TIMESTEP

Understanding the IPSec protocol suite

SECURE VIRTUAL PRIVATE NETWORK SOLUTIONS

Written and designed at:
TimeStep Corporation
362 Terry Fox Drive
Kanata, Ontario K2K 2P5
Phone: (613) 599-3610
Fax: (613) 599-3617

Rev. No. IPSec2.0 — December 1998

Executive summary

IP benefits

The Internet Protocol (IP) underlies the vast majority of large corporate and academic networks as well as the Internet. It is flexible, powerful, and has served people's networking needs well for decades. IP's strength lies in its easily and flexibly routed packets — the small, manageable chunks into which it breaks the data stream for transmission through the network.

IP vulnerabilities

However, IP's strength is also its weakness. The way IP routes packets makes large IP networks vulnerable to a range of security risks:

- spoofing, in which one machine on the network masquerades as another
- sniffing, in which an eavesdropper listens in on a transmission between two other parties
- session highjacking, in which a sophisticated attacker employing both those techniques takes over an established communications session and masquerades as one of the communicating parties

The IPSec protocol suite

Because these vulnerabilities limit and complicate the use of large IP networks (including the Internet) for sensitive communications, an international group organized under the Internet Engineering Task Force (IETF) has developed the IP Security (IPSec) protocol suite — a set of IP extensions that provide security services at the network level. IPSec technology is based on modern cryptographic technologies, making possible very strong data authentication and privacy guarantees.

The secure VPN

The IPSec group's work is conceptually unique in that it seeks to secure the network itself, rather than the applications that use it. In the past, other groups have developed specific application-level methods for protecting communications, including PGP/Web-of-Trust schemes for securing email. While these methods are effective for solving specific security problems, such solutions are by their nature limited to their specific niches. Because it secures the network itself, the IPSec protocol suite guarantees security for *any* application using the network.

IPSec makes possible the realization of the secure virtual private network (or secure VPN) — a private and secure network carved out of a public and/or insecure network. Secure VPNs are as safe as isolated office LANs or WANs run entirely over private lines and far more cost-effective.

The IPSec suite architecture

The IPSec protocol suite provides three overall pieces:

- an authentication header (AH) for IP that lets communicating parties verify that data was not modified in transit and that it genuinely came from its apparent source
- an encapsulation security payload (ESP) format for IP that encrypts data to secure it against eavesdropping during transit
- a protocol negotiation and key exchange protocol (IKE) that allows communicating parties to negotiate methods of secure communication

Negotiation and key exchange — the IPSec difference

Encryption keys are parameters, much like passwords, needed to perform encryption and verification algorithms. To use any encryption in a network environment, communicating parties must first exchange keys.

IPSec's strength at addressing this thorniest of problems for network based encryption sets it apart from all other encryption-based security schemes.

The IKE suite addresses the problem in a unique, powerful, and well-conceived way. Most importantly, IPSec's protocol negotiation and key exchange schemes are truly scalable, so you can build any number and any size of secure VPNs within the larger network.

Unlike anything else, IPSec makes it practical to create secure VPNs over tens of nodes, and over tens of thousands of nodes — easily, conveniently, and transparently. So now you can safely extend your private network through public networks, including the Internet.

Contents

Introduction

The Internet Protocol (IP) is unsurpassed as the base of large networks. It underlies networks from medium-sized corporate LANs up to the Internet itself. More so than any other protocol at any level of network architecture, IP is the universal medium of communications — the *lingua franca* — of computers all around the world.

IP deserves its popularity. It is flexible, powerful, and well-suited to negotiate communications between platforms with divergent capabilities.

The Internet: a dangerous place for data

However, IP-based networks have a weakness, related to the structure of IP itself. IP's original architects had no reason to provide security features at the IP level, and IP's flexibility allows for some creative uses of the protocol that defeat traffic auditing, access control, and many other security measures. IP-based network data is, therefore, wide open to tampering and eavesdropping.

Mass-media reports of the Internet's fundamental unsuitability for sending sensitive communications, while sometimes sensationalized, have their basis in fact. The public and business know about the Internet's weakness, and they should.

Technologies secure Internet communications

Technologies now exist to secure communications over the Internet, but mostly for specific software applications. These technologies generally use powerful new encryption techniques to get the job done. PGP/Web-of-Trust technology encrypts email. Browser-based authentication and encryption between the browser and the web server (SSL) protects commercial web traffic.

While these technologies have their respective strengths and niches, they are limited to specific uses. This doesn't address the challenges faced by the large enterprise and the average Internet Service Provider (ISP) that may never know precisely what applications may be running tomorrow over the networks they're building today.

The importance of the network layer

There is a solution to the security problem, however, and it isn't restricted to a single application. The key is in understanding the nature of the network layer in IP networks.

To isolate the various problems in building networks and making them work, we picture them in layers. Each layer solves specific problems unique to that layer (see *Network layers* in the glossary). Broadly speaking, you can imagine an IP network as having three layers — the physical layer, the IP network layer, and the application layer. Each layer provides services to the level above.

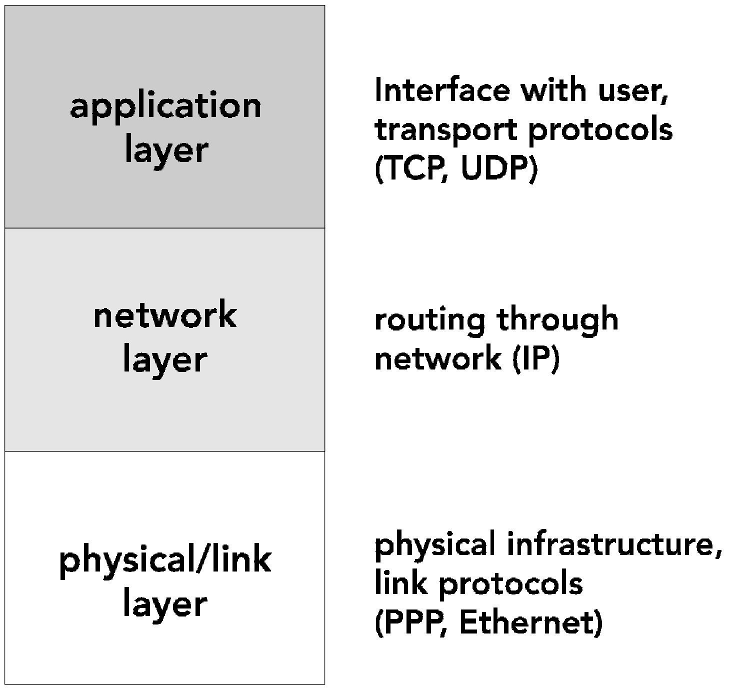

application layer
Interface with user, transport protocols (TCP, UDP)
network layer
routing through network (IP)
physical/link layer
physical infrastructure, link protocols (PPP, Ethernet)

Figure 1: Network layers (simplified)

The physical/link layer (the lowest layer) consists of the electrical cables, network cards, and/or radio links on which information travels, as well as the simple data carrying protocols that provide an interface for the higher level protocols. In an IP network, different parts of the network use different types of physical media — Ethernet[1] in some places, point-to-point lines in others.

Above the physical layer, the network layer (the IP layer in IP networks) deals with getting information from network node to network node across the whole network. It uses the lower level protocols to move the data, and its own routing logic to work out the most sensible subnets through which to send the data.

Above the network layer are a few higher level protocols that set up the links between nodes for different types of communications, and the application layer in which the applications run. Applications use the services of the network layer to figure out how to move data from network node to network node and the network layer in turn uses the physical layer to get the data from one computer's network card to the next.

The significant feature of IP networks is that the network layer in IP networks is entirely homogeneous, and it's the only layer that is. This means that any communication passing through an IP network, including the Internet, *has* to use the IP protocol. In other network layers, different protocols hold sway in different parts for different reasons (depending on the network architecture and the type of communication). But sooner or later everything has to go through the network layer, and there is only one protocol in use in that layer — and that's IP.

So if you secure the IP layer, you secure the network.

Securing the IP layer with IPSec

An international working group organized under the IETF has developed a method of doing exactly that. They call it the IP Security (IPSec) protocol suite. The IPSec protocol suite is based in powerful new encryption technologies, and adds security services to the IP layer in a fashion that is compatible both with the existing (IPv4) IP standard, and which is mandatory in the upcoming one (IPv6[2]).

This means that if you use the IPSec suite where you would normally use IP, you secure all communications in your network for all applications and for all users more transparently than you would using any other approach.

With IPSec, you can build something called a secure virtual private network (VPN) — a secure, private network that is as safe or safer than an isolated office LAN, but built on an unsecured, public network. With IPSec, you can create a secure VPN on the fly, on demand, and with anyone else using the standard.

But because IPSec works with the existing and future IP standards, you can still use regular IP networks in between to carry data. You have to be IPSec-compliant, and the recipient of the information has to be IPSec-compliant, but the rest of the network in between can work just as it works now.

The fundamental strength of the IPSec group's approach is that their security works at a low network level. So just as IP is transparent to the average user, so are IPSec-based security services — unseen servants functioning in the background, ensuring that your communications are secure.

Just as IP's power and flexibility make it universal, IPSec promises to become the new international standard, answering to a diverse range of security needs, allowing vastly different networks around the world to interconnect and to communicate securely.

And just as IP provides services in networks of any size — from LANs with hundreds of nodes to global networks with millions — IPSec promises painless scalability. IPSec promises quiet and reliable service, however complex your security needs.

The following white paper discusses how IPSec does all this.

Security threats in the network environment

To know you have security in the network environment, you want to be confident of three things:

- that the person with whom you're communicating really is that person
- that no one can eavesdrop on your communication
- that the communication you've received has not been altered in any way during transmission

These three security needs, in industry terms, are

- authentication
- confidentiality
- integrity

The architecture of the modern, large, IP-based network, unfortunately, makes all of this difficult to ensure.

Spoofing — counterfeit IP addresses

The first difficulty IP networks pose is that it is difficult to know where information really comes from. A technique called IP spoofing takes advantage of this weakness.

To understand spoofing, you need to understand how information travels across the network. At the IP level, information is broken into small, manageable chunks of data called packets. Each of these packets contains three things:

- the data
- the IP address of the data source
- the IP address of the data destination

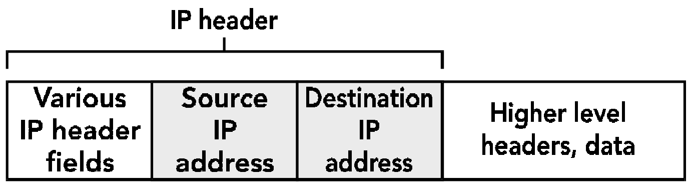


Figure 2: The basic IP packet

The structure of the IP packet, together with the logic of IP subnetting, gives IP much of its adaptability in terms of routing and traffic management, and much of its scalability. When two nodes on an IP network are communicating, the data stream between them is broken up into these packets, and released into the network. The route each packet takes to get to its destination isn't controlled by any particular network element — neither the sending node, nor any single device in between. Nor does any single device on the network have to know all the details of the network architecture to direct the packet to its destination. Most of the devices in between need only know the relative *directions* of the sending and receiving nodes, and need only be responsible for pointing the packet the right way, subject to traffic levels and bandwidth availability. So the packet wends its way through whatever nets and subnets are available to get where it's going. Two adjacent parts of an email message need not even travel the same route. The net can direct them however it wishes, according to how busy the various network segments are. At the other end, the receiving node takes the packets, reads the sending address, and puts it all back together.

The difficulty with this from a security perspective, however, is that source IP addresses in IP packet headers are easily changed. The attack — called spoofing — makes a packet coming from one machine appear to come from somewhere else altogether.

Session hijacking

Spoofing is one level of attack. It makes possible another. If your TCP/IP-based email client program trusts an IP source address to know that it's really communicating with the mail server, nothing prevents someone with sufficiently advanced tools from taking over the connection, sending the mail server a message to cut the link, and thereafter communicating through the same TCP link with your client.

The fact that you've identified the person with whom you're communicating once doesn't mean you can depend on IP to ensure it will be the same person through the rest of the session. You need a scheme that authenticates the data's source throughout the transmission.

Electronic eavesdropping — Ethernet LANs and sniffing

Sniffing is another attack that's possible in Ethernet-based IP networks.

Ethernet LANs make up a large part of most networks. Ethernet technology has the advantages of being cheap, universally available, well-understood, and easy to expand. It has the disadvantage of making sniffing easy.

In most Ethernet LANs, packets are available to every Ethernet node on the network. Conventionally, each node's network interface card (NIC) only listens and responds to packets specifically addressed to it.

It's relatively easy, however, to put many Ethernet NICs in what's called 'promiscuous mode' — meaning they can collect every packet that passes on the wire. There's no way to detect such a NIC from elsewhere on the network, because the NIC doesn't do anything to the packets when it picks them up.

A type of software colloquially called a 'sniffer' (after the original network analysis tool designed to do this — Network General's Sniffer™) can take advantage of this feature of Ethernet technology. Such tools can record *all* the network traffic going past them. As such they are a necessary part of the tool kit of any network diagnostician working with Ethernets — allowing them to determine quickly what's going through any segment of the network. However, in the hands of someone who wants to listen in on sensitive communications, a sniffer is a powerful eavesdropping tool.

The man-in-the-middle

The most obvious solution to the problems of IP security threats is the use of encryption technologies that conceal and authenticate the data passed in IP packets. But there are complications in doing this.

To use encryption, you first have to exchange encryption keys. Encryption keys are bits of information — a little like passwords — that are used with encryption algorithms to scramble and to unscramble data. The point to the key is that once you've encrypted data with that key you need the same key to decrypt it.

But exchanging unprotected keys through the network might easily defeat the whole purpose, since those keys could easily be intercepted and open up yet another type of attack — a threat academic cryptographers refer to as man-in-the-middle attack. A sophisticated attacker employing spoofing, hijacking, and sniffing could actually work his way into such a key exchange, in a system that left the way open. He could plant his own key early in the process so that, while you believed you were communicating with one party's key, you would actually be using a key known to the man-in-the-middle.

IPSec presupposes such an attacker may exist, somewhere in the public network — and provides mechanisms to defeat him.

Authentication, integrity, confidentiality, and scalable key management

IPSec offers three interlocking technologies, that combine to defeat all the traditional threats to IP-based networks:

- **Authentication Header (AH) —** ties data in each packet to a verifiable signature (similar to PGP email signatures) that allows you to verify both the identity of the person sending the data and that the data has not been altered.
- **Encapsulation Payload (ESP) —** scrambles the data (and even certain sensitive IP addresses) in each packet using hard core encryption — so a sniffer somewhere on the network doesn't get anything usable.
- **Internet Key Exchange (IKE)** — a powerful, flexible negotiation protocol that allows users to agree on authentication methods, encryption methods, the keys to use, how long to use the keys before changing them, and that allows smart, secure key exchange.

Secure VPN in an IP world

The IPSec protocol suite provides everything you need for secure communications — authentication, integrity, and confidentiality — and makes key exchange practical even in larger networks.

The end result is that with IPSec-compliant products you can build a secure VPN in any existing IP-based network.

Authentication, integrity, and confidentiality services

The basic building blocks of IPSec, the Encapsulating Security Payload (ESP) and the Authentication Header (AH), use cryptographic techniques for ensuring data confidentiality and digital signatures for authenticating the data's source.

How IPSec embeds encryption in the ESP

The IP datagram, or IP packet, is the fundamental unit of communications in IP networks. IPSec handles the encryption at the packet level. The protocol it uses is called ESP.

ESP supports pretty much any kind of symmetric[3] encryption[4]. The default standard built into ESP that assures basic interoperability is 56-bit DES.

ESP also supports some authentication (as can the AH — the two have been designed with some overlap).

The ESP (see Figure 3) follows the standard IP header in an IP datagram, and contains both the data and all higher level protocol headers relying on IP for routing.[5] The figure does not show the IP header.

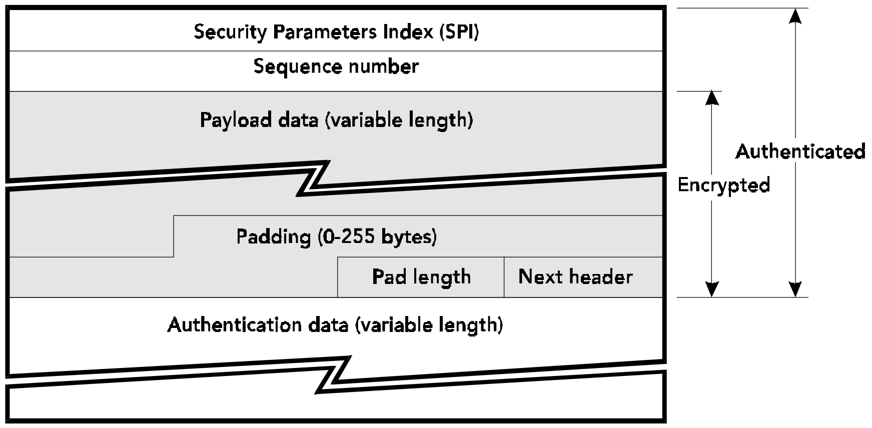

Security Parameters Index (SPI)
Sequence number
Payload data (variable length)
Padding (0-255 bytes)
Pad length
Next header
Authentication data (variable length)
Authenticated
Encrypted

Figure 3: The Encapsulating Security Payload

The ESP contains six parts. The first two parts are not encrypted but are authenticated:

- The **Security Parameter Index (SPI)** is an arbitrary 32-bit number that specifies to the device receiving the packet what group of security protocols the sender is using for communication — which algorithms, which keys, and how long those keys are valid.

- The **Sequence number** is a counter that increases each time a packet is sent to the same address using the same SPI. It indicates which packet is which and how many packets have been sent with the same group of parameters. The sequence number provides protection against replay attacks — in which an attacker copies a packet and sends it out of sequence, to confuse communicating nodes.

The remaining parts (with the exception of the authentication data) are all encrypted during transmission across the network. When unencrypted, they look like this:

- The **Payload data** is the actual data being carried by the packet.
- The **Padding** — from 0 to 255 bytes of data — allows for the fact that certain types of encryption algorithms require the data to be a multiple of a certain number of bytes — to confuse sniffers trying to estimate how much data is being transmitted**.**
- The **Pad length** field specifies how much of the payload is padding as opposed to data.
- The **Next header** field, like a normal IP Next header field, identifies the type of data carried and the protocol above[6].

The final field is an authentication field (see *Authentication within the ESP* in this paper).

Backwards compatibility

Note that the ESP is added after a standard IP header (one that contains as its protocol field a number that says there's an ESP following, instead of a TCP header). Because the packet has a standard IP header, the network can route it with standard IP equipment. So IPSec is backwards-compatible with IP routers and other equipment that isn't yet become IPSec aware.

Any encryption algorithm will do

ESP can support any number of encryption protocols; it's up to the user to decide which ones to use. You can even use different protocols for each party with whom you're communicating. But IPSec specifies a basic DES-CBC (Cipher Block Chaining mode) cipher as its default, to guarantee a minimal interoperability among IPSec networks.

ESP's encryption support is designed for use by symmetric encryption algorithms. IPSec mostly uses asymmetric algorithms for such specialized purposes as negotiating which keys to use for the symmetric encryption — we'll get into that later in the white paper.

Tunneling with the ESP

The ESP also provides for a secure VPN service called tunneling.

Tunneling takes the original IP packet header and encapsulates it within the ESP. Then it adds to the packet a new IP header containing the address of a gateway (a type of secure VPN equipment).

Tunneling allows you to pass illegal IP addresses through a public network (like the Internet) that otherwise wouldn't accept them. Tunneling with ESP also has the advantage of hiding the original source and destination addresses from users on the public network — defeating or at least reducing the power of traffic analysis attacks. A traffic analysis attack uses network monitoring to determine who's saying how much to whom. An attacker doing traffic analysis doesn't know what is being said, but does know how much is being said, and that can be valuable information too (see *Tunneling Services* in this paper).

Authentication[7] within the ESP

The ESP authentication field, an optional field in the ESP, contains something called an Integrity Check Value (ICV) — essentially a digital signature[8], computed over the remaining part of the ESP (minus the authentication field itself). It varies in length depending on the authentication algorithm used. It may also be omitted entirely, if authentication services are not selected for the ESP.

The authentication is calculated on the ESP packet once encryption is complete.

The ICV supports symmetric (classical) authentication schemes. The source encrypts a hash of the data payload (or just composes a keyed hash of the payload), and attaches this as the authentication field. The recipient confirms there has been no tampering and that the payload did come from the expected source by checking the authentication field.

The current IPSec standard specifies HMAC (a symmetric signature scheme) with hashes SHA-1 and MD5 as mandatory algorithms for doing authentication.

AH — authentication without confidentiality

The IPSec suite's second protocol, the Authentication Header, provides authentication services but does not address confidentiality.

The AH may be applied alone, in concert with the ESP, or in a nested fashion when using tunnel mode (see *The ESP and Tunnel mode* and the tunneling discussion in *Putting it all together* in this paper).

Authentication provided by the AH differs from that provided in the ESP in that the ESP's authentication services do not protect the IP header that precedes the ESP. The AH services protect this external IP header, along with the entire contents of the ESP packet.

The AH does not protect all of the fields in the external IP header because some of them change in transit, and the sender cannot predict how they might change. The standard is designed so AH works around these fields.

In the packet, the AH goes after the IP header but before the ESP (if present) or other higher level protocol (like TCP, in the absence of ESP)[9]. The figure below shows the AH format:

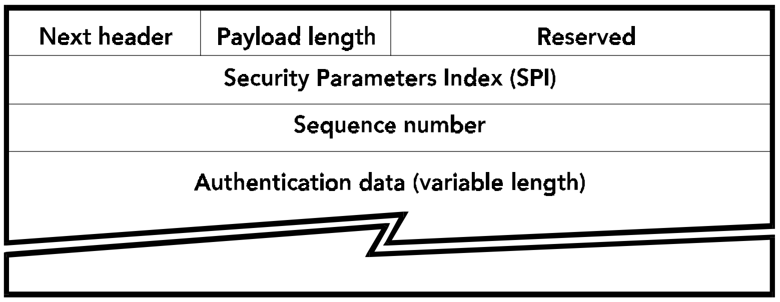

Next header
Payload length
Reserved
Security Parameters Index (SPI)
Sequence number
Authentication data (variable length)

Figure 4: The Authentication Header (AH)

The parts are as follows:

- the **Next header** field indicates what the higher level protocol following the AH is (ESP or TCP, for example)
- the **Payload length** field is an 8-bit field specifying the size of the AH, in 32 bit words (groups of 4 bytes), minus two[10]
- the **Reserved** field is reserved for future use and is currently always set to zero
- the **SPI**, as in the ESP packet, identifies a set of security parameters (algorithms and keys) to use for this connection
- the **Sequence number**, also as in the ESP packet, is a number that increases for each packet sent with a given SPI, for the purposes of keeping track of the order the packets go in, and to make sure that the same set of parameters is not used for too many packets
- finally, the **Authentication data** is the actual ICV, or digital signature, for the packet. It's much the same as the ICV used in the ESP authentication field. It may include padding to bring the length of the header to an integral multiple of 32 bits (in IPv4) or 64 bits (IPv6)

Like the ESP, the AH can be used to implement tunneling mode (see *Tunneling Services* in this paper).

Also as in the ESP, IPSec requires specific algorithms to be available for implementing the AH. These are the minimum any IPSec implementation must support to guarantee minimal interoperability. All IPSec implementations, under the standard, must support at least HMAC-MD5 and HMAC-SHA-1 (the HMAC symmetric authentication scheme supported by MD5 or SHA-1 hashes) for the AH.

AH in IPv4 vs IPv6

IPv6 is the next IP standard coming down the road. The IPv6 header is quite different from the existing IPv4 header. Among the more dramatic changes, IPv6 headers carry 128 bit addresses instead of 32 bit addresses. Among other things, IPv6 is expected to solve the problems of coming up with new IP addresses in an expanding Internet. It is also expected to make a more flexible network architecture.

One of the difficulties with IPv6 headers, and the host of optional headers IPv6 specifies, is that there is more in them that can change in transit through the network. This makes wrapping the AH's authentication around an IPv6 packet a little more complicated.

However, the IPSec group is developing the AH in concert with IPv6 standards, and has developed protocols for flexible ranges of authentication and intelligent placing of the AH header in the IP packet so that it can work under either IPv4 and IPv6.

AH and ESP — only part of the picture

You can view the AH and ESP protocols as the building blocks of IPSec. The two protocols provide the pieces you need to build a secure VPN. They provide the fundamental services you need — confidentiality, authentication, and integrity.

What you need next is the mortar that holds them together. The encryption services provided by the AH and ESP represent a powerful technology for keeping your data secret, for verifying its origin, and for protecting it from undetected tampering. But they mean little without a wisely-designed infrastructure to support them, to distribute keys, and to negotiate protocols between communicating parties. Proposed secure VPN systems succeed or fail based on the strength of this infrastructure and its scalability.

And that's where IPSec's IKE protocol suite comes in.

Protocol negotiation and key management

The password and its numerical analogue, the PIN, are among the simplest ways available of guaranteeing security. Easy to implement and generally suitable to a range of security needs, they're everywhere now. The concept is simple, a password or PIN is something you know that no one else does, that therefore identifies you.

But they're a pain to remember. Probably everyone has at least one embarrassing incident to recount — the day they drew a blank at the bank machine and couldn't remember their PIN. Or the time building security almost called the police.

The same basic difficulty faces anyone trying to implement security measures based on keyed encryption techniques. Keys are like passwords, and you've got to keep track of them. And the more keys you have, the harder it is to keep track.

Factor in that keys often change regularly, and you've got the makings of a real nightmare on your hands.

How many keys? An illustration

Imagine twenty stock traders have interests that are sometimes compatible and sometimes competing. They want to talk to each other privately and safely, through a public network.

If they use a symmetric encryption scheme to do this, they will need a total of 190 keys among them. Each of them has to keep track of 19 keys — one to communicate with each other member of the group.

Add to this that they want to keep changing those keys, at least monthly, and that they have to do it in person (which is safest). So each trader has to meet with each of the 19 other traders monthly, for a total of 190 two-person meetings each month.

Obviously, this is not practical.

The first thing you can do to improve this situation is to use asymmetric encryption instead of symmetric. With asymmetric encryption, each trader issues a public key that all the other traders use. Each trader also has his own private key. So each trader actually has to remember 20 keys (the 19 public keys of the other traders, plus his own private key). But exchanging them is far simpler. They could all meet at one big meeting, and each publicly announce his public key. We're now down from 190 meetings to one meeting a month. Still complicated, but getting easier.

This artificial example illustrates two important points:

- key exchange is fundamentally a complicated business
- key exchange gets more complicated as the group of communicating players expands

So just because a system says it does encryption, that alone doesn't mean it's going to be appropriate for your needs, even if the system does that encryption well. Any proposed secure VPN solution is only as good as its method of key exchange, especially if it serves large enterprise needs.

IPSec does support large enterprise needs and its industrial strength key exchange and protocol negotiation scheme sets it apart from all other security systems.

Key management and exchange

To communicate with someone using authentication and encryption services (like those provided by IPSec's ESP and AH), you need to do three things:

- negotiate with other people the protocols, encryption algorithms, and keys, to use
- exchange keys easily (this might include changing them often)
- keep track of all these agreements

IPSec provides mechanisms to do all three.

The Security Association

The first problem IPSec's designers solved was actually number three, how to keep track of all the details, and which keys and which algorithms to use. They did it by bundling everything together in something called the Security Association (SA).

An SA groups together all the things you need to know about how you communicate securely with someone else. The SA, under IPSec, specifies:

- the mode of the authentication algorithm used in the AH and the keys to that authentication algorithm
- the ESP encryption algorithm mode and the keys to that encryption algorithm
- the presence and size of (or absence of) any cryptographic synchronization to be used in that encryption algorithm
- how you authenticate your communications (using what protocol, what encrypting algorithm, and what key)

- how you make your communications private (again, what algorithm, and what key)
- how often those keys are to be changed
- the authentication algorithm, mode, and transform for use in ESP plus the keys to be used by that algorithm
- the key lifetimes
- the lifetime of the SA itself
- the SA source address
- and a sensitivity level descriptor

You can think of the SA as your secure channel through the public network to a certain person, group of people, or network resource. It's like a contract with whomever is at the other end.

The SA also has the advantage that it lets you construct classes of security channels. If you need to be a little more careful talking to one party than another, the rules of your SA with that party can reflect extra caution — specifying stronger encryption, for example.

The SA applied to business — an example

SAs are good for building multiple secure VPNs. Imagine your company has its own secure VPN. You develop a business relationship with another company which also has a secure VPN. You're perfectly happy to give them some access to your network by linking the two secure VPNs, as this facilitates business, but you don't want them to have full access. Some things do not belong outside the company, or even outside Human Resources' filing cabinets.

So you have specific SAs between your secure VPN and theirs, controlling who has what access to which resources. And you have another set of specific SAs within your secure VPN. And another within Human Resources. And so on.

You can layer your secure VPN like an onion using the SA concept. That's what the SA is for.

How the SA works with the SPI

The SA is a concept. The SPI is more concrete. The SPI is a number that uniquely identifies an SA. The SPI, together with the SA concept, makes keeping track of keys and protocols easy and automatic.

Specifically, the SPI is the 32 bit number we mentioned earlier when describing the ESP and AH packet formats (see *How IPSec embeds encryption in the ESP* and *AH — authentication without confidentiality* in this paper).

The SPI is an arbitrary 32-bit number your system picks to represent that SA whenever someone negotiates an SA with you. It identifies the SA.

The SPI can not be encrypted in the packet because you use it to keep track of how to decrypt the packet.

It works like this. When you negotiate an SA, the recipient node assigns an SPI it isn't already using, and preferably one it hasn't used in a while. It then communicates this SPI to the node with which it negotiated the SA.

From then until that SA expires, whenever that node wishes to communicate with yours using that SA, it uses that SPI to specify it.

Your node, on receipt, would look at the SPI to determine which SA it needs to use. Then it authenticates and/or decrypts the packet according to the rules of that SA, using the agreed-upon keys and algorithms to verify (depending on the terms of the SA) that the data really did come from the node it claims, that the data has not been modified, and that no one between those nodes read the data.

The next thing we need is to work out how to negotiate those SAs in the first place.

IKE — industrial strength key exchange

IKE, the IPSec group's answer to protocol negotiation and key exchange through the Internet, is actually a hybrid protocol. It integrates the Internet Security Association and Key Management Protocol (ISAKMP) with a subset of the Oakley key exchange scheme.

IKE provides a way to:

- agree on which protocols, algorithms, and keys to use (negotiation services)
- ensure from the beginning of the exchange that you're talking to whom you think you're talking to (primary authentication services)
- manage those keys after they've been agreed upon (key management)
- exchange material for generating those keys safely

Key exchange is a closely related service to SA management. When you need to create an SA, you need to exchange keys. So IKE wraps them both up together, and delivers them as an integrated package.

Manual key exchange

There is one other way to exchange keys. IPSec specifies that compliant systems support manual keying as well. That means if you wish to use manual (face-to-face) key exchange for certain situations, you still can. But IPSec's designers also assume that in most situations, for most large enterprises, this would be impractical. So IKE, the safe way to negotiate SAs and exchange keys through public networks, will probably do most of the work for most of the world.

IKE phases

IKE functions in two phases. In phase one, two IKE peers establish a secure channel for doing IKE (called the IKE SA). In phase two, those two peers negotiate general purpose SAs.

An IKE peer is an IPSec-compliant node capable of establishing IKE channels and negotiating SAs. It might be the computer on your desktop, or something called a security gateway that negotiates security services for you.

IKE modes

Oakley provides three modes of exchanging keying information and setting up SAs — two for IKE phase one exchanges, and one for phase two exchanges.

- **Main mode** accomplishes a phase one IKE exchange by establishing a secure channel.
- **Aggressive mode** is another way of accomplishing a phase one exchange — it's a little simpler and a little faster than main mode, but does not provide identity protection for the negotiating nodes, as they must transmit their identities before having negotiated a secure channel through which to do so.
- **Quick mode** accomplishes a phase two exchange by negotiating an SA for general purpose communications.

Establishing a secure channel for negotiation

To establish an IKE SA, the initiating node proposes six things:

- encryption algorithms (to protect data)
- hash algorithms (to reduce data for signing)
- an authentication method (for signing data)
- information about a group over which to do a Diffie-Hellman exchange
- a pseudo-random function (PRF) used to hash certain values during the key exchange for verification purposes (this is optional, you can also just use the hash algorithm)
- the type of protection to use (ESP or AH)

Perfect forward secrecy

One of the most annoying things about passing encrypted data around a public network is the number of opportunities an attacker has to get hold of encrypted material. You can reduce the risk of their ever deciphering it by using larger and larger keys. But the larger the key, the slower and more complex the encryption and this can impair network performance.

A good compromise solution is to use reasonably large keys, and to keep changing them. But this presents difficulties too. You need ways to generate those new keys so that the person at the other end can agree on it as well. But, to generate your new key, you can't use either the key you're changing from, or material used to generate the key you're changing from.

The point being that if you do, and then someone gets hold of the current key, that person can easily deduce your new key.

What you need is a method of generating a new key that is in no way dependent on the value of the current key. So if someone gets hold of your current key, it only gives them a small part of the overall picture, and they would have to break yet another entirely unrelated key to get the next part. Cryptographers call this concept "perfect forward secrecy". IKE uses a scheme called Diffie-Hellman to do this.

Diffie-Hellman

A Diffie-Hellman exchange works like this: two people independently and randomly generate values much like a public/private key pair. Each sends their public value to the other (using authentication to close out the man-in-the-middle). Each then combines the public key they received with the private key they just generated, using the Diffie-Hellman combination algorithm.

The resulting value is the same on both sides, and therefore can be used for fast symmetric encryption by both parties. But no one else in the world can come up with the same value from the two public keys passed through the net, since the final value also depends on the private values, which remain secret.

You can use the derived Diffie-Hellman key either as a session key for subsequent exchanges, or to encrypt yet another randomly generated key, which you can then pass through the net quite safely.

Note that yes, you do need authentication to protect even Diffie-Hellman exchanges against the man-in-the-middle. Diffie-Hellman alone does not solve this problem. It would be complicated, but without authentication a man-in-the-middle could use an active attack to get in on the action and plant his own keys.

But if the key exchange mechanism you're using is protected by an authentication scheme, Diffie-Hellman allows you to generate new shared keys to use for symmetric encryption which are independent of older keys — providing perfect forward secrecy. And since symmetric encryption techniques are a lot faster, this can be quite useful in network communications.

You may wish to agree on a few things to do a Diffie-Hellman exchange in the first place. That's what the Diffie-Hellman parameter in the IKE SA is for. The parameter contains information on a group to perform the Diffie-Hellman exchange. The group consists of generation material used for coming up with keys.

The pseudo-random function

The pseudo-random function (PRF) is also probably worth explaining briefly. A PRF is really just another name for a hash function.

In IKE, you can use the PRF both for authentication purposes and to generate additional key material (as a randomizer).

Main mode

Main mode provides a mechanism for establishing the first phase IKE SA, which is used to negotiate future communications. Remember, the object here is to agree on enough things (authentication and confidentiality algorithms, hashes, and keys) to be able to communicate securely long enough to set up an SA for future communication. The steps in full will be:

(1) use Main mode to bootstrap an IKE SA

(2) use Quick mode within that IKE SA to negotiate a general SA

(3) use that SA to communicate from now until it expires

The first step, securing an IKE SA using Main mode, occurs in three two-way exchanges between the SA initiator and the recipient (see Figure 5). In the first exchange (1 and 2 in the illustration below), the two agree on basic algorithms and hashes. In the second (3 and 4 in the illustration below), they exchange public keys for a Diffie-Hellman exchange, and pass each other nonces — random numbers the other party must sign and return to prove their identify. In the third (5 and 6 in the illustration below), they verify those identities.

So the following is how IKE establishes its own IKE SA, step by step, using Main mode, and established digital signatures for authentication.

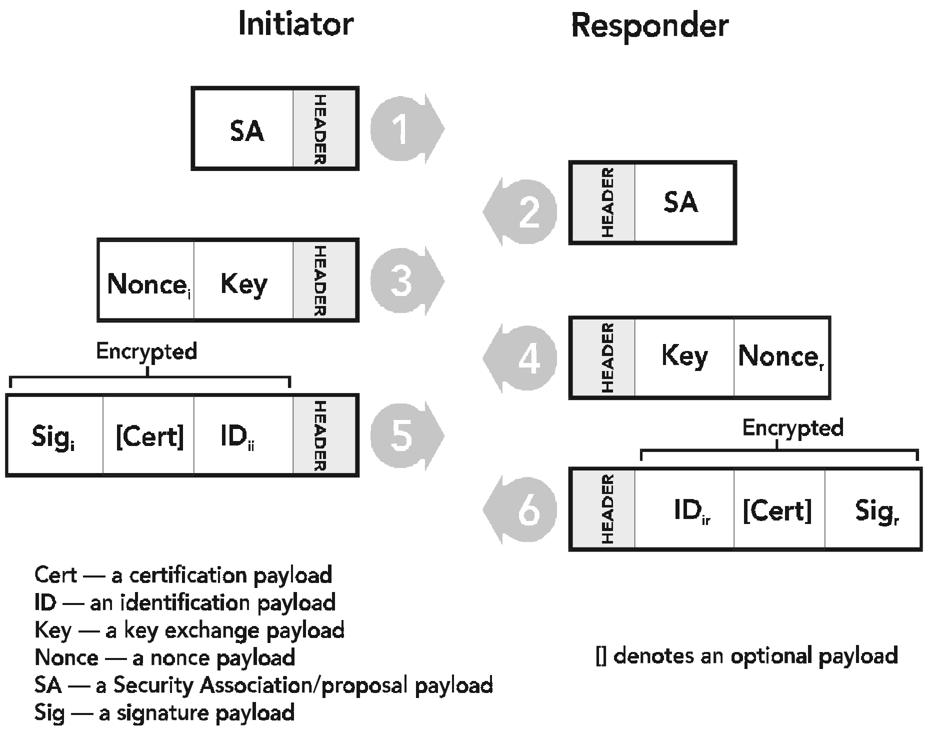

Initiator
Responder
SA
HEADER
1
2
SA
Nonce_i
Key
3
4
Key
Nonce_r
Encrypted
Sig_i
[Cert]
ID_ii
5
6
ID_ir
[Cert]
Sig_r
Cert — a certification payload
ID — an identification payload
Key — a key exchange payload
Nonce — a nonce payload
SA — a Security Association/proposal payload
Sig — a signature payload
[] denotes an optional payload

Figure 5: IKE Main mode

Each of the pieces is carried in its own payload, but you can pack any number of these payloads into a single IKE packet.

The parties actually use the generated shared Diffie-Hellman value in three permutations, once they derive it. Both parties have to hash it three times — generating first a derivation key (to be used later for generating additional keys in Quick mode), then an authentication key (for authentication), and then, finally the encryption key to be used for the IKE SA.

Aggressive mode

Aggressive mode provides the same services as Main mode. It establishes the original IKE SA. It looks much the same as Main mode except that it is accomplished in two exchanges, rather than three, with only one round trip, and a total of three packets rather than six.

In aggressive mode, the proposing party generates a Diffie-Hellman pair at the beginning of the exchange, and does as much as is practical with that first packet — proposing an SA, passing the Diffie-Hellman public value, sending a nonce for the other party to sign, and sending an ID packet which the responder can use to check their identity with a third party. The responder than sends back everything needed to complete the exchange — really an amalgamation of all three response steps in Main mode, and all that's left for the initiator to do is to confirm the exchange (see Figure 6).

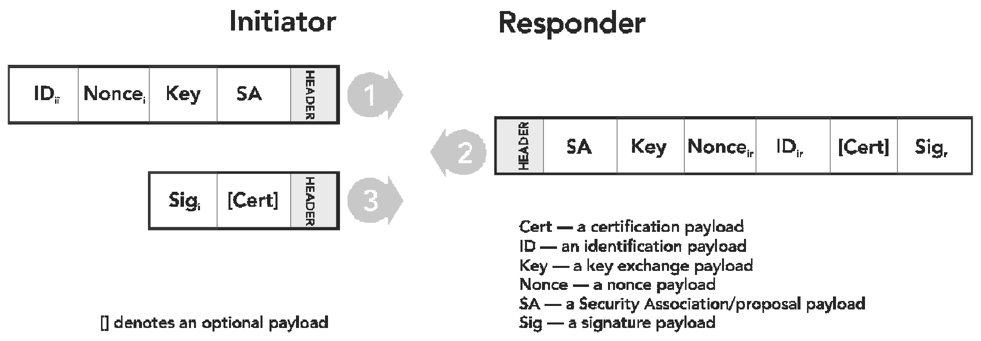

Initiator
Responder
ID_ir
Nonce_i
Key
SA
HEADER
1
2
HEADER
SA
Key
Nonce_ir
ID_ir
[Cert]
Sig_r
Sig_i
[Cert]
HEADER
3
Cert — a certification payload
ID — an identification payload
Key — a key exchange payload
Nonce — a nonce payload
SA — a Security Association/proposal payload
Sig — a signature payload
[] denotes an optional payload

Figure 6: IKE Aggressive mode

The end result is that an Aggressive mode exchange attains the same goal as a Main mode exchange, except that Aggressive mode does not provide identity protection for the communicating parties. That is to say, in Aggressive mode, the parties exchange identification information prior to establishing a secure SA in which to encrypt it. So someone monitoring an aggressive exchange can actually identify who has just formed a new SA.

The advantage of Aggressive mode, however, is speed.

Quick mode

Once two communicating parties have established an IKE SA using Aggressive mode or Main mode, they can use Quick mode.

Quick mode has two purposes: negotiating general IPSec services, and generating fresh keying material.

Quick mode is less complex than either Main or Aggressive mode. Since it's already inside a secure tunnel (every packet is encrypted), it can also afford to be a little more flexible.

Quick mode packets are always encrypted, and always start with a hash payload. The hash payload is composed using the agreed-upon PRF and the derived authentication key for the IKE SA. The hash payload is used to authenticate the rest of the packet. Quick mode defines which parts of the packet are included in the hash.

Key refreshing can be done one of two ways. If you don't want or need perfect forward secrecy, Quick mode can just refresh the keying material already generated (in Main or Aggressive mode) with additional hashing. The two communicating parties can exchange nonces through the secure channel, and use these to hash the existing keys.

If you do want perfect forward secrecy you can still request an additional Diffie-Hellman exchange through the existing SA and change the keys that way.

Basic Quick mode is a three packet exchange, like Aggressive mode.

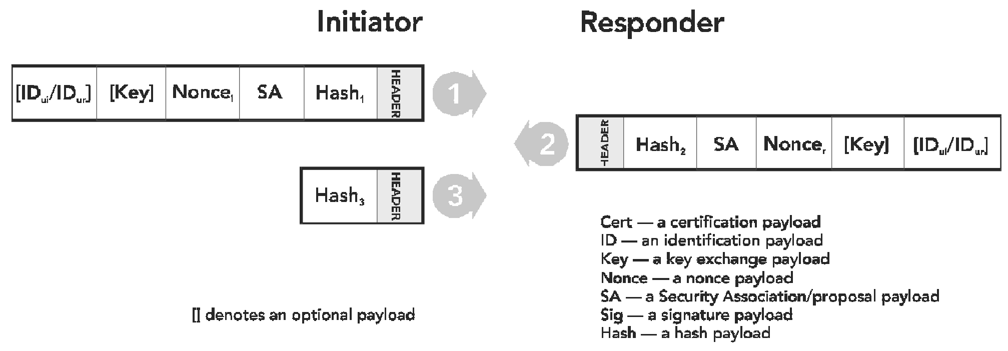

Initiator
Responder
[IDui/IDur]
[Key]
Nonce
SA
Hash1
HEADER
1
2
Hash2
Hash3
3
[] denotes an optional payload
Cert — a certification payload
ID — an identification payload
Key — a key exchange payload
Nonce — a nonce payload
SA — a Security Association/proposal payload
Sig — a signature payload
Hash — a hash payload

Figure 7: IKE Quick mode

If the parties do not require perfect forward secrecy, the initiator sends a packet with the Quick mode hash, with proposals and a nonce. The respondent then replies with a similar packet, but generating their own nonce and including the initiator's nonce in the Quick mode hash for confirmation. The initiator then sends back a confirming Quick mode hash of both nonces, completing the exchange. Finally, both parties perform a hash of a concatenation of the nonces, the SPI, and the protocol values from the ISAKMP header that initiated the exchange, using the derivation key as the key for the hash. The resulting hash becomes the new password for that SA.

If the parties do require perfect forward secrecy, the initiator first generates a public/private key pair, and sends the public key along with the initiation packet (along with the hash and nonce). The recipient then responds with their own public key and nonce, and both parties then generate the shared key using a Diffie-Hellman exchange — again fully protected by the Quick mode hashes, and by full encryption within the IKE SA.

Negotiating the SA

After all those exchanges to generate the IKE SA, actually establishing the general purpose SA is relatively simple.

To generate a new SA, the initiator sends a Quick mode message, protected by the IKE SA, requesting the new SA. A single SA negotiation actually results in two SAs — one inbound, to the initiator, and one outbound. Each IPSec SA is one way and the node on the receiving end of that SA always chooses its own SPI to ensure it is the only SA using that reference.

So, using Quick mode, the initiator tells the respondent which SPI to use in future communications with it, and the respondent follows up with its own selected SPI.

Each SPI, in concert with the destination IP address and the protocol, uniquely identifies a single, IPSec SA. However, these SAs are always formed in pairs (inbound and outbound), and these pairs have identical parameters (keys, authentication and encryption algorithms, and hashes) — apart from the SPI itself.

So how do you know who's who?

We're almost to the end of explaining how the various parts of IPSec work together to make your communications safe. But we haven't yet explained how you verify that people are who they say they are in the first place.

Which brings us to the third party.

Joe sent me – third party verification via a CA

The final component of the IPSec-compliant secure VPN, in most implementations, is something called a Certification Authority (CA).

A CA is a trusted third party — someone whose identity you can already prove, and who can vouch for the identity of people with whom you're trying to communicate.

The idea is easy enough to relate to the real world. The CA is like a public figure (for example, a notary public) who you know well enough to trust, and who vouches for other people.

In the world of online verification, the CA software issues certificates tying three things together:

- someone's identity (anything that uniquely identifies them to you in a meaningful way so you can't confuse them with someone else (i.e. their name and place of residence)
- the public key that a person uses to sign their identity to online documents
- the CA's public key (used to sign and authenticate the entire package)

The CA is the defense against that man-in-the-middle working his way into key exchanges. Whenever you initiate an exchange with someone, he has to sign it with his digital signature. And then you in turn can check that signature against the one on record with the CA. They have to match. You then check that certificate's signature with the CA's signature. They have to match too.

What stops the man-in-the-middle from forging the CA's signature, and turning out his own certificates? Mostly smart distribution. Because it's a highly public entity, and a highly useful value, the CA's public key can be on record in a number of places — making it difficult for anyone to get away with forging it.

You also usually design the CA's signature with strong encryption and an extremely large keyspace. This makes it long lived, so you can distribute it any number of ways, including on the disk on which the verification software is distributed. It makes it difficult indeed (provided the system is vigilant about checking) to introduce counterfeit signatures into the system.

IKE provides for third party verification using the established industry-standard X.509 format certificate.[11]

So when someone initiates an IKE exchange with you, or responds to one you have initiated, you first send the data they have to sign with their digital signature. They then send their certificate, signed by the CA, thereby proving the signature they used legitimately belongs to them. You first check the certificate signature against your copy of the CA's signature to make sure the CA really issued it. Provided it checks out, you then look up the signature attached to the certificate (the one belonging to the person you're trying to initiate an exchange with), and check the data they've just signed against it.

It's really a chain of trust — beginning with your trust in the CA's signature, which is then used to verify someone else's.

All of this may look complicated, but it's part of the beauty of IPSec that this can all be done in the background — quickly, quietly, and painlessly. It's far less trouble then checking the signature on the back of a credit card, in that your verification system checks for forgeries for you. You don't even have to think about it.

Robust, scalable key exchange

Altogether, IPSec's IKE system offers network users a great deal that other schemes have had difficulty delivering. But most critically, IPSec's designers built the system's key exchange from the ground up to be genuinely scalable — so it works with any size of system.

So to put all that together, with an IPSec secure VPN

- you can form SAs with a range of attributes, layering your network
- you can build multiple domains of communication within the same secure VPN
- you can update keys as frequently as you wish, and get the best of both worlds in doing so — perfect forward secrecy, and a silent, painless background mechanism the user doesn't have to think about
- you can do this in any IP network environment, of any size, for any size of enterprise
- and finally, you can do this with anyone using IPSec technology, creating new SAs with whomever you wish to communicate

IPSec as a universal standard

"Rapid advances in communication technology have accentuated the need for security in the Internet. The IP Security Protocol Working Group will develop mechanisms to protect client protocols of IP. A security protocol in the network layer will be developed to provide cryptographic security services that will flexibly support combinations of authentication, integrity, access control, and confidentiality."

— from the IPSec working group's charter

The IPSec working group's goal from the beginning was to provide an optional new version of IP — one that provides security where there was none before.

They have now done so and their work has the potential to change drastically the character of the Internet, making it suitable for carrying secure communications.

We have discussed at length in this paper how IPSec works and why you can trust it to get the job done. But probably one of the most important things about the protocol suite is not its robust design, but the simple fact that it is an open standard and an Internet standard, so any number of vendors and service providers can specialize and cooperate to provide the total range of IPSec equipment and services you need.

Maintaining network infrastructure in an era of rapidly changing technologies has always been a challenge for large and medium-sized enterprises. With the hardware industry constantly on the march, and capabilities and challenges changing by the week, the more you've invested in, the harder it gets to keep up. And for smaller firms, it is frequently impractical for them to build secure networks to bridge the gap between one office and another, or to provide services to their employees on the road, in the first place. Leased lines and dial-up servers just cost too much.

But with IPSec secure VPN you can have secure networking cheaply and without making constant, drastic changes to your network infrastructure.

A laptop with a suitable IPSec-compliant program can connect securely and transparently to your corporate network through a dial-up ISP anywhere in the world.

You no longer need to lease costly, dedicated lines if an Internet connection of suitable available bandwidth will do the same job. And the larger Internet Service Providers (ISPs) are now moving towards providing you with guarantees of service similar to what you'd get with a leased line.

ISPs are also already beginning to offer IPSec services as part of their packages. But where they haven't yet begun, you can still use IPSec technology. It's backwards compatible and travels comfortably over regular IP networks.

Where the ISPs have begun, however, you can make managing your information technology less painful. You can farm out what you'd rather not have to worry about anymore to your ISP — whose business it is to keep up.

The transformation of the Internet

We've come a long way from what you need for security, through why it was so hard to get before, to how IPSec makes things so much easier.

Large IP networks are remarkable entities. Almost like living things, they tend to grow and evolve with the organizations that run them — developing into larger and larger tangles of nodes, until no single person has any way of knowing, nor any hope of ever finding out, just what the network's dimensions are.

This is not, in itself, a bad thing. It's part of the beauty of IP that it's possible in the first place. But it does make keeping track of security difficult. As noted in the introduction, this is why the Internet, and large IP networks, in general, have not in the past been particularly safe places for sensitive data and communications.

That's unfortunate, because large IP networks are remarkable resources. The security weaknesses, however understandable from a historical perspective, can be terribly limiting in terms of the scope of those network's usefulness.

It's a tricky thing adding security to this jungle. A challenging thing. Probably most challenging of all, you don't want to do anything that might limit the net's remarkable mutability, its almost organic quality. Large IP nets are chaotic, nebulous, amorphous things, but it's part of what makes them valuable and powerful.

It's challenging, but not impossible, to add a level of security to this mix without damaging what's already there. And the IPSec suite's designers have answered that challenge.

You can have the network's flexibility. It can grow as it needs to, when it needs to, and do what it does best — open up lines of communications between whomever might happen to need them.

But now, when you need to be sure those communications are secure, you can have that as well. You can build it right into the environment. That's what IPSec gives you.

So welcome to a new world of secure communications and the transformation of the Internet.

Appendix A: Putting it all together — the IPSec secure VPN

To get past the theory to the practical, here's how a typical IPSec secure VPN might look, applied to a firm with a corporate office in one city, a manufacturing center in another, partner companies using compatible equipment, and sales agents spread around the world:

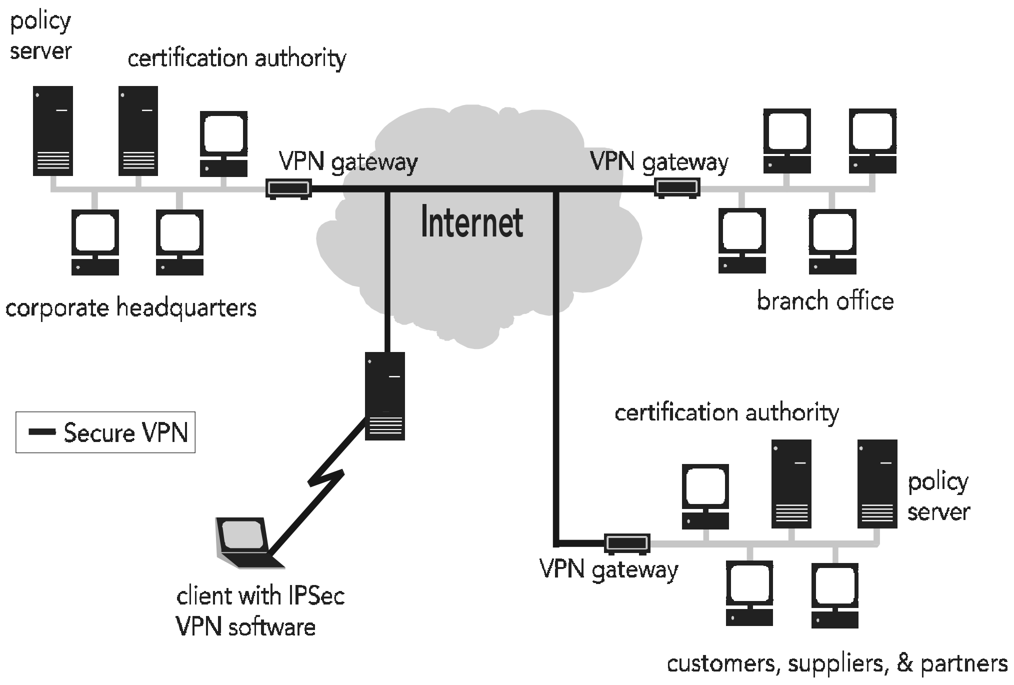


Figure 8: IPSec secure VPN on three continents, with sales agent

The security gateway

You'll note the three little black boxes at the intersections of the Internet (a public network) and private networks. Those are security gateways — a common component of IPSec-compliant secure VPNs.

A security gateway is a gatekeeper device that sits between public and private networks. It functions both as a filter to prevent unauthorized intrusions into the private network, and as a proxy device providing security services to nodes on the private network it protects. The security gateway may also provide tunneling capabilities which allows you to hide internal network addresses, and also allows nodes on the internal network with 'illegal' or unregistered IP addresses to communicate over the public network (see *Tunneling services,* below).

IPSec's protocol suites take into account the possibility that secure VPNs might choose to use such devices — tunneling services in particular — so the protocols support their use.

The security gateways negotiate SAs through the public net for protected nodes — and provide hardware encryption/decryption services — as in TimeStep's PERMIT/Gate family of IPSec-compliant gateways.

A well-built security gateway can do the lion's share of the work of providing IPSec-compliant security services in a typical secure VPN — doing bulk, high-speed encryption, negotiating SAs through the public net, providing tunneling services as required, and generally making sure things get done quickly, painlessly, and transparently, so no one outside your network can see what's going on inside.

The CA and the policy server

A few of the other critical parts of the IPSec-compliant VPN are the CA, and the security policy server.

Note the two CAs in the diagram. CAs run by two differing institutions can be made aware of each other, effectively allowing parties aware of either CA to authenticate users certified by both, by a mechanism known as cross-certification. This is the usual approach taken in extranet secure VPNs between two organizations.

Note the security policy server. IPSec-compliant client software and security gateways are usually set to query a central policy server through a secure channel before negotiating SAs. This allows central administration of who may contact whom, and what level of encryption they should use.

What's it look like to the user?

Using a well-run IPSec secure VPN, from a user's point of view, is exactly the same as using any well-run LAN. The user does not need to worry about whether the node with which they're communicating is down the hall or around the world. The IPSec services built into the network (either in the form of workstation software or a security gateway) determine whether the station needs an SA to communicate with the node or not. If an acceptable SA already exists, they use this one.

If they need an SA and do not have one, the IPSec components negotiate one on the fly — initiating an IKE Main or Aggressive mode exchange with the recipient station, checking the signatures against the node's certificate on the CA, and then putting together an IPSec SA for communications.

They do all this with just a few packets, automatically and in the background. The two nodes can be communicating with industrial strength encryption end-to-end, seconds after initiation, without the user ever having had to think about it.

Effectively, two LANs, one mile or hundreds of miles apart, become one big LAN — and a safe one.

Road warriors

For the user in the field, the process is similar. The user has to find a point-of-presence provider somewhere — and that's probably as complicated as it's ever going to get. They can then connect to the point-of-presence locally, and use the Internet to carry on communications as easily as if they were back in their office. An IPSec client program installed on their laptop goes through the same process as a security gateway on the office LAN — checking permissions, looking for an appropriate SA, and negotiating one (checking certificates for prior authentication) as necessary.

IPSec and tunneling

Tunneling services, another feature of the IPSec-compliant secure VPN, provide for address resolution and address hiding between private networks.

An IPSec-compliant security gateway can take an entire IP packet from a node within the network it protects and encapsulate it inside a new IP packet, before sending it out through the public network (see Figure 9).

This feature has two main applications:

- it can allow nodes that have an illegal IP address (meant only for internal use) to communicate with other nodes across a public network
- it can conceal the addresses of sensitive internal nodes (with legal addresses), protecting them even against denial of service attacks

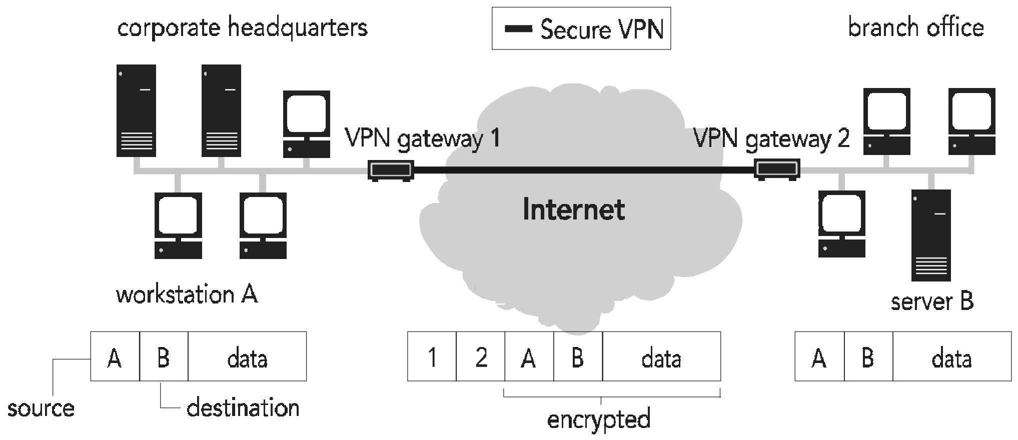


Figure 9: IPSec secure VPN, with tunneling, showing packet schematics

To explain: imagine node A in network 1 is the workstation of a developer in one office of a software firm. Node A does not have a legal, registered IP address. The developer, when he needs to connect to the net outside the office, normally just uses the proxy server, like everyone else.

Gateway 1 is the security gateway handling IPSec services for the nodes on the developer's network.

Server B is a server on which development teams working at a different office place the code they're working on, for coordination purposes. That code is proprietary to the developer's firm, and they don't want just anyone getting hold of it.

So server B's address is also illegal, and irresolvable outside the private network. That's intentional and done for security purposes.

Gateway 2 handles security services for the second office's LAN (network 2).

Suppose, however, the developer at workstation A — and probably other developers in the first office — need to get to the code on server B, to work more easily with development teams in the second office.

You can do this, safely, with tunneling.

It works like this: both gateways have tunnel tables that map the internal, illegal addresses to the gateway behind which they're located.

So whenever gateway 1 receives a packet destined for server B, it knows that server B is behind gateway 2. It takes the whole packet and encloses it in a special type of tunneling packet. Then it attaches a new IP header (one that points the packet at gateway 2) and sends it out into the public net.

When gateway 2 gets that packet, it recognizes it as a tunneled packet. So it takes it apart, finds the original destination address in the encapsulated packet, and sends it on to server B.

The gateway also encapsulates the sending address. So if workstation A is sending the packet to server B, gateway 1 also encrypts workstation A's address within the packet (as the sending address) and, in the IP header the external network sees, gateway 1 then sets itself as the sending address. Gateways 1 and 2 are the public sender and recipient. Internally (inside the encrypted packet), the sender and recipient are workstation A and server B respectively.

The most a sniffer somewhere in the Internet can find out is that someone is sending something between gateways 1 and 2. And the address of server B is always hidden inside the encrypted packet — so no one on the outside network knows how to get to that server, and it is protected against denial of service attacks.

Appendix B: Tunneling in IPSec and in PPTP

Point-to-Point Tunneling Protocol (PPTP) is an extension to the existing Point-to-Point-Protocol (PPP) used in the data link layer for point-to-point lines through much of the Internet (typically, PPP is the link-level protocol that allows computers connecting to the Internet through a dial-up connection to carry IP over the phone line between themselves and the ISP's server). It has surfaced periodically in discussions of virtual private networking as a potential solution for implementing secure VPNs, and solving private address conflicts via tunneling.

PPTP adds tunneling to the PPP, allowing address conflict resolution in a similar fashion to IPSec's tunneling services. PPTP is used on top of IP (as opposed to PPP, which is used below it and only in certain links) to produce a virtual point-to-point line through the IP network, thus providing a tunneling service something like IPSec's.

There is some contrast between the IETF draft PPTP standard (proposed in part by Microsoft), and the Windows NT 4.0 implementation of PPTP:

- The standard references the IPSec protocol suite as the sensible way to provide encryption services in PPTP implementations.
- The Windows NT implementation of PPTP, in contrast, provides built-in, software-based encryption through Windows NT's RAS security function. That means that for any significant number of users using PPTP in the field, you'll probably need a dedicated, high-end NT RAS server somewhere on the network (essentially an IP-based dial-up server) as the PPTP tunnel endpoint, to handle encryption/decryption for multiple users.

The PPTP draft is a reasonable solution for what it proposes to solve but it does not provide secure VPN at the scale the IPSec suite makes possible.

PPTP really just gives you a virtual dial-up line through an IP network — which is rather a different concept from the underlying philosophy of the IPSec protocol suite's architecture, which adds security and tunneling together to the IP layer itself.

PPTP's strength relative to IPSec's is that it can be used to carry protocols other than IP over the public network. IPSec's strength relative to PPTP is that it binds the security features and tunneling services tightly together, so that each reinforces the other, and uses IKE's sophisticated protocol negotiation features to make for a far more scalable solution.

It is technically possible to use both PPTP and IPSec together, but redundant, as both provide tunneling services. The short answer, finally, to the question "should I use PPTP or IPSec for my VPN?" is if you need to tunnel protocols other than IP, and scalability isn't much of a concern, use PPTP. But if you want the flexibility for your secure VPN to grow to thousands, or even millions, of nodes, use IPSec.

Endnotes

[1] Ethernet is a registered trademark of Xerox Corporation.

[2] IPv6 is an upcoming revision to the existing IPv4 standard. It makes a number of maintenance changes and one large one. The big change is that IPv6 will provide for 128-bit IP addresses — greatly extending available address space, which, on the Internet, is starting to get a little crowded.

[3] See the glossary for definitions of asymmetric and symmetric encryption schemes.

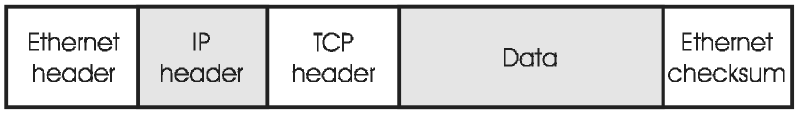
[4] TimeStep's PERMIT Enterprise line of IPSec-compliant products supports DES, 3-DES, RC5, CAST, IDEA, and Blowfish encryption algorithms, to name a few.

[5] Chronologically, the ESP is inserted after the higher level header. Normally, headers are assembled and added to the front of the packet with higher level protocols first, so lower level protocols are earliest in the packet — at the outside, and deciphered first, since the system needs them first. So, for example, in an IP packet carrying TCP data across an Ethernet LAN, it would go like this:

(1) First, the TCP packet header is assembled and appended to the data

(2) Then, the IP header is assembled, and appended to the TCP header (in front of it)

(3) Then the Ethernet header is assembled, and appended to the IP header (in front of it — though there's also a checksum on the end with Ethernet)



A typical TCP/IP packet on an Ethernet LAN

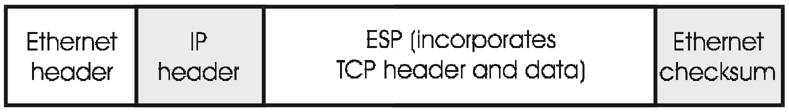
With the ESP, this chain becomes:

(1) First, the TCP packet header is assembled and appended to the data

(2) Second, the ESP packet is assembled. With encryption, it encapsulates the TCP packet in totality, including the header

(3) Third, the IP header is assembled and added to the front of this

(4) Finally, the Ethernet header and checksum are assembled and tacked onto the front and the end



A TCP/IP packet using ESP on an Ethernet LAN

[6] In IPv6 there is also a set of extension headers that may be specified here.

[7] Note that there are other methods of doing authentication in IPSec, including within the Authentication Header (AH). But there are reasons for wanting to do it within ESP too, especially if you're using tunneling.

[8] See *Digital signatures* in the glossary for the theory behind signatures.

[9] In IPv6, to be strictly correct, it goes after the extension (hop by hop, routing, and fragmentation) headers — a few new wrinkles added to make routing that more flexible. And IPv6's destination options header can go on either side of the AH. But that still puts the AH between the IP header and the higher level protocols. There are a few other little wrinkles to how AH authenticates under IPv6 but the critical point is that the IPSec suite does take IPv6's mutable fields into account (see the section *AH in IPv4 vs. IPv6).*

[10] The minus-two is for backwards compatibility with an earlier AH spec that was always at least two words long, and so didn't bother specifying those two words.

[11] But IKE is also an open standard, ready to incorporate whatever certificate standard you might want to use a few years down the road.

References

Bellovin, S.M. "Security Problems in the TCP/IP Protocol Suite." *Computer Communication Review,* Vol 19, No. 2, pp. 32-48, April 1989.

Harkins, D. and D. Carrel. "The Internet Key Exchange (IKE)." (RFC 2409) Internet standards track protocol, November 1998.

Kent, Stephen and Randall Atkinson. "IP Authentication Header." (RFC 2402) Internet standards track protocol, November 1998.

Kent, Stephen and Randall Atkinson. "IP Encapsulating Security Payload." (RFC 2406) Internet standards track protocol, November 1998.

Kent, Stephen, and Randall Atkinson. "Security Architecture for the Internet Protocol." (RFC 2401) Internet standards track protocol, November 1998.

Maughan, Douglas et al. "Internet Security Association and Key Management Protocol." (RFC 2408) Internet standards track protocol, November 1998.

Piper, Derrell. "The Internet IP Security Domain of Interpretation for ISAKMP." (RFC 2407) Internet standards track protocol, November 1998.

Glossary

Aggressive mode — an IPSec term referring to a packet exchange in **IKE** phase one; in **IKE** used to negotiate an **IKE SA**. Alternative to **Main mode**. See also **Quick mode**, **Main mode**.

AH — see **Authentication Header.**

Asymmetric encryption — an encryption scheme in which one key (the public key) and one algorithm is used to encrypt data, and another key (the private key) and another algorithm are used for decryption. The benefit of established asymmetric encryption schemes such as RSA are that you cannot easily find the private key simply from knowing the public one, and the public key can only be used for encryption, not for decryption. In practice, someone using asymmetric encryption for communication generates a public/private key pair, keeps the private key secret, and distributes the public key to anyone who wishes to communicate with them. Those with the public key can then use it to encrypt communications destined for that person, and only that person can decrypt that data. This property of asymmetric encryption schemes makes them particularly valuable in network environments. Asymmetric encryption schemes typically must use a much larger key than symmetric schemes. See also **encryption**, **symmetric decryption**.

Authentication — in cryptography, a way of proving identity. An authentication scheme typically requires a sender to perform manipulations on the data they're sending (with a **digital signature** scheme and/or a **hash function)** to prove they have certain cryptographic keys only they should know, thereby proving their identity. Authentication schemes often guarantee the integrity of the data being sent as well. See **digital signatures** and **hash functions**.

Authentication Header (AH) — part of the **IPSec protocol suite.** It is the header used in IPSec-compliant IP packets to carry authentication data (from a **digital signature scheme** or **keyed hash**), thereby preventing tampering during transmission and permitting verification of the identity of the sending party.

Ciphertext — the coded text generated in an encryption scheme when a **plaintext,** the original message or data, is encrypted. See **encryption**.

Datagram — network term referring to a unit of data. See **packet**.

Data Encryption Standard (DES) — a **symmetric encryption** algorithm certified as a standard for US government departments that use encryption. DES uses a 56 bit key, giving it a 2^{56} **keyspace**. DES has the advantage that it is easily implemented in hardware (and that chips to implement it are readily available), but the disadvantage is that, by computational standards now available, its keyspace may not be large enough for continued use. The **IPSec protocol suite**'s **ESP** protocol uses DES as the minimum fallback standard, but also supports newer schemes with larger keyspaces.

DES — see **Data Encryption Standard**.

Digital signature — a form of authentication. In digital signature schemes, persons proving their identify must encrypt transmitted data with a key only they could have, and then pass both the original data and the **ciphertext** they generate with their key to whomever wishes to verify their identity. Such schemes often offer the benefit that they protect transmitted data against alteration, since doing so would also require that the signature be changed to match it, and the key generating that signature is secret. The **PGP/Web of Trust** system, for example, uses asymmetric **RSA** keys to generate digital signatures on e-mail.

Since most encryption schemes generate ciphertext at least as long as the original **plaintext**, many digital signature schemes also use **hash functions** to reduce the amount of data that must be signed. The sending party first uses the hash to produce a shorter (usually fixed in size) piece of data, then signs this with the signature scheme. To verify, the recipient does the same hash on the data, then runs the opposite encryption operation on the signature, using either the same key as the sender should have (see **symmetric encryption**) or the complementary public key (see **asymmetric encryption**). The output should match the hash if the signature is legitimate and the message has not been altered.

The **IPSec protocol suite** uses digital signature schemes for authentication and data integrity checks throughout the protocol suites.

Encryption — a scheme in which information is rendered unreadable during transport, so that an intercepting party cannot make use of it. In encryption, readable data (the **plaintext**) is encrypted to produce an unreadable **ciphertext**. The ciphertext must then be decrypted (converting it back to the plaintext) before it can be read at the receiving end.

Modern encryption schemes typically use an algorithm and key system. The algorithm is a set of rules for what to do to the data to encrypt it. The key is a parameter — a value the algorithm uses to encrypt the data. Once a certain key has been used to encrypt data, you usually need either the same key (symmetric encryption) or a different but complementary key (asymmetric encryption) to decrypt the data. Typically the rules of the algorithm used are public, but keys used are kept secret between communicating parties, so that only they can decrypt the message.

Standard public algorithms used for encrypting data include **DES**, 3-DES, **RSA**, CAST, and Blowfish.

Encryption key — a parameter used to encrypt and decrypt data. Once data has been encrypted using a given key, either the same key (in **symmetric encryption**), or a complementary key (in **asymmetric encryption**) is then needed to decrypt it. See also **keyspace**.

Encapsulating Security Payload (ESP) — payload format used in IPSec-compliant IP packets to carry encrypted and/or authenticated data, thereby preventing sniffing (eavesdropping) on the network between communicating nodes.

ESP—an IPSec term. See **Encapsulating Security Payload.**

Hash function — a cryptography term. A hash function is a subclass of the encryption functions used to assist **digital signature schemes**, and for cryptographic **authentication** schemes in general. Hashes used for such schemes have the property that their output is of constant length, whatever the input, and that it is extremely difficult to come up with a second input message that would produce the same hash. Therefore in digital signature schemes, rather than signing the message itself, the sending party usually first runs a hash on the message and signs the output (also colloquially called the hash) thus reducing the size of the signature and the amount of data that has to be sent. Hashes may also be keyed. A keyed hash uses a parameter (a key) much like an encryption key, and precisely the same key must be used to generate the same hash from the same data. So a keyed hash can provide authentication of the sending party's identity on its own, by proving they know the appropriate key.

Keyed hashes are similar to encryption functions, except that hashes are reducing processes, so you cannot get the original data back from the hash output. You can only verify that the hash is appropriate output from that data.

A hash is both a **pseudo-random function** (PRF) and a **one-way function**.

Header (or packet header**)** — a network term referring to a portion attached to the beginning of a packet to specify the protocol level data needed to process the rest of the packet — such as addresses to which to deliver it, which **SA rules** to use to decipher it, how to assemble it with the next packet, or which procedures to use to authenticate the packet.

See also **payload**.

IETF — see **Internet Engineering Task Force**.

IKE — part of the **IPSec protocol suite**. IKE is the current IPSec standard for **SA rules** negotiation, key management, and key exchange. IKE uses three modes — **Aggressive mode**, **Quick mode**, and **Main mode**. IKE stands for Internet Key Exchange. IKE was formerly known as **ISAKMP/Oakley**.

IKE SA — an IPSec term referring to an SA negotiated using **IKE**, strictly for the purpose of negotiating general purpose **IPSec SA**s.

Internet Engineering Task Force (IETF) — a multinational group of people working on Internet communications technology issues at the international level. The IETF coordinates working groups, including the **IPSec working group,** developing communications standards.

Internet Key Exchange — see **IKE**.

Internet Protocol (IP) — a network term referring to the basic transmission protocol, immediately above the physical protocols (frequently Ethernet or PPP) in the Internet, and a common standard in many large corporate and academic LANs and WANs. IP is the protocol responsible for delivering **packets** to their destination. Other, higher level protocols (typically **TCP**) are responsible for actually breaking the data into appropriate chunks for transmission, and reassembling them on delivery. IP is a highly flexible scheme designed to transparently negotiate communications between networks of differing capabilities, and for highly flexible, adaptive routing. **TCP** and **UDP** are the two protocols that most commonly call on IP's services. See also **network layers**.

IP is the only IP network protocol used universally throughout the net (and this applies to the Internet) for all communications. This is one of the reasons adding security at the IP level (see **IPSec protocol suite**) has such value.

Internet Security Association and Key Management Protocol — see **ISAKMP**.

IP — see **Internet protocol**.

IP address — a network term referring to a 32 bit address, usually written in four bytes (e.g. 123.145.156.178), used in IP networks to identify a node on the network. Packets are routed by **IP** to the appropriate machine on the network by reading the destination IP address in the packet header. Every machine on a network to which packets may be routed must have a unique **IP address**. See also **domain name**, **Internet protocol**, **IPv4**, and **IPv6**.

IPSec — acronym for Internet Protocol (IP) Security. See also **IPSec protocol suite** and **IPSec working group**.

IPSec protocol suite — a set of extensions to the **Internet Protocol (IP),** adding security services developed by the **IETF**'s **IPSec working group**. The suite consists of protocols for an authentication header (**AH**), encapsulating security protocol (**ESP**), and a key management and exchange protocol (**IKE**).

IPSec SA — term specific to this paper referring to a general purpose SA used for carrying IP data. See also **IKE SA**.

IPSec working group — a subcommittee of the **Internet Engineering Task Force (IETF)** dedicated to developing security extensions for the **Internet Protocol**; designers of the **IPSec protocol suite**.

IPv4 — **IP version 4**. The current standard for IP addresses and routing behavior. IPv4 addresses are 32 bits wide, and are most commonly described in four decimal-separated ordered octets (four numbers from 0-255). For example: 192.168.1.1, (which translates to 0xC0A80101 in hex) is an IP address.

IPv6 — **IP version 6**. The proposed next generation standard for IP addresses, incorporating IPSec security features and other additions. IPv6 addresses are 128 bits wide – four times as long as an IPv4 address, giving an address range that is 2^{96} — or about $8x10^{28}$ — times as large as that provided by IPv4. The most common expression of these (currently) is eight hexadecimal integers in the range 0x0000 to 0xFFFF, ordered and colon-separated. For example, 0001:0001:0001:0001:0001:0001:C0A8:0101 is an IPv6 address.

ISAKMP — the **Internet Security Association** and **Key Management Protocol**. A framework negotiation protocol on top of which **IKE** is designed. You may occasionally find IPSec types who refer to IKE as ISAKMP, but the terms aren't precisely interchangeable. IKE (formerly called **ISAKMP/ Oakley**) is technically an instantiation of ISAKMP, adding functionality for the specific purposes of IPSec key and protocol negotiation.

ISAKMP/Oakley — obsolete term for **IKE**.

Key — a cryptography term. See **encryption key.**

Keyspace — a cryptography term. The total range (or number) of possible **encryption keys** that might be used in a given encryption scheme. Using binary keys, the key space is always two raised to the power of the length of the key. So the keyspace for a 56 bit binary key is 2^{56}, or around $7.2x10^{16}$ (about 72 million billion possible keys). See also **encryption**.

Main mode — an IPSec term referring to the packet exchange protocol used in IKE phase one (in IKE) to negotiate an IKE SA. See also **Aggressive mode**, **Quick mode, IKE SA**.

Network layers — network layers can be discussed in different ways. The ISO Open System Interconnection (OSI) standard, an international standard for network layering, actually specifies seven network layers. However, it's fairly common to describe IP networks as having five layers:

- the physical layer
- the data link layer
- the network (IP) layer
- the transport layer
- the application layer

The physical layer is the actual wire on which the data travels. So Ethernet cable might be the physical layer in an IP network. In other places, the physical layer might be a phone line.

The data link layer is the protocol layer that actually handles the physical layer. So for that Ethernet cable, the data link layer would be the Ethernet protocol. On the phone line, the data link layer would be the PPP protocol.

The network layer is the layer in which **IP** does its work — routing and delivering packets between **IP addresses**.

The transport layer supports a number of higher level protocols running on IP that typically provide certain types of service. **UDP**, for example, delivers data which has to get there on time or not at all, such as live video and audio. **TCP**, by contrast, delivers data that can arrive late, but which must be intact (as in binary executable files).

Finally, the application layer is the actual program using the network for communications. An e-mail client program, for example, is part of the application layer.

The network layer concept simplifies a number of logistic hurdles in building networks — primarily by breaking up big problems into smaller ones. Effectively, each layer's protocol(s) encapsulate(s) the problems encountered in delivering data at that layer, so higher levels can just rely on the lower level protocol to get the job done.

Nonce — a random number usually used either for verification purposes, or for adding randomness to cryptographic key exchanges. In **IKE** exchanges, you send the other communicating party a nonce which they then must sign with their **digital signature** and send back, so you can verify their identity.

Oakley — the mechanism of key exchange/negotiation, used in **IKE**.

One-way function — a mathematical/cryptography term. A one-way function has the often useful property that it is difficult to find what the inputs were from the output alone. A **hash** is a one-way function.

Packet — a network term. A packet is the basic unit of transmission under **IP** (and virtually all network protocols). Data streams are broken into packets (small 'buckets of data') by the transmitting machine, passed through the network in pieces by **IP**, and then reassembled at the receiving end.

You will also see the term **datagram**. A **datagram** is a unit of data, and a **packet** is the physical thing on the wire. But the terms are usually used interchangeably, and for most purposes, this is both convenient and legitimate.

Packet header — see **header**.

Payload — a network term referring to the data portion of a packet following the header. See also **packet** and **header**.

PGP/Web of Trust — a cryptographic authentication scheme typically used by Internet e-mail users to authenticate the identity of the sending party, and the integrity of their message. The scheme uses a variant of the **RSA** asymmetric encryption system to sign data, and a 'Web of trust' system (in which groups of users vouch for each other's identity) to exchange public keys. See **digital signatures**.

PGP stands for Pretty Good Privacy.

Phase one exchange — an **IKE** exchange used to establish the initial **IKE SA**.

Phase two exchange — an **IKE** exchange used to establish the general-purpose **IPSec SA**, or to refresh keying material either for an IPSec SA or an **IKE SA**.

Plaintext — cryptographic term for a message before **encryption** and after decryption. You encrypt the plaintext to generate the **ciphertext**, which you then transmit through the unsecured channel. You then decrypt the ciphertext to get back the plaintext.

Protocol — a general term/network term (1) referring to a way of doing things; (2) in networking, the rules that govern how machines communicate. The **Internet protocol (IP)**, for example, defines the rules for machine addressing and for routing packets between machines in a network.

Pseudo-random function (PRF) — a mathematical/cryptography term. A pseudo-random function is a function in which a small change in the input may result in any magnitude of change in the output. Pseudo-random functions are deterministic — in that the same inputs always result in precisely the same output. However, without actually running the function, it can be difficult to know what will be the output. Random number generators and **hashes** are both pseudo-random functions.

RSA — the original and best-known asymmetric encryption scheme. RSA uses the product of two large primes as a public key, and values derived from those (secret) primes as a secret key. See **asymmetric encryption**.

Quick mode — an IPSec term referring to the packet exchange protocol used in IKE phase two in IKE to establish general purpose **IPSec SAs**, and to refresh keying material for **IKE SAs** or **IPSec SAs**. See also **Main mode**, and **Aggressive mode**.

SA — see **Security Association**.

SA rules — in the **IPSec protocol suite**, the set of algorithms, and keys, and rules for using them in a **Security Association** (SA).

Secure Sockets Layer (SSL) — a technology developed by Netscape, and now standardized, usually used to secure HTTP traffic between a web browser and a web server. Typically the technology in use in shopping-cart applications, it secures the server and client on a session by session basis, generally using widely-publicized certificates for identity verification.

Security Association (SA) — an IPSec term. In the **IPSec protocol suite**, (1) a dedicated secure virtual connection between two nodes; (2) a term casually used to refer to the **SA Rules**.

Secure virtual private network (secure VPN) — a secure private network using unsecured public networks as carriers. The **IPSec protocol suite** provides the capability of building secure VPNs within the context of larger, unsecured public **IP** networks such as the Internet. Users of the secure VPN may use their network as though it were a perfectly secure, isolated LAN, even though it is physically directly connected to unsecured public networks. See also **virtual private network**.

Secure VPN — see **Secure virtual private network**.

Security Parameters Index (SPI) — an IPSec term referring to an arbitrary 32 bit number which, in concert with a destination IP address, uniquely identifies a single **SA**. The headers of IPSec packets carry an SPI, which the recipient node then uses to look up the **SA rules** for a given incoming packet. A node may reassign an SPI to a new SA after the old SA expires, provided it waits long enough that it is unlikely packets from the old SA are still percolating.

SSL — See **Secure Sockets Layer**.

Symmetric encryption — a cryptographic term describing encryption schemes in which the same key is used both for encryption and decryption. Example **DES**.

SPI — see **Security Parameters Index**.

TCP — see **Transmission Control Protocol**.

Transmission Control Protocol (TCP) — an IP-specific network term. A general-purpose protocol designed for carrying data of virtually any type and any size though **IP**-based networks, and the standard protocol for most general-purpose communications in the Internet. TCP is said to 'run on top of' (see **Network layers**) IP, meaning it uses IP for actual delivery and addressing of the data. TCP effectively sets up a dedicated, but temporary channel between two nodes. An e-mail client would typically use TCP for carrying messages back and forth, to and from an e-mail server. Contrast **UDP**.

UDP — see **user datagram protocol**.

User Datagram Protocol (UDP) — an IP-specific network term. A general-purpose protocol used in **IP** networks in situations in which reliable delivery is not required. Like **TCP**, UDP runs "on top of" (see **Network layers**) IP, meaning it uses IP for actual delivery and addressing of data. Internet clients use UDP principally for communications with domain name servers, in resolving Internet addresses. Real-time audio and video traffic also frequently uses UDP. **IPSec's IKE** protocol suite is based on UDP. Contrast **TCP**.

Virtual private network (VPN) — a system in which a public network is used to carry the data of a private one, usually in such a way that the distinctions between the resulting virtual network and a true private LAN or WAN are invisible to the user. Note that the term VPN by itself however does not necessarily imply that the data carried is private (i.e. unreadable by other network users), as does the term **secure virtual private network** (secure VPN). However, the term VPN commonly refers to a secure VPN.

VPN— see **virtual private network**.

TimeStep Corporation, a Newbridge affiliate, is a leading provider of secure virtual private network (VPN) solutions for corporate intranets, extranets, and Internet remote access. Designed for large-scale business communications, TimeStep's award-winning PERMIT Enterprise product suite integrates secure VPN, access control, and authentication technologies in a single solution.

TimeStep Corporation	TimeStep Inc.	Tel: 1-800-383-8211
362 Terry Fox Drive	593 Herndon Parkway	Fax: (613) 599-3617
Kanata, Ontario	Herndon, Virginia	www.timestep.com
Canada K2K 2P5	U.S.A. 20170	info@timestep.com

62-00114-01 97-2552-01-00-B Printed in Canada.