ELSEVIER

Computer Networks 31 (1999) 787–804

COMPUTER NETWORKS

Internet security architecture

Refik Molva [*]

Institut Eurécom, 2229, route des Crêtes, F-06904 Sophia-Antipolis Cedex, France

Abstract

Fear of security breaches has been a major reason for the business world's reluctance to embrace the Internet as a viable means of communication. A widely adopted solution consists of physically separating private networks from the rest of Internet using firewalls. This paper discusses the current cryptographic security measures available for the Internet infrastructure as an alternative to physical segregation. First the IPsec architecture including security protocols in the Internet Layer and the related key management proposals are introduced. The transport layer security protocol and security issues in the network control and management are then presented. The paper is addressed to readers with a basic understanding of common security mechanisms including encryption, authentication and key exchange techniques. © 1999 Elsevier Science B.V. All rights reserved.

Keywords: Internet; Network security; Cryptographic mechanisms; Security protocols

1. Introduction

Apart from increased connectivity and a broad range of new services, the Internet has also given technically advanced intruders the opportunity to carry out a variety of attacks, thereby threatening the integrity of its infrastructure and violating the privacy of its users. Despite the current enthusiasm that supersedes the initial reluctance of business and government users, fear of security breaches on the Internet is forcing most organizations to resort to radical solutions based on physical separation between protected private networks – or intranets – and the public Internet. The resulting segmentation is a major impediment to the accomplishment of the concept of a global Internet. Cryptographic security offers a viable alternative to segmentation by preserving a strongly connected global network. The Internet Engineering Task Force (IETF) recently made significant progress in introducing cryptographic security mechanisms at various layers of the Internet Protocol Suite. These mechanisms allow for the logical protection of information units during their transfer over the global network and eliminate the need for physical segregation of legitimate traffic from potentially harmful network portions. It is hoped that cryptographic security measures will balance the ease and simplicity of solutions based on physical segmentation and provide a practical means of secure communication over the global network for individual users. Nonetheless, segmentation using firewalls and physically separate intranets will probably remain as the only radical solution for globally protecting enterprise networks against malicious traffic.

This article describes the cryptographic security mechanisms of the current Internet architecture in the area of network infrastructure, including Internet and transport layer protocols, routing, directory, and network management functions. Fig. 1 presents the new security components and existing components

[*] E-mail: molva@eurecom.fr; WWW: http://www.eurecom.fr.

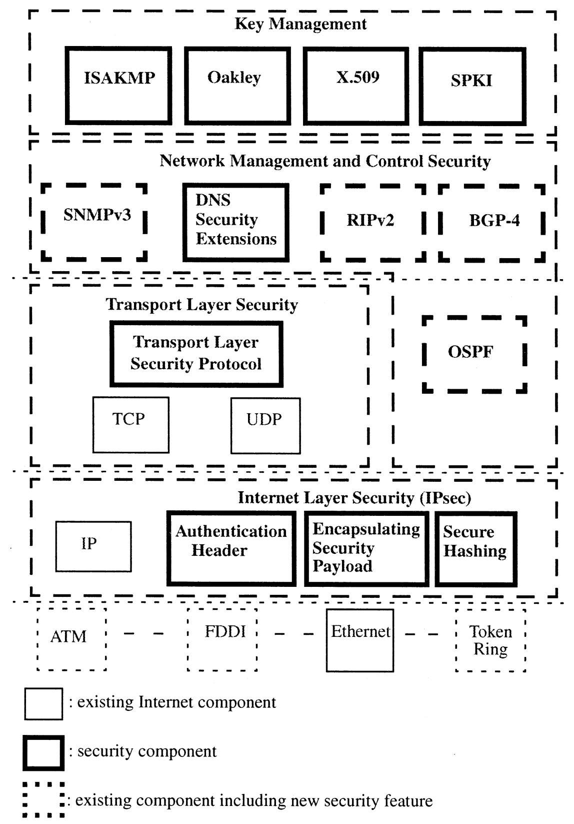

Key Management
ISAKMP
Oakley
X.509
SPKI
Network Management and Control Security
SNMPv3
DNS Security Extensions
RIPv2
BGP-4
Transport Layer Security
Transport Layer Security Protocol
TCP
UDP
OSPF
Internet Layer Security (IPsec)
IP
Authentication Header
Encapsulating Security Payload
Secure Hashing
ATM
FDDI
Ethernet
Token Ring
: existing Internet component
: security component
: existing component including new security feature

Fig. 1. Cryptographic security components of the Internet infrastructure.

enhanced with new security features with respect to the layers of the Internet architecture. Section 2 presents the security architecture for the Internet Protocol, including a detailed description of the two security protocols, IP Authentication Header and IP Encapsulating Payload, a summary of secure hashing techniques adopted by this architecture, and the concept of security associations. The interplay between security protocols and their relationship to security associations are illustrated in a set of typical scenarios pulling together the basic components of the architecture. Section 3 describes the transport layer security protocol and its basic components: the record layer that provides basic security services for the applications and the handshake protocol which assures key exchange and the negotiation of the security functions used by the record layer. Section 4 presents current proposals for key management in the Internet infrastructure. The Internet Security Association and Key Management Protocol and its companion Oakley key exchange protocol are the proposals most likely to become formal standards.

Furthermore, secure data transfer on behalf of users and applications relies on the security of the network control and management protocols that maintain the global connectivity and availability of the network. Among these protocols, the Domain Name System of Internet enjoys the most complete set of security enhancements as presented in Section 5, whereas the other two major functions of the network infrastructure totally lack or only partially enjoy a comprehensive security design. Section 6 discusses the isolated security mechanisms existing in routing protocols and Section 7 summarizes the status of network management security.

2. IP security

The security architecture of the Internet Protocol known as IP Security (IPsec) [1,4] is the most advanced effort in the standardization of Internet security. As the common vehicle for various higher layer protocols, the Internet Protocol (IP) is vulnerable to several attacks threatening either the security of the application payload carried by higher layer protocols like the Transmission Control Protocol (TCP) or the behavior of the network itself through the subversion of network control protocols like the Internet Control Message Protocol (ICMP) or the Border Gateway Protocol (BGP). IPsec covers both the new generation of IP (IPv6) and the current version of IP (IPv4) thanks to the retrofitting of IPv6 security mechanisms into IPv4.

IPsec can be used to protect an IP layer path between a pair of end-systems or hosts, between a pair of intermediate-systems – called security gateways –, or between a host and a security gateway. A security gateway provides the packet forwarding function at the IP layer and thus can be a router, a firewall or a host with IP forwarding capability. IPsec provides the following security functions in the IP layer: data origin authentication, data integrity, replay detection, data confidentiality, limited traffic confidentiality and access control. In addition to the individual security mechanisms that implement these services, IPsec also provides management facilities for the negotiation of services and service parameters between communicating parties, as well as for the exchange of cryptographic keys required by the basic security mechanisms. IPsec mechanisms are designed to be algorithm-independent, in order to accommodate changes in the event of possible evolution of cryptographic algorithms. Nevertheless default algorithms are defined for each service to facilitate interoperability.

IPsec was initially defined in a set of RFC's [1–3]. A substantially revised version was published in a series of Internet drafts [4–6]. Even though the fundamental features of IPsec persisted over the revision, the current IPsec architecture based on the Internet drafts differs significantly from the initial version in several respects. The initial version of IPsec as defined by the RFC's provided a framework that would be completed with possible security mechanisms defined in other documents whereas the current version is a self-contained piece of architecture including a framework and a set of security transforms. Thus message fields previously defined in accompanying documents are now part of the base specification for IPsec. For example, security mechanisms like replay detection, message sequence integrity are now an integral part of the base specification and not a security transform defined in other documents.

The current version of IPsec consists of the following components:

- two security protocols: the IP Authentication Header (IP AH) [5] and the IP Encapsulating Security Payload (IP ESP) [6] that provide the basic security mechanisms within IP;
- security associations (SA) that represent the set of security services and parameters negotiated on each secure IP path;
- algorithms for authentication and encryption.

IP AH and IP ESP may be applied alone or in combination with each other. Each protocol can operate in one of two modes: transport mode or tunnel mode. In transport mode, the security mechanisms of the protocol are applied only to the upper layer data and the information pertaining to IP layer operation as contained in the IP header is left unprotected. In tunnel mode, both the upper layer protocol data and the IP header of the IP packet are protected or 'tunnelled' through encapsulation.

A crucial function closely related to the above mentioned IPsec components is the automatic management of cryptographic keying material and SA's. The Internet Security Association and Key Management Protocol that provides such automatic management functions to security components at the IP layer and above is described in Section 4.

2.1. IP authentication header

The first security protocol in IPsec, IP Authentication Header (IP AH), provides data origin authentication and data integrity for IP datagrams. Replay detection may be selected as an optional service with IP AH. As depicted in Fig. 2, the main fields of IP AH are

- *Security Parameter Index (SPI)*: a random value used in combination with the destination IP address to identify the Security Association for this datagram;
- *Sequence Number*: counter value used to detect replayed IP datagrams in order to assure message sequence integrity;
- *Authentication Data*: integrity check value (ICV) obtained as the result of the secure hash function applied to the integrity protected fields of the original IP datagram.

The AH may be used in two operational modes: transport mode or tunnel mode. In transport mode, the only change in the original IP datagram is the inclusion of the AH field. However, in tunnel mode, in addition to AH, a new IP header is included before the original IP header.

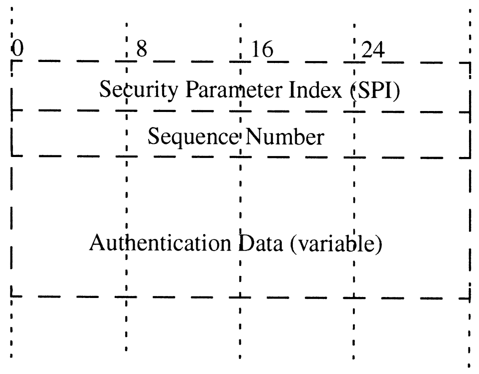

0
8
16
24
Security Parameter Index (SPI)
Sequence Number
Authentication Data (variable)

Fig. 2. Main fields of the IP Authentication Header.

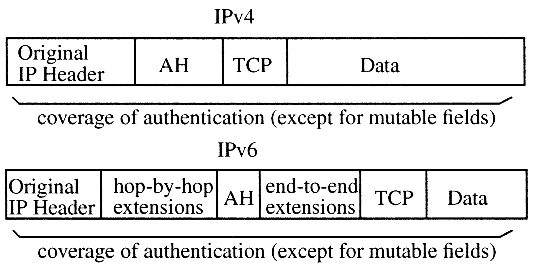

IPv4
Original IP Header
AH
TCP
Data
coverage of authentication (except for mutable fields)
IPv6
Original IP Header
hop-by-hop extensions
AH
end-to-end extensions
TCP
Data
coverage of authentication (except for mutable fields)

Fig. 3. Transport mode AH placement in the IP datagram.

The transport mode is intended for end-to-end protection that can be implemented only by the source and destination hosts of the original IP datagram. Conversely in tunnel mode, source and destination addresses in the new IP header may be different from the ones in the original IP header. Thus, in tunnel mode, the secure path protected by IP AH may be a fraction of the end-to-end path between the source and destination hosts of the original IP header. Hence, the source and destination nodes implementing IP AH on the secure path may either be end-systems (hosts) or intermediate systems (security gateways). The source and destination systems implementing IP AH in either mode are connected through the security association.

The positioning of the AH within the IP packet in transport mode varies depending on the version of the IP as illustrated in Fig. 3. In IPv4, the AH appears after the original IP header and before the upper layer protocol header (TCP). In IPv6, the AH is considered an end-to-end field and thus appears after all the IP header fields required for intermediate node processing (hop-by-hop extension fields) and before the first end-to-end field (end-to-end extension field).

In tunnel mode, both in IPv4 and IPv6, the AH field is placed after the new IP header and before the original IP header as located in the original IP datagram (Fig. 4).

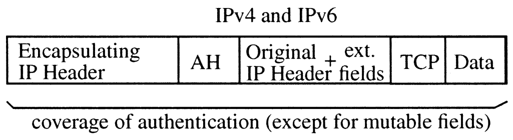


Fig. 4. Tunnel mode AH placement in the IP datagram.

An IP datagram protected by IP AH is processed by the source and destination systems that negotiated a security association prior to the transmission of protected IP datagrams. The outbound processing of an IP AH consists of the generation of the authentication data field. This is performed by calculating a secure hash function (see Section 2.3) on the IP datagram.

The inbound processing consists of the verification of the authentication data field contained in the IP AH with respect to the secure hash value computed by the recipient. If the authentication data field is valid, the integrity of the IP datagram is proved based on the security of the secure hash function. In addition, data origin authentication is assured with respect to the sender since only the sender and recipient of the security association have access to the secure hash function. An attacker can perpetrate a replay attack by sending to the recipient of a security association an IP datagram that has previously been transmitted between the two entities of the security association. If the optional replay detection service is selected by the recipient, then replayed datagrams can be detected based on the sequence number field of the AH.

Some fields of an IP datagram like TTL (time to live) are subject to legitimate modification due to the normal packet forwarding operations performed in intermediate nodes and for such fields qualified as 'mutable' the original value of the field is not known by the node (host or security gateway) at the receiving end of the security association. Thus all mutable fields plus the authentication data field are set to a known value (zero) prior to the computation of the secure hash function both for the generation and the verification of the IP AH header as depicted in Fig. 5. Regardless of the operational mode, the entire IP

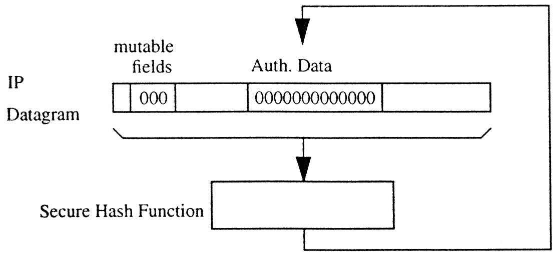


Fig. 5. Computation of the Authentication Data Field for AH.

datagram is considered as the input for the secure hash function except for mutable fields (Figs. 3 and 4).

2.2. IP encapsulating security payload

Encapsulating Security Payload (ESP) is the second IPsec protocol that can be used alone or in combination with IP AH to provide data confidentiality. In its initial design [3], the services provided by ESP were limited to data confidentiality, but this paper refers to the current version of ESP [6] that also includes data origin authentication, data integrity and replay detection services. Data origin authentication and data integrity are joint services that can be selected as an option during the establishment of the security association. Replay detection is another optional service that can be selected if authentication services are selected. Like IP AH, IP ESP may be applied in transport mode or tunnel mode. In tunnel mode the confidentiality service also assures some form of traffic flow secrecy by enabling the security gateways to conceal the identity of the source and destination hosts and the actual size of the IP datagrams.

The ESP header (Fig. 6) includes the security parameter index, sequence number and an optional authentication data field which are handled as similar fields of the AH. The payload field contains the data that is subject to confidentiality protection. Padding is required for 4-byte alignment and to fill the payload data field to the input size required by the encryption algorithm, i.e. the block size of a block cipher, but it may also be viewed as a technique for traffic flow secrecy by keeping the actual length of the protected IP datagram secret.

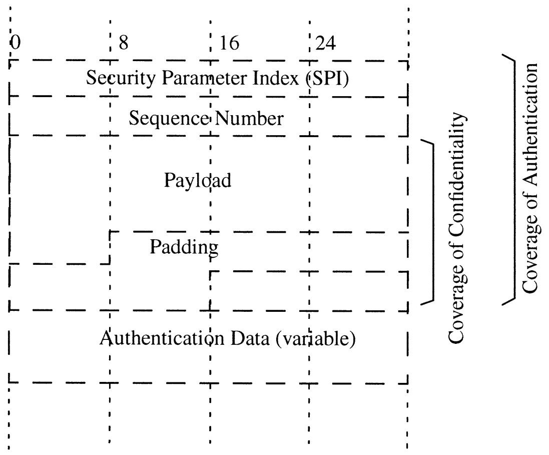
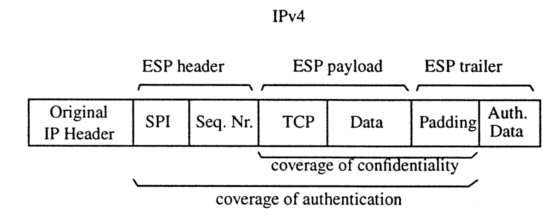

0
8
16
24
Security Parameter Index (SPI)
Sequence Number
Payload
Padding
Authentication Data (variable)
Coverage of Confidentiality
Coverage of Authentication

Fig. 6. Main fields of the ESP Header.



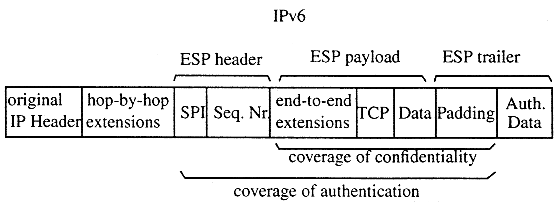


Fig. 7. Structure of IP datagrams in transport mode ESP.

In transport mode ESP (Fig. 7), the encrypted payload includes the upper layer protocol (TCP) information, the user data and the padding. In IPv6, end-to-end extension fields may also be included in the encrypted payload. The original IP header in both IPv4 and IPv6 and the extension fields required by hop-by-hop IPv6 operations are not encrypted. As a result, these fields are positioned in the cleartext part of the IP datagram and before the ESP header. In transport mode the header of the IP datagram, that is, all the information pertaining to the IP protocol including the source and destination addresses, is in cleartext. Hence confidentiality is assured only for the upper layer information. If the entire IP datagram including the protocol specific information also needs to be protected, tunnel mode should be used. In tunnel mode, security gateways acting as intermediate nodes between the ultimate source and destination hosts implement the IP ESP protocol by encapsulating the original IP datagram exchanged between the source and destination with an additional IP header used only on the protected path between the security gateways. The structure of a tunnel mode ESP datagram is depicted in Fig. 8.

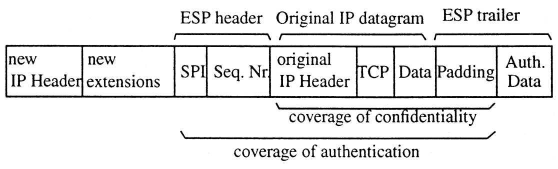


Fig. 8. Structure of IP datagrams in tunnel mode ESP.

Unlike the AH authentication data field, the ESP authentication data field is optional and the authentication provided thereby covers only the ESP header, the ESP payload and the padding fields of the datagram. The IP header (the original one in transport mode or the new one in tunnel mode) is never protected by the ESP authentication service. Thus in cases where data integrity and data confidentiality of the entire IP datagram are required it is recommended to use IP ESP in combination with IP AH.

2.3. Authentication data computation

The authentication service provided by IP AH and IP ESP rely on a secure hashing function to compute the authentication data field that is used for data integrity and data origin authentication. The authentication data can be computed in two different ways:

1. using an encryption algorithm and a message digest function to yield $E_K(H(M))$ where:
   · E is the encryption function using a symmetric or asymmetric algorithm,
   · K is the secret key shared by the source and destination with a symmetric encryption algorithm or the private secret key of the source with an asymmetric algorithm,
   · H is a message digest computed with a secure one-way hash function like MD5 [7] or SHA [8],
2. simply applying the secure one-way hash function (H) on a combination of the message (M) and the secret value (K) shared by the source and destination.

Both methods rely on the security of the one-way hash function, which is evaluated in terms of the frequency of collisions using H on different input messages. By avoiding the use of a cryptographic encryption algorithm, the latter method offers an advantage with respect to government regulations that control export or domestic use of cryptography in various countries.

Even though IPsec protocols are algorithm independent, the current IPsec architecture suggests two different ways to provide secure hashing using the latter technique:

1. keyed hashing: the authentication data is computed as the result of the following expression:
   $$H(K, M, K)$$
   where the cryptographic hash function H is applied to the input message obtained through the concatenation of the shared secret K, the message and K again. K is not transmitted in the datagram since its value is a secret shared by the source and the destination.
2. *HMAC*: the secure hashing expression is
   $$HMAC(K, M) = H(K \oplus P_1, H(K \oplus P_2, M))$$
   where P_1 and P_2 are two different constant strings and $\oplus$ denotes the bit-wise exclusive-or operation.

The main vulnerability of hashing techniques is due to the so called ‘birthday paradox’ that estimates the collision probability for a hash function H with an n-bit output at $2^{-n/2}$ (2^{-64} for H with 128-bit output like MD5). In the case of keyed hashing and *HMAC*, the fact that a secret value is included in the input parameters eliminates the possibility of known-plaintext attacks and the remaining chosen-plaintext search requires on-line collection of $2^{n/2}$ message and authentication data pairs generated by the legitimate parties with the same secret value K. Further justification of keyed hashing and *HMAC* can be found in [9] and [10], respectively. Current IPsec work includes a proposal for each of the above techniques using MD5 as the cryptographic hash function [11,12].

2.4. Security associations

A Security Association (SA) represents an agreement between two IP nodes on a set of security services to be applied to the IP traffic stream between these nodes. An SA is unidirectional in that it defines the services applied to the IP datagrams transmitted in one direction between the pair of nodes that established the SA. Each SA is associated with AH, or ESP services but not both. In cases when both AH and ESP services are to be applied to the same IP traffic stream, two different SA's should be created. The traffic stream associated with an SA can be identified with various levels of granularity. When end-to-end traffic is concerned, the same security services afforded by a single SA can be applied to all IP traffic between two hosts identified by the host IP addresses in the SA, or the traffic pertaining to some higher layer protocol or application as identified by the next protocol field and port numbers. In tunnel mode, all the transit flow between two intermediate nodes or security gateways can be protected by the same set of security services as defined by a single SA.

The SA's of a node are stored in the SA Database (SAD), and each SA is uniquely identified by the tuple

$$\langle \text{destination IP address, IPsec protocol, SPI} \rangle$$

that can be retrieved from the header of each IP datagram protected by an IPsec service. Each SA entry in the SAD stores the following information:

1. list of negotiated values:
   · selected IPsec operational mode (tunnel or transport),
   · list of selected AH or ESP services,
   · types of encryption and hashing algorithms,
   · value of specific parameters for security algorithms like the IV for encryption algorithms or the size of variable fields;
2. keys for authentication and encryption;
3. counter value for message sequence integrity.

The establishment of SA's – either manual or automated – is required prior to the provision of security services between communicating entities. The current solution for the automated management of SA's in the Internet Architecture is presented in Section 4.1.

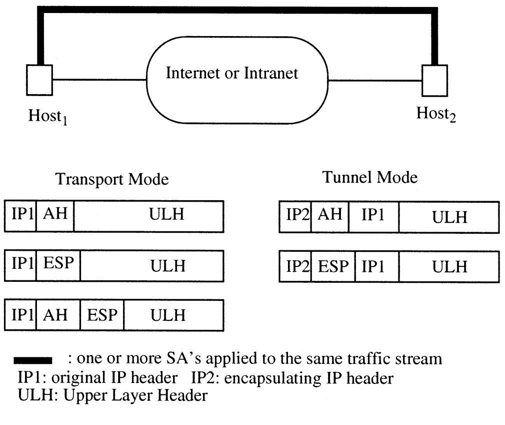


Fig. 9. End-to-end security with IPsec.

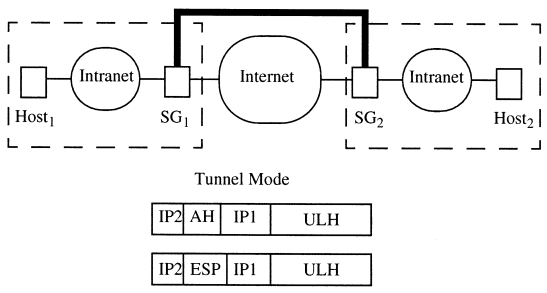


Fig. 10. Simple virtual private network with IPsec.

2.5. IPsec deployment scenarios

This section presents four examples of IPsec deployment scenarios emphasizing the use of SA's and the corresponding IP datagram structure.

The first scenario consists of end-to-end security between two hosts across Internet (or an Intranet). Several SA's, each with different combinations of AH and ESP and different service selections in transport or tunnel mode, can be used between the two hosts in this scenario. Fig. 9 presents transport and tunnel mode IP headers for possible SA combinations. Generalized nesting of more than two SA's is possible but not required.

The second scenario (Fig. 10) illustrates a virtual private network (VPN) built with IPsec. In this case, only tunnel mode is required. AH or ESP protocol can be enforced by the security gateways in order to establish a secure virtual channel between the two Intranet segments. The traffic inside each Intranet, i.e. between Host$_1$ and SG$_1$ and between Host$_2$ and SG$_2$, is not protected.

The third scenario is a combination of the two previous scenarios. As depicted by the possible IP header combinations in Fig. 11, the inner IP datagram exchanged between Host$_1$ and Host$_2$ is encapsulated as a whole by the outer IP header exchanged between the security gateways. The inner header may be protected by AH, ESP, or both in transport and tunnel mode according to the end-to-end SA between the host systems. A different set of SA's is

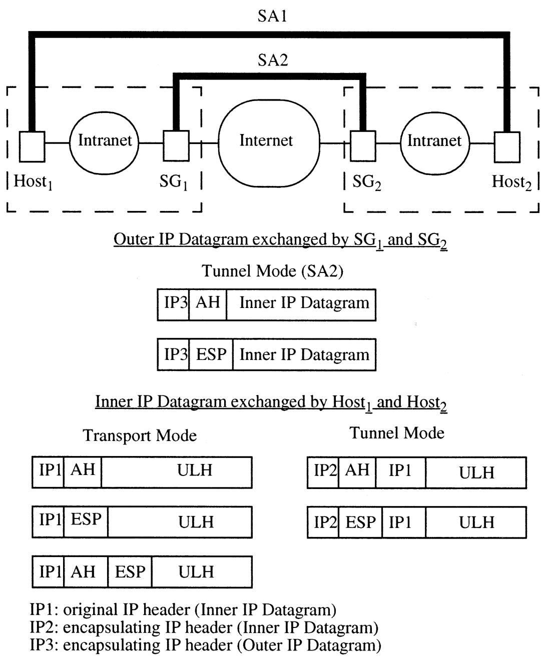

SA1
SA2
Intranet
Internet
Intranet
Host1
SG1
SG2
Host2
Outer IP Datagram exchanged by SG1 and SG2
Tunnel Mode (SA2)
IP3 | AH | Inner IP Datagram
IP3 | ESP | Inner IP Datagram
Inner IP Datagram exchanged by Host1 and Host2
Transport Mode
Tunnel Mode
IP1 | AH | ULH
IP1 | ESP | ULH
IP1 | AH | ESP | ULH
IP2 | AH | IP1 | ULH
IP2 | ESP | IP1 | ULH
IP1: original IP header (Inner IP Datagram)
IP2: encapsulating IP header (Inner IP Datagram)
IP3: encapsulating IP header (Outer IP Datagram)

Fig. 11. Combined VPN and end-to-end security with IPsec.

applied to the outer IP header exchanged between the security gateways across Internet. It should be noted that the support of the end-to-end security across the VPN imposes a new requirement: each security gateway must authorize the transit of IPsec traffic destined to a host behind it.

The fourth scenario depicted in Fig. 12 deals with a remote access situation where an isolated host uses Internet to connect to an Intranet through a security gateway in order to ultimately reach a second host located within the Intranet. Possible choices for the SA between Host$_1$ and SG are identical to the ones between the security gateways of the VPN scenario. Similarly the choices for the end-to-end SA between the remote host and the local one are identical to the ones in the first scenario. The only new requirement in this case is that Host$_1$ must apply the end-to-end transport header before the tunnel header on outbound datagrams.

3. Transport layer security

The main security activity in the area of transport layer is the Transport Layer Security (TLS) Protocol specification [13] based on the Secure Sockets Layer (SSL) Protocol developed by Netscape Communications. Even though TLS is not part of the IPsec architecture, the goal of the TLS effort is to harmonize the TLS Protocol specification with respect to the common key management architecture used by IPsec.

The TLS Protocol operates above a reliable transport protocol like TCP and provides the following security services: peer entity authentication, data confidentiality, data integrity, key generation and distribution, and security parameter negotiation.

The TLS Protocol consists of two layers: the TLS Record Protocol and the TLS Handshake Protocol. The TLS Record Protocol provides basic connection security for various higher layer protocols through encapsulation. One such protocol is the TLS Handshake Protocol that allows the peer entities located at both ends of the secure channel to authenticate one another, to negotiate encryption algorithms and to exchange secret session keys for encryption. Once a transport connection is authenticated and a secret shared key is established with the TLS Handshake Protocol, data exchanged by application protocols can be protected with cryptographic methods by the TLS Record Layer using the keying material derived from the shared secret.

3.1. TLS record layer

The TLS Record Layer affords the following services to the higher layers:

- data encryption using the algorithm selected by the TLS Handshake Protocol. TLS Record Layer supports various encryption algorithms including block ciphers like RC2, Data Encryption Standard (DES), triple DES, 40-bit version of DES (desig-

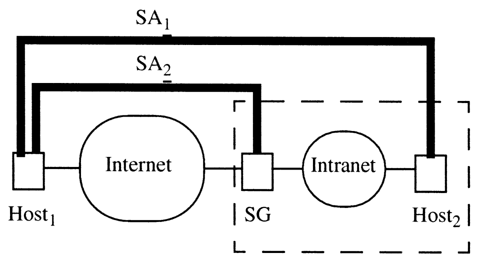


Fig. 12. Remote access with IPsec.

ned to comply with export control regulations), IDEA, and stream ciphers like RC4 (see Refs. [14,15] for further information on encryption algorithms).

- data integrity using a Message Authentication Code (MAC) generated as follows:

  $HMAC\_H(K, s|t|l|m)$

  where

  - $HMAC\_H$ is the HMAC construction for computing the authentication data (see Section 2.3) that is based on the secure hash function H selected by the TLS Handshake Protocol. Possible function types for H are MD5 and SHA.
  - K is the unidirectional data integrity secret established by the TLS Handshake Protocol.
  - $|$ denotes the concatenation.
  - s is a sequence number used for message sequence integrity.
  - t, l and m respectively are the type, length and the content of the higher layer data fragment protected by this MAC.
- replay detection or message sequence integrity using the sequence numbers included in the MAC calculation.
- generation of separate secret keying material for each direction of the data flow and for each security function from the master key established by the TLS Handshake Protocol.

In addition, the TLS Record Layer performs fragmentation and loss-less compression on each higher layer message prior to the application of security mechanisms.

3.2. TLS handshake layer

When a client initiates a connection with a server using the TLS Protocol, they first run the TLS Handshake Protocol to negotiate security algorithms, to authenticate each other and to establish shared cryptographic secrets. The outcome of the initial negotiation by the TLS Handshake Protocol is a session that consists of the following items:

- session identifier: a random byte sequence chosen by the server to identify an active or resumable session state.
- peer certificate: public key certificate of the peer in X.509 version 3 (X.509v3) format [16].
- compression method: the algorithm used to compress data prior to encryption.
- cipher specification: the encryption and MAC algorithms.
- cryptographic attributes such as the hash size.
- master secret: 48-byte secret shared between the client and server and from which various encryption and MAC keys are derived.

These items are then used to create security parameters for use by the Record Layer when protecting application data.

One of three different authentication modes can be negotiated with the TLS Handshake Protocol: authentication of both parties, server authentication with an unauthenticated client, and total anonymity. In conjunction with the authentication modes, the TLS Handshake Protocol supports two different key exchange methods:

1. key distribution with RSA (see Refs. [14,15] for a description of the RSA algorithm); the client generates a secret and sends it to the server after encrypting it with the server's public RSA key.
2. key generation with Diffie–Hellman: the server and the client generate a shared secret key using the Diffie–Hellman algorithm [14,15] and each other's public Diffie–Hellman component. [1] Both the public Diffie–Hellman component and the public RSA key may be either permanent public values or ephemeral values generated for the purpose of a particular key exchange session.

In the anonymous key exchange mode, the public RSA key of the server or the public Diffie–Hellman components are exchanged without authentication. Since intruders do not know the matching secret keys, the resulting shared secret will still be protected from eavesdropping. However, since the communicating parties are not authenticated, active man-in-the-middle attacks [15] are possible.

In the case where only the server is authenticated, the server's public RSA key or its public Diffie–Hellman component can be verified by the client

[1] Each of the peer entities involved in the Diffie–Hellman key exchange pick a random value, x, that is kept secret, and compute $y = g^x \bmod p$, the public value sent to the other party. The shared secret is obtained by each party by computing $y'^x \bmod p$, where y' is the public value received from the other party.

using the certificate sent by the server. The server authentication is complete when the server sends the encryption of all the handshake protocol messages using the shared key distributed during the key exchange (*Finished* message). Thus the server proves its identity by demonstrating its ability to retrieve the shared secret exchanged under its certified public RSA key or through the Diffie–Hellman key generation using its secret component.

In the mutual authentication mode, when Diffie–Hellman key exchange is used, the client is authenticated based on its certified public component and its ability to retrieve the shared secret as the server did in the previous case. If the key exchange is based on RSA, neither a successful key exchange nor the client's ability to retrieve the shared secret assures the client's authentication to the server. In this case, the client is required to sign a hash value derived from the shared secret and all preceding handshake messages. The verification of the signature by the server using the client's certified public key proves the client's identity, and that the secret resulting from the key exchange is shared with the authenticated client (*CertificateVerify* message).

Fig. 13 depicts a typical TLS Handshake message flow. First the client and the server send each other a message containing a random number or a nonce (N_c and N_s respectively) and negotiate the set of attributes and algorithms that will apply to the current session. If the session is not anonymous, the server sends its certificate in X.509v3 format [16]. This certificate contains either the server's public RSA exponent, its Digital Signature Standard [14,15] public key, or its public Diffie–Hellman component depending on the type of algorithm that has been selected for that session. It also contains the certificates of all the certification authorities in the chain through the root certificate. For the purpose of key exchange, if the key corresponding to the server's certificate is not suitable for encryption (signature key or export control limitations) then the server may provide temporary public values signed under the secret key matching with the public key contained in its certificate. The temporary values may be a Diffie–Hellman public component ($g^x \bmod p$) or an RSA public exponent (PK). The server indicates the end of its response by sending the *ServerHelloDone* message.

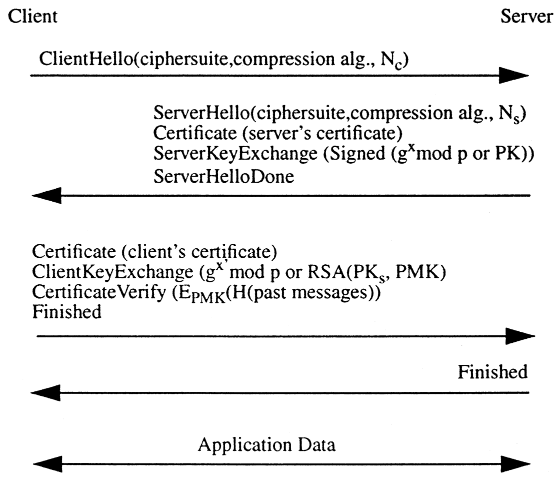

Client
Server
ClientHello(ciphersuite,compression alg., N_c)
ServerHello(ciphersuite,compression alg., N_s)
Certificate (server's certificate)
ServerKeyExchange (Signed (g^xmod p or PK))
ServerHelloDone
Certificate (client's certificate)
ClientKeyExchange ($g^{x'}$ mod p or RSA(PK_s, PMK)
CertificateVerify (E_{PMK}(H(past messages))
Finished
Finished
Application Data

Fig. 13. Message flow for a typical TLS Handshake Exchange.

If client authentication has been negotiated, the client's first reply message is *Certificate* and it contains the client's public key certificate. Next is a key exchange message that is always sent by the client. Depending on the selected key exchange method and authentication type, this message contains either the client's public Diffie–Hellman component ($g^{x'} \bmod p$) if it is not already provided through the client's certificate or the shared secret – called pre-master key (PMK) – generated by the client. PMK is encrypted under the server's public RSA key that is retrieved from the server's certificate. At this point the client and the server can retrieve the shared pre-master key using the selected key exchange method. That is, each can compute it as the Diffie–Hellman shared secret

$$g^{xx'} \bmod p$$

or the server can decrypt the encrypted value sent by the client using its secret RSA key. If client authentication is required and not implicitly assured by the key exchange technique (PMK encrypted with server's public RSA key), the client must send the *CertificateVerify* message including its signature on the bash value of PMK combined with all past messages exchanged in the current session.

In order to reduce the exposure of PMK in the storage of the communicating parties, PMK is substituted with a master secret (K) derived from PMK using the secret hashing technique applied to the concatenation of PMK and the two nonces (N_c and N_s) exchanged in *Hello* messages.

The handshake process terminates with the exchange of the *Finished* message that confirms that the key exchange and the authentication were successful. The *Finished* message includes the secret hash value computed over K and all the past handshake messages.

After completion of the handshake process, application data is protected by the TLS Record Layer using the previously established authenticated secret channel.

4. Key management

Key management is the automated facility that provides communicating parties with symmetric keys required for security services such as authentication, data integrity, and confidentiality. Key management is viewed as a natural component of the basic security architecture in Internet. The two IPsec protocols are tightly coupled with key management via the Security Association (SA) concept. Key management is also considered a complementary mechanism for TLS, routing protocols such as RIP and OSPF (see Section 6), and application protocols. Even though the Internet Architecture Board (IAB) has not yet agreed on a key management architecture among several existing alternatives [18–20], the current work in this area is likely to converge toward a combination of two protocols: the Internet Security Association and Key Management Protocol (ISAKMP) and Oakley key exchange protocol.

ISAKMP [17] is the framework for key exchange and negotiation of SA's (see Section 2.4). ISAKMP is designed to be key exchange independent and can support several key exchange protocols. Oakley [20] describes a series of key exchange methods based on the Diffie–Hellman method that are compatible with the framework defined by ISAKMP. The other alternative key exchange method, that is, key distribution based on a key server like in Kerberos [21], is not supported within the current ISAKMP framework.

Furthermore many Internet protocols rely on public key encryption but the current key management initiative based on ISAKMP and Oakley does not address the management of public keys. Various efforts currently aim at providing a public key infrastructure with different models. The Internet X.509 Public Key Infrastructure work [22] defines public key certificates and certificate management protocols based on the X.509v3 standard. This standard is tightly coupled with the X.500 naming scheme in that each X.509v3 certificate binds a public key with a name expressed in the X.500 format. Lack of support for X.500 names in the Internet community probably has been the main obstacle to the acceptance of the corresponding public key management work. Conversely, an alternative solution using Internet names is provided by the Domain Name System Security Extensions effort, as described in Section 5. Recently, a new direction in public key management was opened in Ref. [23] suggesting a simple public key infrastructure based on the idea that the public key itself can be used as the name of the user, thus avoiding the requirement for an additional naming scheme.

4.1. ISAKMP

A large variety of security services are required depending on each individual network configuration and application scenario. ISAKMP allows peer entities in different communication layers to select and negotiate the security functions suitable to a particular configuration in a pair-wise manner. It also allows them to authenticate one another and to perform key exchanges in a protocol and algorithm independent way.

An important security property assured by ISAKMP is the link between SA establishment, authentication and key exchange. Thus each SA is established between parties that are mutually authenticated and share one or many secrets. Based on the link between the authentication and the shared secrets, the parties can provide the evidence of authentication by mutually demonstrating their ability to encrypt with the shared secret.

Furthermore ISAKMP incorporates a mechanism to counter denial of service attacks in which servers are flooded with bogus request messages. The goal of the attacker perpetrating these attacks is to keep a server busy with the verification of a large number of bogus requests in order to cause abnormal CPU

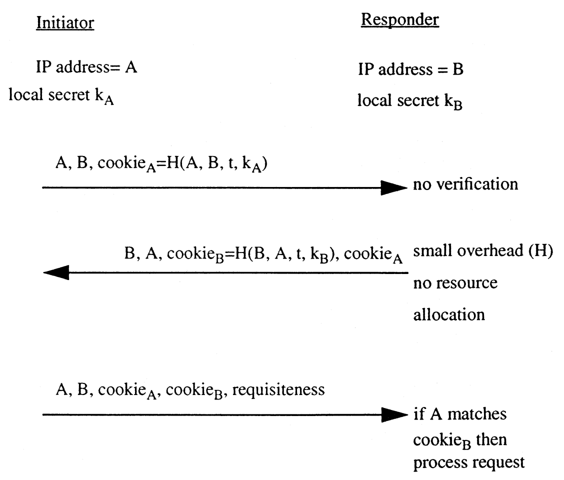

Initiator
Responder
IP address= A
local secret k_A
IP address = B
local secret k_B
A, B, cookie_A=H(A, B, t, k_A)
no verification
B, A, cookie_B=H(B, A, t, k_B), cookie_A
small overhead (H)
no resource allocation
A, B, cookie_A, cookie_B, requisiteness
if A matches cookie_B then process request

Fig. 14. Denial of service protection using the anti-clogging token.

usage and consequently degrade the service provided by the server to legitimate users. To do so, the attacker issues several request messages from his host with bogus user identification information and a different bogus source address is set in each IP datagram carrying the requests. These requests usually get discarded by the application layer authentication mechanism at the server but the CPU and memory consumption required for the verification of these bogus requests can be sufficient to keep most of the server's resources busy thereby causing the denial of service to the legitimate users. Classical authentication mechanisms therefore cannot prevent such denial of service attacks because of the high CPU consumption caused by cryptographic operations used in authentication.

The ISAKMP mechanism to prevent such denial of service attacks is based on the anti-clogging technique introduced by Ref. [18]. The principle of anti-clogging is to perform the exchange of a pair of 'cookies' at the beginning of each client-server connection before initiating any resource-intensive verification (Fig. 14). This initial exchange provides a weak authentication and allows for the verification of the client's presence at the claimed IP address thus thwarting all flooding attempts using bogus IP addresses from a single host. In fact the intruder cannot pursue the protocol using bogus addresses beyond the first message since he cannot get the server's cookie sent in response to the bogus IP source addresses. The computation of the cookie by the server is based on a simple hash function requiring low CPU usage in comparison with CPU-intensive strong authentication and key generation operations and no resource reservation takes place before the completion of the successful cookie exchange. Each ISAKMP message contains the pair of cookies generated by the initiator and the responder based on the anti-clogging technique.

ISAKMP provides protocol exchanges to establish SA's between peer ISAKMP servers (Fig. 15). From the point of view of the protocol suite ISAKMP is an application layer protocol positioned above the transport layer. The typical ISAKMP server operates over UDP at port 500.

First, ISAKMP creates the ISAKMP SA between the ISAKMP servers. Additional SA's on behalf of user protocols like IP AH or IP ESP can then be created by the ISAKMP servers using the security services of the ISAKMP SA to protect subsequent ISAKMP messages.

An ISAKMP message consists of a fixed header followed by a variable number of building blocks named payloads. SA negotiation, certificate exchange, authentication and key exchange are achieved through the exchange of ISAKMP messages using various combinations of basic payload types. These include security association, identification, key exchange, certificate, hash, signature, and nonce. Each payload type can support a variety of techniques for

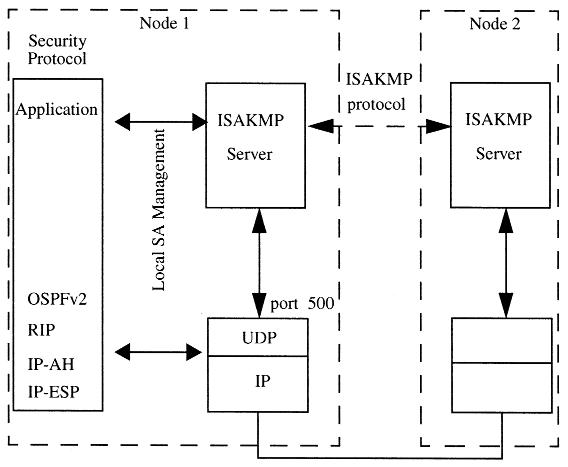


Fig. 15. ISAKMP Model.

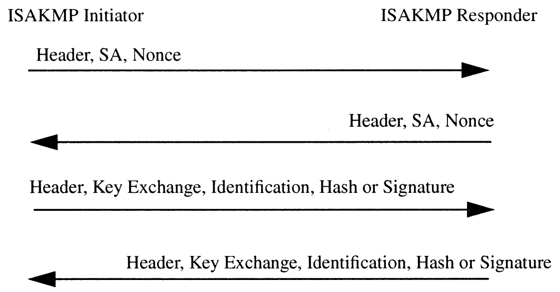

ISAKMP Initiator
ISAKMP Responder
Header, SA, Nonce
Header, SA, Nonce
Header, Key Exchange, Identification, Hash or Signature
Header, Key Exchange, Identification, Hash or Signature

Fig. 16. Message flow for the base ISAKMP exchange.

the corresponding function, i.e. the key exchange payload can support various key exchange protocols including Oakley.

Fig. 16 depicts a simple ISAKMP exchange illustrating the individual ISAKMP payload types included in each message.

In the first message, the initiator generates an SA proposal for the selected security services and parameters it deems adequate for the required protection level. A one-time random number is also transmitted in the *Nonce* payload. This value should be used as a challenge by the authentication mechanism at the responder to generate the *Hash* or *Signature* payload transmitted in the third message.

In the second message, the responder indicates the security services and parameters it has accepted. Again, the responder includes a nonce to be used as a challenge by the authentication mechanism at the client.

In the last two messages, the initiator and the responder mutually exchange keying material using the selected key exchange mechanism to come up with a shared secret and identification information. The payloads in each of these messages are authenticated using the selected authentication mechanism and the challenge sent by the peer entity during the initial SA negotiation. The result of the authentication mechanism may be encoded either as a *Hash* or *Signature* payload depending on the type of the agreed upon mechanism (secure hashing or encryption).

4.2. The Oakley key determination protocol

The Oakley Key Determination Protocol is a key exchange mechanism for establishing shared secrets using the Diffie–Hellman key generation technique. Oakley's main properties are: authenticated key exchange, perfect forward secrecy, and compatibility with ISAKMP.

Oakley incorporates a mandatory authentication mechanism for the verification of identities during key exchange in order to prevent man-in-the-middle attacks. The public components transmitted during the Diffie–Hellman key exchange are signed using a pre-arranged shared secret and secure hashing, a signature using RSA, or a DSS signature.

Perfect forward secrecy as defined by Ref. [24] assures that the compromise of a long-lived master key (such as public and private RSA keys) does not allow the intruder to retrieve the value of the session keys that were exchanged during the lifetime of the master key. The basic rule to achieve perfect forward secrecy is to avoid using master keys to derive session keys either through encryption or algorithmic key generation such as Diffie–Hellman. In Oakley, perfect forward secrecy is achieved by using the master keys only for the authentication of the public Diffie–Hellman component from which the secret session keys are derived. Theft of the master key would thus allow the intruder to impersonate legitimate parties in future key exchanges but the intruder would not be able to retrieve any past session key.

Oakley messages consist of various fields including cookies, public Diffie–Hellman components, nonces, signatures, hash values and identification information. Oakley is compatible with ISAKMP in that each Oakley field can be mapped onto either some ISAKMP header field or ISAKMP payload.

Fig. 17 depicts a typical Oakley exchange using the following notation:

- I, R: the identities of the initiator and the responder, respectively.
- cookie$_i$, cookie$_r$: anti-clogging cookies generated by the initiator and the responder, respectively, using the IP address of the local host.
- N_i, N_r: one-time random numbers or nonces generated by the initiator and the responder, respectively.
- Sign$_K${}: signature or hash computed with secret K.

In the first message of this example, the initiator issues a public Diffie–Hellman component (g^x mod p) using a freshly generated random value (x) that

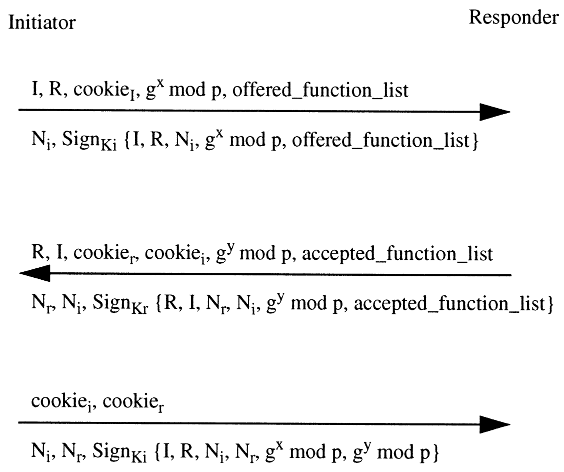

Initiator
Responder
I, R, cookie$_I$, g^x mod p, offered_function_list
N$_i$, Sign$_{Ki}$ {I, R, N$_i$, g^x mod p, offered_function_list}
R, I, cookie$_r$, cookie$_i$, g^y mod p, accepted_function_list
N$_r$, N$_i$, Sign$_{Kr}$ {R, I, N$_r$, N$_i$, g^y mod p, accepted_function_list}
cookie$_i$, cookie$_r$
N$_i$, N$_r$, Sign$_{Ki}$ {I, R, N$_i$, N$_r$, g^x mod p, g^y mod p}

Fig. 17. Oakley key exchange example.

will be kept secret. In the second message, the responder sends his public Diffie–Hellman component (g^y mod p) derived from a secret random value (y). Each party can compute a common shared secret (g^{xy} mod p) using the public component sent by the peer and the local secret. Perfect forward secrecy is achieved through this key exchange because the secret values (x and y) from which the shared session keys are derived are random and not related to any long-lived master key. On the other hand, the resulting public Diffie–Hellman components are not authenticated (as opposed to certified Diffie–Hellman public components). As a result, in the key exchange, fields are accompanied by a signature covering the public Diffie–Hellman component and computed using a long-lived authentication key. The key exchange protocol is also tightly coupled with an authentication exchange using nonces. The signature of the responder on N_i in the second flow and the signature of the initiator on N_i in the third flow authenticate the responder and the initiator, respectively. In addition, the fact that the authentication fields also include the public Diffie–Hellman components assure that the resulting shared session key will be known only by the authenticated parties. Furthermore, anti-clogging cookies included in Oakley messages are also used for the purpose of key identification, each key name being derived from the peers' cookies.

5. Domain name system security extensions

The Domain Name System (DNS) provides host names to IP address mapping. The DNS is organized into a hierarchy of servers each having the responsibility of a particular portion of the DNS database. Current DNS protocols completely lack security mechanisms. A variety of threats on the DNS protocols exist that mainly take advantage of the lack of authentication and data integrity. By exploiting the absence of client authentication or by eavesdropping with bulk data transfers between DNS servers, intruders may cause the leakage of information on the topology of private enterprise networks. The impersonation of DNS servers can cause traffic or mail subversion by injecting bogus addressing information. Moreover, DNS impersonation combined with attacks on the routing system can seriously jeopardize the overall network operation as pointed out by [25].

Current work in the IETF security working groups defines extensions to DNS [26] aiming at the addition of security mechanisms in three areas:

- data origin authentication in order to prevent the tampering with the data stored in the DNS servers,
- transaction authentication to eliminate the possibility of server and client impersonation and data modification during DNS transactions,
- public key certification using DNS as a public key certificate repository.

The DNS extensions do not cover confidentiality, denial of service or any form of access control for DNS requests. In order to assure interoperability between the current DNS protocol and future extensions, the extensions do not require any protocol change other than the support of optional data types to store security information in the basic DNS data structures called 'resource records' or RR (Fig. 18).

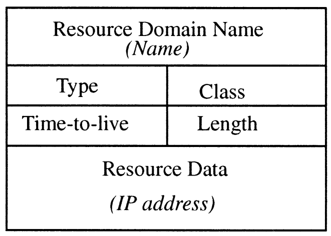


Fig. 18. DNS resource record.

DNS security extensions introduce two new RR types: the KEY RR and the signature or SIG RR.

The SIG RR is the basic building block through which data origin and transaction authentication is assured. A SIG RR stores the value of a signature that covers one or many resource records as identified by the 'Type covered' sub-field in the Resource Data field of the SIG RR (Fig. 19). In addition, the Resource Data field of the SIG RR holds the name of the party that issued the signature, the signature time and its expiration date. The Key footprint subfield contains an algorithm-dependent short value for the rapid verification of the public key that can possibly be used for the verification of the signature. This can consist of the hash or some selected octets of the public key. Although various signature algorithms can be used, RSA encryption of the MD5 hash is incorporated as the default signature mechanism.

Data origin authentication can be provided using a SIG RR including a signature that covers one or many DNS RR's. Through the verification of that signature with the DNS public key, recipients can be assured of the origin of the name to address mapping and thwart impersonation attacks.

The KEY RR stores the public key of a party identified by the Resource Domain Name field (Fig. 20). A DNS public key certificate consists of a KEY RR containing the public key and the name followed by a SIG RR that includes the signature covering the KEY RR. In the case of a SIG RR that is part of a public key certificate, the signature should be computed using the private key associated with the logical portion of the DNS database named 'zone'. The concept of a DNS zone is akin to the role of a certification authority (CA) in X.509 [16]. A DNS public key certificate thus provides a strong binding between a name and a public key based on a trusted zone authority.

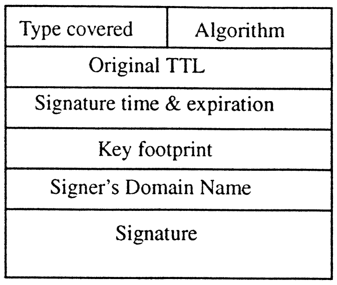


Fig. 19. Resource data field of a SIG RR.

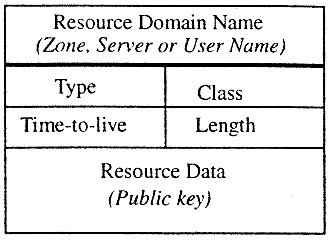


Fig. 20. KEY RR.

DNS servers do not necessarily bear the role of a CA or zone authority with respect to public key certification. Thus the zone private key and the private key of each DNS server managing the corresponding portion of the DNS database are different. Public key signatures stored in the DNS database must therefore be computed off-line using the zone private key that is not stored in the DNS servers. Moreover current DNS extensions do not include the certification chain concept whereby, each public key can be verified using an ordered list of certificates each delivered by a different CA positioned on a path or chain from the local CA through the root CA. In order to validate a public key certificate using such a chain, the certificate, that is signed by the first CA of the chain, is verified using the public key of the first CA. The latter public key is in turn signed by the next CA in the chain. The next step of the certificate chain validation consists of verifying this signature using the public key of the next CA. The public key of each CA is thus verified using the certificate delivered by the next CA on the chain until the root CA is reached. The public key of the root CA is self validated since its value is well known by all parties using the certification system.

DNS transaction authentication is provided by a SIG RR that covers the request or response message. In an authenticated response message, the signature covers both the response and the corresponding request that triggered the former. Unlike the signature that is part of the public key certificates, the signature for authenticating DNS responses is computed by the DNS server that issues the response using the server's private key.

6. Routing security

Routing protocols that are responsible for maintaining network connectivity for all the TCP/IP traffic have recently become one of the main targets of attackers on the Internet. Because of the global impact of such attacks, routing security is a critical issue for the whole Internet infrastructure. Attacks on routing protocols can cause legitimate traffic to flow over unsecure paths and create various types of security exposure for higher layer protocols ranging from eavesdropping to denial of service.

Several routing protocols are used to exchange network topology and routing table information between routers. Commonly used intra-domain routing protocols are the Routing Information Protocol (RIP) and the Open Shortest Path First (OSPF). The Border Gateway Protocol (BGP) is the current inter-domain protocol used between the core routers on the Internet.

The main security threats on routing protocols are route subversion through the exchange of bogus routing information and through the impersonation of routers. The security services required in routing protocols thus include data origin authentication and data integrity to prevent router impersonation and tampering with routing data. RIPv2's password-based authentication scheme that suffered from eavesdropping and masquerade was enhanced with a strong authentication mechanism based on secure hashing using MD5 [27]. Despite a sufficient level of protection against data modification provided by this mechanism, RIPv2 still lacks replay detection. OSPFv2 includes an authentication mechanism that allows communicating routers to use either password-based or cryptographic authentication and replay detection [28]. In IPv6, intra-domain routing protocols rely on the security provided by the default AH and ESP support of IPv6 routers.

In the inter-domain area, the current version of BGP [29] includes an extension for an authentication field in routing protocol messages. Moreover, since BGP messages are carried over the transport layer, unprotected BGP messages are exposed to replay and data tampering in this layer. Some proprietary implementations of BGP, such as the CISCO routers, offer a transport layer protection mechanism for the encapsulated BGP flows. The Inter-Domain Routing Protocol that will replace BGP in the long run includes strong authentication as part of the routing protocol.

Cryptographic mechanisms implemented in routers require a significant amount of secret keys to be shared among routers. Manual key distribution can be afforded, as in the case of OSPF that already involves substantial manual configuration for the routing functions. This becomes a significant burden in case of RIP where the amount of manual configuration for the routing operations is very low. In the inter-domain area, automated key management requires the establishment of common trust between independent domains, putting the accent on public key certification. Automated key management seems to be a strong requirement for both intra-domain and inter-domain routing protocols, but the current authentication solutions in routing protocols are not yet integrated with the forthcoming key management architecture based on ISAKMP and Oakley.

7. Security of network management

The Simple Network Management Protocol (SNMP) that allows network operators to remotely monitor, configure and debug networks is one of the most critical components of the Internet infrastructure. Impersonating various SNMP parties, intruders can gain complete control of a network and totally jeopardize its operation. The current version of SNMP that is widely implemented in commercial products supports a simple identity verification technique based on secret values called 'community names' that are shared by several parties and exchanged in cleartext through the network. By obtaining a community name through eavesdropping or any other form of information leakage, intruders can access the Management Information Base (MIB) on managed network components. Intruders can then subvert the behavior of the network at various layers using the read and write operations on the content of the MIB, including routing tables and security information such as passwords. Several attempts to include strong security features in SNMP version 2 (SNMPv2) have failed. After the demise of SNMPv2, two new pieces of architecture that define authentication and confidentiality mechanisms based on a new approach called ‘user-based security’ [30], and access control mechanisms [31] have recently been proposed as part of SNMP version 3. The authentication scheme suggested in Ref. [30] relies on the HMAC technique for the computation of the authentication data. The new design also includes alternative solutions for detecting replays and assuring the timeliness of network management messages. As part of the user-based security model, each authoritative SNMP engine inherits a cryptographic key derived from the user’s password. The derivation technique is location-dependent in that by computing the key as a function of the password and the identity of the SNMP engine, a different key is obtained for the same user on each different SNMP engine. Despite the comprehensive analysis of security problems and service requirements they offer, these two new pieces of architecture still suffer from the lack of integration with the underlying IPsec architecture.

8. Conclusion

An integrated security architecture exists for the Internet Protocol, including security protocols covering various services and joint management protocols for security association and key exchange. Based on this core architecture, security can be assured for several upper layer protocols that use IPv4 or IPv6 as the basic transport mechanism. Furthermore, in order to provide secure connections that accommodate application specific requirements, the transport-layer security work defines a security protocol positioned immediately below the application layer. Based on a widely used product implementation, the current version of this protocol consists of an independent architecture including its own security management functions.

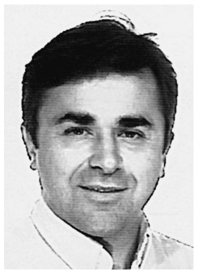
The need for security is even stronger for network control and management functions that are responsible for maintaining the connectivity over the global network. Routing protocols on IPv4 were recently enhanced with isolated authentication mechanisms, but product support for these enhancements and their integration with the core IPsec architecture are still lacking. In IPv6, routing protocols will rely on the security of IPsec like most other protocols using the IP layer. Network management, crucial to the operation of the network, is an area where cryptographic security is severely lacking despite numerous attempts to include security in recent versions of the Simple Network Management Protocol. Conversely, the Domain Name System enjoys a well-defined architecture for security extensions covering the authentication of its database and user transactions.

When security is addressed as a global network problem, a major issue is the management of security services, because of the complexity of interactions between various security mechanisms implemented in the protocols and the need for automatic configuration of these mechanisms. ISAKMP and Oakley offer a suitable solution for the management of security associations and the exchange of shared session keys with IPsec protocols. Other protocols like RIP and OSPF for routing and the TLS protocol are likely to become integrated with the core architecture and make use of ISAKMP and Oakley. Public key management, on the other hand, is being looked at by various competing parties that are far from agreeing on a common solution.

References

[1] R. Atkinson, Security Architecture for the Internet Protocol, RFC 1825, August 1995.
[2] R. Atkinson, IP Authentication Header, RFC 1826, August 1995.
[3] R. Atkinson, IP Encapsulating Security Payload (ESP), RFC 1827, August 1995.
[4] S. Kent, R. Atkinson, Security Architecture for the Internet Protocol, Internet-Draft: draft-ietf-ipsec-arch-sec-07.txt, work in progress, July 1998.
[5] S. Kent, R. Atkinson, IP Authentication Header, Internet-Draft: draft-ietf-ipsec-auth-header-07.txt, work in progress, July 1998.
[6] S. Kent, R. Atkinson, IP Encapsulating Security Payload (ESP), Internet-Draft: draft-ietf-ipsec-esp-v2-07.txt, work in progress, July 1998.
[7] R. Rivest, The MD5 Message-Digest Algorithm, RFC 1321, April 1992.
[8] NIST, FIPS PUB 180-1: Secure Hash Standard, April 1995.
[9] G. Tsudik, Message authentication with one-way hash functions, Proc. Infocom’92, May 1992.
[10] H. Krawczyk, M. Bellare, R. Canetti, HMAC: Keyed-Hashing for Message Authentication, RFC 2104, February 1997.
[11] P. Metzger, W. Simpson, IP Authentication using Keyed MD5, RFC 1828, August 1995.
[12] M. Oehler, R. Glenn, HMAC-MD5 IP Authentication with Replay Prevention, RFC 2085, February 1997.

[13] T. Dierks, C. Allen, The TLS Protocol Version 1.0, Internet-Draft: draft-ietf-tls-protocol-05.txt, work in progress, November 1997.
[14] B. Schneier, Applied Cryptography, 2nd ed., Wiley, New York, 1995.
[15] D. Stinson, Cryptography Theory and Practice, CRC Press, Boca Raton, 1995.
[16] ISO-ITU, The Directory: Authentication Framework, ISO/IEC 9594-8/ITU-T Recommendation X.509, 1997.
[17] D. Maughan, M. Schertler, M. Schneider, J. Turner, Internet Security Association and Key Management Protocol (ISAKMP), Internet-Draft: draft-ietf-ipsec-isakmp-10.txt, work in progress, July, 1998.
[18] P. Karn, B. Simpson, Photuris: Session Key Management Protocol, Internet-Draft: draft-simpson-photuris-15.txt, work in progress, July 1997.
[19] A. Aziz, T. Markson, H. Prafullchandra, Simple Key-Management for Internet Protocols (SKIP), Internet Draft: draft-ietf-ipsec-skip-07.txt, work in progress, August 1996.
[20] H.K. Orman, The Oakley Key Determination Protocol, Internet-Draft: draft-ietf-ipsec-oakley-02.txt, work in progress, July 1997.
[21] J. Steiner, C. Neuman, J. Schiller, Kerberos: an authentication service for open network systems, Proc. USENIX Winter Conf., February 1988.
[22] C. Adams, S. Farell, Internet X.509 Public Key Infrastructure Certificate Management Protocols, Internet-Draft: draft-ietf-pkix-ipki3cmp-08.txt, work in progress, May 1998.
[23] C.M. Ellison, B. Frantz, B. Lampson, R. Rivest, B.M. Thomas, T. Ylonen, SPKI Certificate Theory, Internet-Draft: draft-ietf-spki-cert-theory-02.txt, work in progress, March 1998.
[24] W. Diffie, Authenticated key exchange and secure interactive communication, Securicom '90, Paris, 1990.
[25] S. Bellovin, Security problems in the TCP/IP protocol suite, Computer Communication Reviews, May 1989.
[26] D. Eastlake, C. Kaufman, Domain Name System Security Extensions, RFC 2065, January 1997.
[27] F. Baker, R. Atkinson, Routing Information Protocol Version 2 – MD5 Authentication, RFC 2082, January 1997.
[28] J. Moy, Open Shortest Path First Version 2, RFC 2328, April 1998.
[29] Y. Rekhter, T. Li, A Border Gateway Protocol 4 (BGP-4), Internet-Draft: draft-ietf-idr-bgp4-08.txt, work in progress, August 1998.
[30] U. Blumenthal, B. Wijnen, User-based Security Model (USM) for Version 3 of the Simple Network Management Protocol (SNMPv3), Internet-Draft: draft-ietf-snmpv3-usm-v2-01.txt, work in progress, August 1998.
[31] B. Wijnen, R. Presuhn, K. McCloghrie, View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP), Internet-Draft: draft-ietf-snmpv3-vacm-00.txt, work in progress, August 1998.



Refik Molva is an associate professor at Institut Eurécom in Sophia Antipolis, France. He is currently leading network security research in the area of mobile code security and multipoint security protocols. He has been responsible for research projects on mobile network security, anonymity and intrusion detection. Beside security, he worked on distributed multimedia applications and was responsible for the BETEUS European project on CSCW over a trans-european ATM network. Prior to joining Eurécom, he worked for five years as a Research Staff Member in the Zurich Research Laboratory of IBM where he was one of the key designers of the KryptoKnight security system. He also worked as a network security consultant in the IBM Consulting Group in 1997. Refik Molva has a Ph.D. in Computer Science from the Paul Sabatier University in Toulouse (1986) and a B.Sc. in Computer Science (1981) from the University of Grenoble, France.