# Simple-VPN: Simple IPsec Configuration

Shreyas Srivatsan [*]
Microsoft Corporation
shreyas@microsoft.com

Maritza Johnson
Columbia University
maritzaj@cs.columbia.edu

Steven M. Bellovin
Columbia University
smb@cs.columbia.edu

CUCS-020-10

**Abstract**

The IPsec protocol promised easy, ubiquitous encryption. That has never happened. For the most part, IPsec usage is confined to VPNs for road warriors, largely due to needless configuration complexity and incompatible implementations. We have designed a simple VPN configuration language that hides the unwanted complexities. Virtually no options are necessary or possible. The administrator specifies the absolute minimum of information: the authorized hosts, their operating systems, and a little about the network topology; everything else, including certificate generation, is automatic. Our implementation includes a multitarget compiler, which generates implementation-specific configuration files for three different platforms; others are easy to add.

## 1 Introduction

*It's perfectly appropriate to be upset. I thought of it in a slightly different way — like a space that we were exploring and, in the early days, we figured out this consistent path through the space: IP, TCP, and so on. What's been happening over the last few years is that the IETF is filling the rest of the space with every alternative approach, not necessarily any better. Every possible alternative is now being written down. And it's not useful.*

—Jon Postel, INET '98, Geneva

### 1.1 The Complexity Problem

The IPsec protocol [4, 26, 27] promised easy, ubiquitous encryption. Applications would not need to be changed and all of their traffic would be encrypted. Host-to-host encryption would become ubiquitous. Needless to say, this has not happened.

We assert that complexity — of both the specifications and implementations — is a major reason for this. IPsec is hard to implement and hard to configure, and hence hard to use. Implementations often do not interoperate because implementers understood the specification differently. Also, given different options, different implementers make different choices. The result is the same: a lack of interoperability.

A complex program/application is often, hard to configure. Implementers generally provide knobs to select or tweak every different option. All such knobs must be documented, which results

[*]Work done primarily while at Columbia University, currently at Microsoft Corp.

in configuration instructions that are quite incomprehensible. And to top this, these knobs take different names in different implementations. We know of several skilled individuals, with literally decades of experience with Unix, networking, system administration, and security, who have found IPsec configuration to be extremely complex and challenging. Working configurations (when they are achieved at all) take many trials, web searches for configuration guides, and (not infrequently) use of `tcpdump` and similar tools to see exactly what is being sent over the wire for any particular attempt.

The IPsec problem is not limited to ESP [3, 25, 23] or AH [2, 24, 22] (IPsec standards – background in Section 1.2); the associated key management protocol, IKE (Internet Key Exchange) [25, 21] has at least as much complexity. Further, public key authentication in IKE requires using and hence understanding X.509 certificates [10], which itself is a hideously complex endeavor. As if the latter were not hard enough, the entire subject of certificates is surrounded (and confounded) by a lot of mythology about Public Key Infrastructures (PKIs).

The result is inevitable: IPsec is little-used, with the notable exception of virtual private networks (VPNs) for road warriors (roaming clients). Even there — IPsec's most natural niche — it has been challenged by easier-to-use alternatives such as SSL VPNs. Though SSL VPNs have advantages on small form-factor devices like smartphones, where they secure traffic between applications, they do not scale to more diverse remote access requirements. IPsec VPNs are application-agnostic with all traffic between the end-points encrypted, and hence, provide greater flexibility for a full-scale deployment if only they had been easy to configure.

Complexity is a security problem too; in two different ways. First, if doing something securely is too difficult, people will discard the security rather than the functionality. Second, people will often think their networks are secure when they are not. Both of these cause trouble [42].

We have chosen to discard the complexity. Most options are never used. Making them available to a site administrator, even as options, requires that they be documented and understood. We therefore prefer to leave them out entirely. We rely instead on a strong set of defaults and a compiler that knows how to do complex things.

## 1.2 Background

IPsec is a protocol suite for securing IP communications. It is a dual mode, end to end security scheme that operates at the network layer and hence, is transparent to applications running on top of it.

- **Authentication Header (AH)** [2, 24, 22] This is used to provide *authentication* for IP packets excluding the mutable fields in the header like TTL. Figure 1 shows the IP datagram after applying AH.

- **Encapsulating Security Payload (ESP)** [3, 25, 23] ESP provides *confidentiality* and/or *authentication* for the IP packets. Figure 1 shows the IP datagram after applying ESP.

- **IKE and Manual Keying** Both end points in an IPsec connection need to know the secret values that are used for authentication and encryption. *Manual keys* can be decided upon between the two sides through an out-of-band mechanism. IKE — Internet Key Exchange — [16, 21] is a standard to setup a security association (shared security information to support secure communication) between the end points through means of negotiation.

- **Main Mode and Aggressive Mode** These are the two methods that can be used for the initial key exchange in IKE. Main mode provides complete security during the establishment

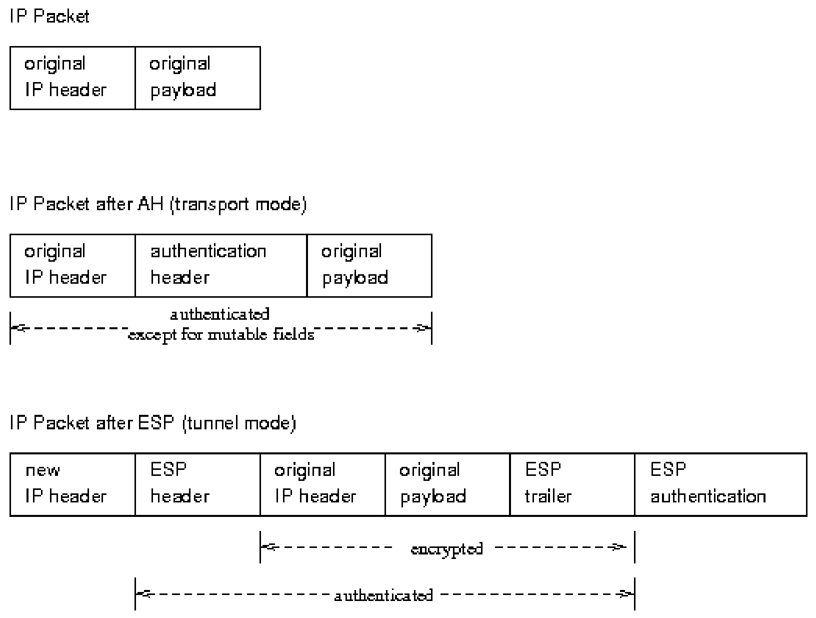

Figure 1: ESP and AH datagram structures, derived from [22, 23].

of an IPsec connection and requires 6 messages. Aggressive mode requires only 3 messages as it sends some messages unencrypted, and hence is less secure.

- **Tunnel Mode and Transport Mode** Transport mode is used for host to host communication. Only the payload of the IP packet is encrypted or authenticated. The routing is intact as the IP header is not changed. In tunnel mode the entire IP packet is encrypted and encapsulated in a new IP packet. Tunnel mode can be used for network-network, network-host and even host-host communications.

- **Road warriors** These are roaming clients who may connect to the VPN using random IP addresses.

IPsec is the dominant standard for VPNs. VPNs combine geographically and topologically distributed users and networks into a single network, and provide confidentiality and integrity to the traffic moving through this combined network. The tunnel mode of IPsec is used to implement VPNs. Gateways act as the link between the different networks that form the VPN. They also accept connections from road warriors who may connect from any random IP address.

## 1.3 Simple-VPN

Simple-VPN is an IPsec configuration compiler. It reads in a very simple configuration file and generates everything necessary. For the most part, where there are choices available in the IPsec or IKE specifications, we have made them, rather than leaving them to the site administrator. The resulting configuration may, to some, be suboptimal; it may consume more CPU time, bandwidth, etc., than a hand-built, site-customized alternative. We disagree. People are expensive (and insecurity even more so). Thanks to Moore's Law, CPU is incredibly cheap; bandwidth, it turns out, has been increasing even faster than CPU power.

We have opted for safe alternatives. Thus, we prefer AES over DES. We ignore AH and always use ESP with authentication enabled. (We feel that AH is an artifact of a time when ESP did not include authentication. It was concluded as early as 1995 that AH's additional protection of part of the IP header added nothing; accordingly, we have chosen not to support it.) Moreover, as we use tunnel mode with ESP we provide confidentiality and integrity to the whole packet.

Authentication is a more complex issue. There are significant real-world differences in authentication practices today. We prefer certificates and smart cards; that said, such an option is not available to the vast majority of sites whose users do not have smart card readers. Many sites prefer tokens such as the RSA SecurID® [39]; again that option is not available to everyone.

Our solution is two-fold. First, we use only public key authentication with IKE. It is more secure and avoids a serious bug in the original IKE specification.[1] Second, Simple-VPN automatically generates the necessary certificates and private keys, thus avoiding the PKI problem. (See Section 4.3 for more discussion of this issue.) Though we have not implemented it as part of this project, we prefer the philosophy espoused in [8]: use outboard means, protected by whatever authenticator is chosen, to obtain a dynamically-generated certificate and its corresponding private key if necessary.

## 1.4 Design Choices

If we want to eliminate the need for the user to select options, we need to make certain decisions. More precisely, we are taking certain choices away from the user. As noted earlier, we feel that the vast majority of networks do not need the full flexibility of IPsec. We choose secure alternatives without compromising performance.

- **AH** We do not use AH.
- **ESP** We always use 128-bit AES in CBC mode. 256-bit AES is generally an overkill.
- **Authentication** ESP authentication is always enabled, using HMAC-SHA1. SHA-2 variants are markedly slower; it is unclear how much better security they provide when used as part of HMAC.
- **Key management** IKE is always used with public key authentication.
- **Modes** We have opted for tunnel mode throughout. Tunnel mode can be used for host-to-host communication; transport mode cannot be used for host-to-gateway communication. For IKE, we use Main Mode exclusively as it provides complete security during the initial key exchange.
- **Selectors** We do not support selective encryption by port numbers, IP addresses, etc. The only exception we allow is how road warrior machines behave: when traffic is destined to a destination outside the VPN, it can either be sent directly or it can be sent to the VPN gateway machine. The latter option is more expensive in terms of bandwidth but provides the remote node with the protection of the organization's firewall.
- **Identities** Whatever name is given in the Simple-VPN configuration file is embedded in the certificates. Note that this may be a host name, an IP address, or a network descriptor.
- **Certificate Lifetimes** We use a one year lifetime for our certificates which is the default value used by most certificates used for IPsec purposes.

[1]IKE Main Mode cannot be used with preshared secrets and dynamic IP addresses. Dynamic IP addresses are the norm for many client computers, especially laptops.

- **Protocol timers** Some IPsec implementations permit the administrator to specify a wearisomely long list of timer values. For example, IPsec implementations provide timers for the maximum time phase 1 and phase 2 of IKE should take. We do not permit such things.

Note that we do not regard all of this flexibility as inherently wrong. In particular, we strongly support the concept of cryptographic algorithm agility; history suggests that over the lifetime of a protocol, people *will* need to use a different algorithm at some point. Indeed, lack of sufficient planning for new algorithms is itself problematic [9]. That said, we think that the choices we have made are the right ones for almost everyone today. We are generally willing to accept other, stronger selections, as long as these are not presented as site options; note, though, that we have to opt for a least common denominator: we cannot standardize on a single algorithm unless it is universally available. That, though, provides the upgrade path: newer algorithms generally come with upgraded versions of IPsec software; an upgraded version of this compiler will know of the improvements, and will automatically select them as soon as they appear on the platforms of interest.

### 1.5 Structure of the Paper

In Section 2, we describe our configuration language; space precludes giving a full language description, but it is only a few pages long. Section 3 describes the compiler implementation, with particular stress on retargetability. We also discuss how to support more features of IPsec, though in general we discourage doing so. Section 4 is an attempt to make future versions of this paper unnecessary: we profer advice on better protocol designs. Related work and concluding thoughts are in Sections 5 and 6.

## 2 The Simple-VPN Configuration Language

### 2.1 Language Goals

Our input language is designed to meet certain principles:

- **Simplicity** The aim is to produce something that is simple to use. A new input language that rivals the complexity of existing ones would be pointless, except perhaps as a way to have a common configuration language.

- **Completeness** Our aim is to generate a configuration language that will be complete, and will suffice for a majority of one's requirements. We support the most used features and leave only the most essential options to the user while making educated choices for the others based on performance and security considerations.

- **Extensible** The goal is to generate a simple language which can be extended to enable support for a richer feature set, if desired, and construct a compiler that is easily retargetable to different IPsec implementations.

### 2.2 Language features

Our main emphasis is to provide only the minimum required options for user choice. We identified the most important features that define a VPN, and used them to create a simple VPN configuration language. We create a strong set of defaults for each target OS and implementation supported.

This allows for easy interoperability between various OS and implementation combinations, by using the subset of options supported by them.

The only required configuration options are the topology of the VPN including the list of nodes that are a part of the VPN, and of course the target IPsec implementation. We provide a few knobs to the user to modify the configuration but these are optional. The node list can be given in the form of a list of hostnames, IP addresses or network descriptors. The compiler provides a simple interface that can be used to add to the configuration options.

The generation module provides functionality for generation of authentication tokens. It also generates scripts to install the configuration files and authentication tokens on the target systems. The generated certificates are only used for the purpose of authorization. If multiple OS/implementations are specified, the compiler selects from the common subset of the defaults. For example, implementation A may support DH groups 1, 2, 5 while implementation B supports DH groups 1, 5, 7. A configuration that will inter-operate with both implementations needs to use either DH group 1 or 5. As DH group 5 is more secure, Simple-VPN will select that as the default option. While selecting from a subset, we always default to (our perception of) the more secure alternative.

The generation module also generates the output topology of the VPN described in the Simple-VPN configuration language. It does so using Graphviz. This topology output allows the user to verify the topology of the VPN generated and match with the actual network topology.

## 2.3 Tutorial

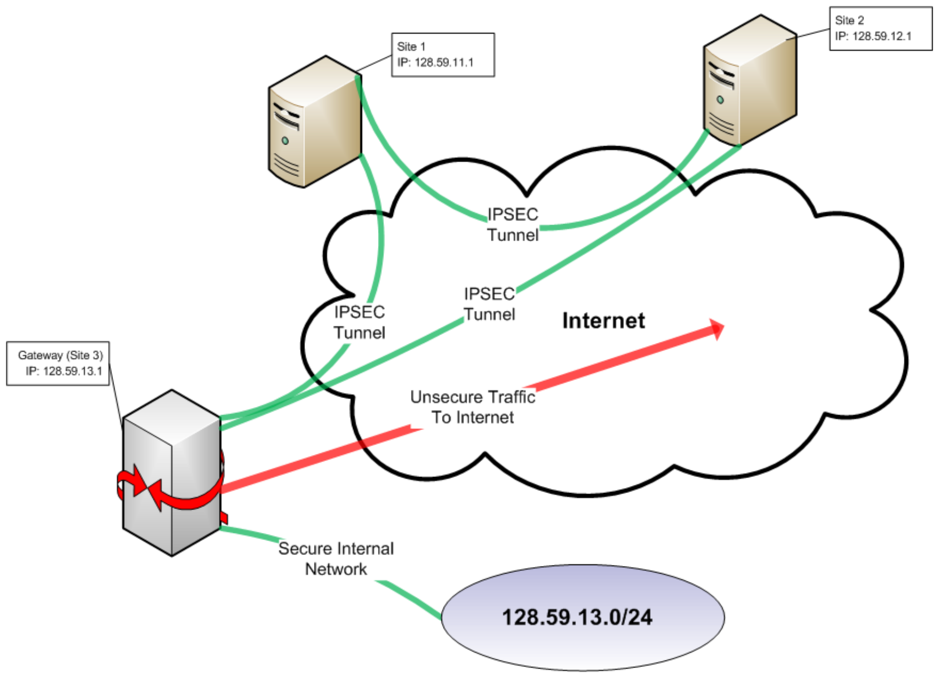

Figure 2: Topology of the network for the sample Simple-VPN configuration file for Config 1.

A single Simple-VPN configuration file can describe multiple VPNs. Each VPN can have multiple node lists. A node list is a set of nodes that share the same properties, like having the same OS, implementation or being a part of the same domain. Domain here refers to a string that is appended to all unqualified host names, thus saving the trouble of retyping the string constantly.

**Config 1** Sample Simple-VPN Configuration File

```
access "direct"   # No triangle routing
type "racoon"     # IPsec implemtnation
authgen           # Generate certficates automatically
vpn sample {
     nodes "ubuntu" {                    # OS for these nodes
          host 128.59.11.1, 128.59.12.1  # Some remote hosts
          gw 128.59.13.1{                # Gateway to these nodes
               subnet 128.59.13.0/24     # An entire protected net
          }
     }
}
```

These also contain information about gateways and nodes that are connected through the gateway. The nodes are specified through an IP address, hostname or a network descriptor. Configuration options can be defined globally, be associated with a VPN, or a node list. The options specific to a node list take precedence over options specific to a VPN, which in turn take precedence over any global options.

A sample Simple-VPN configuration file for the topology shown in Figure 2 is presented to explain the configuration. The sample configuration consists of a single VPN that has one gateway serving a subnet and two other nodes. A sample topology output generated using Graphviz is shown in Figure 3.

## 3 Implementation

### 3.1 Parsing

We created a simple extensible grammar that accepts the VPN definitions. The grammar has been structured to allow a simple human readable configuration file. The parser validates the configuration information and creates an intermediate representation, that is used by the generation component to generate the target configuration files.

The goal is to achieve enough flexibility so that most people can consider this a one stop solution,

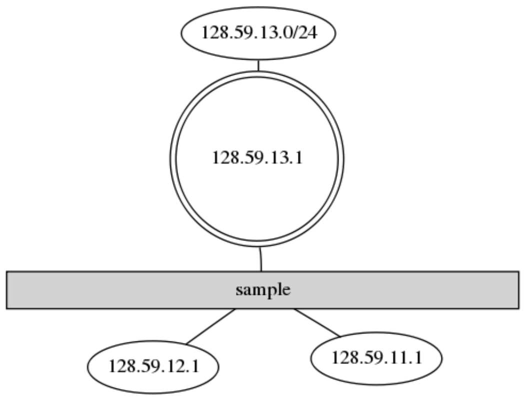

Figure 3: Graphviz generated topology for Config 1.

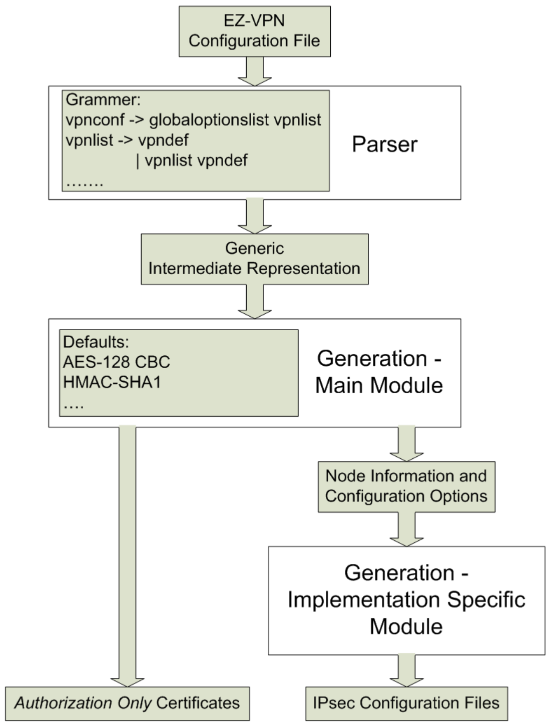

Figure 4: Architecture of Simple-VPN configuration file generation.

but not to create something that needs a lot of instructions to understand. The configuration file needs to be easy to understand and it should be possible to extend the grammar to incorporate new IKE features, if necessary. These goals drive the design decisions we take for the structure of our language.

The intermediate representation has the form of a custom hierarchical data structure that stores information regarding all the VPNs. At the highest level, this data structure holds a list of VPNs that have been defined in the Simple-VPN configuration file. The VPN definition consists of configuration options and the nodes (including gateways) that are a part of the VPN. The intermediate representation is augmented with a set of defaults, which are used for options not specified in the configuration file, and those that are not exposed to the user. The default options have been selected based on performance and security considerations, and also the most common requirements sites have.

To add to the choices mentioned in Section 1, all gateways support connections from authorized road warriors. According to the standards, IPsec and IKE SAs have lifetimes after which they need to be renegotiated. Some implementations permit the administrator to specify these lifetimes in the configuration file, specifically to ease inter-operation of multiple implementations. Many implementations set the IPsec SA lifetime at 1 hour. We choose this value as it will suit most configuration requirements. For the Phase 1 IKE SA lifetime we choose a default value of 8 hours which again is a standard with many IPsec implementations. The above are not settable in our configuration file.

## 3.2 Configuration File Generation

The configuration file generation component consists of two modules.

The first module or the main module consolidates the configuration options using defaults as necessary. It considers each pair of nodes that are present in the VPN, by walking the hierarchical intermediate representation, and calls the implementation specific modules, which generate the configuration file, using this information along with the consolidated configuration options. If a single VPN spans multiple OSes/implementations, the defaults selected by Simple-VPN consider the subset of options that are common to these. This is very important for seamless inter-operation of different implementations.

The main module provides functionality for certificate generation and scripts to install them based on the target operating system. These certificates are tied to the machine on which they are to be installed, rather than to a user, and as mentioned previously, act only as special purpose *authorization* credentials rather than general purpose *identifiers*.

The main module also provides functionality for output topology generation. It achieves this using Graphviz. This assists the user in understanding the topology for which the configuration files have been generated. We note that this is a relatively simple task in our system, since there is a single configuration file. In conventional implementations, with a separate configuration file on each machine, this is much more difficult; apart from the mechanical issue of collecting them all, a program would have to figure out if nodes mentioned in multiple files were identical. This can be difficult, especially if there were multiple instances of the same private address space [37] or other inconsistencies.

Each supported implementation will have an implementation-specific generation module. These are called by the main module, providing information regarding the nodes that are a part of the VPN and the configuration options. As a demonstration modules for Racoon [35], Openswan [33] and Strongswan [40] configuration file generation have been implemented. The implementation specific generation modules take the node pair information, along with the configuration options, and generate the configuration files as per the required format.

## 3.3 Operations

The Simple-VPN configuration process is very simple. Once the administrator knows the topology of the network and the patterns of communication, he uses it along with the host OS/IPsec type information to build the Simple-VPN configuration file. Type information may be drawn from an existing database.

Simple-VPN generates the configuration files and the certificates. These are stored in a hierarchical directory structure. Currently, as a demonstration we provide simple scripts that install these files on the target systems. We suggest the use of bulk management software (e.g., rdist [36], ASD [29], Microsoft Server Management System) to push the files out to each node. The destinations can be kept in sync with the newly generated files, so any update will be automatically pushed out to all the nodes. If a node's configuration is driven from its own database (something like a Cisco concentrator) as per [7], we can push the new files to that database.

For maintenance, or adding or deleting hosts, the Simple-VPN configuration file needs to be modified to reflect the latest topology. To revoke certificates, the administrator can delete the machine's certificate from the generated files, so a new one will be automatically built and the old will no longer appear, and hence will not be accepted.

## 3.4 Adding More IKE features

If it is necessary to support more IKE options — we hope not, but there may be such situations — the grammar and generation modules have been designed for easy addition of new features. The options are divided into three categories: global options that apply to all VPNs described in the Simple-VPN configuration file, VPN options that are limited to a VPN, and node options that apply to a node list in the VPN definition. Based on the scope of the IKE options they can be added to any of these option groups. Addition of more options of IKE will also involve adding information to the default options, and handling them accordingly in the respective generation module.

## 3.5 Adding New Platforms

The compiler has been designed to be retargetable. A new configuration generation module can interpret the generic intermediate representation, and utilize it to generate the configuration files.

As an example, we explored the addition of the Cisco VPN based on IPsec [41]. The Cisco VPN provides support for a subset of the encryption algorithms supported by IKE. The generated configuration files have additional options that are not present in the more basic generation modules of Racoon, Openswan or Strongswan that we have implemented. As is the general case, many of the options can be configured without requiring user choice. The Cisco Concentrators and the later Advanced Security Appliances (ASA) require us to define public and private interfaces. Some also provide interfaces to connect to additional LANs. Although we can provide this option through Simple-VPN, we believe that conceptually it is not needed as this is subsumed by the routing information. That said, we support platform-specific commands to supply such additional information, though as of now we have not defined any.

## 3.6 Useful extensions

A useful addition to our system would be the addition of version numbers to the implementation type, which would simplify supporting new versions of the implementations. Small changes to the implementation or its generated configuration files, such as the list of algorithms supported, might be table-driven; other changes might require entirely new code. Regardless, this is a property of the implementation-specific module; different implementations could handle this differently.

One more useful extension would be to allow different classes of defaults instead of having a single set of defaults for all users. Users can be provided with the option of different levels of security and hence, performance. These will govern the defaults options chosen, like the encryption algorithm used and the default SA lifetimes and hence, provide more flexibility to the end user.

# 4 Lessons Learned

*Everything should be made as simple as possible — but no simpler.*

—Albert Einstein

## 4.1 Protocol Design

The evaluation [14] of IPsec talks about the general properties that security protocols should conform to, and stresses the need for having simple protocols. It is necessary to be able to understand a protocol thoroughly before one can have a perfectly secure implementation. IPsec falls short in this respect due to its complexity and the difficulty in gaining a thorough understanding of the protocol.

Complex protocols targeting a variety of scenarios have an extensive list of configuration options and seldom work as they are meant to. This is because understanding the rationale behind each feature, and implementing the same correctly is almost an impossible task. Not only is implementing such a system difficult, it is also difficult to configure and manage it. The end user rarely understands each of the options described. This may lead to configurations with combinations of options that were not meant to be used together.

The necessity of encryption with authentication is well known. [6] talks about authenticated encryption. The encryption only-ESP configuration of IPsec which most implementations still allow is insecure. [34] shows attacks that recover the complete contents of IPsec protected datagrams, not only in this case, but also in some scenarios where it is used with AH authentication. Here the encryption and the integrity checking occur at different points, which opens the door to attacks. Unsuspecting end users may very easily configure their VPN with such options and believe their network is secure. Hence, having abundant options, can very easily lead to scenarios that were not taken into account while designing the protocol, and can adversely affect the security of the system. Such situations make it necessary that the configuration options be kept to a minimum. A protocol that targets a few scenarios and works well for them has higher chances of being a very secure protocol.

## 4.2 IPsec Implementions

One main reason for the complexity of IPsec is that it provides a framework, that allows two parties to use any set of algorithms they agree upon, rather than defining which algorithms should be used. With such an enriched feature set, getting different IPsec implementations to interoperate is not a very straightforward process. IPsec lacks de facto default standards that all implementations are expected to use. Though most IPsec implementations will necessarily have some common algorithms, figuring what these common algorithms are, is not always an easy task and, means going through pages of the configuration guide. This can especially be a long drawn process, when one needs to look at more than two implementations. Having a set of default standards that all implementations should use, will go a long way in allowing easy inter-operability.

Along with the complexity of IPsec, the associated mechanisms used by IPsec come with their own challenges. Deploying Public key authentication means using and understanding X.509 certificates; that, coupled with PKIs, makes it a very challenging endeavor. We have made this process much easier by having *authorization only* certificates and handling the certificate generation completely through Simple-VPN. We discuss this further in Section 4.3.

It can be seen that over a period of time IPsec implementations have evolved and become more secure and complete. It is far easier to manage multiple implementations in an organization now than was possible a few years ago. This can also be attributed to the improvements to IPsec through the revamped standards in [27] and the improvements in IKEv2 through [21]. More drastic changes, though desirable, are infeasible due to interoperability concerns with older versions of the protocol. We believe we can make secure IPsec configuration an easy task by providing an abstraction, that has been developed with a thorough understanding of the underlying systems, and making choices for the various options with proper security considerations.

## 4.3 The PKI Myth

Public key infrastructures (PKIs) are surrounded by a great mystique. Organizations are regularly told that they are complex, require ultra-high security, and perhaps are best outsourced to competent parties. Setting up a certificate authority (CA) requires a "ceremony", a term with a technical meaning [13] but nevertheless redolent of high priests in robes, acolytes with censers, and more. This may or may not be true in general; for most IPsec uses, however, little of this is accurate. (High priests and censers are definitely not needed; we are uncertain about the need for acolytes. . . )

Much of the mystique is due to the general-purpose nature of PKIs and certificates. If a certificate is intended to attest to a person's identity, a lot of process may be necessary. The real danger from a compromised root key comes from the attacker's ability to create arbitrarily many fraudulent credentials. As we shall see, however, the actual risks when using IPsec are considerably less.

The complexity stems from a familiar source. If a PKI can serve many different purposes, the software must be very flexible. If a certificate is used for identification, its format must accommodate many different forms of identity, ranging from a person's name and physical address to a particular role within a large organization to a computer with a given domain name and IP address. Understanding how to use PKI software packages, such as OpenSSL, is every bit as complex as understanding how to configure IPsec.

The philosophical basis for our solution rests in a simple observation: the certificates we will use are not general-purpose *identifiers*; rather, they are special-purpose *authorization* credentials. They are used only for IPsec within a given domain; they confer no other privileges. If such a certificate is compromised, the attacker gains no other abilities. If the root key is compromised, all the attacker can do is create fraudulent VPN endpoints, but this is the same ability an attacker would gain from compromising the shared secret database in a non-public key IPsec implementation. In fact, the public key solution is more secure, since at least in principle a CA can operate offline; a shared secrete database must be accessible online.

Dealing with compromised private keys is difficult, even conceptually [38]. We have not implemented any mechanism for certificate revocation. At least for small installations, the simplest solution is to delete the compromised key (or node) from the VPN definition, rerun the compiler, and push out new databases to all nodes within the VPN. This scheme will not scale to large VPNs, so some form of certificate revocation list (CRL) server or online certificate status protocol (OCSP) [32] server would likely be needed. While adding a suitable command to the configuration language is easy, doing so would require the administrator to install, configure, and run appropriate software on that machine. We fear this is no simpler than running any other form of standard PKI software, but reserve this problem for future work.

The notion of using certificates for authorization rather than identification is not new, of course. [28] suggests it for both security and privacy. SPKI [11, 12] in fact abandons any pretense of identification, and uses certificates solely for authorization. Roughly speaking, if you possess a certificate and the corresponding private key, you are entitled to certain forms of access; your identity is irrelevant. Such certificates are analogous to cash; conventional identity-linked certificates are more like checks.

## 5 Related Work

Complexity concerns surrounding IPsec and IKE are long standing. Indeed, the complexity of ISAKMP [31] (the protocol framework for IKE) was raised quite often in the IETF's IPsec working group. It was only adopted when Photuris [20, 19] and SKIP [5] were rejected.[2]

[2]The saga of Photuris, SKIP, and ISAKMP was composed of equal parts technical considerations, organizational politics, personalities, and more. The details are not relevant to this paper. It is undisputed, though, that Photuris and SKIP were significantly simpler than ISAKMP.

An independent evaluation of IPsec [14] came to similar conclusions about IPsec. They found flaws in all of IPsec, not just IKE:

> Our main criticism of IPsec is its complexity. IPsec contains too many options and too much flexibility; there are often several ways of doing the same or similar things. . . As we all know, this additional complexity and bloat is seriously detrimental to a normal (functional) standard. However, it has a devastating effect on a security standard.

They also noted that "from various discussions with the people involved, we learned that virtually nobody is satisfied with the process or the result".

IKEv2 [21] is a replacement for IKE. Both it and the other replacement candidate, JFK [1], are significantly simpler than IKE. Perhaps not surprisingly, given the overlap in authorship, JFK shares many philosophical similarities to this work.

More generally, the ill effects of complexity on the successful use and even the adoption of cryptography have long been known. The classic reference in computer science is "Why Johnny Can't Encrypt" [42]; a follow-up study [15] came to similar conclusions. A review of the history of cryptography shows similar concerns; see, for example, [30]. Kahn's monumental history of cryptography [17] makes similar points. For that matter, the difficulty the Germans had in correctly using their Enigma machine was a crucial factor in the British success in cracking it [18].

## 6 Conclusions

In this paper, we present Simple-VPN an IPsec configuration compiler that tackles the complexities of IPsec configuration for VPNs. The IPsec protocol, though having the advantage of operating transparently to applications running in the host system, has never taken off; one of the main reasons being the complexity of the protocol. This complexity has also led to interoperability problems between the various implementations of the protocol. Public key authentication used by IKE brings with it its own set of complexities.

We solve the problem by creating a compiler that hides all these complexities from the end user. Based on security and performance considerations and the requirements of most people today, we make educated choices for most of the options provided, allowing the user to make very few choices. We also argue that public key authentication becomes complex when the certificates used are multi-purpose. But in the situation where they are used solely for the purpose of authorization, and provide no other privileges, the stakes are not as high. Simple-VPN generates *authorization only* certificates which are tied to the machine that is part of the VPN, rather than to the identity of the user. This also allows for easy deployment of the VPN. Simple-VPN has been implemented as an extensible language, which can allow easy support for newer features and IPsec implementations.

## References

[1] William Aiello, Steven M. Bellovin, Matt Blaze, Ran Canetti, John Ioannidis, Angelos D. Keromytis, and Omer Reingold. Just fast keying: Key agreement in a hostile Internet. *ACM Transactions on Information and System Security (TISSEC)*, 7(2):1–32, May 2004.

[2] R. Atkinson. IP Authentication Header. RFC 1826, Internet Engineering Task Force, August 1995.

[3] R. Atkinson. IP Encapsulating Security Payload (ESP). RFC 1827, Internet Engineering Task Force, August 1995.

[4] R. Atkinson. Security Architecture for the Internet Protocol. RFC 1825, Internet Engineering Task Force, August 1995.

[5] Ashar Aziz, Tom Markson, and Hemma Prafullchandra. Simple key-management for Internet protocols (SKIP). Internet draft; work in progress, December 21, 1995. (draft-ietf-ipsec-skip-06.txt).

[6] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. Cryptology ePrint Archive, Report 2000/025, 2000. `http://eprint.iacr.org/`.

[7] Steven M. Bellovin and Randy Bush. Configuration management and security. *IEEE Journal on Selected Areas in Communications*, 27(3):268–274, April 2009.

[8] Steven M. Bellovin and Robert G. Moskowitz. Client certificate and key retrieval for IKE. Obsolete Internet draft, November 2000. `http://www.cs.columbia.edu/~smb/papers/draft-ietf-ipsra-getcert-00.txt`.

[9] Steven M. Bellovin and Eric K. Rescorla. Deploying a new hash algorithm. In *Procedings of NDSS '06*, 2006.

[10] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280, Internet Engineering Task Force, May 2008.

[11] C. Ellison. SPKI Requirements. RFC 2692, Internet Engineering Task Force, September 1999.

[12] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. SPKI Certificate Theory. RFC 2693, Internet Engineering Task Force, September 1999.

[13] Carl Ellison. Ceremony design and analysis. IACR eprint archive 2007/399, 2007.

[14] Niels Ferguson and Bruce Schneier. A cryptographic evaluation of IPsec, 2003.

[15] Simson L. Garfinkel and Robert C. Miller. Johnny 2: a user test of key continuity management with S/MIME and Outlook Express. In *SOUPS '05: Proceedings of the 2005 symposium on Usable privacy and security*, pages 13–24, New York, NY, USA, 2005. ACM.

[16] D. Harkins and D. Carrel. The Internet Key Exchange (IKE). RFC 2409, Internet Engineering Task Force, November 1998.

[17] David Kahn. *The CodeBreakers*. Macmillan, New York, 1967.

[18] David Kahn. *Seizing the Enigma: The Race to Break the German U-Boat Codes, 1939–1943*. Houghton Mifflin, Boston, 1991.

[19] P. Karn and W. Simpson. Photuris: Extended Schemes and Attributes. RFC 2523, Internet Engineering Task Force, March 1999.

[20] P. Karn and W. Simpson. Photuris: Session-Key Management Protocol. RFC 2522, Internet Engineering Task Force, March 1999.

[21] C. Kaufman. Internet Key Exchange (IKEv2) Protocol. RFC 4306, Internet Engineering Task Force, December 2005.

[22] S. Kent. IP Authentication Header. RFC 4302, Internet Engineering Task Force, December 2005.

[23] S. Kent. IP Encapsulating Security Payload (ESP). RFC 4303, Internet Engineering Task Force, December 2005.

[24] S. Kent and R. Atkinson. IP Authentication Header. RFC 2402, Internet Engineering Task Force, November 1998.

[25] S. Kent and R. Atkinson. IP Encapsulating Security Payload (ESP). RFC 2406, Internet Engineering Task Force, November 1998.

[26] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. RFC 2401, Internet Engineering Task Force, November 1998.

[27] S. Kent and K. Seo. Security Architecture for the Internet Protocol. RFC 4301, Internet Engineering Task Force, December 2005.

[28] Stephen T. Kent and Lynette I. Millett, editors. *Who Goes There? Authentication Through the Lens of Privacy*. National Academies Press, 2003.

[29] Andrew Koenig. Automatic software distribution. In *USENIX Conference Proceedings*, pages 312–322, Salt Lake City, UT, Summer 1984.

[30] Leo Marks. *Between Silk and Cyanide: A Codemaker's War, 1941–1945*. Free Press, New York, NY, 1998.

[31] D. Maughan, M. Schertler, M. Schneider, and J. Turner. Internet Security Association and Key Management Protocol (ISAKMP). RFC 2408, Internet Engineering Task Force, November 1998.

[32] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. RFC 2560, Internet Engineering Task Force, June 1999.

[33] Openswan. http://www.openswan.org/.

[34] Kenneth G. Paterson and Arnold K.L. Yau. Cryptography in theory and practice: The case of encryption in ipsec. Cryptology ePrint Archive, Report 2005/416, 2005. `http://eprint.iacr.org/`.

[35] Racoon. http://ipsec-tools.sourceforge.net/.

[36] RDist. http://www.magnicomp.com/rdist/.

[37] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear. Address Allocation for Private Internets. RFC 1918, Internet Engineering Task Force, February 1996.

[38] Fred B. Schneider, editor. *Trust in Cyberspace*. National Academy Press, 1999.

[39] RSA SecurID. http://www.rsa.com/node.aspx?id=1156.

[40] Strongswan. http://strongswan.org/.

[41] Cisco Systems. Vpn 3000 series concentrator reference.

[42] Alma Whitten and J.D. Tygar. Why Johnny can't encrypt: A usability evaluation of PGP 5.0. In *Proceedings of Usenix Security Symposium*, 1999.

## A Language Reference Manual

### A.1 EZ-VPN Grammar

We present the complete grammar and explain the various options provided. We feel that the current grammar is sufficient for most users, but we have designed it in a way that it is easy to allow support for newer features in the future.

**The EZ-VPN Configuration Language**

| | | |
|---|---|---|
| **vpnconf** | → | globaloptionslist vpnlist |
| **vpnlist** | → | vpndef |
| | | vpnlist vpndef |
| **nodelist** | → | nodedef |
| | | nodelist nodedef |
| **vpndef** | → | VPN WORD '{' vpnoptionslist nodelist '}' |
| **nodedef** | → | NODE OSNAME '{' nodeoptionslist nodes '}' |
| **nodes** | → | nodetypedescription |
| | | nodes nodetypedescription |
| **nodetypedescription** | → | HOST nodeaddresslist |
| | | GATEWAY nodeaddress '{' gwdef '}' |
| **nodeaddresslist** | → | nodeaddress |
| | | nodeaddresslist ',' nodeaddress |
| **gwdef** | → | gwoptions |
| | | gwdef gwoptions |
| **nodeaddress** | → | IP |
| | | HOSTNAME |
| | | WORD |
| **gwoptions** | → | HOST nodeaddresslist |
| | | NET subnetlist |
| **subnetlist** | → | SUBNET |
| | | subnetlist ',' SUBNET |
| **globaloptionslist** | → | globaloptions |
| **vpnoptionslist** | → | vpnoptions |
| **nodeoptionslist** | → | nodeoptions |
| **globaloptions** | → | globalcommands |
| | | globaloptions globalcommands |
| **vpnoptions** | → | vpncommands |
| | | vpnoptions vpncommands |
| **nodeoptions** | → | nodecommands |
| | | nodeoptions nodecommands |
| **globalcommands** | → | vpncommands |

**The EZ-VPN Configuration Language – continued**

| | | |
|---|---|---|
| **vpncommands** | → | AUTHGEN AUTHTYPE |
| | | ENCRYPTIONALGORITHM ENCALGO |
| | | nodecommands |
| **nodecommands** | → | DOMAIN HOSTNAME |
| | | ACCESS ROUTINGMODE |
| | | TYPE TYPENAME |
| | | TYPE TYPENAME '{' platformspecoptions '}' |
| **platformspecoptions** | → | |

## A.2 Configuration Commands

### A.2.1 Options

**domain** `domain_suffix`

The **domain** command is a form of syntactic sugar. It is a string appended to all unqualified host names, thus saving the administrator the trouble of retyping the string constantly. **domain** may appear as a global, VPN, or node option.

**type** `IPsec_implementation [{platform_commands...}]`

**type** specifies the implementation type for all nodes in its scope. It may appear as a global, VPN, or node option.

Some target platforms require platform-specific commands that cannot be intuited by the compiler. If a { follows the `IPsec_implementation`, a new scope is created for such commands. None are currently defined.

**access** `access_type`

**access** selects between `direct` and `triangle` routing. With `direct` access, traffic from a node to a non-VPN destination is sent directly. If `triangle` is specified, all traffic is routed to the VPN gateway; forwarding from there to non-VPN destinations is at the whim of other policies. **access** may appear as a global, VPN, or node option.

**authgen** `[auth_mechanism]`

**authgen** causes what type of authentication token should be generated by the compiler. The default if `auth_mechanism` is not specified is `certificate`, in which case standard X.509 certificates and their corresponding private keys are produced. If `none` is specified, certificates are presumed to be provided by private mechanisms. **authgen** may appear as a global or VPN option.

**encryption** `encryption_algorithm`

**encryption** specifies what encryption algorithm should be used. `AES128CBC`, the default if **encryption** is not given, is the only value supported. **encryption** may appear as a global or VPN option.

Implementation note: **encryption** is provided primarily as an example of how to extend the compiler. Since it cannot be used to specify any non-mandatory values, it is not anticipated that it will ever appear in any real configuration files.

### A.2.2 Scoping Commands

**vpn** `vpn_name`

**vpn** defines a named virtual private network. Nodes within the scope of this command will talk to each other via IPsec. All other destinations are outside the VPN; routing to them is controlled by the **access** command. The `vpn_name` has no semantic significance. The **vpn** command may occur only at global level.

**nodes** `os_type`

**nodes** specifies the operating system for a group of VPN nodes. The values of `type` permissible are `IPsec_implementation`-specific. **nodes** may only occur at VPN level.

### A.2.3 Node Commands

**host** *`host_list`*

**host** specifies a list of hosts within the VPN. A *`host_list`* is a comma-separated list of host names or IP addresses in standard dotted quad notation. **host** may only appear at node level.

**subnet** *`subnet_list`*

**subnet** specifies a list of subnets within the VPN. A *`subnet_list`* is a comma-separated list of network adddresses: a dotted quad with "/`prefix_length`" appended. **subnet** may only appear as a part of gateway definition at node level.

**gw** `gateway` {*`host_list`*`,` *`subnet_list`*`, ...`}

**gw** specifies the gateway to a group of other hosts or networks. All traffic intended for *`host_list`* and *`subnet_list`* is sent to `gateway` via IPsec; it, in turn, must forward it to the listed destinations. **gw** may only appear at node level.