Steve Friedl UNIXWIZ.NET

Steve Friedl's Unixwiz.net Tech Tips

An Illustrated Guide to IPsec

IPsec is a suite of protocols for securing network connections, but the details and many variations quickly become overwhelming. This is particularly the case when trying to interoperate between disparate systems, causing more than one engineer to just mindlessly turn the knobs when attempting to bring up a new connection.

Table of Contents

This Tech Tip means to give bottom-up coverage of the low-level protocols used in an IPv4 context (we provide no coverage of IPv6). This is *not* a deployment guide or best-practices document — we're looking at it strictly at the protocol level on up, rather than from the big picture on down.

NOTE: originally this was to be a pair of papers, with the second covering Key Exchange and the like, but it appears that this was not meant to be. Sorry.

So many flavors...

One of the first things that one notices when trying to set up IPsec is that there are *so many* knobs and settings: even a pair of entirely standards-conforming implementations sports a bewildering number of ways to impede a successful connection. It's just an astonishingly-complex suite of protocols.

One cause of the complexity is that IPsec provides mechanism, not policy: rather than define such-and-such encryption algorithm or a certain authentication function, it provides a framework that allows an implementation to provide nearly anything that both ends agree upon.

In this section, we'll touch on some of the items in the form of a glossary, with a compare-and-contrast to show which terms relate to which other terms. This is not even remotely complete.

- **AH *versus* ESP**

  "Authentication Header" (AH) and "Encapsulating Security Payload" (ESP) are the two main wire-level protocols used by IPsec, and they authenticate (AH) and encrypt+authenticate (ESP) the data flowing over that connection. They are typically used independently, though it's possible (but uncommon) to use them both together.

- **Tunnel mode *versus* Transport mode**

  Transport Mode provides a secure connection between two endpoints as it encapsulates IP's payload, while Tunnel Mode encapsulates the *entire* IP packet to provide a virtual "secure hop" between two gateways. The latter is used to form a traditional VPN, where the tunnel generally creates a secure tunnel across an untrusted Internet.

- **MD5 *versus* SHA-1 *versus* DES *versus* 3DES *versus* AES *versus* blah blah blah**

  Setting up an IPsec connection involves all kinds of crypto choices, but this is simplified substantially by the fact that any given connection can use at most two or (rarely) three at a time.

  Authentication calculates an Integrity Check Value (ICV) over the packet's contents, and it's usually built on top of a cryptographic hash such as MD5 or SHA-1. It incorporates a secret key known to both ends, and this allows the recipient to compute the ICV in the same way. If the recipient gets the same value, the sender has effectively authenticated itself (relying on the property that cryptographic hashes can't practically be reversed). AH always provides authentication, and ESP does so optionally.

  Encryption uses a secret key to encrypt the data before transmission, and this hides the actual contents of the packet from eavesdroppers. There are quite a few choices for algorithms here, with DES, 3DES, Blowfish and AES being common. Others are possible too.

- **IKE *versus* manual keys**

  Since both sides of the conversation need to know the secret values used in hashing or encryption, there is the question of just how this data is exchanged. *Manual keys* require manual entry of the secret values on both ends, presumably conveyed by some out-of-band mechanism, and IKE (Internet Key Exchange) is a sophisticated mechanism for doing this online.

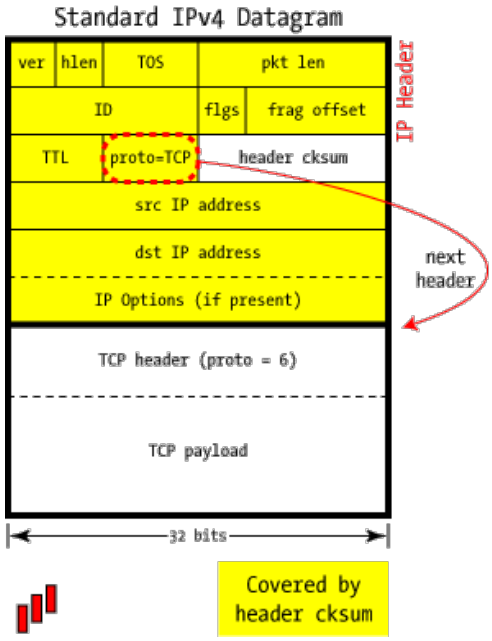
- **Main mode *versus* aggressive mode**

  These modes control an efficiency-*versus*-security tradeoff during initial IKE key exchange. "Main mode" requires six packets back and forth, but affords complete security during the establishment of an IPsec connection, while Aggressive mode uses half the exchanges providing a bit less security because some information is transmitted in cleartext.

We'll certainly face more options as we unwrap IPsec.

The IP Datagram

Since we're looking at IPsec from the bottom up, we must first take a brief detour to revisit the IP Header itself, which carries all of the traffic we'll be considering. Note that we are not trying to provide comprehensive coverage to the IP header — there are other excellent resources for that (the best being *TCP/IP Illustrated, vol 1*).



- **ver**

  This is the version of the protocol, which is now 4=IPv4

- **hlen**

  IP Header length, as a four-bit number of 32-bit words ranging from 0..15. A standard IPv4 header is always 20 bytes long (5 words), and IP Options — if any — are indicated by a larger **`hlen`** field up to at most 60 bytes. This header length never includes the size of payload or other headers that follow.

- **TOS**

  Type of Service

  This field is a bitmask that gives some clues as to the type of service this datagram should receive: optimize for bandwidth? Latency? Low cost? Reliability?

- **pkt len**

  Overall packet length in bytes, up to 65535. This count includes the bytes of the header, so this suggests that the maximum size of any payload is at least 20 bytes less. The vast majority of IP datagrams are much, much smaller.

- **ID**

  The ID field is used to associate related packets that have been fragmented (large packets broken up into smaller ones).

- **flgs**

  These are small flags that mainly control fragmentation: one marks the packet as ineligible for fragmentation, and the other says that more fragments follow.

- **frag offset**

  When a packet is fragmented, this shows where in the overall "virtual" packet this fragment belongs.

- **TTL**

  This is the Time to Live, and is decremented by each router that passes this packet. When the value reaches zero, it suggests some kind of routing loop, so it's discarded to prevent it from running around the Internet forever.

- **proto**

  This represents the protocol carried within this packet, and it's going to be central to most of our discussions. Though the datagram itself is IP, it *always* encapsulates a subsidiary protocol (TCP, UDP, ICMP, etc. — see the chart below) within. It can be thought of as giving the type of the header that follows.

- **header cksum**

  This holds a checksum of the entire IP header, and it's designed to detect errors in transit. This is not a *cryptographic* checksum, and it doesn't cover any part of the datagram that follow the IP header.

- **src IP address**

  The 32-bit source IP address, which the recipient uses to reply to this datagram. Generally speaking, it's possible to spoof these addresses (i.e., lie about where the datagram is coming from).

- **dst IP address**

  The 32-bit destination IP address, which is where the packet is intended to arrive.

- **IP Options**

  These are an optional part of the IP header that contains application-specific information, though they are not commonly used for routine traffic. The presence of IP options is indicated by a **`hlen`** greater than 5, and they (if present) are included in the header checksum.

- **Payload**

  Each protocol type implies its own format for what follows the IP header, and we've used TCP here just to show an example.

These **proto** codes are defined by IANA — the Internet Assigned Numbers Authority — and there are many more than would ever be used by any single installation, but most will ring a bell with a network-savvy technician. These representative types are taken from the IANA website listing protocols:

Some IP protocol codes

Protocol code	Protocol Description
1	ICMP — Internet Control Message Protocol
2	IGMP — Internet Group Management Protocol
4	IP within IP (a kind of encapsulation)
6	TCP — Transmission Control Protocol
17	UDP — User Datagram Protocol
41	IPv6 — next-generation TCP/IP
47	GRE — Generic Router Encapsulation (used by PPTP)
50	IPsec: ESP — Encapsulating Security Payload
51	IPsec: AH — Authentication Header

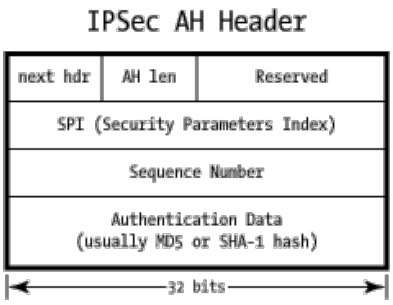
We'll be studying the last two in detail.

AH: Authentication Only

AH is used to authenticate — but not encrypt — IP traffic, and this serves the treble purpose of ensuring that we're really talking to who we think we are, detecting alteration of data while in transit, and (optionally) to guard against replay by attackers who capture data from the wire and attempt to re-inject that data back onto the wire at a later date.

Authentication is performed by computing a cryptographic hash-based message authentication code over nearly all the fields of the IP packet (excluding those which might be modified in transit, such as TTL or the header checksum), and stores this in a newly-added AH header and sent to the other end.

This AH header contains just five interesting fields, and it's injected between the original IP header and the payload. We'll touch on each of the fields here, though their utility may not be fully apparent until we see how they're used in the larger picture.



- **next hdr**

  This identifies the protocol type of the following payload, and it's the original packet type being encapsulated: this is how the IPsec header(s) are linked together.

- **AH len**

  This defines the length, in 32-bit words, of the whole AH header, minus two words (this "minus two words" proviso springs from the format of IPv6's RFC 1883 Extension Headers, of which AH is one).

- **Reserved**

  This field is reserved for future use and must be zero.

- **Security Parameters Index**

  This is an opaque 32-bit identifier that helps the recipient select which of possibly many ongoing conversations this packet applies. Each AH-protected connection implies a hash algorithm (MD5, SHA-1, etc.), some kind of secret data, and a host of other parameters. The SPI can be thought of as an index into a table of these settings, allowing for easy association of packet with parameter.

- **Sequence Number**

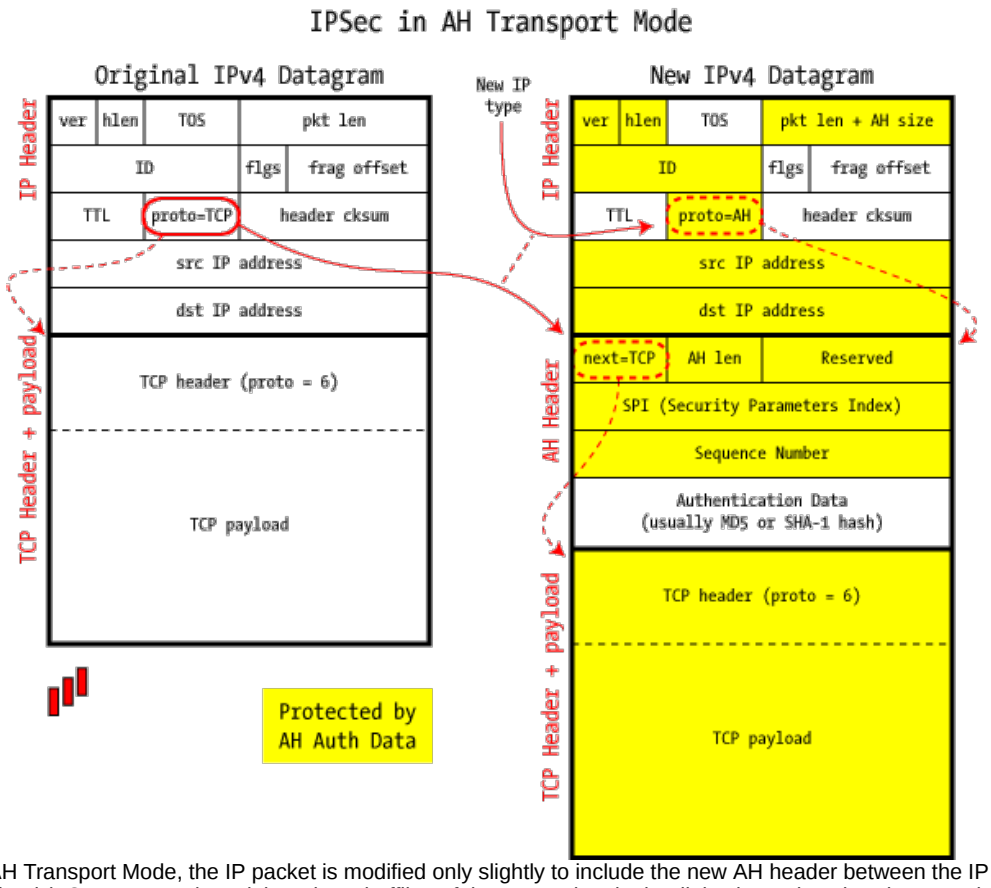
  This is a monotonically increasing identifier that's used to assist in antireplay protection. This value is included in the authentication data, so modifications (intentional or otherwise) are detected.

- **Authentication Data**

  This is the Integrity Check Value calculated over the entire packet — including most of the headers — The recipient recomputes the same hash; Mismatched values mark the packet as either damaged in transit, or not having the proper secret key. These are discarded.

Transport Mode

The easiest mode to understand is *Transport Mode*, which is used to protect an end-to-end conversation between two hosts. This protection is either authentication or encryption (or both), but it is not a tunneling protocol. It has nothing to do with a traditional VPN: it's simply a secured IP connection.



In AH Transport Mode, the IP packet is modified only slightly to include the new AH header between the IP header and the protocol payload (TCP, UDP, etc.), and there is a shuffling of the protocol code that links the various headers together.

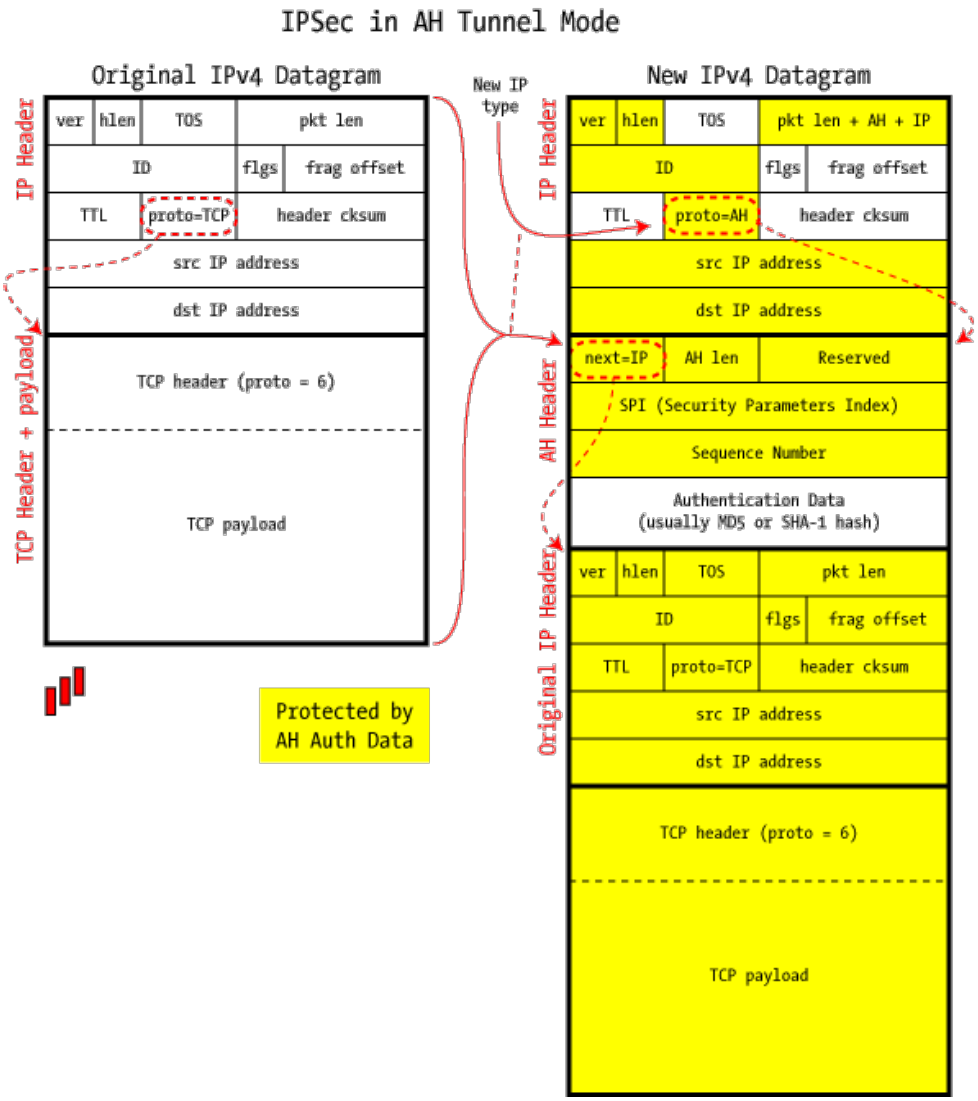
This protocol shuffling is required to allow the original IP packet to be reconstituted at the other end: after the IPsec headers have been validated upon receipt, they're stripped off, and the original protocol type (TCP, UDP, etc.) is stored back in the IP header. We'll see this chain of **`next header`** fields again and again as we examine IPsec.

When the packet arrives at its destination and passes the authentication check, the AH header is removed and the Proto=AH field in the IP header is replaced with the saved "Next Protocol". This puts the IP datagram back to its original state, and it can be delivered to the waiting process.

Tunnel Mode

Tunnel Mode forms the more familiar VPN functionality, where entire IP packets are encapsulated inside another and delivered to the destination.

Like Transport mode, the packet is sealed with an Integrity Check Value to authenticate the sender and to prevent modification in transit. But unlike Transport mode, it encapsulates the *full IP header* as well as the payload, and this allows the source and destination addresses to be different from those of the encompassing packet: This allows formation of a tunnel.



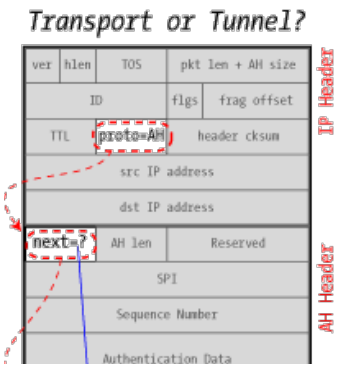
When a Tunnel-mode packet arrives at its destination, it goes through the same authentication check as any AH-type packet, and those passing the check have their entire IP and AH headers stripped off. This effectively reconstitutes the original IP datagram, which is then injected into the usual routing process.

Most implementations treat the Tunnel-mode endpoint as a virtual network interface — just like an Ethernet interface or localhost — and the traffic entering or leaving it is subject to all the ordinary routing decisions.

The reconstituted packet could be delivered to the local machine or routed elsewhere (according to the destination IP address found in the encapsulated packet), though in any case is no longer subject to the protections of IPsec. At this point, it's just a regular IP datagram.

Though Transport mode is used strictly to secure an end-to-end connection between two computers, Tunnel mode is more typically used between gateways (routers, firewalls, or standalone VPN devices) to provide a Virtual Private Network (VPN).

Transport or Tunnel?



Curiously, there is no explicit "Mode" field in IPsec: what distinguishes Transport mode from Tunnel mode is the next header field in the AH header.

When the next-header value is *IP*, it means that this packet encapsulates an entire IP datagram (including the independent source and destination IP addresses that allow separate routing after de-encapsulation). This is Tunnel mode.

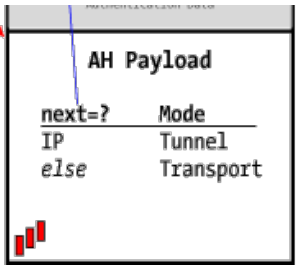
Any other value (TCP, UDP, ICMP, etc.) means that it's Transport mode and is securing an endpoint-to-endpoint connection.

The top-level of the IP datagram is structured the same way regardless of mode, and intermediate routers treat all flavors IPsec/AH traffic identically without deeper inspection.

We'll note that a host — as opposed to a gateway — is required to support both Transport and Tunnel modes, but when creating a host-to-host connection, it seems a little superfluous to use Tunnel mode.

Furthermore, a gateway (router, firewall, etc.) is only required to support Tunnel mode, though

supporting Transport mode is useful only when creating an endpoint to the gateway itself, as in the case of network management functions.

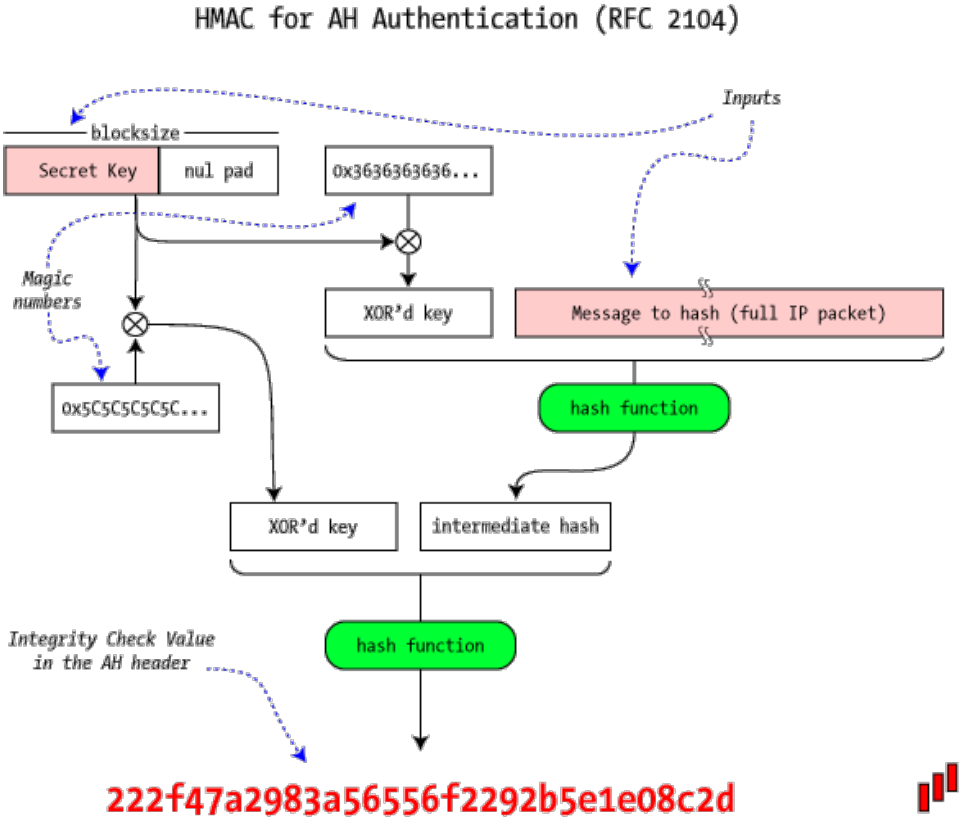


Authentication Algorithms

AH carries an Integrity Check Value in the Authentication Data portion of the header, and it's typically (but not always) built on top of standard cryptographic hash algorithms such as MD5 or SHA-1.

Rather than use a straight checksum, which would provide no real security against intentional tampering, it uses a Hashed Message Authentication Code (HMAC) which incorporates a secret value while creating the ICV. Though an attacker can easily recompute a hash, without the secret value he won't be able to recreate the proper ICV.

HMAC is described by RFC 2104, and this illustration shows how the message data and secret contribute to the final Integrity Check Value:



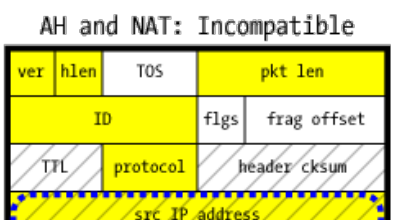
We'll note that IPsec/AH doesn't define what the authentication function must be, it instead provides a framework which allows any reasonable implementation *agreed to by both ends* to use. It's possible to use other authentication functions, such as a digital signature or an encryption function as long as both sides provide for it.

AH and NAT — Not Gonna Happen

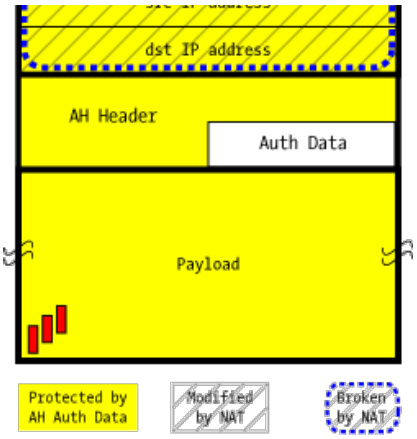
Though AH provides very strong protection of a packet's contents because it covers *everything* that can be possibly considered immutable, this protection comes at a cost: AH is incompatible with NAT (Network Address Translation).

NAT is used to map a range of private addresses (say, 192.168.1.X) to and from a (usually) smaller set of public address, thereby reducing the demand for routable, public IP space. In this process, the IP header is actually modified on the fly by the NAT device to change the source and/or destination IP address.

When the appropriate source or header IP address is changed, it forces a recalculation of the header checksum. This has to be done anyway, because the NAT device typically serves as one "hop" in the path from source to destination, and this requires the decrement of the TTL (Time To Live) field.



Because the TTL and header checksum fields are *always* modified in flight, AH knows to excludes them from coverage, but this does not apply to the IP addresses. These are *included* in the Integrity Check Value, and any modification will cause the check to fail when verified by the recipient. Because the ICV incorporates a secret key which is unknown by intermediate parties, the NAT router is not able to recompute the ICV.



This same difficulty also applies to PAT (Port Address Translation), which maps multiple private IP addresses into a single external IP address. Not only are the IP addresses modified on the fly, but the UDP and TCP port numbers (and sometimes even to payload). This requires much more intelligence on the part of the NAT device, and more extensive modifications to the whole IP datagram.

For this reason, AH — whether in Tunnel or Transport mode — is entirely incompatible with NAT, and it may only be employed when the source and destination networks are reachable without translation.

We'll note that this particular difficulty doesn't apply to ESP, as its authentication and encryption do *not* incorporate the IP header being modified by NAT. Nevertheless, NAT does impose some challenges even on ESP.

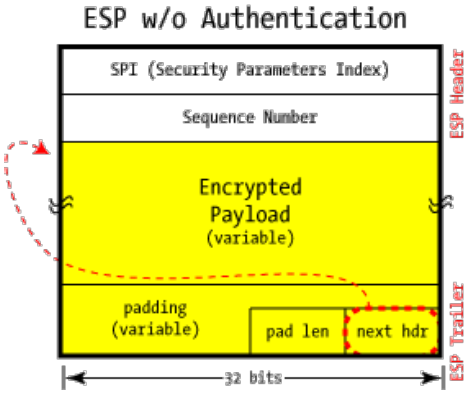
NAT translates IP addresses on the fly — but it has to keep track of which connections are flowing through it so that replies can be properly associated with sources. When using TCP or UDP, this is commonly done with port numbers (whether rewritten on the fly or not), but IPsec provides no hook to allow this.

At first one might suspect the SPI, which appears to be a useful identifier, but because the SPI is different in both directions, the NAT device has no way to associate the returning packet with the outgoing connection.

Addressing this requires special facilities for NAT traversal, something not covered in this paper.

ESP — Encapsulating Security Payload

Adding encryption makes ESP a bit more complicated because the encapsulation *surrounds* the payload rather than *precedes* it as with AH: ESP includes header and trailer fields to support the encryption and optional authentication. It also provides Tunnel and Transport modes which are used in by-now familiar ways.



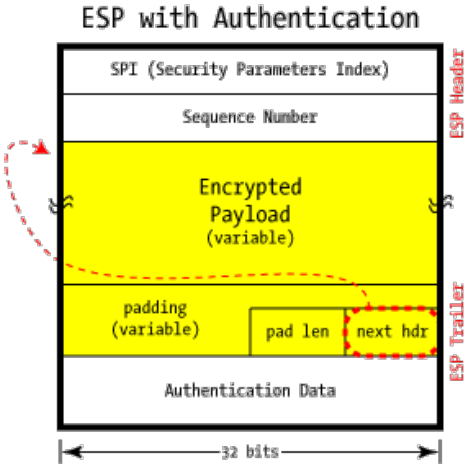
The IPsec RFCs don't insist upon any particular encryption algorithms, but we find DES, triple-DES, AES, and Blowfish in common use to shield the payload from prying eyes. The algorithm used for a particular connection is specified by the Security Association (covered in a later section), and this SA includes not only the algorithm, but the key used.

Unlike AH, which provides a small header *before* the payload, ESP *surrounds* the payload it's protecting. The Security Parameters Index and Sequence Number serve the same purpose as in AH, but we find padding, the next header, and the optional Authentication Data at the end, in the ESP Trailer.

It's possible to use ESP without any actual encryption (to use a NULL algorithm), which nonetheless structures the packet the same way. This provides no confidentiality, and it only makes sense if combined with ESP authentication. It's pointless to use ESP without either encryption or authentication (unless one is simply doing protocol testing).

Padding is provided to allow block-oriented encryption algorithms room for multiples of their blocksize, and the length of that padding is provided in the **`pad len`** field. The **`next hdr`** field gives the type (IP, TCP, UDP, etc.) of the payload in the usual way, though it can be thought of as pointing "backwards" into the packet rather than forward as we've seen in AH.

In addition to encryption, ESP can also optionally provide authentication, with the same HMAC as found in AH. Unlike AH, however, this authentication is *only for the ESP header and encrypted payload*: it does not cover the full IP packet. Surprisingly, this does not substantially weaken the security of the authentication, but it does provide some important benefits.



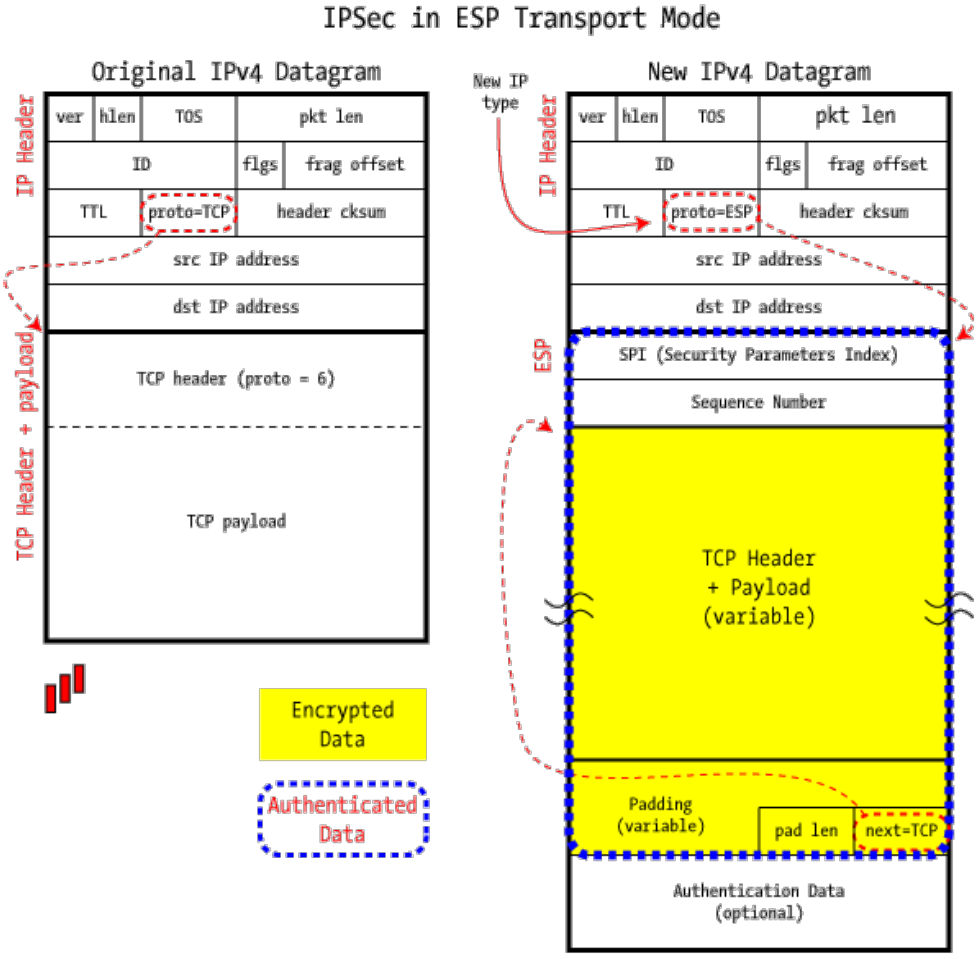
When an outsider examines an IP packet containing ESP data, it's essentially impossible to make any real guesses about what's inside save for the usual data found in the IP header (particularly the source and destination IP addresses). The attacker will certainly know that it's ESP data — that's also in the header — but the type of the payload is *encrypted with the payload*.

Even the presence or absence of Authentication Data can't be determined by looking at the packet itself (this determination is made by using the Security Parameters Index to reference the preshared set of parameters and algorithms for this connection).

However, it should be noted that sometimes *the envelope* provides hints that the *payload* does not. With more people sending VoIP inside ESP over the internet, the QoS taggings are in the outside header and is fairly obvious what traffic is VoIP signaling (IP precedence 3) and what is RTP traffic (IP precedence 5). It's not a sure thing, but it might be enough of a clue to matter in some circumstances.

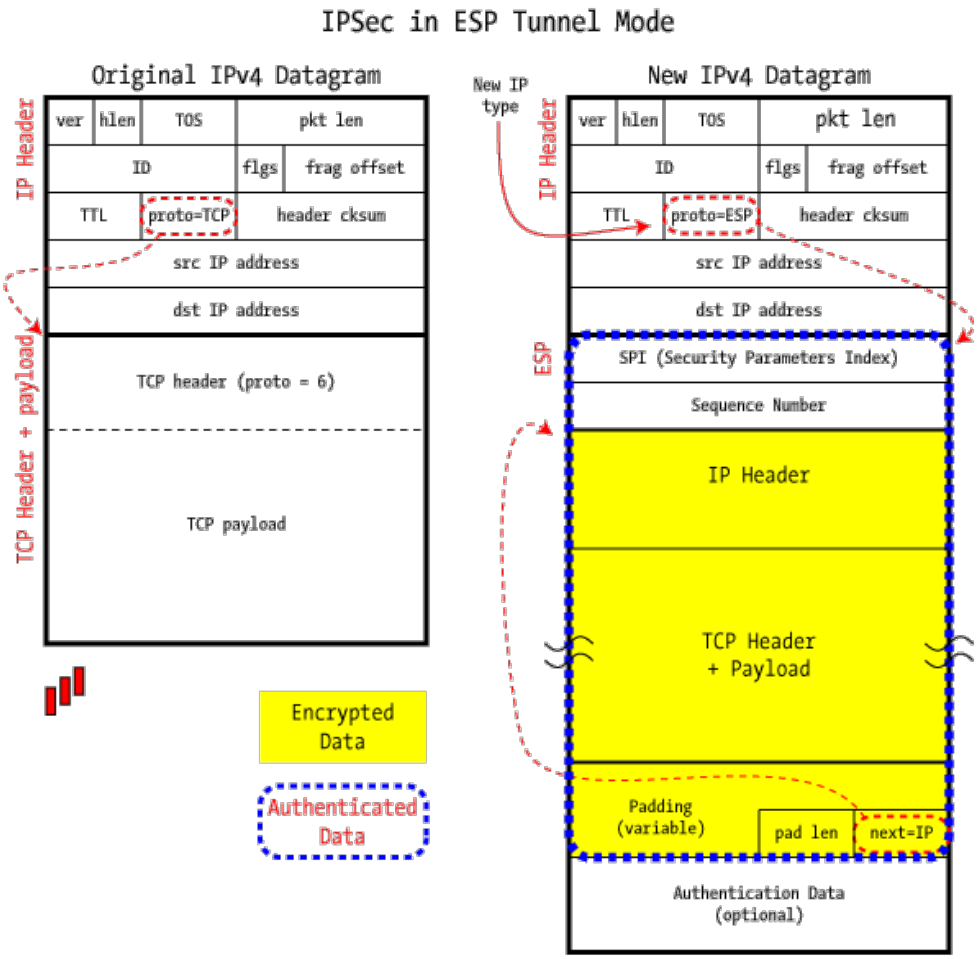
ESP in Transport Mode

As with AH, Transport Mode encapsulates just the datagram's payload and is designed strictly for host-to-host communications. The original IP header is left in place (except for the shuffled Protocol field), and it means that — among other things — the source and destination IP addresses are unchanged.



ESP in Tunnel Mode

Our final look of standalone ESP is in Tunnel mode, which encapsulates an entire IP datagram inside the encrypted shell:

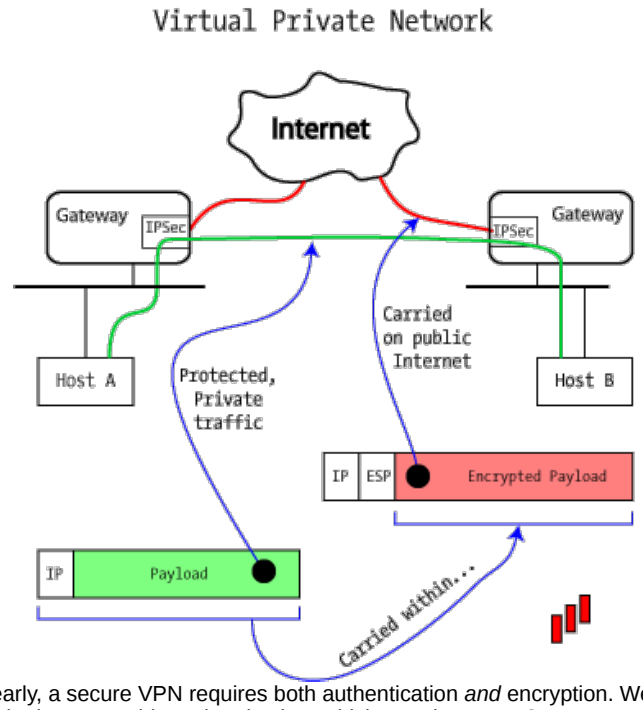


Providing an encrypted Tunnel Mode connection is getting very close to the traditional VPN that springs to mind when most of us think about IPsec, but we have to add authentication of one type or another to complete the picture: this is covered in the following section.

Unlike AH, where an onlooker can easily tell whether traffic is in Tunnel or Transport mode, this information is unavailable here: the fact that this is Tunnel mode (via **`next=IP`**) is part of the encrypted payload, and is simply not visible to one unable to decrypt the packet.

Putting it all together: Building a real VPN

With coverage of the Authenticating Header and Encapsulating Security Payload complete, we're ready to enable both encryption and authentication to build a real VPN. The whole purpose of a Virtual Private Network is to join two trusted networks across an untrusted intermediate network, as is by stringing a very long Ethernet cable between the two. This is commonly used to connect branch offices with company headquarters, allowing all users to share sensitive resources without fear of interception.



Clearly, a secure VPN requires both authentication *and* encryption. We know that ESP is the only way to provide encryption, but ESP and AH both can provide authentication: which one do we use?

The obvious solution of wrapping ESP inside of AH is technically possible, but in practice is not commonly used because of AH's limitations with respect to Network Address Translation. By using AH+ESP, this tunnel could never successfully traverse a NAT'ed device.

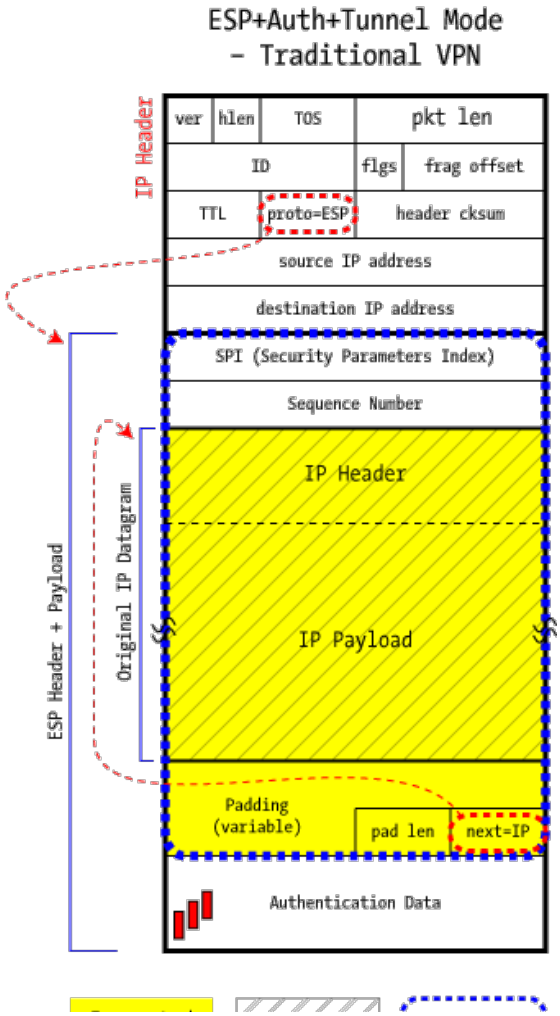
Instead, ESP+Auth is used in Tunnel mode to fully encapsulate the traffic on its way across an untrusted network, protected by both encryption and authentication in the same thing.

Traffic protected in this manner yields nearly no useful information to an interloper save for the fact that the two sites *are* connected by a VPN. This information might help an attacker understand trust relationships, but *nothing* about the actual traffic itself is revealed. Even the type of encapsulated protocol — TCP, UDP, or ICMP — is hidden from outsiders.

What's particularly nice about this mode of operation is that the end-user hosts generally know nothing about the VPN or other security measures in place. Since a VPN implemented by a gateway device treats the VPN as yet another interface, traffic destined for the other end is routed normally.

This packet-in-a-packet can actually be nested yet more levels: Host A and Host B can establish their own authenticated connection (via AH), and have this routed over the VPN. This would put an AH inner packet inside an enclosing ESP+Auth packet.

Update - it's important to use authentication even if encryption is used, because encrypt-only implementations are subject to effective attack as described in the paper *Cryptography in Theory and Practice: The Case of Encryption in IPsec*; see the Resources section for more information.



Touching on Other Matters

IPsec is a very complex suite of protocols, and this Tech Tip cannot possibly give proper justice to more than a small part of it. In this section we'll mention a few areas that beg for more coverage.

Security Associations and the SPI

It seems self-evident that if two endpoints or gateways are going to establish a secure connection, some kind of shared secret is required to seed the authentication function and/or key the encryption algorithm. The matter of just how these are secrets are established is a substantial topic to be addressed elsewhere, and for the purposes of this discussion we shall just assume that the keys have magically landed where they belong.

When an IPsec datagram — either AH or ESP — arrives at an interface, just how does the interface know which set of parameters (key, algorithm, and policies) to use? Any host could have many ongoing conversations, each with a different set of keys and algorithms, and *something* must be able to direct this processing.

This is specified by the *Security Association* (SA), a collection of connection-specific parameters, and each partner can have one or more Security Associations. When a datagram arrives, three pieces of data are used to locate the correct SA inside the Security Associations Database (SADB):

- Partner IP address
- IPsec Protocol (ESP or AH)
- Security Parameters Index

In many ways this triple can be likened to an IP socket, which is uniquely denoted by the remote IP address, protocol, and port number.

> ***Unsure!***
> It's been pointed out that the SADB only uses the protocol type and SPI to select an entry, not the partner IP address; we simply don't know.
>
> This might depend on whether the association is configured with main mode or aggressive mode, but we welcome clarifications.

Security Associations are *one way*, so a two-way connection (the typical case) requires at least two. Furthermore, each protocol (ESP/AH) has its own SA in each direction, so a full AH+ESP VPN requires four Security Associations. These are all kept in the Security Associations Database.

A tremendous amount of information is kept in the SADB, and we can only touch on a few of them:

- AH: authentication algorithm
- AH: authentication secret
- ESP: encryption algorithm
- ESP: encryption secret key
- ESP: authentication enabled yes/no
- *Many* key-exchange parameters
- Routing restrictions
- IP filtering policy

Some implementations maintain the SPD (Security Policy Database) with command-line tools, others with a GUI, while others provide a web-based interface over the network. The amount of detail maintained by any particular implementation depends on the facilities offered, as well as whether it's operating in Host or Gateway mode (or both).

Key Management

Finally, we briefly visit the very complex matter of key management. This area includes several protocols and many options, and the bulk of this will be covered in a future paper. This section is necessarily *highly* incomplete.

IPsec would be nearly useless without the cryptographic facilities of authentication and encryption, and these require the use of secret keys known to the participants but not to anyone else.

The most obvious and straightforward way to establish these secrets is via manual configuration: one party generates a set of secrets, and conveys them to all the partners. All parties install these secrets in their appropriate Security Associations in the SPD.

But this process does not scale well, nor is it always terribly secure: the mere act of conveying the secrets to another site's SPD may well expose them in transit. In a larger installation with many devices using the same preshared key, compromise of that key makes for a very disruptive re-deployment of new keys.

IKE — Internet Key Exchange — exists to allow two endpoints to properly set up their Security Associations, including the secrets to be used. IKE uses the ISAKMP (Internet Security Association Key Management Protocol) as a framework to support establishment of a security association compatible with both ends.

Multiple key-exchange protocols themselves are supported, with Oakley being the most widely used. We'll note that IPsec key exchange typically takes place over port 500/udp.

Resources

The Internet has a great many resources surrounding IPsec, some better than others. The starting point, of course, is always with the RFCs (Requests for Comment) that form the Internet standards defining the protocols. These are the main reference works upon which all other documentation — including this one — is based.

Update: In December 2005, a whole new set of RFCs was issued by IETF, and the 43xx series largely obsoleted the 24xx series. We included references to all the RFCs (old and new) below, though this document has not really been updated for the new ones.

- **RFC 2401 — Security Architecture for IPsec —** obsolete
- **RFC 4301 — Security Architecture for IPsec —** new Dec 2005

  This is the overview of the entire IPsec protocol suite from the point of view of the RFCs. This, and the Documentation Roadmap (RFC 2411) are good places to start.

- **RFC 2402 — AH: Authentication Header —** obsolete
- **RFC 4302 — AH: Authentication Header —** new Dec 2005

  This defines the format of the IPsec Authentication Header, in both Tunnel and Transport modes.

- **RFC 2403 — Use of HMAC-MD5-96 within ESP and AH**
- **RFC 2404 — Use of HMAC-SHA-1-96 within ESP and AH**

  These two RFCs define authentication algorithms used in AH and ESP: MD5 and SHA-1 are both cryptographic hashes, and they are part of a Hashed Message Authentication Code. AH *always* performs authentication, while ESP does so optionally.

- **RFC 2104 — HMAC: Keyed-Hashing for Message Authentication**

  This RFC defines the authentication algorithm that uses a cryptographic hash along with a secret to verify the integrity and authenticity of a message. It's not written to be part of IPsec, but it's referenced in RFC 2403 and RFC 2404.

- **RFC 2405 — The ESP DES-CBC Cipher Algorithm With Explicit IV**

  This defines the use of DES (the Data Encryption Standard) as a confidentiality algorithm in the context of ESP.

- **RFC 2406 — ESP: Encapsulating Security Payload** obsolete
- **RFC 4303 — ESP: Encapsulating Security Payload** new Dec 2005

  ESP is the encrypting companion to AH, and it affords confidentiality to the contents of its payload. ESP by itself does not define any particular encryption algorithms but provides a framework for them.

- **RFC 2407 — The Internet IP Security Domain of Interpretation for ISAKMP**

  This RFC describes the use of ISAKMP — Internet Security Association and Key Management Protocol — in the context of IPsec. It's a framework for key exchange at the start of a conversation, and its use obviates the poor practice of using manual keys.

- **RFC 2408 — Internet Security Association and Key Management Protocol (ISAKMP)**

  Hand in hand with RFC 2407, this RFC dives into much more detail on the ISAKMP protocol used to support key exchange (though it doesn't define the key exchange protocols themselves).

- **RFC 2409 — The Internet Key Exchange (IKE) Protocol** obsolete
- **RFC 4306 — The Internet Key Exchange (IKE) Protocol** new Dec 2005

  Though ISAKMP provides a framework for key-exchange, it doesn't define the protocols themselves: this RFC does that. IKE includes initial authentication, as well as Oakley key exchange.

- **RFC 2410 — The NULL Encryption Algorithm and Its Use With IPsec**

  IPsec's ESP protocol performs encryption of payload using one of several available algorithms, but a NULL encryption algorithm is typically made available for testing. Of course, this provides no confidentiality for the "protected" data, but it may be useful for developers or those attempting to understand IPsec by sniffing the wire. This RFC is written humorously and could have been (but was not) written on April 1.

- **RFC 2411 — IP Security Document Roadmap**

  This RFC provides an layout of the various IPsec-related RFCs, as well as provides a framework for new RFCs of particular types ("authentication algorithms", "encryption algorithms"). It's a good starting point.

- **RFC 2412 — The OAKLEY Key Determination Protocol**

  OAKLEY forms part of IKE (Internet Key Exchange), and it provides a service where two authenticated parties can agree on the secrets required for IPsec communications.

- **An Illustrated Guide to Cryptographic Hashes - Unixwiz.net Tech Tip**

  An introductory paper on the use of cryptographic hashes such as MD5 or SHA-1, which are used in AH's HMAC for authentication.

- **IPsec Technical Reference by Microsoft**

  This provides information on Microsoft's implementation of IPsec in the Windows Server 2003 product, including a great deal about the larger infrastructure required to support IPsec in the enterprise.

- **TCP/IP Illustrated, Volume 1, by W. Richard Stevens.**

  This is the classic textbook on the TCP/IP protocol, covering down to the packet header in exquisite detail: This is an extraordinary resource.

- **A Cryptographic Evaluation of IPsec, by Bruce Schneier and Niels Ferguson.**

  An interesting paper on the security of IPsec, whose main point is that IPsec is far too complex to ever really be secure (something which has crossed our minds as well). Among their proposals are to eliminate both Transport Mode and AH: ESP in Tunnel mode can provide all this same functionality.

- **RFC 3884 — Use of IPsec Transport Mode for Dynamic Routing**

  In contrast to the Schneier paper, it's also been suggested that *Transport Mode* is the only one that's strictly required to accomplish everything, and RFC 3884 shows a way of providing tunnel mode. It's been suggested to us that this makes some implementation issues much easier, though we've not really investigated any of it.

- **Cryptography in Theory and Practice: The Case of Encryption in IPsec ; Paterson & Yau**

  This very interesting paper discusses some of the dangers of encrypted but not authenticated IPsec connections, with effective attacks on real systems (including the Linux kernel's implementation of IPsec). It's a very clever paper.

- **Protocolo IPsec — Alexandru Ionut Grama**

  This paper was translated into Spanish!

N.B. — We are *not* IPsec experts, and though we've spent a great deal of time researching these matters, we may have some details wrong. Feedback and corrections are very much welcome. We're particularly grateful for the extensive technical feedback provided by IPsec architects at Sun and Microsoft.

These original figures were produced by the author using Adobe Illustrator.

First published: 2005-08-24

Home | Stephen J. Friedl | Software Consultant | Orange County, CA USA | steve@unixwiz.net