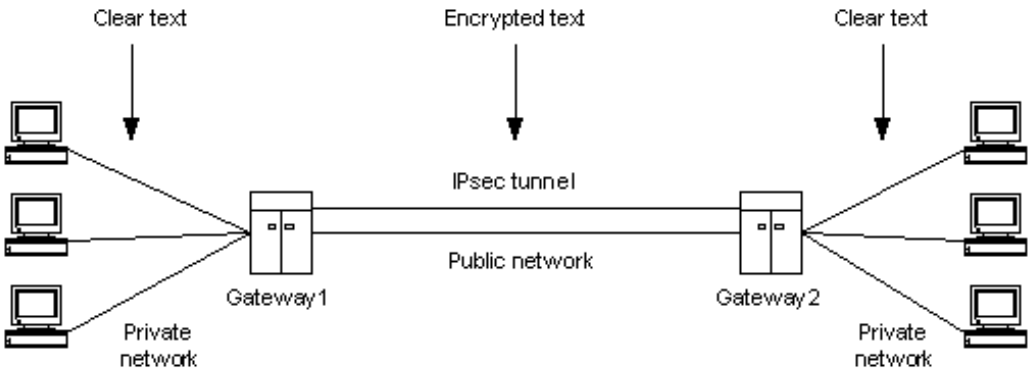

Figure 1. Site-to-Site VPN Overview

In this setup, the users transferring data through the tunnel, probably don't even know they are going through a tunnel. This is a configuration that is set up by the router or firewall administrator(s), not the user. The user may know that they are connecting to the corporate office two thousand miles away, but not be aware of how.

Basically, one system on a private network is trying to communicate with another system on another private network, but travelling through the public network to get there. The gateways are actually doing the work, which is transparent to the user, to provide a secure tunnel between the two private networks.

IPSec tunnel Overview

The rules and processes by which an IPSec tunnel is constructed have been standardized by the IETF (Internet). The current governing document for this is the IKE document (Braden et al. 24).

In general, the process to create an IPSec tunnel is to first establish a preparatory tunnel, encrypted and secure, then from within that secure tunnel, negotiate the encryption keys and parameters for the IPSec tunnel. This first tunnel, or this secure and authenticated channel (Carrel and Harkins 4) is commonly referred to as the ISAKMP tunnel. IKE is actually the protocol and is used to negotiate and authenticate keying material for the tunnel. It is important to note that IKE is a hybrid protocol made up of a combination of ISAKMP (Internet Security Association Key Management Protocol), Oakley, and SKEME (Secure Key Exchange Mechanism for Internet) protocols (Carrel and Harkins 1). IKE does not implement the entire ISAKMP, Oakley, or SKEME protocols (Carrel and Harkins 2). These protocols have specific rules for use within IKE.

This ISAKMP tunnel, once created, continues in session until it expires. Note that there is no expiration specified in any of the RFCs for this, it is vendor-dependent.

For example, on the Cisco PIX Firewall, it can be set from two minutes to twenty-four hours (Cisco 6-25). It creates the IPSec tunnel for a specified interval of time (determined and pre-programmed by the administrator) and then will re-create a new tunnel every time this interval is close to expiring again. See Figure 2 for a visual overview.

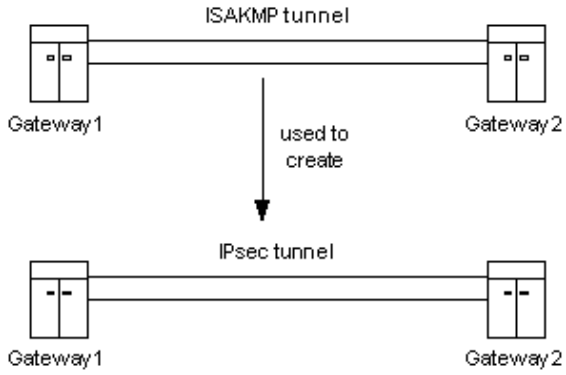

Figure 2. IPsec Tunnel Overview

ISAKMP Tunnel Creation

IKE specifies a two phase method for tunnel creation (Carrel and Harkins 4). A secure ISAKMP tunnel is created upon completion of Phase One. This tunnel is not used until Phase Two begins. Then, this new ISAKMP tunnel is used to create the secure IPSec tunnel, which is in place when Phase Two is complete.

By creating this initial or preparatory tunnel, you are assured that your negotiation of security parameters for the IPSec tunnel is going through a secure channel. So, you are safe there. But this first tunnel was not negotiated from within a secure tunnel, so what about here? You can create as many secure tunnels as you want, but what's ultimately going to matter is how secure was that first tunnel that was created. If the first tunnel was not created securely, then it doesn't matter how secure your IPSec tunnel is. The reality is that your IPSec tunnel is really only as secure as the first tunnel that created it was. Therefore, this really adds an importance to how this first tunnel is created and thus the reason for the interest in this paper.

The security parameters involved in the first tunnel are actually much more secure than those in the IPSec tunnel, as should be. So, here we have a secure tunnel (first tunnel). But, if we have a secure tunnel, more secure than the IPSec tunnel, why should we create a second tunnel in the first place for our IPSec traffic? Don't we want the most secure tunnel we can get? There are two reasons I have found. First, the first tunnel is so secure that it will slow down the traffic. This is made under the assumption that you are choosing much longer key lengths for your ISAKMP

keys. Secondly, having two tunnels allows the first one the ability to periodically "refresh" the keys (negotiate new keys) used in the second tunnel. These features help the security of the IPSec tunnel. If the keys are refreshed, a brute force attack would have to start over because a new key would have to be found. The timing of the refreshes can be set so that a brute force attack on the keys would never have enough time to exhaust all possible keys (Schneier 152). Also, the processor is freed up in the second tunnel because less computation is required and keys don't need to be derived.

ISAKMP Phase One

In Phase One, there are six messages exchanged between the peers (Carrel and Harkins 7). Note that there is an Aggressive Mode also using less messages which I will not be going over here. Three messages are from the Initiator peer and three are from the Responder peer. These are also thought of as three round trips of messages. This round trip method of reference is helpful because there are three types of exchanges going on. Let's define the Initiator as the peer that wants to start an IPSec session with the other peer, whom will be the Responder. In general, when Message One is sent, the two peers are communicating in clear text. By the time the sixth message is sent, their communication is encrypted and both of their identities have been verified (authenticated). At this point, the ISAKMP tunnel is built and ready to use by the peers in the beginning of Phase Two.

In Message One and Two (Round Trip One), the peers negotiate a security policy for their upcoming private session in Phase Two. In Round Trip Two, the peers exchange some Diffie-Hellman values and some secondary values. Finally, in round trip three, the data exchanged in Round Two is manipulated by formulas, then exchanged between the peers and used to authenticate the other peer. Also, the keys are created that will be used to encrypt data in the new secure tunnel in Phase Two. See Figure 3 for a graphic depiction of this.

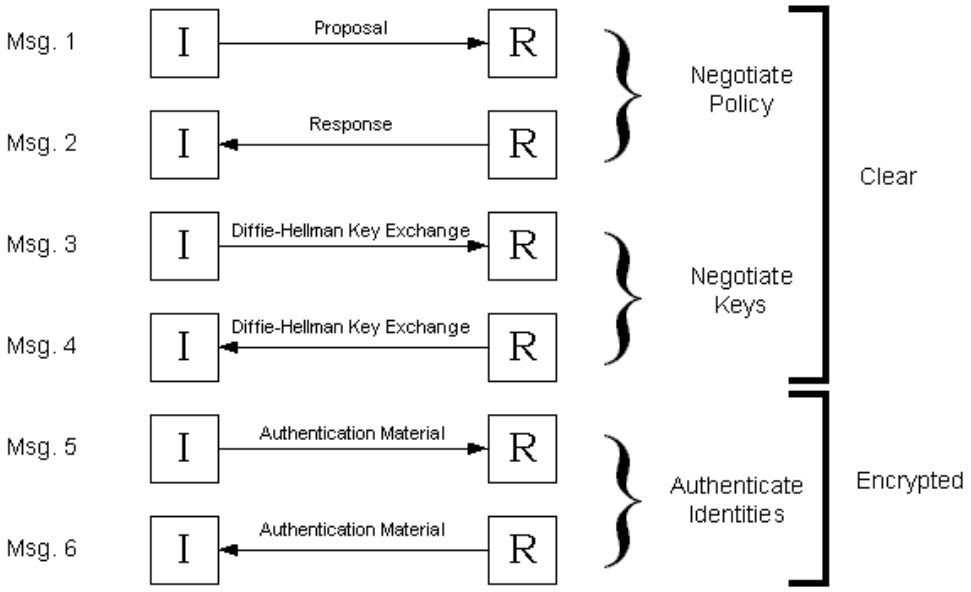

Figure 3. ISAKMP Phase 1 Overview

This 'tunnel' and also the IPSec 'tunnel' are actually just a group of Security Associations agreed to between the peers. These are more commonly referred to as 'SA's. A tunnel has two SAs, one for each direction of data travel. (Atkinson and Kent, Security 7)

It is important to note that up until the authentication round, either peer could be an imposter disguised as one of the peers. The exchange that takes place in Round Three is designed to stop these kind of imposters who should not pass authentication. This adds to the importance of this round trip.

Round Two is arguably as important as Round Three. Round Three is important because you don't want to create a secure tunnel with the wrong person. Round Two is important because you don't want someone to break your encryption and be able to see all your data. They are different concerns but equally important.

Before Round One, policies to be negotiated need to be pre-programmed into the gateways. Usually this is done upon installation of the device by the system administrator. The four main negotiated policy parameters of the ISAKMP SAs are as follows: Encryption algorithm, Authentication method, Hash algorithm, and Diffie-Hellman Group (Carrel and Harkins 6). Below, as I describe each round trip, I describe how and where these parameters are used. The specific attributes available are dependent on the hardware being used. There are some attributes

that are required on all hardware and the rest are optional (Carrel and Harkins 6). See Figure 4 for some common examples. Note that all of these options are not available on all hardware. The hi-lighted items will be used in the example that follows.

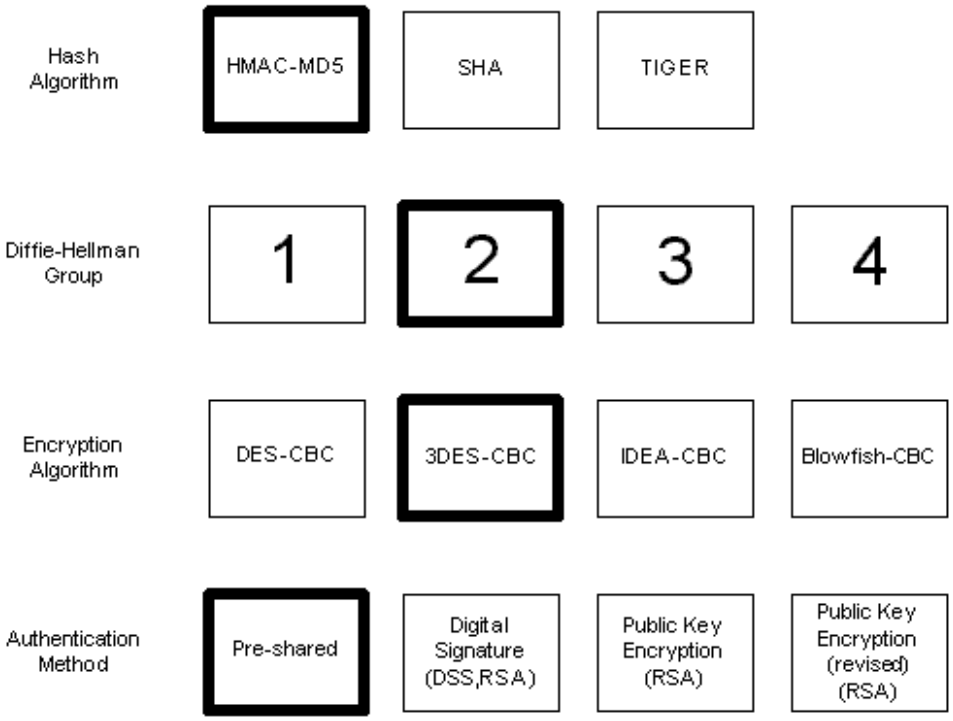

Figure 4. Sample ISAKMP parameters

Phase One, Round One

The purpose of this round is to agree on parameters for the upcoming ISAKMP tunnel. In this round, the Initiator peer will send a suite of proposed security parameters and the Responder will choose one (if its on its acceptable list) and send its choice back. Message One is the proposal and Message Two is the response. After this round is over, both peers have an agreed suite of parameters for the ISAKMP tunnel that will be protecting traffic in Phase Two.

One of the important pieces of this round is the generation and use of the Initiator and Responder cookies. Each peer generates a 64-bit cookie that is included in every subsequent message (Maughan et al. 20). In Message One, the Initiator cookie is sent, and in Message Two, the Responder cookie is sent along with the Initiator cookie (Maughan et al. 17). In subsequent messages, both cookies are sent

in the ISAKMP header. These cookies are used throughout Phase One to prevent Denial of Service type attacks (Maughan et al. 11), and later in keying material generation (see Phase Two below). For clarity, the symbolic representations throughout this paper will not necessarily correspond to the symbols used in the official documents of these protocols. My symbolic representations for the cookies are as follows:

CKY_I = Initiator's cookie
CKY_R = Responder's cookie

The payloads in the following diagrams (Figures 5a, 5b) are symbolic and are not perfectly representative of the actual complete IP packet. All of the packets in Phase One are required to use UDP port 500 for transport (Piper 18). For brevity, only the relevant items to this paper will be shown. I have also used the data from Figure 4 above for one of the proposed security suites in Message One and the actual chosen one in Message Two. The boxes represent encapsulation of payloads and are representative of actual packet construction.

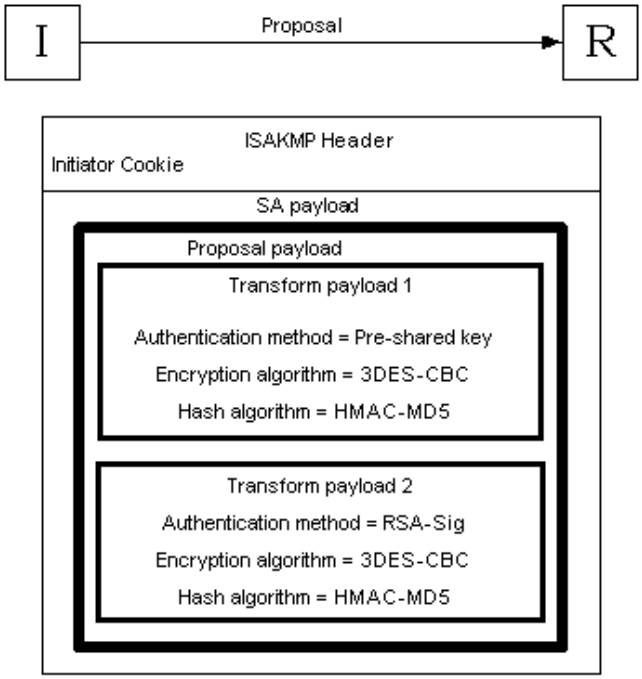

Figure 5a. ISAKMP round trip #1 (message 1)

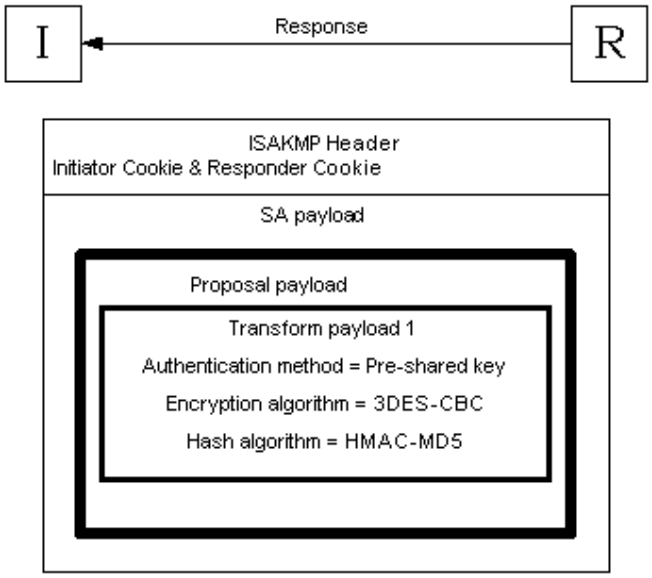

Figure 5b. ISAKMP round trip #1 (message 2)

Phase One, Round Two

The purpose of this round is to come up with a shared secret value only known to the Initiator and Responder, while communicating in clear text. This will later be used to create the keys for the ISAKMP SAs.

This is done by exchanging four non-secret (public) values. Each side will calculate four values. First, each side generates a private value from scratch that doesn't get sent across the channel. Based on this value, the first public value is generated. A second public value is generated from scratch. Then, these two public values are exchanged with the other peer's public values. Then, the two received values from the opposite peer, together with the first private value, are used to generate a fourth value which becomes the shared secret.

Note that this new shared secret has never gone across the wire. Also note that the two peers amazingly come up with the same value for the shared secret. This is amazing because their private inputs are different!

A value has been created here using three variables. Two out of three of these variables are available to an attacker/eavesdropper. Without this third variable, he cannot calculate this shared value. Note that there are two different values for this third variable. The Initiator uses one and the Responder uses another. Therefore an attacker has two values that he can attempt to obtain instead of just one. This third missing variable is the Diffie-Hellman private value described later and is pseudo-randomly generated and very tough to obtain.

Round Two Pre-exchange Calculations

First, each side computes two pseudo-random values. These will be called the nonce (Orman 3) and the Diffie-Hellman private value (Van Oorschot and Wiener 1). The nonce is sent across the channel in clear text and the Diffie-Hellman private value is kept secret. The Diffie-Helman private value is used to calculate the Diffie-Hellman public value, and then this public value is sent across the channel in clear text. These are described in detail below.

Pseudo-Random Numbers

Pseudo-random numbers are used here versus true random numbers. A mathematician would say that true random numbers only occur in nature (Schneier 44). The next best numbers come from hardware that is designed and dedicated to creating random numbers (Schneier 44). The next best numbers after that come from software-based Pseudo-Random Number Generators (PNG) based on Pseudo-Random Functions (PRF). The best of these work by obtaining random input from various uncorrelated sources and mixing them together with strong mixing functions (Crocker, Eastlake, and Schiller 13). This method is employed here.

This PRF is negotiated and agreed to as with the other items above in Figure 4. But, this is an optional negotiation item (Carrel and Harkins 6) and is often not specified by the system administrator. When this occurs, IKE requires the peers to use the chosen hash algorithm as the Pseudo-Random Function (Carrel and Harkins 6).

Pseudo-Random Function (PRF)

In order for a hash algorithm to be used as a PRF, it has to be ran in HMAC (Hashed Message Authentication Codes) mode. The negotiated hash algorithm can be ran in two modes: native and HMAC (Carrel and Harkins 6). In this paper, MD5 (Message Digest 5) is the chosen algorithm as shown in Figure 4.

In normal mode, MD5 is a function that accepts one arbitrary length input, which is converted to a multiple of 512 bits, and outputs a fixed-length output of 128 bits (Rivest 1). It cannot be reverse-engineered because of the mathematical properties of the algorithm. It is these mathematical properties that make it such a good choice to use for encryption or confidentiality. This function is also referred to as a one-way hash function and a compression function. Normal mode is represented symbolically as follows:

MD5 (message)

The simplest way to key a function would be to simply add a key to the front of the standard input message and concatenate it together and then run it through the formula as if it were one input, see below. The comma represents concatenation.

Simple-Keyed-MD5 = MD5 (key, message)

Bellare, Canetti, and Krawczyk explain in their paper (1), this method is not secure or random enough for pseudo-random number generation. So, they created the HMAC method. The HMAC mode is an adaptation of the normal MD5 function. There are two separate, but dependent, iterations of MD5 for each one iteration of HMAC-MD5. The core algorithm used is not changed, it is just re-used, if you will, using the output of the first one. HMAC acts as a front-end to MD5. The formula for HMAC-MD5 follows (Bellare, Canetti, and Krawczyk 14):

HMAC-MD5 = MD5 ([K xor OPAD] , [MD5 (K xor IPAD, X)])

This function is parsed and computed like a mathematical function. The internal brackets are computed first and become the equivalent of the key and message above. Then, like above, these are concatenated together and ran through the MD5 function. This mixing function combines logical operations using XOR, more concatenation, bit padding (extending), and iterated function operations to create this random number transformation. Note that the last function that occurs is MD5, which makes the final output 128 bits. See Bellare, Canetti, and Krawczyk's paper for more details on this process.

Basically, the mixing details ensure the pseudo-randomness of the result. The padding ensures the length of all the inputs are fixed (Rivest 2). The XOR is the logical bit-wise exclusive-OR operator. Concatenation hides the beginning and ending of the data from each variable. Iterated operations ensure the function against collision attacks. Because of the complex mixing techniques, the collision attacks used on the normal MD5 algorithm are infeasible on the HMAC version (Bellare, Canetti, and Krawczyk 15).

Nonce

The nonce is a Pseudo-Random number generated locally by the designated PRF (Orman 3), in this case, HMAC-MD5. The purpose of the nonce is to guarantee liveness and protect against replay attacks (Maughan et al. 36). It will also be used to derive the Diffie-Hellman Shared Secret (see below).

The interesting thing about the nonce is that if you think about the fact that it will be sent over the open wire, in clear text, then why the worry about security. Actually, there is no security in the nonce. The important thing about the nonce is that it is as random as possible for its use in the upcoming equations.

The nonce is required to have a minimal amount of randomness (Orman 8). Entropy describes the level of randomness. The nonce's entropy level must be equivalent to the strength of the negotiated Diffie-Hellman group (Orman 8). In this case, Group Two is based on a 1024 bit prime number, but its strength, instead, depends on a

few factors (Carrel and Harkins 28). This makes its entropy somewhere between 60 and 180 bits. Below are symbolic representations of the nonces.

N_I = Pseudo-random number from Initiator (from HMAC-MD5 algorithm)
N_R = Pseudo-random number from Responder (from HMAC-MD5 algorithm)

Diffie-Hellman Values

The Diffie-Hellman (DH) private value is another pseudo-random value generated by the PRF, HMAC-MD5, but will never get sent across the channel. This is the only value that is kept secret during this round. This is probably the most important value to keep secret in all of IKE, and therefore IPSec. Each peer generates one of these values. If these values become known by an attacker, its all over. Everything else is already in the open, now this. An attacker would easily be able to create the same keys that you're about to create and any IPSec keys that you create from within your ISAKMP tunnel.

After the private DH value is generated, the DH public value can be generated. These must be generated in sequence, first the private value, then the public value. This is because the private value is a required variable in the formula to calculate the public value. Mathematically speaking, the public value is a function of the private value. This public value is then sent across the channel with the nonce above. Once again, just as it is important to keep the DH private value secret, it is also important that the algorithm used to create the public value from the private value not be reversible.

Private DH value generation

The private value is a pseudo-random number and is just created as stated above using the current PRF. Below are symbolic representations of the DH private values:

X_I = Private DH value for Initiator
X_R = Private DH value for Responder

Diffie-Hellman Groups

When speaking of Diffie-Hellman groups, one is referring to Diffie-Hellman public values. These groups are also called Oakley groups because they originated with the Oakley protocol (Carrel and Harkins 20). But do not confuse the group numbers. The original Oakley groups were different for groups three, four, and others. The group numbers now are defined by IKE standards.

The two main types of groups used today are Modular Exponentiation Groups (MODP) and Elliptic Curve Groups (EC2N). Both of these have a connection with prime numbers. The whole security of these groups is based on solving some

intractable math problem. An intractable math problem is a problem that is believed to not be solvable in polynomial time (SANS 2-5). This is agreed to by the worldwide mathematical community that is active in researching issues in the field of computation complexity (SANS 2-5). Polynomial time is approximately equivalent to your lifetime. It is important to note that there is no mathematical proof that these problems can't be solved easily, we only have the years of public scrutiny that suggests that (SANS 2-5). For the MODP groups, the intractable problem is solving the Discrete Logarithm Problem over finite fields (SANS 2-7). For the EC2N groups, its also solving the Discrete Logarithm Problem but as applied to elliptic curves (SANS 2-8).

Public DH value generation

The public DH value is derived from a formula (Orman 36): $2^X \pmod P$. The X is the DH private value derived above and the P is a special prime number. The 2 and the P are both hard-coded constants, with the only variable being X (Carrel and Harkins 21). The P is different depending on the DH group that is being used. In our case, Group 2 uses a 1024-bit prime number (Carrel and Harkins 21). Recall that X is only 128 bits long because its an output from HMAC-MD5. The X is also referred to as the Diffie-Hellman exponent.

The basic function employed here is exponentiation but using modular arithmetic. First, 2 is raised to the X^{th} power, then modulus is ran on the result. Modulus is division, but instead of looking for the quotient, you look for the remainder. For example, (8 / 4 = 2), but (8 mod 4 = 0). Note that when you raise 2 to the X^{th} power, you are not raising it to the 128^{th} power, but to the decimal equivalent of the 128-bit number. This ensures that 2^X is always larger than P and that there is a remainder. The following are symbolic representations of the public DH value. Note that 'g' is always 2 in our example.

g^{X_I} short for $[g^{X_I} \pmod P]$ = Public DH value for Initiator
g^{X_R} short for $[g^{X_R} \pmod P]$ = Public DH value for Responder

Round Two Exchanges

At this point, each side has generated a Diffie-Hellman public value, or half-key (Orman 12), and a unique nonce. These are now passed across the wire first from the Initiator in Message Three and second from the Responder in Message Four. See Figure 6a and 6b for visual depictions.

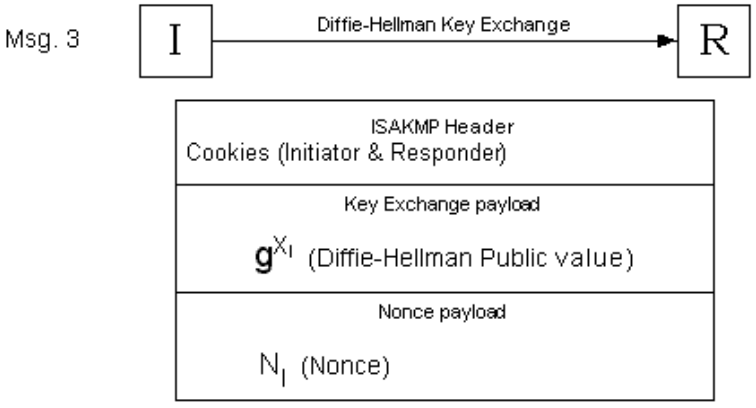

Figure 6a. ISAKMP round trip #2 (message 3)

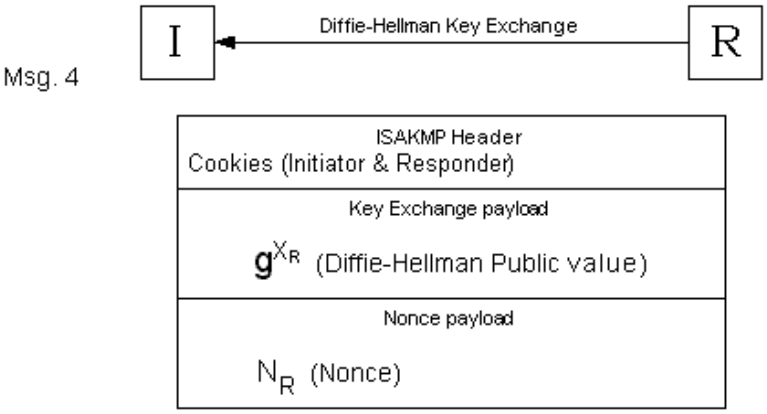

Figure 6b. ISAKMP round trip #2 (message 4)

Round Three Pre-Calculations

After the two messages are exchanged, a number of things are calculated. Each side will calculate keying material called the Diffie-Hellman Shared Secret. This will be used to create the keys. If a pre-shared key is used, as in our case, it is also used here.

Diffie-Hellman Shared Secret

The Diffie-Hellman Shared Secret must be computed first as one of the other formulas depend on its value. Note in the formulas below that the first formula after the equal sign is shown how it is commonly written. To the right, I've included the complete mathematical operations for a clear idea of how they are derived (Van Oorschot and Wiener 1).

$$\mathbf{DHSS} = (\text{other peer's public value})^{\text{your own private value}}$$

$$\mathbf{DHSS}_I = g^{X_R X_I} = (g^{X_R})^{X_I} \text{ short for } [(g^{X_R})^{X_I} \bmod P]$$

$$\mathbf{DHSS}_R = g^{X_I X_R} = (g^{X_I})^{X_R} \text{ short for } [(g^{X_I})^{X_R} \bmod P]$$

$$\mathbf{DHSS} = \mathbf{DHSS}_I = \mathbf{DHSS}_R$$

This fourth equation states that the previous two are equal to each other. They are equal because of the mathematical properties of exponents.

Pre-Shared Key

For pre-shared key authentication, the other shared secret is the pre-shared key. Remember, this key is pre-determined and manually programmed into each peer and is never passed across the line.

Keying Material Derivation

Four pieces of keying material must be calculated and all must be calculated in order. SKEYID is the first of these, and its formula follows (Carrel and Harkins 9):

$$\mathbf{SKEYID} = \text{HMAC-MD5}[(\text{pre\_shared\_key}), (N_I \mid N_R)]$$

Recall that HMAC-MD5 is used as a keyed MD5 function. The "|" also symbolizes concatenation of values. If you recall the definition of HMAC-MD5, the left is the key side, represented here by the pre_shared_key. The right is the message side, represented here by the concatenation of nonces. Note that the order of the nonces is the same for both peers, N_I is always first. This is true because there is only one formula for SKEYID. The comma would normally symbolize concatenation, but don't confuse simple-keyed-MD5 with HMAC-MD5. The comma separates the two sides which are plugged into the HMAC-MD5 function defined earlier.

It is important to note that the formula for SKEYID changes depending on the authentication method. If digital signatures or public keys are used, there is a different formula to derive SKEYID. There would also be a difference in Phase One messages Three and Four (Carrel and Harkins).

Note that this value is secret only because the pre-shared key is secret. Both nonces were sent across the channel in the clear. Also, note that SKEYID is never sent across the channel. This is why it is important to choose a strong value for your pre-shared key.

At this point, the following keying material is generated from the SKEYID (Carrel and Harkins 9). Like SKEYID, these are all 128-bit values (because of HMAC-MD5). Unlike SKEYID above, these formulas are the same for all types of authentication. This is true because they are all generated after SKEYID.

SKEYID_D = HMAC-MD5[(SKEYID) , (DHSS | CKY_I | CKY_R | 0)]

SKEYID_A = HMAC-MD5[(SKEYID) , (SKEYID_D | DHSS | CKY_I | CKY_R | 1)]

SKEYID_E = HMAC-MD5[(SKEYID) , (SKEYID_A | DHSS | CKY_I | CKY_R | 2)]

The process for processing these formulas is just like SKEYID above. Everything in parenthesis is concatenated first, then the right and left sides are processed through the HMAC-MD5 mixing process. The numbers at the end are just hard-coded constants(Carrel and Harkins 9).

If you examine these formulas, you will notice that they have to be generated in the sequence listed above. This is because part of the formula for one is dependent on the previous one. You will also notice that by the time you get to SKEYID_E, the numbers have been through many iterations, adding to the security of the SKEYID_E keying material.

Although these keying materials must be generated in order, they don't have to be used in any particular order, but rather for different purposes. The different letters represent the different purposes of these keying materials (Carrel and Harkins 3). The 'D' stands for derive because it is used to derive the keys for the IPSec Security Associations. The 'A' stands for authenticate because it is used to authenticate ISAKMP messages. The 'E' stands for encryption because it is used to encrypt ISAKMP messages.

At this point, everything is in place for creating the keys for Phase Two and encrypting the channel, except the peers need to be authenticated.

Phase One, Round Three

This round provides the authentication of the peers, so that the first secure tunnel (ISAKMP SA) can be created. All the material is now in place to create a secure tunnel and it can be created right now, but you would have no guarantee that the tunnel is going to where you think its going. To have this guarantee, authentication is performed. To authenticate each other using pre-shared keys, a Hash Payload and an Identification Payload are exchanged (Carrel and Harkins 15). Also, the Hash Payload along with the Identification Payload are each encrypted (Carrel and Harkins 2). The algorithm used is 3DES-CBC, from Figure 4, and the key is SKEYID_E from above (Carrel and Harkins 3).

Hash Payload

The hash consists of some secret values and some public values. The hash formulas differ for each peer. In the hash formulas shown below, the SKEYID derived earlier is used to key the function. The message data (on the right) is mostly public information and has been sent across the wire before. The exception is the ID_I and ID_R, which are being sent across the wire now in the ID Payload and in Message Six. The SA_I is the entire SA payload offered in Message One by the Initiator excluding the ISAKMP header (Carrel and Harkins 2). The ID_I is the entire Identification Payload without the generic ID Payload header (Carrel and Harkins 3).

Identification Payload

The ID payload includes information about the Initiator of this tunnel request. It includes information such as the IP address or range of IP addresses to apply tunnel parameters to, the protocol being used, and port being used (Piper 18).

Following are the formulas for the two hashes (Carrel and Harkins 9).

$$\textbf{HASH\_I} = \text{HMAC-MD5}\left[(\text{SKEYID}), (g^{X_I} \mid g^{X_R} \mid CKY_I \mid CKY_R \mid SA_I \mid ID_I)\right]$$

$$\textbf{HASH\_R} = \text{HMAC-MD5}\left[(\text{SKEYID}), (g^{X_R} \mid g^{X_I} \mid CKY_R \mid CKY_I \mid SA_I \mid ID_R)\right]$$

The following Figures (7a, 7b) show a visual depiction of Round Three.

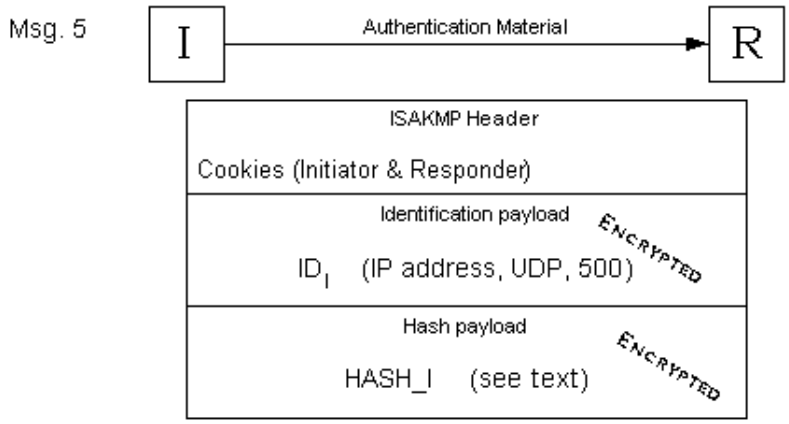

Figure 7a. ISAKMP round trip #3 (message 5)

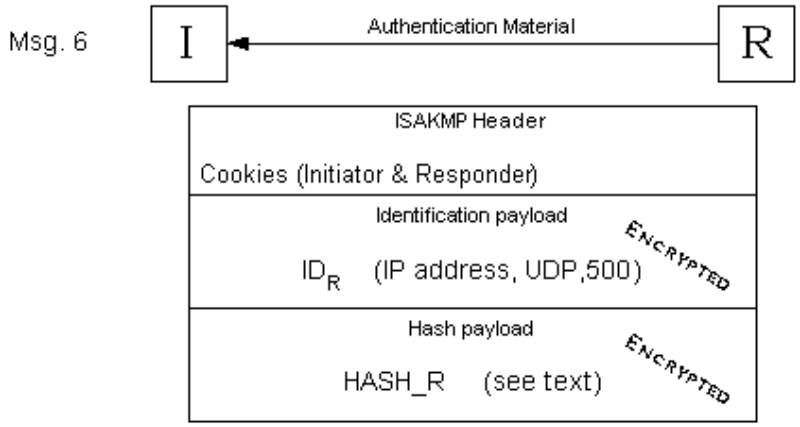

Figure 7b. ISAKMP round trip #3 (message 6)

Round Three Post-Calculations

As of now, both sides have exchanged a hash and an ID payload. Authentication is accomplished by the fact that the other peer can reproduce the hash that you sent to it (Carrel and Harkins 9). This should not be possible unless the other peer knows the value of SKEYID, which was never sent over the wire. The other peer does know this value because it also calculated it between Rounds Two and Three. Note that the last of the inputs to be sent across the open wire, ID_I and ID_R , have just been sent across in this round. Therefore, each peer now reproduces the hash and compares it to the one sent to it. If it matches, then it will consider the other's identity true.

ISAKMP Phase Two

The purpose of Phase Two is to negotiate the parameters for an IPSec SA (Carrel and Harkins 4). These are also known as "Non-ISAKMP SAs" (Carrel and Harkins 7). These SAs will be the channel that the actual data will be travelling through for the two gateways. There are three messages involved in Phase Two (Carrel and Harkins 17). All three messages are secure and use keys derived in Phase One (Carrel and Harkins 9). By the end of Message Three, the keys will be created for the IPSec SAs. Nonces are also exchanged here for replay protection (Carrel and Harkins 16). See Figure 8 for a visual overview of this.

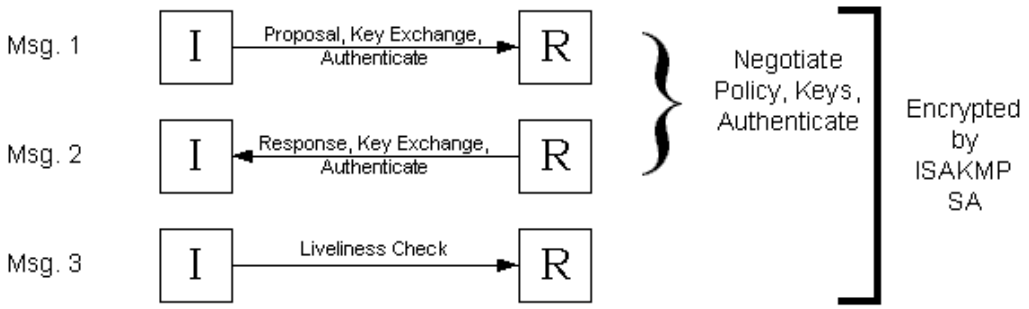

Figure 8. ISAKMP Phase 2 Overview

Keep in mind that at this point, we are communicating through a secure (encrypted) tunnel. There will be more independent keys generated during this phase, but keep in mind that someone would first have to break into the ISAKMP tunnel that you are now communicating in.

Phase Two, Message One

In Message One, a proposal is made for a protection suite. For our example, we will pick two protection suites: ESP-DES-MD5 and AH-MD5. Five things are sent across from the Initiator to the Responder: ISAKMP header, a Hash Payload, an SA Payload, a Nonce Payload, and a Key Exchange Payload (Carrel and Harkins 17). The Key Exchange is optional, but required if you want PFS (Perfect Forward Secrecy) (Carrel and Harkins 16). The reason why its not mandatory is that this mode (Quick Mode) is also used for refreshing keys. When refreshing keys, its much faster to skip the Key Exchange. All of these payloads are encrypted except for the ISAKMP header (Carrel and Harkins 15) using the same method described in Phase One, Round Three. See Figure 9 for a visual explanation.

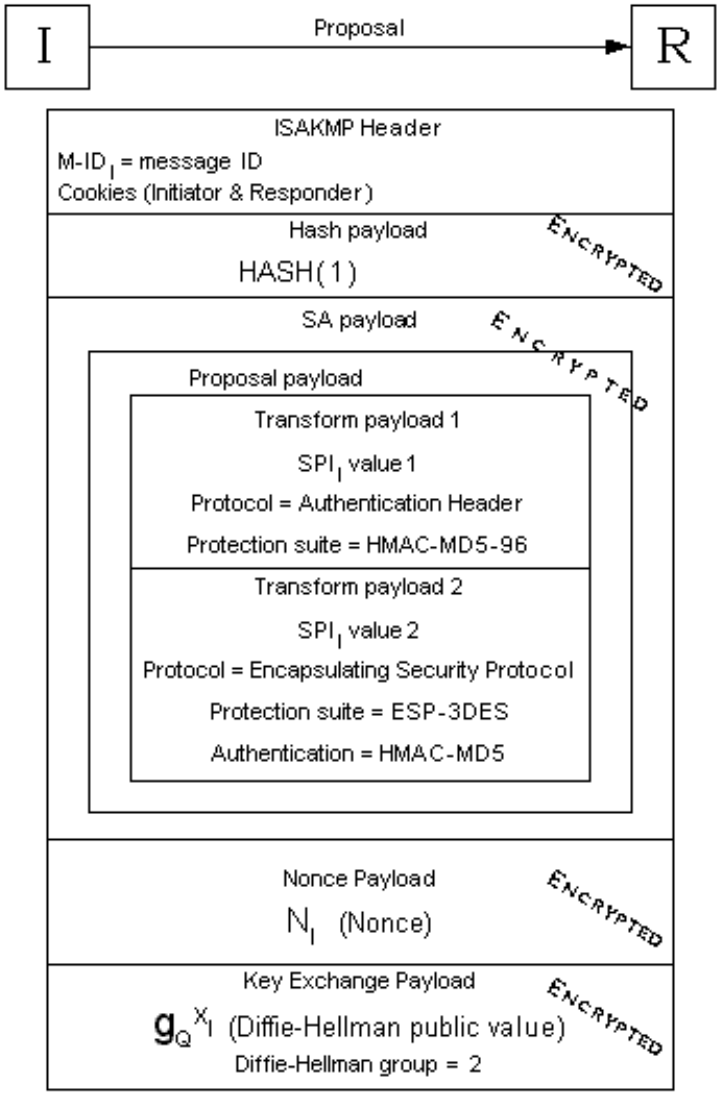

Figure 9. Phase 2, Message 1

Hash Payload

The purpose of this Hash is to authenticate the message and provide proof to the other peer that it is alive (Carrel and Harkins 15). The authentication is done in the same manner as in Phase One, Round Three. The Hash Payload uses the negotiated PRF from Phase One to perform the mixing of its inputs.

Here, the SA, Nonce, and Key Exchange payloads, along with the Message ID (found in the ISAKMP header) are mixed using the SKEYID_A key derived previously in Phase One. Notice that all payloads in this packet are included in the input of this hash including only part of the ISAKMP header, the Message ID. The following is the formula for the hash (Carrel and Harkins 18):

HASH(1) = HMAC-MD5 $[($ SKEYID_A $) , ($ M-ID | SA | N_I | KE $)]$

Note that this hash does not include the variables that were input into it, but only a hash of them. The actual variables are transported in their separate, respective payloads below. Also note that the hash must be constructed last because it depends on those other variables.

SA Payload

The SA Payload includes the Proposal Payload. More than one Proposal Payload can be included, but only for negotiating multiple non-ISAKMP SAs, which we are not doing here (Carrel and Harkins 8). The Proposal Payload includes the Transform Payloads. These are similar to the ones offered in Phase One, Message One, except these are destined for the final IPSec tunnel. The general options being offered here are the AH (Authentication Header) protocol and the ESP (Encapsulating Security Payload) protocol. The specific options offered here for AH are to use the HMAC-MD5-96 protection suite. The specific options here for ESP are to use the 3DES protection suite and use HMAC-MD5 for authentication.

The main difference between AH and ESP here is that ESP offers confidentiality through encryption. For more detailed information on these specific protocols see (Atkinson and Kent, IP Authentication 1) and (Atkinson and Kent, IP Encapsulating 2). Each transform also includes a unique pseudo-randomly generated value called the SPI (Security Parameter Index) (Maughan et al. 62). Note that this is a 32-bit value. The Protocol ID for each proposed protocol (ESP-DES, AH-MD5) is also included in the payload.

Nonce Payload

The Nonce Payload includes a pseudo-random number generated by the agreed upon PRF in Phase One (HMAC-MD5). The purpose of this nonce is to provide

Replay Protection (Carrel and Harkins 17). The nonce provides a freshly generated value to combat imposters that try use previous packets to create fake SAs.

Key Exchange Payload

The Key Exchange Payload is an optional payload, but should be included for PFS (Carrel and Harkins 16). Perfect Forward Secrecy is the idea that if a single key is compromised, only data protected by that key is also compromised. If that single key is used to derive new keys, or the key which derived that single key is re-used to derive new keys, the new keys could be compromised if you knew the first single key (Carrel and Harkins 4). With PFS, the keys are independently derived and can't be compromised. By performing a key exchange, we are duplicating what was done in Phase One, Round Two and basically deriving a key from scratch.

The contents of the Key Exchange Payload are the same as in Phase One. The values passed are the Diffie-Hellman public values and the Diffie-Hellman group number. As in Phase One, this Diffie-Hellman public value must be computed and is based on a random Diffie-Hellman private value.

After each payload above is generated, each is individually encrypted using the agreed encryption algorithm from Phase One, 3DES-CBC, including the hash. The payloads are put together and then sent as Message One.

Phase Two, Message Two

Message Two is a response to the proposal in Message One. A proposal is chosen and sent back to the Initiator. Message Two is just like Message One in that a Hash, SA, Nonce, and Key Exchange Payload are sent (Carrel and Harkins 17). The difference is that the SA payload will contain the choice of transforms from the proposal in Message One. The nonce will be the Responder's nonce and the Key Exchange Payload will contain the Responder's Diffie-Hellman public value. See Figure 10 for a visual explanation.

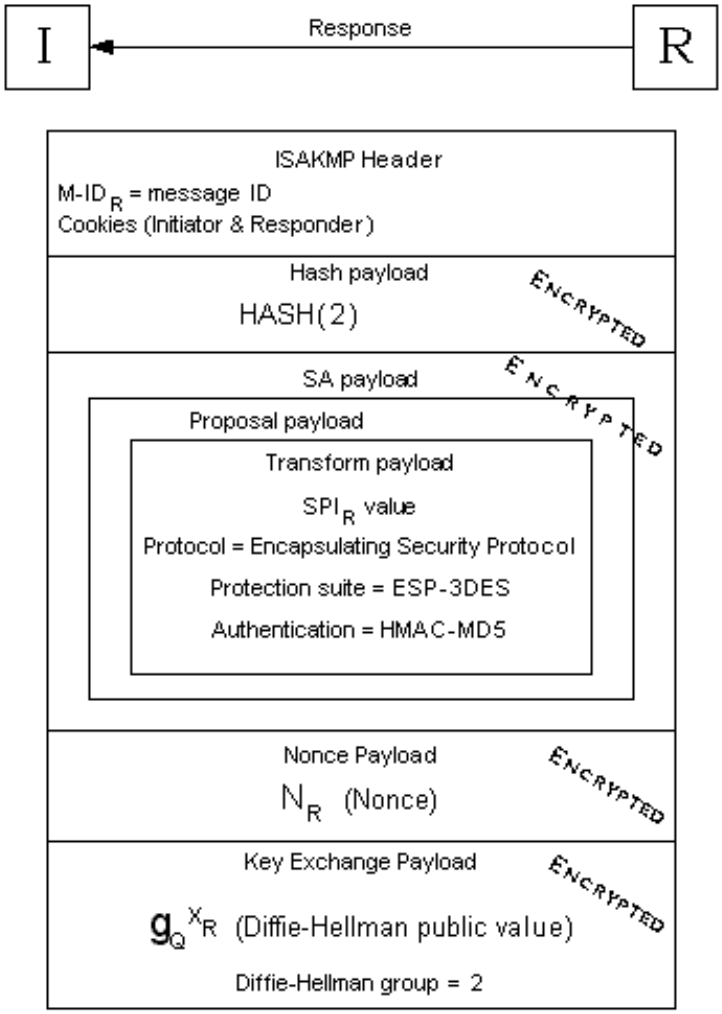

Figure 10. Phase 2, Message 2

Hash Payload

In this hash, an additional nonce is added to the concatenation on the message side of the hash input. The "N_{IB}" is the additional value put in for liveliness proof (Carrel and Harkins 17). This is actually the Initiator's bare nonce; the nonce payload without the payload header. The formula for this hash is as follows (Carrel and Harkins 17):

HASH(2) = HMAC-MD5 $\left[\left(\text{SKEYID\_A} \right) , \left(\text{M-ID} \mid N_{IB} \mid \text{SA} \mid N_R \mid \text{KE} \right)\right]$

SA Payload

The SA payload is a response to the SA payload received in Message One. Here, the Responder chooses a transform and sends the choice back to the Initiator. Our

choice will be ESP with 3DES encryption and HMAC-MD5 authentication. The Responder also generates its own pseudo-random SPI and includes it in the payload) (Maughan et al. 62). The protocol ID for the chosen protocol is also included in the payload (Carrel and Harkins 17).

Nonce payload

This is the Responder's pseudo-random number generated by the agreed upon PRF, HMAC-MD5.

Key Exchange payload

This is the Responder's Diffie-Hellman Public Value and group number. Just as in Message One, first a pseudo-random value is generated and called the Diffie-Hellman private value. Then, this value uses the formula from the agreed upon DH Group number and derives the DH public value.

Like Message One, all the above payloads including the hash payload are individually encrypted and then brought together to form Message Two. Message Two is then sent.

Message Three Pre-Calculations

At this point, both peers have received and authenticated each other by using Hash(1) and Hash(2). Also, Diffie-Hellman public values and nonces have been exchanged. These values are now used to derive keying material for the non-ISAKMP SAs. First, the new Diffie-Hellman Shared Secret is derived ($DHSS_q$), then the keying material (KEYMAT). The "q"s denote that these are Quick Mode variables. The Diffie-Hellman Shared Secret will be the same for both peers. The keying material will not be the same for both peers. There will be two sets of keying material, one for each direction of traffic. One will be for traffic going from Initiator to Responder ($KEYMAT_{IR}$) and one will be from Responder to Initiator ($KEYMAT_{RI}$). The formulas for these are as follows (Carrel and Harkins 17):

$$\mathbf{DHSS}_{QI} = g_q^{X_R X_I} = (g_q^{X_R})^{X_I} \text{ short for } [(g_q^{X_R})^{X_I} \bmod P]$$

$$\mathbf{DHSS}_{QR} = g_q^{X_I X_R} = (g_q^{X_I})^{X_R} \text{ short for } [(g_q^{X_I})^{X_R} \bmod P]$$

$$\mathbf{DHSS}_{Q} = \mathbf{DHSS}_{QI} = \mathbf{DHSS}_{QR}$$

$$\mathbf{KEYMAT}_{IR} = \text{HMAC-MD5}\,[(SKEYID\_D), (DHSS_Q \mid protocol \mid SPI_I \mid N_I \mid N_R)]$$

$$\mathbf{KEYMAT}_{RI} = \text{HMAC-MD5}\,[(SKEYID\_D), (DHSS_Q \mid protocol \mid SPI_R \mid N_I \mid N_R)]$$

The SKEYID_D is one of the resulting key materials from Phase One. The protocol and SPIs are from the proposal payloads. The SPIs will be used to identify each SA, one for each direction. Note that even though the differences are slight in the inputs of the two KEYMAT formulas, their results will be quite different. The nonces are from Message One and Message Two above.

Note that the KEYMAT formulas change if PFS is not needed (Carrel and Harkins 17). There would not be a Key Exchange element ($DHSS_Q$).

Phase Two, Message Three

The purpose of Message Three is for the Initiator to prove its liveliness to the Responder (Carrel and Harkins 17). It does this by sending a hash only, HASH(3). The hash is keyed with the same key as HASH(1) and (2), but on the message side is an 8-bit zero value, the Message-ID (from the ISAKMP header), and both of the bare nonces from messages One and Two (Carrel and Harkins 17). Note that only this hash is sent and not the additional payloads as in Messages One and Two. The Responder peer receives this message and verifies it by reproducing the same hash independently, then negotiations are over and peers can start using the SAs. See Figure 11 for a visual representation. The formula for HASH(3) follows:

HASH(3) = HMAC-MD5 $\left[\left(\text{SKEYID\_A} \right) , \left(0 \mid \text{M-ID} \mid N_{IB} \mid N_{RB} \right)\right]$

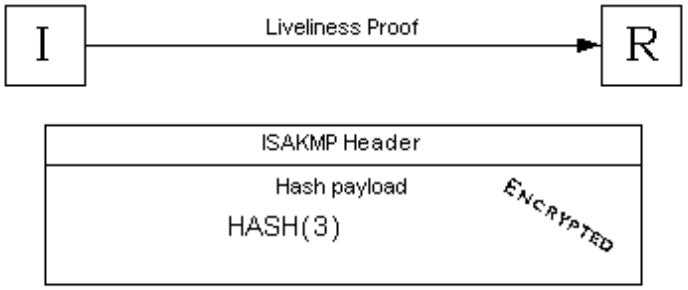

Figure 11. Phase 2, Message 3

After Message Three

The IPSec tunnel is now complete. The security protocol is now put into action. In this case, the derived keying material $KEYMAT_{IR}$ and $KEYMAT_{RI}$ will be used to create the keys needed in the IPSec tunnel (Carrel and Harkins 18). User data will now pass through and be transformed by these keys using the ESP protocol. This continues until the SAs expire or are renewed.

Conclusion

IPSec is a very secure way to create a VPN. The technology behind the creation of secret keys and encryption is very complex, as you can see from this paper, but on the surface, the average person doesn't see this, nor do they need to. ISAKMP and IKE are mature, industry proven standards that have made it through the real-world challenges facing security protocols. Today's IPSec construction standards have taken into account all of the various attacks that have surfaced since the time of its inception. The complexity of these protocols demonstrate the time and thought that has gone into their development. After studying this, I have a deeper respect for the ingenuity that goes into these processes and the theories behind them. One thing that stands out here is the modular construction of the protocols. The ability for a user to choose their own algorithms, etc. is an example of a well-designed and well-thought-out framework with the future in mind. The structure is set for integration with future algorithms, key generation, and key transport methods.

Works Cited

Atkinson, R., Kent, S. "IP Authentication Header." RFC 2402, The Internet Society (Nov 1998). <http://www.ietf.org/rfc/rfc2402.txt>

---. "IP Encapsulating Security Payload (ESP)." RFC 2406, The Internet Society (Nov 1998). <http://www.ietf.org/rfc/rfc2406.txt>

---. "Security Architecture for the Internet Protocol." RFC 2401, The Internet Society (Nov 1998). <http://www.ietf.org/rfc/rfc2401.txt>

Bellare, Mihir, Canetti, Ran, Krawczyk, Hugo. "Keying Hash Functions for Message Authentication." Lecture Notes in Computer Science. Crypto 96 Proceedings (June 1996).

Braden, R., De La Cruz, A., Ginoza, S., Reynolds, J. "Internet Official Protocol Standards." RFC 3300, The Internet Society (Nov 2002).

Carrel, D., Harkins, D. "The Internet Key Exchange (IKE)." RFC 2409, The Internet Society (Nov 1998). <http://www.ietf.org/rfc/rfc2409.txt>

Cisco PIX Firewall Command Reference, Version 6.2. Cisco Systems (Jan 2003). <http://www.cisco.com/univercd/cc/td/doc/product/iaabu/pix/pix_62/cmdref/gl.htm>

Crocker, S., Eastlake, D., III, Schiller, J. "Randomness Recommendations for Security." RFC 1750, IETF Network Working Group (Dec 1994). <http://www.ietf.org/rfc/rfc1750.txt>

How Virtual Private Networks Work. Cisco Systems (Nov. 2002). <http://www.cisco.com/warp/public/471/how_vpn_works.shtml>

Internet Engineering Task Force Web Site (Feb 2003). <http://www.ietf.org/>

Maughan, D., Schertler, M., Schneider, M., Turner, J. "Internet Security Association and Key Management Protocol (ISAKMP)." RFC 2408, The Internet Society (Nov 1998). <http://www.ietf.org/rfc/rfc2408.txt>

Orman, H. "The OAKLEY Key Determination Protocol." RFC 2412, The Internet Society (Nov 1998). <http://www.ietf.org/rfc/rfc2412.txt>

Piper, D. "The Internet IP Security Domain of Interpretation for ISAKMP." RFC 2407, The Internet Society (Nov 1998). <http://www.ietf.org/rfc/rfc2407.txt>

Rivest, R. "The MD5 Message-Digest Algorithm." RFC 1321, The IETF Network Working Group (Apr 1992). <http://www.ietf.org/rfc/rfc1321.txt>

SANS Institute. "Introduction into Encryption II", SANS Security Essentials IV: Encryption and Exploits, Course Material (Apr 2002)

Schneier, Bruce. Applied Cryptography Second Edition: Protocols, Algorithms, and Source Code in C. New York: Wiley, 1996.

Van Oorschot, Paul C., Wiener, Michael J. "On Diffie-Hellman Key Agreement with Short Exponents." Eurocrypt '96, LNCS vol. 1070, pp. 332-343, Springer-Verlag 1996.

Last Updated: September 24th, 2017

Upcoming SANS Training

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

Rocky Mountain Fall 2017	**Denver, COUS**	**Sep 25, 2017 - Sep 30, 2017**	**Live Event**
SANS Baltimore Fall 2017	**Baltimore, MDUS**	**Sep 25, 2017 - Sep 30, 2017**	**Live Event**
Data Breach Summit & Training	**Chicago, ILUS**	**Sep 25, 2017 - Oct 02, 2017**	**Live Event**
SANS Copenhagen 2017	**Copenhagen, DK**	**Sep 25, 2017 - Sep 30, 2017**	**Live Event**
SANS London September 2017	**London, GB**	**Sep 25, 2017 - Sep 30, 2017**	**Live Event**
SANS Oslo Autumn 2017	**Oslo, NO**	**Oct 02, 2017 - Oct 07, 2017**	**Live Event**
SANS DFIR Prague Summit & Training 2017	**Prague, CZ**	**Oct 02, 2017 - Oct 08, 2017**	**Live Event**
SANS Phoenix-Mesa 2017	**Mesa, AZUS**	**Oct 09, 2017 - Oct 14, 2017**	**Live Event**
SANS October Singapore 2017	**Singapore, SG**	**Oct 09, 2017 - Oct 28, 2017**	**Live Event**
Secure DevOps Summit & Training	**Denver, COUS**	**Oct 10, 2017 - Oct 17, 2017**	**Live Event**
SANS Tysons Corner Fall 2017	**McLean, VAUS**	**Oct 14, 2017 - Oct 21, 2017**	**Live Event**
SANS Brussels Autumn 2017	**Brussels, BE**	**Oct 16, 2017 - Oct 21, 2017**	**Live Event**
SANS Tokyo Autumn 2017	**Tokyo, JP**	**Oct 16, 2017 - Oct 28, 2017**	**Live Event**
SANS Berlin 2017	**Berlin, DE**	**Oct 23, 2017 - Oct 28, 2017**	**Live Event**
SANS Seattle 2017	**Seattle, WAUS**	**Oct 30, 2017 - Nov 04, 2017**	**Live Event**
SANS San Diego 2017	**San Diego, CAUS**	**Oct 30, 2017 - Nov 04, 2017**	**Live Event**
SANS Gulf Region 2017	**Dubai, AE**	**Nov 04, 2017 - Nov 16, 2017**	**Live Event**
SANS Miami 2017	**Miami, FLUS**	**Nov 06, 2017 - Nov 11, 2017**	**Live Event**
SANS Milan November 2017	**Milan, IT**	**Nov 06, 2017 - Nov 11, 2017**	**Live Event**
SANS Amsterdam 2017	**Amsterdam, NL**	**Nov 06, 2017 - Nov 11, 2017**	**Live Event**
SANS Paris November 2017	**Paris, FR**	**Nov 13, 2017 - Nov 18, 2017**	**Live Event**
Pen Test Hackfest Summit & Training 2017	**Bethesda, MDUS**	**Nov 13, 2017 - Nov 20, 2017**	**Live Event**
SANS Sydney 2017	**Sydney, AU**	**Nov 13, 2017 - Nov 25, 2017**	**Live Event**
SANS London November 2017	**London, GB**	**Nov 27, 2017 - Dec 02, 2017**	**Live Event**
SANS San Francisco Winter 2017	**San Francisco, CAUS**	**Nov 27, 2017 - Dec 02, 2017**	**Live Event**
SIEM & Tactical Analytics Summit & Training	**Scottsdale, AZUS**	**Nov 28, 2017 - Dec 05, 2017**	**Live Event**
SANS Khobar 2017	**Khobar, SA**	**Dec 02, 2017 - Dec 07, 2017**	**Live Event**
SANS Munich December 2017	**Munich, DE**	**Dec 04, 2017 - Dec 09, 2017**	**Live Event**
European Security Awareness Summit & Training 2017	**London, GB**	**Dec 04, 2017 - Dec 07, 2017**	**Live Event**
SANS Austin Winter 2017	**Austin, TXUS**	**Dec 04, 2017 - Dec 09, 2017**	**Live Event**
SANS Frankfurt 2017	**Frankfurt, DE**	**Dec 11, 2017 - Dec 16, 2017**	**Live Event**
SANS Bangalore 2017	**Bangalore, IN**	**Dec 11, 2017 - Dec 16, 2017**	**Live Event**
SANS SEC504 at Cyber Security Week 2017	**OnlineNL**	**Sep 25, 2017 - Sep 30, 2017**	**Live Event**
SANS OnDemand	**Books & MP3s OnlyUS**	**Anytime**	**Self Paced**