EasyVPN: IPsec Remote Access Made Easy

Mark C. Benvenuto, Angelos D. Keromytis
{markb,angelos}@cs.columbia.edu
Computer Science Department, Columbia University, USA

Abstract

Telecommuting and access over a Wireless LAN require strong security at the network level. Although IPsec is well-suited for this task, it is difficult to configure and operate a large number of clients. To address this problem, we leverage the almost universal deployment and use of web browsers capable of SSL/TLS connections to web servers and the familiarity of users with such an interface. We use this mechanism to create configurations and certificates that will be downloaded to the user's machine and be used by a program to perform all configuration on the user's system.

Our system builds on common security protocols and standards such as IKE, X.509, and SSL/TLS to provide users with a secure-access environment that "just works." One of the main goals of the system is ease of use both for the users and the system administrators that maintain the infrastructure. We describe our implementation that uses Linux FreeS/WAN and Windows to show the practicality of the approach.

1 Introduction/Purpose

The growth of untrusted networks, *e.g.*, the Internet and Wireless LANs, has created a problem for system administrators who need to provide secure access to corporate LANs for their users. While these problems superficially seem different, they can both be solved by using a virtual private network.

Native support for IPsec [12] in most operating systems allows most groups to create their own IPsec VPN solutions. Operating systems such as Windows 2000/XP, Linux, OpenBSD, and others support the ability to be IPsec clients and servers. While these systems are powerful and flexible, the configuration is often confusing and time-consuming. A tool that simplifies configuration is particularly important, because it allows system administrators to enforce policy in a transparent fashion, and it prevents user misconfiguration problems which are time consuming and expensive to fix.

The proposed framework is designed to eliminate those problems while remaining adaptable to the requirements of different sites. Because of the interoperability of IPsec, the server platform can be any platform, such as Linux or OpenBSD. The client platform chosen was Windows 2000/XP, because it has native IPsec support unlike Windows 9x/ME, and because it is the most common commodity OS platform. Our tool, like most VPN systems, models a remote access server (RAS) that is both simple and familiar to most users.

2 Background

Using IPsec for Remote Access is becoming important, but requires additional extensions and application support to become truly useful. IPsec is an appealing technology for VPN, remote access, and other uses because it is a proven standard that can be used as a trusted component in a larger security infrastructure.

The concluded IETF Working Group for IP Security Remote Access (IPSRA) [7] investigated the application of IPsec for Remote Access. This resulted in the PIC [14] proposal based on ISAKMP, and the (now expired) GetCert work [3]. The IPSRA working group also produced a set of requirements for IPsec Remote Access [16] to serve as guidelines for any standard. These guidelines divide remote access into a few categories, according to requirements such as location and level of security. This paper presents a system designed to fulfill the requirements for the "Telecommuters (Dialup/DSL/Cablemodem)" scenario.

The proposed GetCert uses a similar design to this paper, but it differs in that it uses a subset of the Simple Certificate Enrollment Protocol (SCEP) [13] as the certificate protocol which has support for root certificate, and CRL retrieval. Authentication is handled using a RADIUS server over a HTTP/TLS connection. This authentication scheme supports a variety of authentication techniques, but the protocol has no support for exchanging IPsec policy information with the client.

One early solution to the remote access problem was Moat [4], which was created by AT&T Laboratories and the FreeS/WAN project. Instead of requiring the client machine to run VPN software, Moat is a VPN/NAT appliance that contains an *x86* machine running Linux and FreeS/WAN designed to provide a dedicated VPN to a pre-configured corporate network. While it is a good solution for telecommuters who always work from the same place, it does not help in the "roaming salesman" access scenario. Also, because of its dedicated hardware and configuration, it is difficult to upgrade the software or use at other locations.

Another way to create IPsec tunnels is by using DNS KEY records such as the FreeS/WAN Opportunistic Encryption (OE). This system will initiate IPsec connections with a potential host that has KEY and TXT records in its forward and reverse DNS zone files. This system makes creating tunnels easy as the system will make tunnels to any machine it can without any user involvement after the initial setup. The difficulty with this system is that it requires access to a forward domain to initiate connections, and the reverse DNS for it to be able to receive connections. These are steep initial setup requirements for inexperienced users.

Every time a connection is made to a new machine, the originator will have to do a reverse DNS lookup for the destination host to determine if a new connection can be made, a process that can be time consuming. Because of the need for frequent DNS lookups, the documentation recommends using a local caching DNS server for performance reasons. Finally, it is difficult for the user to determine which connections are protected without using FreeS/WAN status commands.

Wireless LANs (WLANs) have become increasingly popular in the last few years. Developed by the IEEE in standard 802.11, the WLANs have little security due to weaknesses in the wireless encryption [18], and other security features of the protocol. Because of the lack of security at the link-level, application and/or network level security protocols are used to address the problem, such as SSH, SSL/TLS, and IPsec. WLAN also lacks the benefits of some of the inherent physical security associated with wired networks since anyone in range of a wireless Access Point can potentially sniff network traffic or launch attacks.

One solution to the WLAN authentication problem was developed by the NASA Advanced Supercomputing Division. The "Wireless Firewall Gateway" (WFG) [1] serves the dual role of protecting the corporate LAN via a firewall and providing network configuration to clients via DHCP.

The WFG is an OpenBSD PC which runs a web-server that authenticates users over HTTPS against a RADIUS server with MD5 digest encryption, and then modifies firewall rules to allow client machines through the firewall. This machine relies on providing DHCP to the wireless clients so that it can be notified as clients connect and disconnect from the network. When a client releases or loses its DHCP lease, the WFG machine will remove the associated firewall rules. The system does not provide any additional protections to the clients because it is designed to protect access to the corporate network. WFG uses mutual authentication using an SSL certificate signed by VeriSign so clients can be reasonably assured that they are connecting to the server.

The power of this system is the hierarchy of security levels that are allowed. It is configured so that unauthenticated clients can be given limited access to only use email, VPN, and web access, while authenticated clients are given full access. The authentication relies on a modified ISC DHCPv3 server that will automatically update firewall rules as clients disconnect. One issue is that there is a small window of time between when a client that disconnects without properly notifying the DHCP server, and the DHCP server terminating the lease. In this window of time, it is possible for another client to spoof the identity of the recently disconnected user, and therefore gain authenticated access. Finally, this system relies on the user to properly use the network connection, which does not always happen as users often will use insecure protocols such as unencrypted email over the unsecured wireless network. This system is similar to the freely distributed NoCatAuth project from the NoCatNet project (`http://nocat.net/`).

WAVElan SECurity using IPsec (WAVEsec) (`http://www.wavesec.org/`) is a system designed to secure WLAN traffic by creating IPsec tunnels to a gateway on a local LAN. This system was designed for conferences to provide encrypted connections over WLAN to the public Internet. It does not authenticate users and requires custom software to configure connections. It provides initial IPsec connection information over DHCP packets such as the gateway IP. The server then sends its public key to the DHCP client as part of a dynamic DNS update request. Finally, the client creates the tunnel using the provided information, and the public key for the gateway from a DNS KEY record.

This system requires modifications to both the DHCP client and server to exchange configuration information and keys. It is designed to make it easy to create tunnels without exchanging keys or configuration out-of-band after the software is setup. It is designed to be used by experienced Linux users since the client setup requires installing custom scripts. Finally, this system is designed to only provide secure unauthenticated connections.

3 Problem

IPsec is a good solution for both remote access and wireless LAN access. Those two domains are similar enough that a common solution is feasible, without introducing undue complexity. This allows a system administrator to use a common set of software and hardware to provide IPsec for both sets of clients while being able to add additional security features as needed in the form of IPsec policies, firewalls, and DHCP servers, depending on the domain.

IPsec will only be practically useful to users if they can simply just use it without worrying about how it works. Ideally, users will be presented with some login screen where they will provide authentication information which is passed to an authentication server, and then the client machine will create a tunnel with the IPsec server. This ideal system will fit the two conflicting goals of providing a secure connection, yet be easy to use. The user should be confident that he is connecting to the correct server without danger of man-in-the-middle or impersonation attacks.

A current problem with IPsec is the many different implementations which all implement a different set of features for Internet Key Exchange (IKE) [11], and have distinctly different ways of configuring them. Implementations support some or all of the following authentication methods: private shared key, RSA public/private key, X.509, and manual keying. Many implement private key exchange, but this can be difficult to manage as the number of distinct users grows, difficult to work with in dynamic environments, and requires out-of-band cooperation from both client and server to setup. X.509 certificates solve many of these problems, but also require cooperation to configure.

The biggest problem in the configuration process is configuring the correct set of certificates to use, especially for someone who is not familiar with the details of X.509 certificates [2] and the intricacies of the different implementations. One of the authors spent many hours dealing with certificate problems in order to make an IPsec tunnel. There problems provided good evidence of the problems people may have and motivation to solve them. The complexity in the process can be eliminated with a client-side tool that will do all the certificate configuration and additional sanity checking as needed. On the server side, scripts can validate the setup with useful and instructive error messages about incorrect certificates without requiring that the system administrator become an IPsec expert to diagnose the logs. Such tools will only check each system in isolation for configuration problems but not large end-to-end problems such as firewalls.

To verify the security of the network configuration, the system administrator should use a tool such as IPSEC-validate [17] or other verification approaches [5]. Verifying the actual configuration is an important part of setting up an IPsec solution so users can correct security lapses, and it prevents unwanted access to the corporate network.

Other important features for system administrators include easy manageability, such as a system that will warn users about expiring tickets and performance metrics so usage growth can be anticipated. SNMP is the traditional mechanism used to monitor systems and MIBS for both IKE [6] and IPsec [8] [9] are in development.

4 Architecture

The system architecture, shown in Figure 1, is composed of three systems: client, gateway, and VPN Server. In our approach, we leverage the almost universal deployment and use of web browsers capable of SSL/TLS connections to web servers, and the familiarity of users with such an interface. We use this mechanism to create configurations and certificates that will be downloaded to the user's machine. This information is used by a program that will perform all IPsec configuration on the user's system using the server-provided configuration. The client and servers do all authentication via X.509 certificates in both IKE and SSL/TLS.

The *client* is the IPsec client that will create a certificate with the gateway, retrieve IPsec tunnel configura-

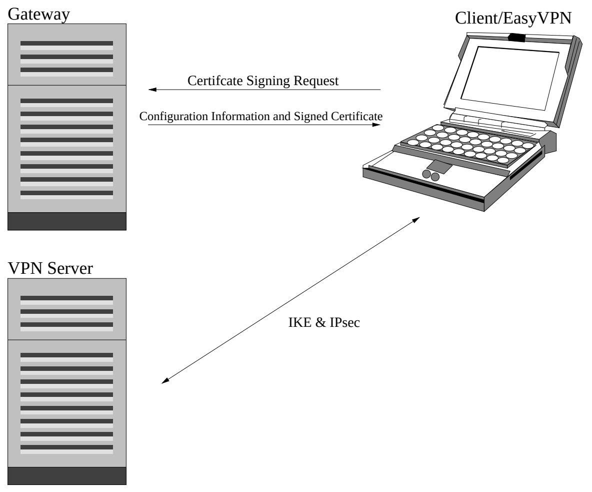
Gateway
Client/EasyVPN
Certifcate Signing Request
Configuration Information and Signed Certificate
VPN Server
IKE & IPsec

Figure 1: System Architecture

tion information, and then create the IPsec tunnel with the VPN Server. The client program will generate a certificate signing request (CSR) and private key. The client chooses the gateway either by a hard-coded IP or hostname, or via an SRV DNS record for the destination domain. An SSL/TLS connection is then made to the gateway machine that receives the user's username, password, and CSR as part of a HTTP POST operation. The client then receives the signed certificate back in a HTTP response, with configuration information indicating various network parameters such as the tunnel destination IP address and network mask. The client is responsible for initializing the IPsec connection to the server with these network parameters and the freshly-minted certificate.

There are two important details. First, the SSL/TLS connection establishes the identity of the gateway server either via the system certificate store in the case of Windows or via an embedded certificate in the application. If a system certificate store is used, then either a third-party certificate authority such as VeriSign or a local certificate authority must sign the gateway certificate. In the second case, either the user must configure the new CA or the application can configure the CA for the user. If the application needs to configure the certificate, then the CA certificate needs to be embedded in the application, which requires the user to be able to initially obtain the client in a secure manner such as a floppy disk from friends or via a trusted network before trying remote access. Second, the configuration information is extensible. Currently, it is a simple text format, but other formats can be used, such as XML. There are no restrictions on the content of the policy passed to the client. Useful information could be preferred ciphers, a message of the day, or other additional parameters needed to configure the tunnel. This allows the protocol to adapt to the needs of the client/server configurations.

The *gateway* is running an HTTPS-enabled web server that serves as a Certificate Authority. Its task is to produce X.509 certificates for use in user authentication during IKE exchanges. The requests are received via HTTP POST requests, which are then processed by a CGI script to authenticate the user and sign the certificate after successful authentication. The user can be authenticated in any appropriate manner such as Unix PAM, or a RADIUS server.

The gateway also serves as a policy server such that it provides configuration information for the client to use to create the IPsec tunnel with the IPsec server. This allows it to dynamically load-balance clients between servers, favor certain users, or restrict users differently according to predefined classes such as user groups or location. While it is desirable to simply enforce these policies only on the client side, because of the open nature of the protocol, the policies need to be enforced on the server side in the form of firewall rules, additional X.509 certificate rules, *etc*.

The *VPN Server* is simply required to negotiate with clients that present valid certificates signed by the CA. Clients verify the server's authenticity via a certificate signed by the certificate authority. The VPN server also authenticates the client in the same manner. This means that the VPN server is not required to keep a list of clients or exchange any state explicitly with the gateway.

This architecture is very powerful because neither the gateway nor the VPN server need to keep state about the allowed users. The VPN server will accept users because their certificates are signed by the CA, and the client accepts the server because its certificate is signed by the same CA. The certificates can be used to implement policies that require state by treating it as a token. Since the certificate is passed to the server for new connections, digitally signed by the CA, and supports custom fields, it provides a secure extensible token to use to exchange simple policy information. For instance, a timeout for the IPsec connection can be implemented by using a short lifetime for the certificate to force the VPN server to deny the client connections after the timeout or attempts by the client to re-key the connection.

This stateless interaction between the gateway and the VPN server allows for multiple gateways and servers. Since the gateway is simply an HTTPS server, the servers can be load balanced with fail-over support using existing techniques to improve the reliability of the system. For the VPN servers, the CGI script used to generate the policy can choose a VPN server for the client to use from a collection of such servers. The criteria for choosing the VPN server to use may be round robin, smallest number of IPsec connections, network locality, *etc.*

4.1 Application Models

There are two models for building the client. The client can either be a separate application that resides on the client machine or be a browser-based plug-in that interacts with the system only when the user accesses a specific page. Each model has specific advantages and disadvantages, but only the application model is easily adaptable to multiple operating systems. A common problem to both models is developing new client software. Since IPsec configuration is platform specific, developers are still faced with creating different solutions for each platform. The IETF has started the process of developing an API to solve this problem [15].

4.1.1 Standalone Application

The stand-alone application is the original design of the application. The core requirements are to keep the binary size small and to minimize the installation complexity. In our implementation, the size was kept to a minimum by using only native Win32 API for all GUI, cryptography, certificate and HTTPS tasks, instead of external libraries or frameworks.

This approach has two advantages. First, the client resides on the client machine, making it easy for the user to access it without requiring a web browser. Second, the application model is a portable concept that can apply to other operating systems and user environments without forcing the user to use a particular web browser.

There are also two disadvantages. First, in order to build the first stage of trust, the client has to trust the CA certificate used by the system which either needs to be signed by a well-known CA so that the CA already exists in the system store or embedded so that the client application can create the trust itself. If an embedded certificate is used, then that means the client itself has to be trusted beforehand. Second, updating the client becomes difficult since this requires special client support to do automatic updates or user intervention to download a new client.

4.1.2 Browser Plug-in

The browser-based plug-in is an extension of the stand-alone application. It operates in a different way than the stand-alone application, but must meet the same requirements.

The most important requirement for a browser plug-in is what browser and operating system combination to use. Two of the more popular browsers are Microsoft Internet Explorer and Mozilla, which implement different approaches for supporting plug-ins, each with different limitations. Both support ActiveX components on Windows that can be downloaded from the network and installed by the browser. These ActiveX components have full access to the operating system services after the digital signature of the component is verified. On other platforms, such as Linux, Mozilla is not a suitable platform for this task. The only dynamic plug-in support is through its JavaScript and XPCOM system. Unfortunately, the system requires lower level access to system functionality than provided by the Mozilla platform.

This approach has three advantages. First, updates are easy as the client accesses the web site to get the client each time. Second, the user is required to trust the web site he is accessing and this trust has been setup out-of-band via a third party CA or other arrangement. Third, the plug-in can be more adaptable by simply trusting the location of its download as a trusted source instead of requiring hard-coded configuration parameters.

There are also two disadvantages. First, we require the user to establish trust with the gateway web server, which can be difficult to do correctly if the user misconfigures their web browser to trust the wrong set of certificates. This flexibility may be too much for system administrators to give to users for fear of users incorrectly configuring their certificates, and therefore being open to attacks from imposter servers. Second, the flexibility required to support the plug-in approach is limited to browsers on Windows because no other web browser and operating system pair supports dynamic plug-ins that can access systems services. This is also difficult on multi-user operating systems because users usually need an elevated level of access to be able to manipulate the IPsec subsystem.

5 Implementation

The architecture is a framework in which any implementation can fit as long as it uses the correct set of protocols. For instance, there are several operating systems that support IPsec and IKE, and the choice may be made simply because of familiarity or availability. The client, on the other hand, depends on the individual user. For our implementation, we used Linux with FreeS/WAN on the server because of our familiarity with it and its availability. For the client, we chose Windows 2000/XP because of its support for IPsec and its popularity as a client platform. We believe that the ability to support such diverse platforms demonstrates the flexibility of the architecture.

5.1 Client

This implementation supports any *x86* system running Windows 2000 SP2 or Windows XP. These clients support IPsec tunneling natively as opposed to the Windows *9x* series. In order to configure the IPsec subsystem for Windows, the user must either manipulate it with dialog boxes and property pages via the Microsoft Management Console (MMC) or with the Internet Protocol Security Policies Tool (Ipsecpol.exe). There is no public API to access the IPsec subsystem and so the only practical way to configure IPsec is through the command line tool [10].

A unique advantage of using Windows as the client is the centralized Certificate Store. The certificate store can be accessed through MMC, and is used by both Internet Explorer and the IPsec subsystem. This makes it easy to manage certificates since there is one comprehensive place to establish and analyze trust relationships. This was useful while developing the client because it allowed flexibility in testing and made it easy to check what the system trusted. Once a certificate is placed on the certificate store, it can be accessed by the WinInet API, Internet Explorer, and the IPsec subsystem.

5.2 Server

The IPsec server can be any system that supports IPsec and has an IKE daemon capable of supporting X.509 authentication via a Certificate Authority certificate. Systems that support this include Linux/FreeS/WAN, OpenBSD, and {Net,Free}BSD with the Raccoon IKE engine.

For our implementation, a Linux *x86* Redhat 7.3 system with FreeS/WAN and the FreeS/WAN X.509 patch was used. To simplify the configuration, the gateway and VPN server ran on the same machine. For the gateway web server, we used Apache with the mod_ssl module. The Certificate Authority was a Perl script that manipulated the certificates using OpenSSL.

6 Security

The security of this system depends on its design, its implementation, and use. The design uses well-understood protocols as building blocks with a minimal trust relationship between the various machines comprising the framework. Each of the protocols has been evaluated for several years, so their weaknesses are well-understood. The implementation is built on proven system libraries. Overall, the system is secure when used by intelligent users.

SSLv3 and TLS have been resistant to many attacks and there are several implementations such as OpenSSL, Microsoft WinInet API, *etc.* Its security relies on the user being acquiring and using valid certificates.

X.509 has also not been successfully attacked, but its security relies on the ability to establish trust and the secrecy of the private keys. While it is recommended that the Certificate Authority be kept separate from the network, our system requires the CA to be connected. The security risks involved can be mitigated by keeping even the CA separate from the gateway, and by creating a separate certificate hierarchy to handle certificates for IPsec. For instance, the CA used to generate certificates for the IPsec gateway and server should not be used to sign the certificate for a company's main web site. Either a separate subtree or disjoint certification tree should be used, instead of the normal CA tree.

Finally, IPsec is composed of the IPsec protocol, and the key exchange. The IPsec protocol is built on simple symmetric cipher and hash operations. While its security is well understood, IKE is a complex protocol, whose security properties are still not entirely understood. This complexity also leads to large and bug-prone software implementations, which introduce additional risks.

While the protocols can be shown to be secure against cryptanalytic attacks, the system is still vulnerable to problems created by careless users. For instance, by allowing users' computers access to the internal network, we could allow worms to spread, or increase the network's exposure to attack. Many windows viruses spread via HTTP (Code Red, Nimda) or SMB file shares (Klez, BugBear). Also, the new tunnel is another potential liability, since it presents another computer which an attacker can compromise and thereby access the corporate network.

7 Conclusions and Future Work

We presented a framework for providing strong security at the network layer to telecommuters and Wireless LAN users by using IPsec. Our contribution is in the ease of use for end users and administrators, by using familiar tools and interfaces (such as a web browser and SSL authentication), which we use to automatically configure the end-user's system with the appropriate parameters. Our system thus removes a large impediment to the use of secure protocols such as IPsec, which are otherwise difficult to configure. The proposed architecture can support any combination of server and client platforms, and can be tailored to the specific needs of individual sites and organizations.

There is more work to be done on implementing this system for other operating systems. Also, the protocol needs to be improved and formalized possibly by using a language like XML. Finally, the system would be integrated with a firewall and/or network appliance, to provide a turn-key solution for telecommuters and users of wireless LANs.

Author Information

Mark Benvenuto received his B.S. in Computer Science from Columbia University (2003). While a student, he worked part-time as a junior system administrator for the Computer Science Department. He currently works as a Software Design Engineer in the SQL Server Engine group at Microsoft. He can be reached at markb@cs.columbia.edu.

Angelos Keromytis has been an Assistant Professor in the Computer Science Department at Columbia University since 2001. He received his M.Sc. and Ph.D. in Computer Science from the University of Pennsylvania (2001), and his B.Sc. from the University of Crete, Greece (1996). His research revolves around end-point security mechanisms, cryptographic protocols, and operating system support for security. He can be reached at angelos@cs.columbia.edu.

Acknowledgements

This work was supported in part by NSF under Contract CCR-TC-0208972.

References

[1] N. Boscia, D. Shaw "Wireless Firewall Gateway White Paper", NASA Advanced Supercomputing Division, White Paper, `http://www.nas.nasa.gov/Groups/Networks/Projects/Wireless/index.html`, March 2003.

[2] CCITT. *X.509: The Directory Authentication Framework.* International Telecommunications Union, 1989.

[3] "Client Certificate and Key Retrieval for IKE", IETF draft, work in progress, `http://www.ietf.org/proceedings/01mar/I-D/ipsra-getcert-00.txt`

[4] John S. Denker, Steven M. Bellovin, Hugh Daniel, Nancy L. Mintz, Tom Killian, Mark Plotnick. "Moat: a Virtual Private Network Appliance and Services Platform" Proceedings of LISA, pp. 251-260, 1999.

[5] Z. Fu, S. F. Wu, H. Huang, K. Loh, F. Gong, "IPsec/VPN Security Policy: Correctness, Conflict Detection and Resolution", IEEE Policy 2001 Workshop, Jan. 2001.

[6] "Internet Key Exchange (IKE) Monitoring MIB", IETF draft, work in progress, `http://www.ietf.org/internet-drafts/draft-ietf-ipsec-ike-monitor-mib-04.txt`

[7] IP Security Remote Access (ipsra) Working Group, `http://www.ietf.org/html.charters/OLD/ipsra-charter.html`

[8] "IPsec Flow Monitoring MIB", IETF draft, work in progress, `http://www.ietf.org/internet-drafts/`

draft-ietf-ipsec-flow-monitoring-mib-02.txt

[9] "IPsec Monitoring MIB", IETF draft, work in progress, http://www.ietf.org/internet-drafts/draft-ietf-ipsec-monitor-mib-06.txt

[10] "KB265112: IPsec and L2TP Implementation in Windows 2000." Microsoft Knowledge Base. June 2003.

[11] S. Kent and R. Atkinson. " The Internet Key Exchange (IKE)." RFC (Proposed Standard) 2409, IETF, November 1998.

[12] S. Kent and R. Atkinson. "Security Architecture for the Internet Protocol." RFC (Proposed Standard) 2401, IETF, November 1998.

[13] X. Liu, C. Madsen, D. McDrew, A. Nourse. "Cisco Systems' Simple Certificate Enrollment Protocol (SCEP)", IETF draft, work in progress, http://www.ietf.org/internet-drafts/draft-nourse-scep-06.txt, May 2002.

[14] "PIC, A Pre-IKE Credential Provisioning Protocol", IETF draft, work in progress, http://www.ietf.org/internet-drafts/draft-ietf-ipsra-pic-06.txt

[15] "Requirements for an IPsec API", IETF draft, work in progress, http://www.ietf.org/internet-drafts/draft-ietf-ipsp-ipsec-apireq-00.txt

[16] "RFC 3457: Requirements for IPsec Remote Access Scenarios"http://www.ietf.org/rfc/rfc3457.txt

[17] Sailer, R., *et al.* "IPSECvalidate - A Tool to Validate IPsec Configurations". Proceedings of LISA, 2001.

[18] Adam Stubblefield, John Ioannidis, Aviel D. Rubin. "Using the Fluhrer, Mantin, and Shamir Attack to Break WEP." Network and Distributed System Security Symposium Conference Proceedings, 2002.