DECLARATION OF SANDY GINOZA FOR IETF

RFC 2401: SECURITY ARCHITECTURE FOR THE INTERNET PROTOCOL
RFC 2408: INTERNET SECURITY ASSOCIATION AND KEY MANAGEMENT PROTOCOL (ISAKMP)
RFC 3102: REALM SPECIFIC IP: FRAMEWORK
RFC 3103: REAL SPECIFIC IP: PROTOCOL SPECIFICATION
RFC 3104: RSIP SUPPORT FOR END-TO-END IPSEC

I, Sandy Ginoza, hereby declare that all statements made herein are of my own knowledge and are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code:

1. I am an employee of Association Management Solutions, LLC (AMS), which acts under contract to the IETF Administration LLC (IETF) as the operator of the RFC Production Center. The RFC Production Center is part of the "RFC Editor" function, which prepares documents for publication and places files in an online repository for the authoritative Request for Comments (RFC) series of documents (RFC Series), and preserves records relating to these documents. The RFC Series includes, among other things, the series of Internet standards developed by the IETF. I hold the position of Director of the RFC Production Center. I began employment with AMS in this capacity on 6 January 2010.

2. Among my responsibilities as Director of the RFC Production Center, I act as the custodian of records relating to the RFC Series, and I am familiar with the record keeping practices relating to the RFC Series, including the creation and maintenance of such records.

3. From June 1999 to 5 January 2010, I was an employee of the Information Sciences Institute at University of Southern California (ISI). I held various position titles with the RFC Editor project at ISI, ending with Senior Editor.

4. The RFC Editor function was conducted by ISI under contract to the United States government prior to 1998. In 1998, ISOC, in furtherance of its IETF activity, entered into the first in a series of contracts with ISI providing for ISI's performance of the RFC Editor function. Beginning in 2010, certain aspects of the RFC Editor function were assumed by the RFC Production Center operation of AMS under contract to ISOC (acting through its IETF function and, in particular, the IETF Administrative Oversight Committee (now the IETF Administration LLC (IETF)). The business records of the RFC Editor function as it was conducted by ISI are currently housed on the computer systems of AMS, as contractor to the IETF.

5. I make this declaration based on my personal knowledge and information contained in the business records of the RFC Editor as they are currently housed at AMS, or confirmation with other responsible RFC Editor personnel with such knowledge.

6. Prior to 1998, the RFC Editor's regular practice was to publish RFCs, making them available from a repository via FTP. When a new RFC was published, an announcement of its publication, with information on how to access the RFC, would be typically sent out within 24 hours of the publication.

7. Any RFC published on the RFC Editor website or via FTP was reasonably accessible to the public and was disseminated or otherwise available to the extent that persons interested and ordinarily skilled in the subject matter or art exercising reasonable diligence could have located it. In particular, the RFCs were indexed and placed in a public repository.

8. The RFCs are kept in an online repository in the course of the RFC Editor's regularly conducted activity and ordinary course of business. The records are made pursuant to established procedures and are relied upon by the RFC Editor in the performance of its functions.

9. It is the regular practice of the RFC Editor to make and keep the RFC records.

10. Based on the business records for the RFC Editor and the RFC Editor's course of conduct in publishing RFCs, I have determined that the publication date of RFC 2401 and RFC 2408 was no later than November 1998, at which time they were reasonably accessible to the public either on the RFC Editor website or via FTP from a repository. Copies of these RFCs are attached to this declaration as exhibits.

11. Based on the business records for the RFC Editor and the RFC Editor's course of conduct in publishing RFCs, I have determined that the publication date of RFC 3102, RFC 3103, and RFC 3104 was no later than October 2001, at which time they were reasonably accessible to the public either on the RFC Editor website or via FTP from a repository. Copies of these RFCs are attached to this declaration as exhibits.

Pursuant to Section 1746 of Title 28 of United States Code, I declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct and that the foregoing is based upon personal knowledge and information and is believed to be true.

Date: 14 February 2019

By: ______________________
Sandy Ginoza

4847-8253-0696.1

Network Working Group S. Kent
Request for Comments: 2401 BBN Corp
Obsoletes: 1825 R. Atkinson
Category: Standards Track @Home Network
November 1998

Security Architecture for the Internet Protocol

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Table of Contents

1. Introduction

1.1 Summary of Contents of Document

This memo specifies the base architecture for IPsec compliant systems. The goal of the architecture is to provide various security services for traffic at the IP layer, in both the IPv4 and IPv6 environments. This document describes the goals of such systems, their components and how they fit together with each other and into the IP environment. It also describes the security services offered by the IPsec protocols, and how these services can be employed in the IP environment. This document does not address all aspects of IPsec architecture. Subsequent documents will address additional architectural details of a more advanced nature, e.g., use of IPsec in NAT environments and more complete support for IP multicast. The following fundamental components of the IPsec security architecture are discussed in terms of their underlying, required functionality. Additional RFCs (see Section 1.3 for pointers to other documents) define the protocols in (a), (c), and (d).

a. Security Protocols -- Authentication Header (AH) and Encapsulating Security Payload (ESP)
b. Security Associations -- what they are and how they work, how they are managed, associated processing
c. Key Management -- manual and automatic (The Internet Key Exchange (IKE))
d. Algorithms for authentication and encryption

This document is not an overall Security Architecture for the Internet; it addresses security only at the IP layer, provided through the use of a combination of cryptographic and protocol security mechanisms.

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as described in RFC 2119 [Bra97].

1.2 Audience

The target audience for this document includes implementers of this IP security technology and others interested in gaining a general background understanding of this system. In particular, prospective users of this technology (end users or system administrators) are part of the target audience. A glossary is provided as an appendix to help fill in gaps in background/vocabulary. This document assumes that the reader is familiar with the Internet Protocol, related networking technology, and general security terms and concepts.

1.3 Related Documents

As mentioned above, other documents provide detailed definitions of some of the components of IPsec and of their inter-relationship. They include RFCs on the following topics:

a. "IP Security Document Roadmap" [TDG97] -- a document providing guidelines for specifications describing encryption and authentication algorithms used in this system.
b. security protocols -- RFCs describing the Authentication Header (AH) [KA98a] and Encapsulating Security Payload (ESP) [KA98b] protocols.
c. algorithms for authentication and encryption -- a separate RFC for each algorithm.
d. automatic key management -- RFCs on "The Internet Key Exchange (IKE)" [HC98], "Internet Security Association and Key Management Protocol (ISAKMP)" [MSST97],"The OAKLEY Key Determination Protocol" [Orm97], and "The Internet IP Security Domain of Interpretation for ISAKMP" [Pip98].

2. Design Objectives

2.1 Goals/Objectives/Requirements/Problem Description

IPsec is designed to provide interoperable, high quality, cryptographically-based security for IPv4 and IPv6. The set of security services offered includes access control, connectionless integrity, data origin authentication, protection against replays (a form of partial sequence integrity), confidentiality (encryption), and limited traffic flow confidentiality. These services are provided at the IP layer, offering protection for IP and/or upper layer protocols.

These objectives are met through the use of two traffic security protocols, the Authentication Header (AH) and the Encapsulating Security Payload (ESP), and through the use of cryptographic key management procedures and protocols. The set of IPsec protocols employed in any context, and the ways in which they are employed, will be determined by the security and system requirements of users, applications, and/or sites/organizations.

When these mechanisms are correctly implemented and deployed, they ought not to adversely affect users, hosts, and other Internet components that do not employ these security mechanisms for protection of their traffic. These mechanisms also are designed to be algorithm-independent. This modularity permits selection of different sets of algorithms without affecting the other parts of the implementation. For example, different user communities may select different sets of algorithms (creating cliques) if required.

A standard set of default algorithms is specified to facilitate interoperability in the global Internet. The use of these algorithms, in conjunction with IPsec traffic protection and key management protocols, is intended to permit system and application developers to deploy high quality, Internet layer, cryptographic security technology.

2.2 Caveats and Assumptions

The suite of IPsec protocols and associated default algorithms are designed to provide high quality security for Internet traffic. However, the security offered by use of these protocols ultimately depends on the quality of the their implementation, which is outside the scope of this set of standards. Moreover, the security of a computer system or network is a function of many factors, including personnel, physical, procedural, compromising emanations, and computer security practices. Thus IPsec is only one part of an overall system security architecture.

Finally, the security afforded by the use of IPsec is critically dependent on many aspects of the operating environment in which the IPsec implementation executes. For example, defects in OS security, poor quality of random number sources, sloppy system management protocols and practices, etc. can all degrade the security provided by IPsec. As above, none of these environmental attributes are within the scope of this or other IPsec standards.

3. System Overview

This section provides a high level description of how IPsec works, the components of the system, and how they fit together to provide the security services noted above. The goal of this description is to enable the reader to "picture" the overall process/system, see how it fits into the IP environment, and to provide context for later sections of this document, which describe each of the components in more detail.

An IPsec implementation operates in a host or a security gateway environment, affording protection to IP traffic. The protection offered is based on requirements defined by a Security Policy Database (SPD) established and maintained by a user or system administrator, or by an application operating within constraints established by either of the above. In general, packets are selected for one of three processing modes based on IP and transport layer header information (Selectors, Section 4.4.2) matched against entries in the database (SPD). Each packet is either afforded IPsec security services, discarded, or allowed to bypass IPsec, based on the applicable database policies identified by the Selectors.

3.1 What IPsec Does

IPsec provides security services at the IP layer by enabling a system to select required security protocols, determine the algorithm(s) to use for the service(s), and put in place any cryptographic keys required to provide the requested services. IPsec can be used to protect one or more "paths" between a pair of hosts, between a pair of security gateways, or between a security gateway and a host. (The term "security gateway" is used throughout the IPsec documents to refer to an intermediate system that implements IPsec protocols. For example, a router or a firewall implementing IPsec is a security gateway.)

The set of security services that IPsec can provide includes access control, connectionless integrity, data origin authentication, rejection of replayed packets (a form of partial sequence integrity), confidentiality (encryption), and limited traffic flow confidentiality. Because these services are provided at the IP layer, they can be used by any higher layer protocol, e.g., TCP, UDP, ICMP, BGP, etc.

The IPsec DOI also supports negotiation of IP compression [SMPT98], motivated in part by the observation that when encryption is employed within IPsec, it prevents effective compression by lower protocol layers.

3.2 How IPsec Works

IPsec uses two protocols to provide traffic security -- Authentication Header (AH) and Encapsulating Security Payload (ESP). Both protocols are described in more detail in their respective RFCs [KA98a, KA98b].

- o The IP Authentication Header (AH) [KA98a] provides connectionless integrity, data origin authentication, and an optional anti-replay service.
- o The Encapsulating Security Payload (ESP) protocol [KA98b] may provide confidentiality (encryption), and limited traffic flow confidentiality. It also may provide connectionless

integrity, data origin authentication, and an anti-replay service. (One or the other set of these security services must be applied whenever ESP is invoked.)

- o Both AH and ESP are vehicles for access control, based on the distribution of cryptographic keys and the management of traffic flows relative to these security protocols.

These protocols may be applied alone or in combination with each other to provide a desired set of security services in IPv4 and IPv6. Each protocol supports two modes of use: transport mode and tunnel mode. In transport mode the protocols provide protection primarily for upper layer protocols; in tunnel mode, the protocols are applied to tunneled IP packets. The differences between the two modes are discussed in Section 4.

IPsec allows the user (or system administrator) to control the granularity at which a security service is offered. For example, one can create a single encrypted tunnel to carry all the traffic between two security gateways or a separate encrypted tunnel can be created for each TCP connection between each pair of hosts communicating across these gateways. IPsec management must incorporate facilities for specifying:

- o which security services to use and in what combinations
- o the granularity at which a given security protection should be applied
- o the algorithms used to effect cryptographic-based security

Because these security services use shared secret values (cryptographic keys), IPsec relies on a separate set of mechanisms for putting these keys in place. (The keys are used for authentication/integrity and encryption services.) This document requires support for both manual and automatic distribution of keys. It specifies a specific public-key based approach (IKE -- [MSST97, Orm97, HC98]) for automatic key management, but other automated key distribution techniques MAY be used. For example, KDC-based systems such as Kerberos and other public-key systems such as SKIP could be employed.

3.3 Where IPsec May Be Implemented

There are several ways in which IPsec may be implemented in a host or in conjunction with a router or firewall (to create a security gateway). Several common examples are provided below:

a. Integration of IPsec into the native IP implementation. This requires access to the IP source code and is applicable to both hosts and security gateways.

b. "Bump-in-the-stack" (BITS) implementations, where IPsec is implemented "underneath" an existing implementation of an IP protocol stack, between the native IP and the local network drivers. Source code access for the IP stack is not required in this context, making this implementation approach appropriate for use with legacy systems. This approach, when it is adopted, is usually employed in hosts.

c. The use of an outboard crypto processor is a common design feature of network security systems used by the military, and of some commercial systems as well. It is sometimes referred to as a "Bump-in-the-wire" (BITW) implementation. Such implementations may be designed to serve either a host or a gateway (or both). Usually the BITW device is IP addressable. When supporting a single host, it may be quite analogous to a BITS implementation, but in supporting a router or firewall, it must operate like a security gateway.

4. Security Associations

This section defines Security Association management requirements for all IPv6 implementations and for those IPv4 implementations that implement AH, ESP, or both. The concept of a "Security Association" (SA) is fundamental to IPsec. Both AH and ESP make use of SAs and a major function of IKE is the establishment and maintenance of Security Associations. All implementations of AH or ESP MUST support the concept of a Security Association as described below. The remainder of this section describes various aspects of Security Association management, defining required characteristics for SA policy management, traffic processing, and SA management techniques.

4.1 Definition and Scope

A Security Association (SA) is a simplex "connection" that affords security services to the traffic carried by it. Security services are afforded to an SA by the use of AH, or ESP, but not both. If both AH and ESP protection is applied to a traffic stream, then two (or more) SAs are created to afford protection to the traffic stream. To secure typical, bi-directional communication between two hosts, or between two security gateways, two Security Associations (one in each direction) are required.

A security association is uniquely identified by a triple consisting of a Security Parameter Index (SPI), an IP Destination Address, and a security protocol (AH or ESP) identifier. In principle, the Destination Address may be a unicast address, an IP broadcast address, or a multicast group address. However, IPsec SA management mechanisms currently are defined only for unicast SAs. Hence, in the discussions that follow, SAs will be described in the context of point-to-point communication, even though the concept is applicable in the point-to-multipoint case as well.

As noted above, two types of SAs are defined: transport mode and tunnel mode. A transport mode SA is a security association between two hosts. In IPv4, a transport mode security protocol header appears immediately after the IP header and any options, and before any higher layer protocols (e.g., TCP or UDP). In IPv6, the security protocol header appears after the base IP header and extensions, but may appear before or after destination options, and before higher layer protocols. In the case of ESP, a transport mode SA provides security services only for these higher layer protocols, not for the IP header or any extension headers preceding the ESP header. In the case of AH, the protection is also extended to selected portions of the IP header, selected portions of extension headers, and selected options (contained in the IPv4 header, IPv6 Hop-by-Hop extension header, or IPv6 Destination extension headers). For more details on the coverage afforded by AH, see the AH specification [KA98a].

A tunnel mode SA is essentially an SA applied to an IP tunnel. Whenever either end of a security association is a security gateway, the SA MUST be tunnel mode. Thus an SA between two security gateways is always a tunnel mode SA, as is an SA between a host and a security gateway. Note that for the case where traffic is destined for a security gateway, e.g., SNMP commands, the security gateway is acting as a host and transport mode is allowed. But in that case, the security gateway is not acting as a gateway, i.e., not transiting traffic. Two hosts MAY establish a tunnel mode SA between themselves. The requirement for any (transit traffic) SA involving a security gateway to be a tunnel SA arises due to the need to avoid potential problems with regard to fragmentation and reassembly of IPsec packets, and in circumstances where multiple paths (e.g., via different security gateways) exist to the same destination behind the security gateways.

For a tunnel mode SA, there is an "outer" IP header that specifies the IPsec processing destination, plus an "inner" IP header that specifies the (apparently) ultimate destination for the packet. The security protocol header appears after the outer IP header, and before the inner IP header. If AH is employed in tunnel mode, portions of the outer IP header are afforded protection (as above), as well as all of the tunneled IP packet (i.e., all of the inner IP header is protected, as well as higher layer protocols). If ESP is employed, the protection is afforded only to the tunneled packet, not to the outer header.

In summary,

a) A host MUST support both transport and tunnel mode.

b) A security gateway is required to support only tunnel mode. If it supports transport mode, that should be used only when the security gateway is acting as a host, e.g., for network management.

4.2 Security Association Functionality

The set of security services offered by an SA depends on the security protocol selected, the SA mode, the endpoints of the SA, and on the election of optional services within the protocol. For example, AH provides data origin authentication and connectionless integrity for IP datagrams (hereafter referred to as just "authentication"). The "precision" of the authentication service is a function of the granularity of the security association with which AH is employed, as discussed in Section 4.4.2, "Selectors".

AH also offers an anti-replay (partial sequence integrity) service at the discretion of the receiver, to help counter denial of service attacks. AH is an appropriate protocol to employ when confidentiality is not required (or is not permitted, e.g , due to government restrictions on use of encryption). AH also provides authentication for selected portions of the IP header, which may be necessary in some contexts. For example, if the integrity of an IPv4 option or IPv6 extension header must be protected en route between sender and receiver, AH can provide this service (except for the non-predictable but mutable parts of the IP header.)

ESP optionally provides confidentiality for traffic. (The strength of the confidentiality service depends in part, on the encryption algorithm employed.) ESP also may optionally provide authentication (as defined above). If authentication is negotiated for an ESP SA, the receiver also may elect to enforce an anti-replay service with the same features as the AH anti-replay service. The scope of the authentication offered by ESP is narrower than for AH, i.e., the IP header(s) "outside" the ESP header is(are) not protected. If only the upper layer protocols need to be authenticated, then ESP authentication is an appropriate choice and is more space efficient than use of AH encapsulating ESP. Note that although both confidentiality and authentication are optional, they cannot both be omitted. At least one of them MUST be selected.

If confidentiality service is selected, then an ESP (tunnel mode) SA between two security gateways can offer partial traffic flow confidentiality. The use of tunnel mode allows the inner IP headers to be encrypted, concealing the identities of the (ultimate) traffic source and destination. Moreover, ESP payload padding also can be invoked to hide the size of the packets, further concealing the external characteristics of the traffic. Similar traffic flow confidentiality services may be offered when a mobile user is assigned a dynamic IP address in a dialup context, and establishes a (tunnel mode) ESP SA to a corporate firewall (acting as a security gateway). Note that fine granularity SAs generally are more vulnerable to traffic analysis than coarse granularity ones which are carrying traffic from many subscribers.

4.3 Combining Security Associations

The IP datagrams transmitted over an individual SA are afforded protection by exactly one security protocol, either AH or ESP, but not both. Sometimes a security policy may call for a combination of services for a particular traffic flow that is not achievable with a single SA. In such instances it will be necessary to employ multiple SAs to implement the required security policy. The term "security association bundle" or "SA bundle" is applied to a sequence of SAs through which traffic must be processed to satisfy a security policy. The order of the sequence is defined by the policy. (Note that the SAs that comprise a bundle may terminate at different endpoints. For example, one SA may extend between a mobile host and a security gateway and a second, nested SA may extend to a host behind the gateway.)

Security associations may be combined into bundles in two ways: transport adjacency and iterated tunneling.

- o Transport adjacency refers to applying more than one security protocol to the same IP datagram, without invoking tunneling. This approach to combining AH and ESP allows for only one level of combination; further nesting yields no added benefit (assuming use of adequately strong algorithms in each protocol) since the processing is performed at one IPsec instance at the (ultimate) destination.

```
Host 1 --- Security ---- Internet -- Security --- Host 2
 | |        Gwy 1                     Gwy 2        | |
 | |                                               | |
 | -----Security Association 1 (ESP transport)------- |
 |                                                    |
 -------Security Association 2 (AH transport)----------
```

- o Iterated tunneling refers to the application of multiple layers of security protocols effected through IP tunneling. This approach allows for multiple levels of nesting, since each tunnel can originate or terminate at a different IPsec site along the path. No special treatment is expected for ISAKMP traffic at intermediate security gateways other than what can be specified through appropriate SPD entries (See Case 3 in Section 4.5)

There are 3 basic cases of iterated tunneling -- support is required only for cases 2 and 3.:

1. both endpoints for the SAs are the same -- The inner and outer tunnels could each be either AH or ESP, though it is unlikely that Host 1 would specify both to be the same, i.e., AH inside of AH or ESP inside of ESP.

```
Host 1 --- Security ---- Internet -- Security --- Host 2
 | |        Gwy 1                     Gwy 2        | |
 | |                                               | |
 | -------Security Association 1 (tunnel)---------- | |
 |                                                   |
 ---------Security Association 2 (tunnel)--------------
```

2. one endpoint of the SAs is the same -- The inner and uter tunnels could each be either AH or ESP.

```
Host 1 --- Security ---- Internet -- Security --- Host 2
 | |        Gwy 1                     Gwy 2          |
 | |                                    |            |
 | ----Security Association 1 (tunnel)----           |
 |                                                   |
 ---------Security Association 2 (tunnel)-------------
```

3. neither endpoint is the same -- The inner and outer tunnels could each be either AH or ESP.

```
Host 1 --- Security ---- Internet -- Security --- Host 2
 |          Gwy 1                     Gwy 2          |
 |            |                         |            |
 |            --Security Assoc 1 (tunnel)-           |
 |                                                   |
 -----------Security Association 2 (tunnel)-----------
```

These two approaches also can be combined, e.g., an SA bundle could be constructed from one tunnel mode SA and one or two transport mode SAs, applied in sequence. (See Section 4.5 "Basic Combinations of Security Associations.") Note that nested tunnels can also occur where neither the source nor the destination endpoints of any of the tunnels are the same. In that case, there would be no host or security gateway with a bundle corresponding to the nested tunnels.

For transport mode SAs, only one ordering of security protocols seems appropriate. AH is applied to both the upper layer protocols and (parts of) the IP header. Thus if AH is used in a transport mode, in conjunction with ESP, AH SHOULD appear as the first header after IP, prior to the appearance of ESP. In that context, AH is applied to the ciphertext output of ESP. In contrast, for tunnel mode SAs, one can imagine uses for various orderings of AH and ESP. The required set of SA bundle types that MUST be supported by a compliant IPsec implementation is described in Section 4.5.

4.4 Security Association Databases

Many of the details associated with processing IP traffic in an IPsec implementation are largely a local matter, not subject to standardization. However, some external aspects of the processing must be standardized, to ensure interoperability and to provide a minimum management capability that is essential for productive use of IPsec. This section describes a general model for processing IP traffic relative to security associations, in support of these interoperability and functionality goals. The model described below is nominal; compliant implementations need not match details of this model as presented, but the external behavior of such implementations must be mappable to the externally observable characteristics of this model.

There are two nominal databases in this model: the Security Policy Database and the Security Association Database. The former specifies the policies that determine the disposition of all IP traffic inbound or outbound from a host, security gateway, or BITS or BITW IPsec implementation. The latter database contains parameters that are associated with each (active) security association. This section also defines the concept of a Selector, a set of IP and upper layer protocol field values that is used by the Security Policy Database to map traffic to a policy, i.e., an SA (or SA bundle).

Each interface for which IPsec is enabled requires nominally separate inbound vs. outbound databases (SAD and SPD), because of the directionality of many of the fields that are used as selectors. Typically there is just one such interface, for a host or security gateway (SG). Note that an SG would always have at least 2 interfaces, but the "internal" one to the corporate net, usually would not have IPsec enabled and so only one pair of SADs and one pair of SPDs would be needed. On the other hand, if a host had multiple interfaces or an SG had multiple external interfaces, it might be necessary to have separate SAD and SPD pairs for each interface.

4.4.1 The Security Policy Database (SPD)

Ultimately, a security association is a management construct used to enforce a security policy in the IPsec environment. Thus an essential element of SA processing is an underlying Security Policy Database (SPD) that specifies what services are to be offered to IP datagrams and in what fashion. The form of the database and its interface are outside the scope of this specification. However, this section does specify certain minimum management functionality that must be provided, to allow a user or system administrator to control how IPsec is applied to traffic transmitted or received by a host or transiting a security gateway.

The SPD must be consulted during the processing of all traffic (INBOUND and OUTBOUND), including non-IPsec traffic. In order to support this, the SPD requires distinct entries for inbound and outbound traffic. One can think of this as separate SPDs (inbound vs. outbound). In addition, a nominally separate SPD must be provided for each IPsec-enabled interface.

An SPD must discriminate among traffic that is afforded IPsec protection and traffic that is allowed to bypass IPsec. This applies to the IPsec protection to be applied by a sender and to the IPsec protection that must be present at the receiver. For any outbound or inbound datagram, three processing choices are possible: discard, bypass IPsec, or apply IPsec. The first choice refers to traffic that is not allowed to exit the host, traverse the security gateway, or be delivered to an application at all. The second choice refers to traffic that is allowed to pass without additional IPsec protection. The third choice refers to traffic that is afforded IPsec protection, and for such traffic the SPD must specify the security services to be provided, protocols to be employed, algorithms to be used, etc.

For every IPsec implementation, there MUST be an administrative interface that allows a user or system administrator to manage the SPD. Specifically, every inbound or outbound packet is subject to processing by IPsec and the SPD must specify what action will be taken in each case. Thus the administrative interface must allow the user (or system administrator) to specify the security processing to be applied to any packet entering or exiting the system, on a packet by packet basis. (In a host IPsec implementation making use of a socket interface, the SPD may not need to be consulted on a per packet basis, but the effect is still the same.) The management interface for the SPD MUST allow creation of entries consistent with the selectors defined in Section 4.4.2, and MUST support (total) ordering of these entries. It is expected that through the use of wildcards in various selector fields, and because all packets on a single UDP or TCP connection will tend to match a single SPD entry, this requirement will not impose an unreasonably detailed level of SPD specification. The selectors are analogous to what are found in a stateless firewall or filtering router and which are currently manageable this way.

In host systems, applications MAY be allowed to select what security processing is to be applied to the traffic they generate and consume. (Means of signalling such requests to the IPsec implementation are outside the scope of this standard.) However, the system administrator MUST be able to specify whether or not a user or application can override (default) system policies. Note that application specified policies may satisfy system requirements, so that the system may not need to do additional IPsec processing beyond that needed to meet an application's requirements. The form of the management interface is not specified by this document and may differ for hosts vs. security gateways, and within hosts the interface may differ for socket-based vs. BITS implementations. However, this document does specify a standard set of SPD elements that all IPsec implementations MUST support.

The SPD contains an ordered list of policy entries. Each policy entry is keyed by one or more selectors that define the set of IP traffic encompassed by this policy entry. (The required selector types are defined in Section 4.4.2.) These define the granularity of policies or SAs. Each entry includes an indication of whether traffic matching this policy will be bypassed, discarded, or subject to IPsec processing. If IPsec processing is to be applied, the entry includes an SA (or SA bundle) specification, listing the IPsec protocols, modes, and algorithms to be employed, including any nesting requirements. For example, an entry may call for all matching traffic to be protected by ESP in transport mode using 3DES-CBC with an explicit IV, nested inside of AH in tunnel mode using HMAC/SHA-1. For each selector, the policy entry specifies how to derive the corresponding values for a new Security Association Database (SAD, see Section 4.4.3) entry from those in the SPD and the packet (Note that at present, ranges are only supported for IP addresses; but wildcarding can be expressed for all selectors):

a. use the value in the packet itself -- This will limit use of the SA to those packets which have this packet's value for the selector even if the selector for the policy entry has a range of allowed values or a wildcard for this selector.
b. use the value associated with the policy entry -- If this were to be just a single value, then there would be no difference between (b) and (a). However, if the allowed values for the selector are a range (for IP addresses) or

wildcard, then in the case of a range,(b) would enable use of the SA by any packet with a selector value within the range not just by packets with the selector value of the packet that triggered the creation of the SA. In the case of a wildcard, (b) would allow use of the SA by packets with any value for this selector.

For example, suppose there is an SPD entry where the allowed value for source address is any of a range of hosts (192.168.2.1 to 192.168.2.10). And suppose that a packet is to be sent that has a source address of 192.168.2.3. The value to be used for the SA could be any of the sample values below depending on what the policy entry for this selector says is the source of the selector value:

source for the value to be used in the SA	example of new SAD selector value
a. packet	192.168.2.3 (one host)
b. SPD entry	192.168.2.1 to 192.168.2.10 (range of hosts)

Note that if the SPD entry had an allowed value of wildcard for the source address, then the SAD selector value could be wildcard (any host). Case (a) can be used to prohibit sharing, even among packets that match the same SPD entry.

As described below in Section 4.4.3, selectors may include "wildcard" entries and hence the selectors for two entries may overlap. (This is analogous to the overlap that arises with ACLs or filter entries in routers or packet filtering firewalls.) Thus, to ensure consistent, predictable processing, SPD entries MUST be ordered and the SPD MUST always be searched in the same order, so that the first matching entry is consistently selected. (This requirement is necessary as the effect of processing traffic against SPD entries must be deterministic, but there is no way to canonicalize SPD entries given the use of wildcards for some selectors.) More detail on matching of packets against SPD entries is provided in Section 5.

Note that if ESP is specified, either (but not both) authentication or encryption can be omitted. So it MUST be possible to configure the SPD value for the authentication or encryption algorithms to be "NULL". However, at least one of these services MUST be selected, i.e., it MUST NOT be possible to configure both of them as "NULL".

The SPD can be used to map traffic to specific SAs or SA bundles. Thus it can function both as the reference database for security policy and as the map to existing SAs (or SA bundles). (To accommodate the bypass and discard policies cited above, the SPD also

MUST provide a means of mapping traffic to these functions, even though they are not, per se, IPsec processing.) The way in which the SPD operates is different for inbound vs. outbound traffic and it also may differ for host vs. security gateway, BITS, and BITW implementations. Sections 5.1 and 5.2 describe the use of the SPD for outbound and inbound processing, respectively.

Because a security policy may require that more than one SA be applied to a specified set of traffic, in a specific order, the policy entry in the SPD must preserve these ordering requirements, when present. Thus, it must be possible for an IPsec implementation to determine that an outbound or inbound packet must be processed thorough a sequence of SAs. Conceptually, for outbound processing, one might imagine links (to the SAD) from an SPD entry for which there are active SAs, and each entry would consist of either a single SA or an ordered list of SAs that comprise an SA bundle. When a packet is matched against an SPD entry and there is an existing SA or SA bundle that can be used to carry the traffic, the processing of the packet is controlled by the SA or SA bundle entry on the list. For an inbound IPsec packet for which multiple IPsec SAs are to be applied, the lookup based on destination address, IPsec protocol, and SPI should identify a single SA.

The SPD is used to control the flow of ALL traffic through an IPsec system, including security and key management traffic (e.g., ISAKMP) from/to entities behind a security gateway. This means that ISAKMP traffic must be explicitly accounted for in the SPD, else it will be discarded. Note that a security gateway could prohibit traversal of encrypted packets in various ways, e.g., having a DISCARD entry in the SPD for ESP packets or providing proxy key exchange. In the latter case, the traffic would be internally routed to the key management module in the security gateway.

4.4.2 Selectors

An SA (or SA bundle) may be fine-grained or coarse-grained, depending on the selectors used to define the set of traffic for the SA. For example, all traffic between two hosts may be carried via a single SA, and afforded a uniform set of security services. Alternatively, traffic between a pair of hosts might be spread over multiple SAs, depending on the applications being used (as defined by the Next Protocol and Port fields), with different security services offered by different SAs. Similarly, all traffic between a pair of security gateways could be carried on a single SA, or one SA could be assigned for each communicating host pair. The following selector parameters MUST be supported for SA management to facilitate control of SA granularity. Note that in the case of receipt of a packet with an ESP header, e.g., at an encapsulating security gateway or BITW implementation, the transport layer protocol, source/destination ports, and Name (if present) may be "OPAQUE", i.e., inaccessible because of encryption or fragmentation. Note also that both Source and Destination addresses should either be IPv4 or IPv6.

- Destination IP Address (IPv4 or IPv6): this may be a single IP address (unicast, anycast, broadcast (IPv4 only), or multicast group), a range of addresses (high and low values (inclusive), address + mask, or a wildcard address. The last three are used to support more than one destination system sharing the same SA (e.g., behind a security gateway). Note that this selector is conceptually different from the "Destination IP Address" field in the <Destination IP Address, IPsec Protocol, SPI> tuple used to uniquely identify an SA. When a tunneled packet arrives at the tunnel endpoint, its SPI/Destination address/Protocol are used to look up the SA for this packet in the SAD. This destination address comes from the encapsulating IP header. Once the packet has been processed according to the tunnel SA and has come out of the tunnel, its selectors are "looked up" in the Inbound SPD. The Inbound SPD has a selector called destination address. This IP destination address is the one in the inner (encapsulated) IP header. In the case of a transport'd packet, there will be only one IP header and this ambiguity does not exist. [REQUIRED for all implementations]

- Source IP Address(es) (IPv4 or IPv6): this may be a single IP address (unicast, anycast, broadcast (IPv4 only), or multicast group), range of addresses (high and low values inclusive), address + mask, or a wildcard address. The last three are used to support more than one source system sharing the same SA (e.g., behind a security gateway or in a multihomed host). [REQUIRED for all implementations]

- Name: There are 2 cases (Note that these name forms are supported in the IPsec DOI.)
    1. User ID
        a. a fully qualified user name string (DNS), e.g., mozart@foo.bar.com
        b. X.500 distinguished name, e.g., C = US, SP = MA, O = GTE Internetworking, CN = Stephen T. Kent.
    2. System name (host, security gateway, etc.)
        a. a fully qualified DNS name, e.g., foo.bar.com
        b. X.500 distinguished name
        c. X.500 general name

  NOTE: One of the possible values of this selector is "OPAQUE".

[REQUIRED for the following cases. Note that support for name forms other than addresses is not required for manually keyed SAs.

  - o User ID
    - native host implementations
    - BITW and BITS implementations acting as HOSTS with only one user
    - security gateway implementations for INBOUND processing.
  - o System names -- all implementations]

- Data sensitivity level: (IPSO/CIPSO labels)
  [REQUIRED for all systems providing information flow security as per Section 8, OPTIONAL for all other systems.]

- Transport Layer Protocol: Obtained from the IPv4 "Protocol" or the IPv6 "Next Header" fields. This may be an individual protocol number. These packet fields may not contain the Transport Protocol due to the presence of IP extension headers, e.g., a Routing Header, AH, ESP, Fragmentation Header, Destination Options, Hop-by-hop options, etc. Note that the Transport Protocol may not be available in the case of receipt of a packet with an ESP header, thus a value of "OPAQUE" SHOULD be supported.
  [REQUIRED for all implementations]

  NOTE: To locate the transport protocol, a system has to chain through the packet headers checking the "Protocol" or "Next Header" field until it encounters either one it recognizes as a transport protocol, or until it reaches one that isn't on its list of extension headers, or until it encounters an ESP header that renders the transport protocol opaque.

- Source and Destination (e.g., TCP/UDP) Ports: These may be individual UDP or TCP port values or a wildcard port. (The use of the Next Protocol field and the Source and/or Destination Port fields (in conjunction with the Source and/or Destination Address fields), as an SA selector is sometimes referred to as "session-oriented keying."). Note that the source and destination ports may not be available in the case of receipt of a packet with an ESP header, thus a value of "OPAQUE" SHOULD be supported.

  The following table summarizes the relationship between the "Next Header" value in the packet and SPD and the derived Port Selector value for the SPD and SAD.

Next Hdr in Packet	Transport Layer Protocol in SPD	Derived Port Selector Field Value in SPD and SAD
ESP	ESP or ANY	ANY (i.e., don't look at it)
-don't care-	ANY	ANY (i.e., don't look at it)
specific value fragment	specific value	NOT ANY (i.e., drop packet)
specific value not fragment	specific value	actual port selector field

If the packet has been fragmented, then the port information may not be available in the current fragment. If so, discard the fragment. An ICMP PMTU should be sent for the first fragment, which will have the port information. [MAY be supported]

The IPsec implementation context determines how selectors are used. For example, a host implementation integrated into the stack may make use of a socket interface. When a new connection is established the SPD can be consulted and an SA (or SA bundle) bound to the socket. Thus traffic sent via that socket need not result in additional lookups to the SPD/SAD. In contrast, a BITS, BITW, or security gateway implementation needs to look at each packet and perform an SPD/SAD lookup based on the selectors. The allowable values for the selector fields differ between the traffic flow, the security association, and the security policy.

The following table summarizes the kinds of entries that one needs to be able to express in the SPD and SAD. It shows how they relate to the fields in data traffic being subjected to IPsec screening. (Note: the "wild" or "wildcard" entry for src and dst addresses includes a mask, range, etc.)

Field	Traffic Value	SAD Entry	SPD Entry
src addr	single IP addr	single,range,wild	single,range,wildcard
dst addr	single IP addr	single,range,wild	single,range,wildcard
xpt protocol*	xpt protocol	single,wildcard	single,wildcard
src port*	single src port	single,wildcard	single,wildcard
dst port*	single dst port	single,wildcard	single,wildcard
user id*	single user id	single,wildcard	single,wildcard
sec. labels	single value	single,wildcard	single,wildcard

* The SAD and SPD entries for these fields could be "OPAQUE" because the traffic value is encrypted.

NOTE: In principle, one could have selectors and/or selector values in the SPD which cannot be negotiated for an SA or SA bundle. Examples might include selector values used to select traffic for discarding or enumerated lists which cause a separate SA to be created for each item on the list. For now, this is left for future versions of this document and the list of required selectors and selector values is the same for the SPD and the SAD. However, it is acceptable to have an administrative interface that supports use of selector values which cannot be negotiated provided that it does not mislead the user into believing it is creating an SA with these selector values. For example, the interface may allow the user to specify an enumerated list of values but would result in the creation of a separate policy and SA for each item on the list. A vendor might support such an interface to make it easier for its customers to specify clear and concise policy specifications.

4.4.3 Security Association Database (SAD)

In each IPsec implementation there is a nominal Security Association Database, in which each entry defines the parameters associated with one SA. Each SA has an entry in the SAD. For outbound processing, entries are pointed to by entries in the SPD. Note that if an SPD entry does not currently point to an SA that is appropriate for the packet, the implementation creates an appropriate SA (or SA Bundle) and links the SPD entry to the SAD entry (see Section 5.1.1). For inbound processing, each entry in the SAD is indexed by a destination IP address, IPsec protocol type, and SPI. The following parameters are associated with each entry in the SAD. This description does not purport to be a MIB, but only a specification of the minimal data items required to support an SA in an IPsec implementation.

For inbound processing: The following packet fields are used to look up the SA in the SAD:

- o Outer Header's Destination IP address: the IPv4 or IPv6 Destination address.
  [REQUIRED for all implementations]
- o IPsec Protocol: AH or ESP, used as an index for SA lookup in this database. Specifies the IPsec protocol to be applied to the traffic on this SA.
  [REQUIRED for all implementations]
- o SPI: the 32-bit value used to distinguish among different SAs terminating at the same destination and using the same IPsec protocol.
  [REQUIRED for all implementations]

For each of the selectors defined in Section 4.4.2, the SA entry in the SAD MUST contain the value or values which were negotiated at the time the SA was created. For the sender, these values are used to decide whether a given SA is appropriate for use with an outbound packet. This is part of checking to see if there is an existing SA that can be used. For the receiver, these values are used to check that the selector values in an inbound packet match those for the SA (and thus indirectly those for the matching policy). For the receiver, this is part of verifying that the SA was appropriate for this packet. (See Section 6 for rules for ICMP messages.) These fields can have the form of specific values, ranges, wildcards, or "OPAQUE" as described in section 4.4.2, "Selectors". Note that for an ESP SA, the encryption algorithm or the authentication algorithm could be "NULL". However they MUST not both be "NULL".

The following SAD fields are used in doing IPsec processing:

- o Sequence Number Counter: a 32-bit value used to generate the Sequence Number field in AH or ESP headers.
  [REQUIRED for all implementations, but used only for outbound traffic.]
- o Sequence Counter Overflow: a flag indicating whether overflow of the Sequence Number Counter should generate an auditable event and prevent transmission of additional packets on the SA.
  [REQUIRED for all implementations, but used only for outbound traffic.]
- o Anti-Replay Window: a 32-bit counter and a bit-map (or equivalent) used to determine whether an inbound AH or ESP packet is a replay.
  [REQUIRED for all implementations but used only for inbound traffic. NOTE: If anti-replay has been disabled by the receiver, e.g., in the case of a manually keyed SA, then the Anti-Replay Window is not used.]
- o AH Authentication algorithm, keys, etc.
  [REQUIRED for AH implementations]
- o ESP Encryption algorithm, keys, IV mode, IV, etc.
  [REQUIRED for ESP implementations]
- o ESP authentication algorithm, keys, etc. If the authentication service is not selected, this field will be null.
  [REQUIRED for ESP implementations]
- o Lifetime of this Security Association: a time interval after which an SA must be replaced with a new SA (and new SPI) or terminated, plus an indication of which of these actions should occur. This may be expressed as a time or byte count, or a simultaneous use of both, the first lifetime to expire taking precedence. A compliant implementation MUST support both types of lifetimes, and must support a simultaneous use of both. If time is employed, and if IKE employs X.509 certificates for SA establishment, the SA lifetime must be constrained by the validity intervals of the certificates, and the NextIssueDate of the CRLs used in the IKE exchange

for the SA. Both initiator and responder are responsible for constraining SA lifetime in this fashion.
[REQUIRED for all implementations]

NOTE: The details of how to handle the refreshing of keys when SAs expire is a local matter. However, one reasonable approach is:

(a) If byte count is used, then the implementation SHOULD count the number of bytes to which the IPsec algorithm is applied. For ESP, this is the encryption algorithm (including Null encryption) and for AH, this is the authentication algorithm. This includes pad bytes, etc. Note that implementations SHOULD be able to handle having the counters at the ends of an SA get out of synch, e.g., because of packet loss or because the implementations at each end of the SA aren't doing things the same way.

(b) There SHOULD be two kinds of lifetime -- a soft lifetime which warns the implementation to initiate action such as setting up a replacement SA and a hard lifetime when the current SA ends.

(c) If the entire packet does not get delivered during the SAs lifetime, the packet SHOULD be discarded.

o IPsec protocol mode: tunnel, transport or wildcard. Indicates which mode of AH or ESP is applied to traffic on this SA. Note that if this field is "wildcard" at the sending end of the SA, then the application has to specify the mode to the IPsec implementation. This use of wildcard allows the same SA to be used for either tunnel or transport mode traffic on a per packet basis, e.g., by different sockets. The receiver does not need to know the mode in order to properly process the packet's IPsec headers.

[REQUIRED as follows, unless implicitly defined by context:
- host implementations must support all modes
- gateway implementations must support tunnel mode]

NOTE: The use of wildcard for the protocol mode of an inbound SA may add complexity to the situation in the receiver (host only). Since the packets on such an SA could be delivered in either tunnel or transport mode, the security of an incoming packet could depend in part on which mode had been used to deliver it. If, as a result, an application cared about the SA mode of a given packet, then the application would need a mechanism to obtain this mode information.

o Path MTU: any observed path MTU and aging variables. See Section 6.1.2.4
[REQUIRED for all implementations but used only for outbound traffic]

4.5 Basic Combinations of Security Associations

This section describes four examples of combinations of security associations that MUST be supported by compliant IPsec hosts or security gateways. Additional combinations of AH and/or ESP in tunnel and/or transport modes MAY be supported at the discretion of the implementor. Compliant implementations MUST be capable of generating these four combinations and on receipt, of processing them, but SHOULD be able to receive and process any combination. The diagrams and text below describe the basic cases. The legend for the diagrams is:

```
==== = one or more security associations (AH or ESP, transport
       or tunnel)
---- = connectivity (or if so labelled, administrative boundary)
Hx   = host x
SGx  = security gateway x
X*   = X supports IPsec
```

NOTE: The security associations below can be either AH or ESP. The mode (tunnel vs transport) is determined by the nature of the endpoints. For host-to-host SAs, the mode can be either transport or tunnel.

Case 1. The case of providing end-to-end security between 2 hosts across the Internet (or an Intranet).

```
 ====================================
 |                                  |
H1* ------ (Inter/Intranet) ------ H2*
```

Note that either transport or tunnel mode can be selected by the hosts. So the headers in a packet between H1 and H2 could look like any of the following:

```
     Transport               Tunnel
-----------------      ---------------------
1. [IP1][AH][upper]        4. [IP2][AH][IP1][upper]
2. [IP1][ESP][upper]       5. [IP2][ESP][IP1][upper]
3. [IP1][AH][ESP][upper]
```

Note that there is no requirement to support general nesting, but in transport mode, both AH and ESP can be applied to the packet. In this event, the SA establishment procedure MUST ensure that first ESP, then AH are applied to the packet.

Case 2. This case illustrates simple virtual private networks support.

```
                      ===========================
                      |                         |
  ---------------------|----               ---|-----------------------
  |                    |   |               |  |                      |
  |  H1 -- (Local --- SG1* |--- (Internet) ---| SG2* --- (Local --- H2  |
  |        Intranet)       |               |          Intranet)      |
  --------------------------               ---------------------------
      admin. boundary                              admin. boundary
```

Only tunnel mode is required here. So the headers in a packet between SG1 and SG2 could look like either of the following:

```
                Tunnel
        ---------------------
        4. [IP2][AH][IP1][upper]
        5. [IP2][ESP][IP1][upper]
```

Case 3. This case combines cases 1 and 2, adding end-to-end security between the sending and receiving hosts. It imposes no new requirements on the hosts or security gateways, other than a requirement for a security gateway to be configurable to pass IPsec traffic (including ISAKMP traffic) for hosts behind it.

```
   ===============================================================
   |                                                             |
   |                 =========================                   |
   |                 |                       |                   |
  ---|-----------------|----             ---|-------------------|---
  |  |                 |   |             |  |                   |  |
  | H1* -- (Local --- SG1* |-- (Internet) --| SG2* --- (Local --- H2* |
  |        Intranet)       |             |          Intranet)      |
  --------------------------             ---------------------------
       admin. boundary                          admin. boundary
```

Case 4. This covers the situation where a remote host (H1) uses the Internet to reach an organization's firewall (SG2) and to then gain access to some server or other machine (H2). The remote host could be a mobile host (H1) dialing up to a local PPP/ARA server (not shown) on the Internet and then crossing the Internet to the home organization's firewall (SG2), etc. The details of support for this case, (how H1 locates SG2, authenticates it, and verifies its authorization to represent H2) are discussed in Section 4.6.3, "Locating a Security Gateway".

```
======================================================
|                                                    |
|==============================                      |
||                             |                     |
||                          ---|----------------------|---
||                          |  |                      |  |
H1* ----- (Internet) ------| SG2* ---- (Local ----- H2* |
      ^                     |           Intranet)        |
      |                     ------------------------------
could be dialup              admin. boundary (optional)
to PPP/ARA server
```

Only tunnel mode is required between H1 and SG2. So the choices for the SA between H1 and SG2 would be one of the ones in case 2. The choices for the SA between H1 and H2 would be one of the ones in case 1.

Note that in this case, the sender MUST apply the transport header before the tunnel header. Therefore the management interface to the IPsec implementation MUST support configuration of the SPD and SAD to ensure this ordering of IPsec header application.

As noted above, support for additional combinations of AH and ESP is optional. Use of other, optional combinations may adversely affect interoperability.

4.6 SA and Key Management

IPsec mandates support for both manual and automated SA and cryptographic key management. The IPsec protocols, AH and ESP, are largely independent of the associated SA management techniques, although the techniques involved do affect some of the security services offered by the protocols. For example, the optional anti-replay services available for AH and ESP require automated SA management. Moreover, the granularity of key distribution employed with IPsec determines the granularity of authentication provided. (See also a discussion of this issue in Section 4.7.) In general, data origin authentication in AH and ESP is limited by the extent to which secrets used with the authentication algorithm (or with a key management protocol that creates such secrets) are shared among multiple possible sources.

The following text describes the minimum requirements for both types of SA management.

4.6.1 Manual Techniques

The simplest form of management is manual management, in which a person manually configures each system with keying material and security association management data relevant to secure communication with other systems. Manual techniques are practical in small, static environments but they do not scale well. For example, a company could create a Virtual Private Network (VPN) using IPsec in security gateways at several sites. If the number of sites is small, and since all the sites come under the purview of a single administrative domain, this is likely to be a feasible context for manual management techniques. In this case, the security gateway might selectively protect traffic to and from other sites within the organization using a manually configured key, while not protecting traffic for other destinations. It also might be appropriate when only selected communications need to be secured. A similar argument might apply to use of IPsec entirely within an organization for a small number of hosts and/or gateways. Manual management techniques often employ statically configured, symmetric keys, though other options also exist.

4.6.2 Automated SA and Key Management

Widespread deployment and use of IPsec requires an Internet-standard, scalable, automated, SA management protocol. Such support is required to facilitate use of the anti-replay features of AH and ESP, and to accommodate on-demand creation of SAs, e.g., for user- and session-oriented keying. (Note that the notion of "rekeying" an SA actually implies creation of a new SA with a new SPI, a process that generally implies use of an automated SA/key management protocol.)

The default automated key management protocol selected for use with IPsec is IKE [MSST97, Orm97, HC98] under the IPsec domain of interpretation [Pip98]. Other automated SA management protocols MAY be employed.

When an automated SA/key management protocol is employed, the output from this protocol may be used to generate multiple keys, e.g., for a single ESP SA. This may arise because:

- o the encryption algorithm uses multiple keys (e.g., triple DES)
- o the authentication algorithm uses multiple keys
- o both encryption and authentication algorithms are employed

The Key Management System may provide a separate string of bits for each key or it may generate one string of bits from which all of them are extracted. If a single string of bits is provided, care needs to be taken to ensure that the parts of the system that map the string of bits to the required keys do so in the same fashion at both ends of the SA. To ensure that the IPsec implementations at each end of the SA use the same bits for the same keys, and irrespective of which part of the system divides the string of bits into individual keys, the encryption key(s) MUST be taken from the first (left-most, high-order) bits and the authentication key(s) MUST be taken from the remaining bits. The number of bits for each key is defined in the relevant algorithm specification RFC. In the case of multiple encryption keys or multiple authentication keys, the specification for the algorithm must specify the order in which they are to be selected from a single string of bits provided to the algorithm.

4.6.3 Locating a Security Gateway

This section discusses issues relating to how a host learns about the existence of relevant security gateways and once a host has contacted these security gateways, how it knows that these are the correct security gateways. The details of where the required information is stored is a local matter.

Consider a situation in which a remote host (H1) is using the Internet to gain access to a server or other machine (H2) and there is a security gateway (SG2), e.g., a firewall, through which H1's traffic must pass. An example of this situation would be a mobile host (Road Warrior) crossing the Internet to the home organization's firewall (SG2). (See Case 4 in the section 4.5 Basic Combinations of Security Associations.) This situation raises several issues:

1. How does H1 know/learn about the existence of the security gateway SG2?
2. How does it authenticate SG2, and once it has authenticated SG2, how does it confirm that SG2 has been authorized to represent H2?
3. How does SG2 authenticate H1 and verify that H1 is authorized to contact H2?
4. How does H1 know/learn about backup gateways which provide alternate paths to H2?

To address these problems, a host or security gateway MUST have an administrative interface that allows the user/administrator to configure the address of a security gateway for any sets of destination addresses that require its use. This includes the ability to configure:

- o the requisite information for locating and authenticating the security gateway and verifying its authorization to represent the destination host.
- o the requisite information for locating and authenticating any backup gateways and verifying their authorization to represent the destination host.

It is assumed that the SPD is also configured with policy information that covers any other IPsec requirements for the path to the security gateway and the destination host.

This document does not address the issue of how to automate the discovery/verification of security gateways.

4.7 Security Associations and Multicast

The receiver-orientation of the Security Association implies that, in the case of unicast traffic, the destination system will normally select the SPI value. By having the destination select the SPI value, there is no potential for manually configured Security Associations to conflict with automatically configured (e.g., via a key management protocol) Security Associations or for Security Associations from multiple sources to conflict with each other. For multicast traffic, there are multiple destination systems per multicast group. So some system or person will need to coordinate among all multicast groups to select an SPI or SPIs on behalf of each multicast group and then communicate the group's IPsec information to all of the legitimate members of that multicast group via mechanisms not defined here.

Multiple senders to a multicast group SHOULD use a single Security Association (and hence Security Parameter Index) for all traffic to that group when a symmetric key encryption or authentication algorithm is employed. In such circumstances, the receiver knows only that the message came from a system possessing the key for that multicast group. In such circumstances, a receiver generally will not be able to authenticate which system sent the multicast traffic. Specifications for other, more general multicast cases are deferred to later IPsec documents.

At the time this specification was published, automated protocols for multicast key distribution were not considered adequately mature for standardization. For multicast groups having relatively few members, manual key distribution or multiple use of existing unicast key distribution algorithms such as modified Diffie-Hellman appears feasible. For very large groups, new scalable techniques will be needed. An example of current work in this area is the Group Key Management Protocol (GKMP) [HM97].

5. IP Traffic Processing

As mentioned in Section 4.4.1 "The Security Policy Database (SPD)", the SPD must be consulted during the processing of all traffic (INBOUND and OUTBOUND), including non-IPsec traffic. If no policy is found in the SPD that matches the packet (for either inbound or outbound traffic), the packet MUST be discarded.

NOTE: All of the cryptographic algorithms used in IPsec expect their input in canonical network byte order (see Appendix in RFC 791) and generate their output in canonical network byte order. IP packets are also transmitted in network byte order.

5.1 Outbound IP Traffic Processing

5.1.1 Selecting and Using an SA or SA Bundle

In a security gateway or BITW implementation (and in many BITS implementations), each outbound packet is compared against the SPD to determine what processing is required for the packet. If the packet is to be discarded, this is an auditable event. If the traffic is allowed to bypass IPsec processing, the packet continues through "normal" processing for the environment in which the IPsec processing is taking place. If IPsec processing is required, the packet is either mapped to an existing SA (or SA bundle), or a new SA (or SA bundle) is created for the packet. Since a packet's selectors might match multiple policies or multiple extant SAs and since the SPD is ordered, but the SAD is not, IPsec MUST:

1. Match the packet's selector fields against the outbound policies in the SPD to locate the first appropriate policy, which will point to zero or more SA bundles in the SAD.

2. Match the packet's selector fields against those in the SA bundles found in (1) to locate the first SA bundle that matches. If no SAs were found or none match, create an appropriate SA bundle and link the SPD entry to the SAD entry. If no key management entity is found, drop the packet.

3. Use the SA bundle found/created in (2) to do the required IPsec processing, e.g., authenticate and encrypt.

In a host IPsec implementation based on sockets, the SPD will be consulted whenever a new socket is created, to determine what, if any, IPsec processing will be applied to the traffic that will flow on that socket.

NOTE: A compliant implementation MUST not allow instantiation of an ESP SA that employs both a NULL encryption and a NULL authentication algorithm. An attempt to negotiate such an SA is an auditable event.

5.1.2 Header Construction for Tunnel Mode

This section describes the handling of the inner and outer IP headers, extension headers, and options for AH and ESP tunnels. This includes how to construct the encapsulating (outer) IP header, how to handle fields in the inner IP header, and what other actions should be taken. The general idea is modeled after the one used in RFC 2003, "IP Encapsulation with IP":

- o The outer IP header Source Address and Destination Address identify the "endpoints" of the tunnel (the encapsulator and decapsulator). The inner IP header Source Address and Destination Addresses identify the original sender and recipient of the datagram, (from the perspective of this tunnel), respectively. (see footnote 3 after the table in 5.1.2.1 for more details on the encapsulating source IP address.)
- o The inner IP header is not changed except to decrement the TTL as noted below, and remains unchanged during its delivery to the tunnel exit point.
- o No change to IP options or extension headers in the inner header occurs during delivery of the encapsulated datagram through the tunnel.
- o If need be, other protocol headers such as the IP Authentication header may be inserted between the outer IP header and the inner IP header.

The tables in the following sub-sections show the handling for the different header/option fields (constructed = the value in the outer field is constructed independently of the value in the inner).

5.1.2.1 IPv4 -- Header Construction for Tunnel Mode

```
                      <-- How Outer Hdr Relates to Inner Hdr -->
                      Outer Hdr at                 Inner Hdr at
IPv4                  Encapsulator                 Decapsulator
  Header fields:      --------------------         ------------
    version           4 (1)                        no change
    header length     constructed                  no change
    TOS               copied from inner hdr (5)    no change
    total length      constructed                  no change
    ID                constructed                  no change
    flags (DF,MF)     constructed, DF (4)          no change
    fragmt offset     constructed                  no change
```

TTL	constructed (2)	decrement (2)
protocol	AH, ESP, routing hdr	no change
checksum	constructed	constructed (2)
src address	constructed (3)	no change
dest address	constructed (3)	no change
Options	never copied	no change

1. The IP version in the encapsulating header can be different from the value in the inner header.

2. The TTL in the inner header is decremented by the encapsulator prior to forwarding and by the decapsulator if it forwards the packet. (The checksum changes when the TTL changes.)

   Note: The decrementing of the TTL is one of the usual actions that takes place when forwarding a packet. Packets originating from the same node as the encapsulator do not have their TTL's decremented, as the sending node is originating the packet rather than forwarding it.

3. src and dest addresses depend on the SA, which is used to determine the dest address which in turn determines which src address (net interface) is used to forward the packet.

   NOTE: In principle, the encapsulating IP source address can be any of the encapsulator's interface addresses or even an address different from any of the encapsulator's IP addresses, (e.g., if it's acting as a NAT box) so long as the address is reachable through the encapsulator from the environment into which the packet is sent. This does not cause a problem because IPsec does not currently have any INBOUND processing requirement that involves the Source Address of the encapsulating IP header. So while the receiving tunnel endpoint looks at the Destination Address in the encapsulating IP header, it only looks at the Source Address in the inner (encapsulated) IP header.

4. configuration determines whether to copy from the inner header (IPv4 only), clear or set the DF.

5. If Inner Hdr is IPv4 (Protocol = 4), copy the TOS. If Inner Hdr is IPv6 (Protocol = 41), map the Class to TOS.

5.1.2.2 IPv6 -- Header Construction for Tunnel Mode

See previous section 5.1.2 for notes 1-5 indicated by (footnote number).

IPv6	Outer Hdr at Encapsulator	Inner Hdr at Decapsulator
Header fields:		
version	6 (1)	no change
class	copied or configured (6)	no change
flow id	copied or configured	no change
len	constructed	no change
next header	AH,ESP,routing hdr	no change
hop limit	constructed (2)	decrement (2)
src address	constructed (3)	no change
dest address	constructed (3)	no change
Extension headers	never copied	no change

<-- How Outer Hdr Relates Inner Hdr --->

6. If Inner Hdr is IPv6 (Next Header = 41), copy the Class. If Inner Hdr is IPv4 (Next Header = 4), map the TOS to Class.

5.2 Processing Inbound IP Traffic

Prior to performing AH or ESP processing, any IP fragments are reassembled. Each inbound IP datagram to which IPsec processing will be applied is identified by the appearance of the AH or ESP values in the IP Next Protocol field (or of AH or ESP as an extension header in the IPv6 context).

Note: Appendix C contains sample code for a bitmask check for a 32 packet window that can be used for implementing anti-replay service.

5.2.1 Selecting and Using an SA or SA Bundle

Mapping the IP datagram to the appropriate SA is simplified because of the presence of the SPI in the AH or ESP header. Note that the selector checks are made on the inner headers not the outer (tunnel) headers. The steps followed are:

1. Use the packet's destination address (outer IP header), IPsec protocol, and SPI to look up the SA in the SAD. If the SA lookup fails, drop the packet and log/report the error.

2. Use the SA found in (1) to do the IPsec processing, e.g., authenticate and decrypt. This step includes matching the packet's (Inner Header if tunneled) selectors to the selectors in the SA. Local policy determines the specificity of the SA selectors (single value, list, range, wildcard). In general, a packet's source address MUST match the SA selector value. However, an ICMP packet received on a tunnel mode SA may have a source address

other than that bound to the SA and thus such packets should be permitted as exceptions to this check. For an ICMP packet, the selectors from the enclosed problem packet (the source and destination addresses and ports should be swapped) should be checked against the selectors for the SA. Note that some or all of these selectors may be inaccessible because of limitations on how many bits of the problem packet the ICMP packet is allowed to carry or due to encryption. See Section 6.

Do (1) and (2) for every IPsec header until a Transport Protocol Header or an IP header that is NOT for this system is encountered. Keep track of what SAs have been used and their order of application.

3. Find an incoming policy in the SPD that matches the packet. This could be done, for example, by use of backpointers from the SAs to the SPD or by matching the packet's selectors (Inner Header if tunneled) against those of the policy entries in the SPD.

4. Check whether the required IPsec processing has been applied, i.e., verify that the SA's found in (1) and (2) match the kind and order of SAs required by the policy found in (3).

   NOTE: The correct "matching" policy will not necessarily be the first inbound policy found. If the check in (4) fails, steps (3) and (4) are repeated until all policy entries have been checked or until the check succeeds.

At the end of these steps, pass the resulting packet to the Transport Layer or forward the packet. Note that any IPsec headers processed in these steps may have been removed, but that this information, i.e., what SAs were used and the order of their application, may be needed for subsequent IPsec or firewall processing.

Note that in the case of a security gateway, if forwarding causes a packet to exit via an IPsec-enabled interface, then additional IPsec processing may be applied.

5.2.2 Handling of AH and ESP tunnels

The handling of the inner and outer IP headers, extension headers, and options for AH and ESP tunnels should be performed as described in the tables in Section 5.1.

6. ICMP Processing (relevant to IPsec)

The focus of this section is on the handling of ICMP error messages. Other ICMP traffic, e.g., Echo/Reply, should be treated like other traffic and can be protected on an end-to-end basis using SAs in the usual fashion.

An ICMP error message protected by AH or ESP and generated by a router SHOULD be processed and forwarded in a tunnel mode SA. Local policy determines whether or not it is subjected to source address checks by the router at the destination end of the tunnel. Note that if the router at the originating end of the tunnel is forwarding an ICMP error message from another router, the source address check would fail. An ICMP message protected by AH or ESP and generated by a router MUST NOT be forwarded on a transport mode SA (unless the SA has been established to the router acting as a host, e.g., a Telnet connection used to manage a router). An ICMP message generated by a host SHOULD be checked against the source IP address selectors bound to the SA in which the message arrives. Note that even if the source of an ICMP error message is authenticated, the returned IP header could be invalid. Accordingly, the selector values in the IP header SHOULD also be checked to be sure that they are consistent with the selectors for the SA over which the ICMP message was received.

The table in Appendix D characterize ICMP messages as being either host generated, router generated, both, unknown/unassigned. ICMP messages falling into the last two categories should be handled as determined by the receiver's policy.

An ICMP message not protected by AH or ESP is unauthenticated and its processing and/or forwarding may result in denial of service. This suggests that, in general, it would be desirable to ignore such messages. However, it is expected that many routers (vs. security gateways) will not implement IPsec for transit traffic and thus strict adherence to this rule would cause many ICMP messages to be discarded. The result is that some critical IP functions would be lost, e.g., redirection and PMTU processing. Thus it MUST be possible to configure an IPsec implementation to accept or reject (router) ICMP traffic as per local security policy.

The remainder of this section addresses how PMTU processing MUST be performed at hosts and security gateways. It addresses processing of both authenticated and unauthenticated ICMP PMTU messages. However, as noted above, unauthenticated ICMP messages MAY be discarded based on local policy.

6.1 PMTU/DF Processing

6.1.1 DF Bit

In cases where a system (host or gateway) adds an encapsulating header (ESP tunnel or AH tunnel), it MUST support the option of copying the DF bit from the original packet to the encapsulating header (and processing ICMP PMTU messages). This means that it MUST be possible to configure the system's treatment of the DF bit (set, clear, copy from encapsulated header) for each interface. (See Appendix B for rationale.)

6.1.2 Path MTU Discovery (PMTU)

This section discusses IPsec handling for Path MTU Discovery messages. ICMP PMTU is used here to refer to an ICMP message for:

IPv4 (RFC 792):
- Type = 3 (Destination Unreachable)
- Code = 4 (Fragmentation needed and DF set)
- Next-Hop MTU in the low-order 16 bits of the second word of the ICMP header (labelled "unused" in RFC 792), with high-order 16 bits set to zero

IPv6 (RFC 1885):
- Type = 2 (Packet Too Big)
- Code = 0 (Fragmentation needed)
- Next-Hop MTU in the 32 bit MTU field of the ICMP6 message

6.1.2.1 Propagation of PMTU

The amount of information returned with the ICMP PMTU message (IPv4 or IPv6) is limited and this affects what selectors are available for use in further propagating the PMTU information. (See Appendix B for more detailed discussion of this topic.)

o PMTU message with 64 bits of IPsec header -- If the ICMP PMTU message contains only 64 bits of the IPsec header (minimum for IPv4), then a security gateway MUST support the following options on a per SPI/SA basis:

   a. if the originating host can be determined (or the possible sources narrowed down to a manageable number), send the PM information to all the possible originating hosts.
   b. if the originating host cannot be determined, store the PMTU with the SA and wait until the next packet(s) arrive from the originating host for the relevant security association. If

the packet(s) are bigger than the PMTU, drop the packet(s), and compose ICMP PMTU message(s) with the new packet(s) and the updated PMTU, and send the ICMP message(s) about the problem to the originating host. Retain the PMTU information for any message that might arrive subsequently (see Section 6.1.2.4, "PMTU Aging").

- o PMTU message with >64 bits of IPsec header -- If the ICMP message contains more information from the original packet then there may be enough non-opaque information to immediately determine to which host to propagate the ICMP/PMTU message and to provide that system with the 5 fields (source address, destination address, source port, destination port, transport protocol) needed to determine where to store/update the PMTU. Under such circumstances, a security gateway MUST generate an ICMP PMTU message immediately upon receipt of an ICMP PMTU from further down the path.

- o Distributing the PMTU to the Transport Layer -- The host mechanism for getting the updated PMTU to the transport layer is unchanged, as specified in RFC 1191 (Path MTU Discovery).

6.1.2.2 Calculation of PMTU

The calculation of PMTU from an ICMP PMTU MUST take into account the addition of any IPsec header -- AH transport, ESP transport, AH/ESP transport, ESP tunnel, AH tunnel. (See Appendix B for discussion of implementation issues.)

Note: In some situations the addition of IPsec headers could result in an effective PMTU (as seen by the host or application) that is unacceptably small. To avoid this problem, the implementation may establish a threshold below which it will not report a reduced PMTU. In such cases, the implementation would apply IPsec and then fragment the resulting packet according to the PMTU. This would result in a more efficient use of the available bandwidth.

6.1.2.3 Granularity of PMTU Processing

In hosts, the granularity with which ICMP PMTU processing can be done differs depending on the implementation situation. Looking at a host, there are 3 situations that are of interest with respect to PMTU issues (See Appendix B for additional details on this topic.):

a. Integration of IPsec into the native IP implementation
b. Bump-in-the-stack implementations, where IPsec is implemented "underneath" an existing implementation of a TCP/IP protocol stack, between the native IP and the local network drivers

c. No IPsec implementation -- This case is included because it is relevant in cases where a security gateway is sending PMTU information back to a host.

Only in case (a) can the PMTU data be maintained at the same granularity as communication associations. In (b) and (c), the IP layer will only be able to maintain PMTU data at the granularity of source and destination IP addresses (and optionally TOS), as described in RFC 1191. This is an important difference, because more than one communication association may map to the same source and destination IP addresses, and each communication association may have a different amount of IPsec header overhead (e.g., due to use of different transforms or different algorithms).

Implementation of the calculation of PMTU and support for PMTUs at the granularity of individual communication associations is a local matter. However, a socket-based implementation of IPsec in a host SHOULD maintain the information on a per socket basis. Bump in the stack systems MUST pass an ICMP PMTU to the host IP implementation, after adjusting it for any IPsec header overhead added by these systems. The calculation of the overhead SHOULD be determined by analysis of the SPI and any other selector information present in a returned ICMP PMTU message.

6.1.2.4 PMTU Aging

In all systems (host or gateway) implementing IPsec and maintaining PMTU information, the PMTU associated with a security association (transport or tunnel) MUST be "aged" and some mechanism put in place for updating the PMTU in a timely manner, especially for discovering if the PMTU is smaller than it needs to be. A given PMTU has to remain in place long enough for a packet to get from the source end of the security association to the system at the other end of the security association and propagate back an ICMP error message if the current PMTU is too big. Note that if there are nested tunnels, multiple packets and round trip times might be required to get an ICMP message back to an encapsulator or originating host.

Systems SHOULD use the approach described in the Path MTU Discovery document (RFC 1191, Section 6.3), which suggests periodically resetting the PMTU to the first-hop data-link MTU and then letting the normal PMTU Discovery processes update the PMTU as necessary. The period SHOULD be configurable.

7. Auditing

Not all systems that implement IPsec will implement auditing. For the most part, the granularity of auditing is a local matter. However, several auditable events are identified in the AH and ESP specifications and for each of these events a minimum set of information that SHOULD be included in an audit log is defined. Additional information also MAY be included in the audit log for each of these events, and additional events, not explicitly called out in this specification, also MAY result in audit log entries. There is no requirement for the receiver to transmit any message to the purported transmitter in response to the detection of an auditable event, because of the potential to induce denial of service via such action.

8. Use in Systems Supporting Information Flow Security

Information of various sensitivity levels may be carried over a single network. Information labels (e.g., Unclassified, Company Proprietary, Secret) [DoD85, DoD87] are often employed to distinguish such information. The use of labels facilitates segregation of information, in support of information flow security models, e.g., the Bell-LaPadula model [BL73]. Such models, and corresponding supporting technology, are designed to prevent the unauthorized flow of sensitive information, even in the face of Trojan Horse attacks. Conventional, discretionary access control (DAC) mechanisms, e.g., based on access control lists, generally are not sufficient to support such policies, and thus facilities such as the SPD do not suffice in such environments.

In the military context, technology that supports such models is often referred to as multi-level security (MLS). Computers and networks often are designated "multi-level secure" if they support the separation of labelled data in conjunction with information flow security policies. Although such technology is more broadly applicable than just military applications, this document uses the acronym "MLS" to designate the technology, consistent with much extant literature.

IPsec mechanisms can easily support MLS networking. MLS networking requires the use of strong Mandatory Access Controls (MAC), which unprivileged users or unprivileged processes are incapable of controlling or violating. This section pertains only to the use of these IP security mechanisms in MLS (information flow security policy) environments. Nothing in this section applies to systems not claiming to provide MLS.

As used in this section, "sensitivity information" might include implementation-defined hierarchic levels, categories, and/or releasability information.

AH can be used to provide strong authentication in support of mandatory access control decisions in MLS environments. If explicit IP sensitivity information (e.g., IPSO [Ken91]) is used and confidentiality is not considered necessary within the particular operational environment, AH can be used to authenticate the binding between sensitivity labels in the IP header and the IP payload (including user data). This is a significant improvement over labeled IPv4 networks where the sensitivity information is trusted even though there is no authentication or cryptographic binding of the information to the IP header and user data. IPv4 networks might or might not use explicit labelling. IPv6 will normally use implicit sensitivity information that is part of the IPsec Security Association but not transmitted with each packet instead of using explicit sensitivity information. All explicit IP sensitivity information MUST be authenticated using either ESP, AH, or both.

Encryption is useful and can be desirable even when all of the hosts are within a protected environment, for example, behind a firewall or disjoint from any external connectivity. ESP can be used, in conjunction with appropriate key management and encryption algorithms, in support of both DAC and MAC. (The choice of encryption and authentication algorithms, and the assurance level of an IPsec implementation will determine the environments in which an implementation may be deemed sufficient to satisfy MLS requirements.) Key management can make use of sensitivity information to provide MAC. IPsec implementations on systems claiming to provide MLS SHOULD be capable of using IPsec to provide MAC for IP-based communications.

8.1 Relationship Between Security Associations and Data Sensitivity

Both the Encapsulating Security Payload and the Authentication Header can be combined with appropriate Security Association policies to provide multi-level secure networking. In this case each SA (or SA bundle) is normally used for only a single instance of sensitivity information. For example, "PROPRIETARY - Internet Engineering" must be associated with a different SA (or SA bundle) from "PROPRIETARY - Finance".

8.2 Sensitivity Consistency Checking

An MLS implementation (both host and router) MAY associate sensitivity information, or a range of sensitivity information with an interface, or a configured IP address with its associated prefix (the latter is sometimes referred to as a logical interface, or an interface alias). If such properties exist, an implementation SHOULD compare the sensitivity information associated with the packet against the sensitivity information associated with the interface or address/prefix from which the packet arrived, or through which the packet will depart. This check will either verify that the sensitivities match, or that the packet's sensitivity falls within the range of the interface or address/prefix.

The checking SHOULD be done on both inbound and outbound processing.

8.3 Additional MLS Attributes for Security Association Databases

Section 4.4 discussed two Security Association databases (the Security Policy Database (SPD) and the Security Association Database (SAD)) and the associated policy selectors and SA attributes. MLS networking introduces an additional selector/attribute:

- Sensitivity information.

The Sensitivity information aids in selecting the appropriate algorithms and key strength, so that the traffic gets a level of protection appropriate to its importance or sensitivity as described in section 8.1. The exact syntax of the sensitivity information is implementation defined.

8.4 Additional Inbound Processing Steps for MLS Networking

After an inbound packet has passed through IPsec processing, an MLS implementation SHOULD first check the packet's sensitivity (as defined by the SA (or SA bundle) used for the packet) with the interface or address/prefix as described in section 8.2 before delivering the datagram to an upper-layer protocol or forwarding it.

The MLS system MUST retain the binding between the data received in an IPsec protected packet and the sensitivity information in the SA or SAs used for processing, so appropriate policy decisions can be made when delivering the datagram to an application or forwarding engine. The means for maintaining this binding are implementation specific.

8.5 Additional Outbound Processing Steps for MLS Networking

An MLS implementation of IPsec MUST perform two additional checks besides the normal steps detailed in section 5.1.1. When consulting the SPD or the SAD to find an outbound security association, the MLS implementation MUST use the sensitivity of the data to select an appropriate outbound SA or SA bundle. The second check comes before forwarding the packet out to its destination, and is the sensitivity consistency checking described in section 8.2.

8.6 Additional MLS Processing for Security Gateways

An MLS security gateway MUST follow the previously mentioned inbound and outbound processing rules as well as perform some additional processing specific to the intermediate protection of packets in an MLS environment.

A security gateway MAY act as an outbound proxy, creating SAs for MLS systems that originate packets forwarded by the gateway. These MLS systems may explicitly label the packets to be forwarded, or the whole originating network may have sensitivity characteristics associated with it. The security gateway MUST create and use appropriate SAs for AH, ESP, or both, to protect such traffic it forwards.

Similarly such a gateway SHOULD accept and process inbound AH and/or ESP packets and forward appropriately, using explicit packet labeling, or relying on the sensitivity characteristics of the destination network.

9. Performance Issues

The use of IPsec imposes computational performance costs on the hosts or security gateways that implement these protocols. These costs are associated with the memory needed for IPsec code and data structures, and the computation of integrity check values, encryption and decryption, and added per-packet handling. The per-packet computational costs will be manifested by increased latency and, possibly, reduced throughout. Use of SA/key management protocols, especially ones that employ public key cryptography, also adds computational performance costs to use of IPsec. These per-association computational costs will be manifested in terms of increased latency in association establishment. For many hosts, it is anticipated that software-based cryptography will not appreciably reduce throughput, but hardware may be required for security gateways (since they represent aggregation points), and for some hosts.

The use of IPsec also imposes bandwidth utilization costs on transmission, switching, and routing components of the Internet infrastructure, components not implementing IPsec. This is due to the increase in the packet size resulting from the addition of AH and/or ESP headers, AH and ESP tunneling (which adds a second IP header), and the increased packet traffic associated with key management protocols. It is anticipated that, in most instances, this increased bandwidth demand will not noticeably affect the Internet infrastructure. However, in some instances, the effects may be significant, e.g., transmission of ESP encrypted traffic over a dialup link that otherwise would have compressed the traffic.

Note: The initial SA establishment overhead will be felt in the first packet. This delay could impact the transport layer and application. For example, it could cause TCP to retransmit the SYN before the ISAKMP exchange is done. The effect of the delay would be different on UDP than TCP because TCP shouldn't transmit anything other than the SYN until the connection is set up whereas UDP will go ahead and transmit data beyond the first packet.

Note: As discussed earlier, compression can still be employed at layers above IP. There is an IETF working group (IP Payload Compression Protocol (ippcp)) working on "protocol specifications that make it possible to perform lossless compression on individual payloads before the payload is processed by a protocol that encrypts it. These specifications will allow for compression operations to be performed prior to the encryption of a payload by IPsec protocols."

10. Conformance Requirements

All IPv4 systems that claim to implement IPsec MUST comply with all requirements of the Security Architecture document. All IPv6 systems MUST comply with all requirements of the Security Architecture document.

11. Security Considerations

The focus of this document is security; hence security considerations permeate this specification.

12. Differences from RFC 1825

This architecture document differs substantially from RFC 1825 in detail and in organization, but the fundamental notions are unchanged. This document provides considerable additional detail in terms of compliance specifications. It introduces the SPD and SAD, and the notion of SA selectors. It is aligned with the new versions of AH and ESP, which also differ from their predecessors. Specific requirements for supported combinations of AH and ESP are newly added, as are details of PMTU management.

Acknowledgements

Many of the concepts embodied in this specification were derived from or influenced by the US Government's SP3 security protocol, ISO/IEC's NLSP, the proposed swIPe security protocol [SDNS, ISO, IB93, IBK93], and the work done for SNMP Security and SNMPv2 Security.

For over 3 years (although it sometimes seems *much* longer), this document has evolved through multiple versions and iterations. During this time, many people have contributed significant ideas and energy to the process and the documents themselves. The authors would like to thank Karen Seo for providing extensive help in the review, editing, background research, and coordination for this version of the specification. The authors would also like to thank the members of the IPsec and IPng working groups, with special mention of the efforts of (in alphabetic order): Steve Bellovin, Steve Deering, James Hughes, Phil Karn, Frank Kastenholz, Perry Metzger, David Mihelcic, Hilarie Orman, Norman Shulman, William Simpson, Harry Varnis, and Nina Yuan.

Appendix A -- Glossary

This section provides definitions for several key terms that are employed in this document. Other documents provide additional definitions and background information relevant to this technology, e.g., [VK83, HA94]. Included in this glossary are generic security service and security mechanism terms, plus IPsec-specific terms.

Access Control
Access control is a security service that prevents unauthorized use of a resource, including the prevention of use of a resource in an unauthorized manner. In the IPsec context, the resource to which access is being controlled is often:

- o for a host, computing cycles or data
- o for a security gateway, a network behind the gateway or bandwidth on that network.

Anti-replay
[See "Integrity" below]

Authentication
This term is used informally to refer to the combination of two nominally distinct security services, data origin authentication and connectionless integrity. See the definitions below for each of these services.

Availability
Availability, when viewed as a security service, addresses the security concerns engendered by attacks against networks that deny or degrade service. For example, in the IPsec context, the use of anti-replay mechanisms in AH and ESP support availability.

Confidentiality
Confidentiality is the security service that protects data from unauthorized disclosure. The primary confidentiality concern in most instances is unauthorized disclosure of application level data, but disclosure of the external characteristics of communication also can be a concern in some circumstances. Traffic flow confidentiality is the service that addresses this latter concern by concealing source and destination addresses, message length, or frequency of communication. In the IPsec context, using ESP in tunnel mode, especially at a security gateway, can provide some level of traffic flow confidentiality. (See also traffic analysis, below.)

Encryption
Encryption is a security mechanism used to transform data from an intelligible form (plaintext) into an unintelligible form (ciphertext), to provide confidentiality. The inverse transformation process is designated "decryption". Oftimes the term "encryption" is used to generically refer to both processes.

Data Origin Authentication
Data origin authentication is a security service that verifies the identity of the claimed source of data. This service is usually bundled with connectionless integrity service.

Integrity
Integrity is a security service that ensures that modifications to data are detectable. Integrity comes in various flavors to match application requirements. IPsec supports two forms of integrity: connectionless and a form of partial sequence integrity. Connectionless integrity is a service that detects modification of an individual IP datagram, without regard to the ordering of the datagram in a stream of traffic. The form of partial sequence integrity offered in IPsec is referred to as anti-replay integrity, and it detects arrival of duplicate IP datagrams (within a constrained window). This is in contrast to connection-oriented integrity, which imposes more stringent sequencing requirements on traffic, e.g., to be able to detect lost or re-ordered messages. Although authentication and integrity services often are cited separately, in practice they are intimately connected and almost always offered in tandem.

Security Association (SA)
A simplex (uni-directional) logical connection, created for security purposes. All traffic traversing an SA is provided the same security processing. In IPsec, an SA is an internet layer abstraction implemented through the use of AH or ESP.

Security Gateway
A security gateway is an intermediate system that acts as the communications interface between two networks. The set of hosts (and networks) on the external side of the security gateway is viewed as untrusted (or less trusted), while the networks and hosts and on the internal side are viewed as trusted (or more trusted). The internal subnets and hosts served by a security gateway are presumed to be trusted by virtue of sharing a common, local, security administration. (See "Trusted Subnetwork" below.) In the IPsec context, a security gateway is a point at which AH and/or ESP is implemented in order to serve

a set of internal hosts, providing security services for these hosts when they communicate with external hosts also employing IPsec (either directly or via another security gateway).

SPI

Acronym for "Security Parameters Index". The combination of a destination address, a security protocol, and an SPI uniquely identifies a security association (SA, see above). The SPI is carried in AH and ESP protocols to enable the receiving system to select the SA under which a received packet will be processed. An SPI has only local significance, as defined by the creator of the SA (usually the receiver of the packet carrying the SPI); thus an SPI is generally viewed as an opaque bit string. However, the creator of an SA may choose to interpret the bits in an SPI to facilitate local processing.

Traffic Analysis

The analysis of network traffic flow for the purpose of deducing information that is useful to an adversary. Examples of such information are frequency of transmission, the identities of the conversing parties, sizes of packets, flow identifiers, etc. [Sch94]

Trusted Subnetwork

A subnetwork containing hosts and routers that trust each other not to engage in active or passive attacks. There also is an assumption that the underlying communications channel (e.g., a LAN or CAN) isn't being attacked by other means.

Appendix B -- Analysis/Discussion of PMTU/DF/Fragmentation Issues

B.1 DF bit

In cases where a system (host or gateway) adds an encapsulating header (e.g., ESP tunnel), should/must the DF bit in the original packet be copied to the encapsulating header?

Fragmenting seems correct for some situations, e.g., it might be appropriate to fragment packets over a network with a very small MTU, e.g., a packet radio network, or a cellular phone hop to mobile node, rather than propagate back a very small PMTU for use over the rest of the path. In other situations, it might be appropriate to set the DF bit in order to get feedback from later routers about PMTU constraints which require fragmentation. The existence of both of these situations argues for enabling a system to decide whether or not to fragment over a particular network "link", i.e., for requiring an implementation to be able to copy the DF bit (and to process ICMP PMTU messages), but making it an option to be selected on a per interface basis. In other words, an administrator should be able to configure the router's treatment of the DF bit (set, clear, copy from encapsulated header) for each interface.

Note: If a bump-in-the-stack implementation of IPsec attempts to apply different IPsec algorithms based on source/destination ports, it will be difficult to apply Path MTU adjustments.

B.2 Fragmentation

If required, IP fragmentation occurs after IPsec processing within an IPsec implementation. Thus, transport mode AH or ESP is applied only to whole IP datagrams (not to IP fragments). An IP packet to which AH or ESP has been applied may itself be fragmented by routers en route, and such fragments MUST be reassembled prior to IPsec processing at a receiver. In tunnel mode, AH or ESP is applied to an IP packet, the payload of which may be a fragmented IP packet. For example, a security gateway, "bump-in-the-stack" (BITS), or "bump-in-the-wire" (BITW) IPsec implementation may apply tunnel mode AH to such fragments. Note that BITS or BITW implementations are examples of where a host IPsec implementation might receive fragments to which tunnel mode is to be applied. However, if transport mode is to be applied, then these implementations MUST reassemble the fragments prior to applying IPsec.

NOTE: IPsec always has to figure out what the encapsulating IP header fields are. This is independent of where you insert IPsec and is intrinsic to the definition of IPsec. Therefore any IPsec implementation that is not integrated into an IP implementation must include code to construct the necessary IP headers (e.g., IP2):

- o AH-tunnel --> IP2-AH-IP1-Transport-Data
- o ESP-tunnel --> IP2-ESP_hdr-IP1-Transport-Data-ESP_trailer

****

Overall, the fragmentation/reassembly approach described above works for all cases examined.

Implementation approach	AH Xport IPv4	AH Xport IPv6	AH Tunnel IPv4	AH Tunnel IPv6	ESP Xport IPv4	ESP Xport IPv6	ESP Tunnel IPv4	ESP Tunnel IPv6
Hosts (integr w/ IP stack)	Y	Y	Y	Y	Y	Y	Y	Y
Hosts (betw/ IP and drivers)	Y	Y	Y	Y	Y	Y	Y	Y
S. Gwy (integr w/ IP stack)			Y	Y			Y	Y
Outboard crypto processor *								

* If the crypto processor system has its own IP address, then it is covered by the security gateway case. This box receives the packet from the host and performs IPsec processing. It has to be able to handle the same AH, ESP, and related IPv4/IPv6 tunnel processing that a security gateway would have to handle. If it doesn't have it's own address, then it is similar to the bump-in-the stack implementation between IP and the network drivers.

The following analysis assumes that:

1. There is only one IPsec module in a given system's stack. There isn't an IPsec module A (adding ESP/encryption and thus) hiding the transport protocol, SRC port, and DEST port from IPsec module B.
2. There are several places where IPsec could be implemented (as shown in the table above).
   a. Hosts with integration of IPsec into the native IP implementation. Implementer has access to the source for the stack.
   b. Hosts with bump-in-the-stack implementations, where IPsec is implemented between IP and the local network drivers. Source access for stack is not available; but there are well-defined interfaces that allows the IPsec code to be incorporated into the system.

c. Security gateways and outboard crypto processors with integration of IPsec into the stack.
3. Not all of the above approaches are feasible in all hosts. But it was assumed that for each approach, there are some hosts for whom the approach is feasible.

For each of the above 3 categories, there are IPv4 and IPv6, AH transport and tunnel modes, and ESP transport and tunnel modes -- for a total of 24 cases (3 x 2 x 4).

Some header fields and interface fields are listed here for ease of reference -- they're not in the header order, but instead listed to allow comparison between the columns. (* = not covered by AH authentication. ESP authentication doesn't cover any headers that precede it.)

IPv4	IPv6	IP/Transport Interface (RFC 1122 -- Sec 3.4)
Version = 4	Version = 6	
Header Len		
*TOS	Class,Flow Lbl	TOS
Packet Len	Payload Len	Len
ID		ID (optional)
*Flags		DF
*Offset		
*TTL	*Hop Limit	TTL
Protocol	Next Header	
*Checksum		
Src Address	Src Address	Src Address
Dst Address	Dst Address	Dst Address
Options?	Options?	Opt

? = AH covers Option-Type and Option-Length, but might not cover Option-Data.

The results for each of the 20 cases is shown below ("works" = will work if system fragments after outbound IPsec processing, reassembles before inbound IPsec processing). Notes indicate implementation issues.

a. Hosts (integrated into IP stack)
   - o AH-transport --> (IP1-AH-Transport-Data)
     - - IPv4 -- works
     - - IPv6 -- works
   - o AH-tunnel --> (IP2-AH-IP1-Transport-Data)
     - - IPv4 -- works
     - - IPv6 -- works

```
        o ESP-transport --> (IP1-ESP_hdr-Transport-Data-ESP_trailer)
                  - IPv4 -- works
                  - IPv6 -- works
        o ESP-tunnel -->  (IP2-ESP_hdr-IP1-Transport-Data-ESP_trailer)
                  - IPv4 -- works
                  - IPv6 -- works
```

b. Hosts (Bump-in-the-stack) -- put IPsec between IP layer and network drivers. In this case, the IPsec module would have to do something like one of the following for fragmentation and reassembly.

- do the fragmentation/reassembly work itself and send/receive the packet directly to/from the network layer. In AH or ESP transport mode, this is fine. In AH or ESP tunnel mode where the tunnel end is at the ultimate destination, this is fine. But in AH or ESP tunnel modes where the tunnel end is different from the ultimate destination and where the source host is multi-homed, this approach could result in sub-optimal routing because the IPsec module may be unable to obtain the information needed (LAN interface and next-hop gateway) to direct the packet to the appropriate network interface. This is not a problem if the interface and next-hop gateway are the same for the ultimate destination and for the tunnel end. But if they are different, then IPsec would need to know the LAN interface and the next-hop gateway for the tunnel end. (Note: The tunnel end (security gateway) is highly likely to be on the regular path to the ultimate destination. But there could also be more than one path to the destination, e.g., the host could be at an organization with 2 firewalls. And the path being used could involve the less commonly chosen firewall.) OR
- pass the IPsec'd packet back to the IP layer where an extra IP header would end up being pre-pended and the IPsec module would have to check and let IPsec'd fragments go by.

  OR
- pass the packet contents to the IP layer in a form such that the IP layer recreates an appropriate IP header

At the network layer, the IPsec module will have access to the following selectors from the packet -- SRC address, DST address, Next Protocol, and if there's a transport layer header --> SRC port and DST port. One cannot assume IPsec has access to the Name. It is assumed that the available selector information is sufficient to figure out the relevant Security Policy entry and Security Association(s).

```
      o AH-transport  --> (IP1-AH-Transport-Data)
                - IPv4 -- works
                - IPv6 -- works
      o AH-tunnel --> (IP2-AH-IP1-Transport-Data)
                - IPv4 -- works
                - IPv6 -- works
      o ESP-transport --> (IP1-ESP_hdr-Transport-Data-ESP_trailer)
                - IPv4 -- works
                - IPv6 -- works
      o ESP-tunnel -->  (IP2-ESP_hdr-IP1-Transport-Data-ESP_trailer)
                - IPv4 -- works
                - IPv6 -- works
```

c. Security gateways -- integrate IPsec into the IP stack

NOTE: The IPsec module will have access to the following selectors from the packet -- SRC address, DST address, Next Protocol, and if there's a transport layer header --> SRC port and DST port. It won't have access to the User ID (only Hosts have access to User ID information.) Unlike some Bump-in-the-stack implementations, security gateways may be able to look up the Source Address in the DNS to provide a System Name, e.g., in situations involving use of dynamically assigned IP addresses in conjunction with dynamically updated DNS entries. It also won't have access to the transport layer information if there is an ESP header, or if it's not the first fragment of a fragmented message. It is assumed that the available selector information is sufficient to figure out the relevant Security Policy entry and Security Association(s).

```
      o AH-tunnel --> (IP2-AH-IP1-Transport-Data)
                - IPv4 -- works
                - IPv6 -- works
      o ESP-tunnel -->  (IP2-ESP_hdr-IP1-Transport-Data-ESP_trailer)
                - IPv4 -- works
                - IPv6 -- works
```

****

B.3 Path MTU Discovery

As mentioned earlier, "ICMP PMTU" refers to an ICMP message used for Path MTU Discovery.

The legend for the diagrams below in B.3.1 and B.3.3 (but not B.3.2) is:

```
     ==== = security association (AH or ESP, transport or tunnel)
```

```
---- = connectivity (or if so labelled, administrative boundary)
.... = ICMP message (hereafter referred to as ICMP PMTU) for

        IPv4:
        - Type = 3 (Destination Unreachable)
        - Code = 4 (Fragmentation needed and DF set)
        - Next-Hop MTU in the low-order 16 bits of the second
          word of the ICMP header (labelled unused in RFC 792),
          with high-order 16 bits set to zero

        IPv6 (RFC 1885):
        - Type = 2 (Packet Too Big)
        - Code = 0 (Fragmentation needed and DF set)
        - Next-Hop MTU in the 32 bit MTU field of the ICMP6

Hx   = host x
Rx   = router x
SGx  = security gateway x
X*   = X supports IPsec
```

B.3.1 Identifying the Originating Host(s)

The amount of information returned with the ICMP message is limited and this affects what selectors are available to identify security associations, originating hosts, etc. for use in further propagating the PMTU information.

In brief... An ICMP message must contain the following information from the "offending" packet:

- IPv4 (RFC 792) -- IP header plus a minimum of 64 bits

Accordingly, in the IPv4 context, an ICMP PMTU may identify only the first (outermost) security association. This is because the ICMP PMTU may contain only 64 bits of the "offending" packet beyond the IP header, which would capture only the first SPI from AH or ESP. In the IPv6 context, an ICMP PMTU will probably provide all the SPIs and the selectors in the IP header, but maybe not the SRC/DST ports (in the transport header) or the encapsulated (TCP, UDP, etc.) protocol. Moreover, if ESP is used, the transport ports and protocol selectors may be encrypted.

Looking at the diagram below of a security gateway tunnel (as mentioned elsewhere, security gateways do not use transport mode)...

```
     H1   ===================          H3
       \  |                 |         /
   H0 -- SG1* ---- R1 ---- SG2* ---- R2 -- H5
       /  ^        |                  \
     H2   |........|                   H4
```

Suppose that the security policy for SG1 is to use a single SA to SG2 for all the traffic between hosts H0, H1, and H2 and hosts H3, H4, and H5. And suppose H0 sends a data packet to H5 which causes R1 to send an ICMP PMTU message to SG1. If the PMTU message has only the SPI, SG1 will be able to look up the SA and find the list of possible hosts (H0, H1, H2, wildcard); but SG1 will have no way to figure out that H0 sent the traffic that triggered the ICMP PMTU message.

```
   original        after IPsec     ICMP
   packet          processing      packet
   --------        -----------     ------
                                   IP-3 header (S = R1, D = SG1)
                                   ICMP header (includes PMTU)
                   IP-2 header     IP-2 header (S = SG1, D = SG2)
                   ESP header      minimum of 64 bits of ESP hdr (*)
   IP-1 header     IP-1 header
   TCP header      TCP header
   TCP data        TCP data
                   ESP trailer

   (*) The 64 bits will include enough of the ESP (or AH) header to
       include the SPI.
           - ESP -- SPI (32 bits), Seq number (32 bits)
           - AH -- Next header (8 bits), Payload Len (8 bits),
             Reserved (16 bits), SPI (32 bits)
```

This limitation on the amount of information returned with an ICMP message creates a problem in identifying the originating hosts for the packet (so as to know where to further propagate the ICMP PMTU information). If the ICMP message contains only 64 bits of the IPsec header (minimum for IPv4), then the IPsec selectors (e.g., Source and Destination addresses, Next Protocol, Source and Destination ports, etc.) will have been lost. But the ICMP error message will still provide SG1 with the SPI, the PMTU information and the source and destination gateways for the relevant security association.

The destination security gateway and SPI uniquely define a security association which in turn defines a set of possible originating hosts. At this point, SG1 could:

a. send the PMTU information to all the possible originating hosts. This would not work well if the host list is a wild card or if many/most of the hosts weren't sending to SG1; but it might work if the SPI/destination/etc mapped to just one or a small number of hosts.
b. store the PMTU with the SPI/etc and wait until the next packet(s) arrive from the originating host(s) for the relevant security association. If it/they are bigger than the PMTU, drop the packet(s), and compose ICMP PMTU message(s) with the new packet(s) and the updated PMTU, and send the originating host(s) the ICMP message(s) about the problem. This involves a delay in notifying the originating host(s), but avoids the problems of (a).

Since only the latter approach is feasible in all instances, a security gateway MUST provide such support, as an option. However, if the ICMP message contains more information from the original packet, then there may be enough information to immediately determine to which host to propagate the ICMP/PMTU message and to provide that system with the 5 fields (source address, destination address, source port, destination port, and transport protocol) needed to determine where to store/update the PMTU. Under such circumstances, a security gateway MUST generate an ICMP PMTU message immediately upon receipt of an ICMP PMTU from further down the path. NOTE: The Next Protocol field may not be contained in the ICMP message and the use of ESP encryption may hide the selector fields that have been encrypted.

B.3.2 Calculation of PMTU

The calculation of PMTU from an ICMP PMTU has to take into account the addition of any IPsec header by H1 -- AH and/or ESP transport, or ESP or AH tunnel. Within a single host, multiple applications may share an SPI and nesting of security associations may occur. (See Section 4.5 Basic Combinations of Security Associations for description of the combinations that MUST be supported). The diagram below illustrates an example of security associations between a pair of hosts (as viewed from the perspective of one of the hosts.) (ESPx or AHx = transport mode)

```
        Socket 1 -------------------------|
                                          |
        Socket 2 (ESPx/SPI-A) ---------- AHx (SPI-B) -- Internet
```

In order to figure out the PMTU for each socket that maps to SPI-B, it will be necessary to have backpointers from SPI-B to each of the 2 paths that lead to it -- Socket 1 and Socket 2/SPI-A.

B.3.3 Granularity of Maintaining PMTU Data

In hosts, the granularity with which PMTU ICMP processing can be done differs depending on the implementation situation. Looking at a host, there are three situations that are of interest with respect to PMTU issues:

a. Integration of IPsec into the native IP implementation
b. Bump-in-the-stack implementations, where IPsec is implemented "underneath" an existing implementation of a TCP/IP protocol stack, between the native IP and the local network drivers
c. No IPsec implementation -- This case is included because it is relevant in cases where a security gateway is sending PMTU information back to a host.

Only in case (a) can the PMTU data be maintained at the same granularity as communication associations. In the other cases, the IP layer will maintain PMTU data at the granularity of Source and Destination IP addresses (and optionally TOS/Class), as described in RFC 1191. This is an important difference, because more than one communication association may map to the same source and destination IP addresses, and each communication association may have a different amount of IPsec header overhead (e.g., due to use of different transforms or different algorithms). The examples below illustrate this.

In cases (a) and (b)... Suppose you have the following situation. H1 is sending to H2 and the packet to be sent from R1 to R2 exceeds the PMTU of the network hop between them.

```
 ==================================
 |                                |
H1* --- R1 ----- R2 ---- R3 ---- H2*
 ^       |
 |.......|
```

If R1 is configured to not fragment subscriber traffic, then R1 sends an ICMP PMTU message with the appropriate PMTU to H1. H1's processing would vary with the nature of the implementation. In case (a) (native IP), the security services are bound to sockets or the equivalent. Here the IP/IPsec implementation in H1 can store/update the PMTU for the associated socket. In case (b), the IP layer in H1 can store/update the PMTU but only at the granularity of Source and Destination addresses and possibly TOS/Class, as noted above. So the result may be sub-optimal, since the PMTU for a given SRC/DST/TOS/Class will be the subtraction of the largest amount of IPsec header used for any communication association between a given source and destination.

In case (c), there has to be a security gateway to have any IPsec processing. So suppose you have the following situation. H1 is sending to H2 and the packet to be sent from SG1 to R exceeds the PMTU of the network hop between them.

```
                  ================
                  |              |
    H1 ---- SG1* --- R --- SG2* ---- H2
     ^       |
     |.......|
```

As described above for case (b), the IP layer in H1 can store/update the PMTU but only at the granularity of Source and Destination addresses, and possibly TOS/Class. So the result may be sub-optimal, since the PMTU for a given SRC/DST/TOS/Class will be the subtraction of the largest amount of IPsec header used for any communication association between a given source and destination.

B.3.4 Per Socket Maintenance of PMTU Data

Implementation of the calculation of PMTU (Section B.3.2) and support for PMTUs at the granularity of individual "communication associations" (Section B.3.3) is a local matter. However, a socket-based implementation of IPsec in a host SHOULD maintain the information on a per socket basis. Bump in the stack systems MUST pass an ICMP PMTU to the host IP implementation, after adjusting it for any IPsec header overhead added by these systems. The determination of the overhead SHOULD be determined by analysis of the SPI and any other selector information present in a returned ICMP PMTU message.

B.3.5 Delivery of PMTU Data to the Transport Layer

The host mechanism for getting the updated PMTU to the transport layer is unchanged, as specified in RFC 1191 (Path MTU Discovery).

B.3.6 Aging of PMTU Data

This topic is covered in Section 6.1.2.4.

Appendix C -- Sequence Space Window Code Example

This appendix contains a routine that implements a bitmask check for a 32 packet window. It was provided by James Hughes (jim_hughes@stortek.com) and Harry Varnis (hgv@anubis.network.com) and is intended as an implementation example. Note that this code both checks for a replay and updates the window. Thus the algorithm, as shown, should only be called AFTER the packet has been authenticated. Implementers might wish to consider splitting the code to do the check for replays before computing the ICV. If the packet is not a replay, the code would then compute the ICV, (discard any bad packets), and if the packet is OK, update the window.

```
#include <stdio.h>
#include <stdlib.h>
typedef unsigned long u_long;

enum {
    ReplayWindowSize = 32
};

u_long bitmap = 0;                 /* session state - must be 32 bits */
u_long lastSeq = 0;                     /* session state */

/* Returns 0 if packet disallowed, 1 if packet permitted */
int ChkReplayWindow(u_long seq);

int ChkReplayWindow(u_long seq) {
    u_long diff;

    if (seq == 0) return 0;             /* first == 0 or wrapped */
    if (seq > lastSeq) {                /* new larger sequence number */
        diff = seq - lastSeq;
        if (diff < ReplayWindowSize) {  /* In window */
            bitmap <<= diff;
            bitmap |= 1;                /* set bit for this packet */
        } else bitmap = 1;          /* This packet has a "way larger" */
        lastSeq = seq;
        return 1;                       /* larger is good */
    }
    diff = lastSeq - seq;
    if (diff >= ReplayWindowSize) return 0; /* too old or wrapped */
    if (bitmap & ((u_long)1 << diff)) return 0; /* already seen */
    bitmap |= ((u_long)1 << diff);              /* mark as seen */
    return 1;                           /* out of order but good */
}

char string_buffer[512];
```

```
#define STRING_BUFFER_SIZE sizeof(string_buffer)

int main() {
    int result;
    u_long last, current, bits;

    printf("Input initial state (bits in hex, last msgnum):\n");
    if (!fgets(string_buffer, STRING_BUFFER_SIZE, stdin)) exit(0);
    sscanf(string_buffer, "%lx %lu", &bits, &last);
    if (last != 0)
    bits |= 1;
    bitmap = bits;
    lastSeq = last;
    printf("bits:%08lx last:%lu\n", bitmap, lastSeq);
    printf("Input value to test (current):\n");

    while (1) {
        if (!fgets(string_buffer, STRING_BUFFER_SIZE, stdin)) break;
        sscanf(string_buffer, "%lu", &current);
        result = ChkReplayWindow(current);
        printf("%-3s", result ? "OK" : "BAD");
        printf(" bits:%08lx last:%lu\n", bitmap, lastSeq);
    }
    return 0;
}
```

Appendix D -- Categorization of ICMP messages

The tables below characterize ICMP messages as being either host generated, router generated, both, unassigned/unknown. The first set are IPv4. The second set are IPv6.

IPv4

Type	Code	Name/Codes	Reference
HOST GENERATED:			
3		Destination Unreachable	
	2	Protocol Unreachable	[RFC792]
	3	Port Unreachable	[RFC792]
	8	Source Host Isolated	[RFC792]
	14	Host Precedence Violation	[RFC1812]
10		Router Selection	[RFC1256]

Type	Code	Name/Codes	Reference
ROUTER GENERATED:			
3		Destination Unreachable	
	0	Net Unreachable	[RFC792]
	4	Fragmentation Needed, Don't Fragment was Set	[RFC792]
	5	Source Route Failed	[RFC792]
	6	Destination Network Unknown	[RFC792]
	7	Destination Host Unknown	[RFC792]
	9	Comm. w/Dest. Net. is Administratively Prohibited	[RFC792]
	11	Destination Network Unreachable for Type of Service	[RFC792]
5		Redirect	
	0	Redirect Datagram for the Network (or subnet)	[RFC792]
	2	Redirect Datagram for the Type of Service & Network	[RFC792]
9		Router Advertisement	[RFC1256]
18		Address Mask Reply	[RFC950]

IPv4

Type	Name/Codes	Reference
BOTH ROUTER AND HOST GENERATED:		
0	Echo Reply	[RFC792]
3	Destination Unreachable	
	1 Host Unreachable	[RFC792]
	10 Comm. w/Dest. Host is Administratively Prohibited	[RFC792]
	12 Destination Host Unreachable for Type of Service	[RFC792]
	13 Communication Administratively Prohibited	[RFC1812]
	15 Precedence cutoff in effect	[RFC1812]
4	Source Quench	[RFC792]
5	Redirect	
	1 Redirect Datagram for the Host	[RFC792]
	3 Redirect Datagram for the Type of Service and Host	[RFC792]
6	Alternate Host Address	[JBP]
8	Echo	[RFC792]
11	Time Exceeded	[RFC792]
12	Parameter Problem	[RFC792,RFC1108]
13	Timestamp	[RFC792]
14	Timestamp Reply	[RFC792]
15	Information Request	[RFC792]
16	Information Reply	[RFC792]
17	Address Mask Request	[RFC950]
30	Traceroute	[RFC1393]
31	Datagram Conversion Error	[RFC1475]
32	Mobile Host Redirect	[Johnson]
39	SKIP	[Markson]
40	Photuris	[Simpson]

Type	Name/Codes	Reference
UNASSIGNED TYPE OR UNKNOWN GENERATOR:		
1	Unassigned	[JBP]
2	Unassigned	[JBP]
7	Unassigned	[JBP]
19	Reserved (for Security)	[Solo]
20-29	Reserved (for Robustness Experiment)	[ZSu]
33	IPv6 Where-Are-You	[Simpson]
34	IPv6 I-Am-Here	[Simpson]
35	Mobile Registration Request	[Simpson]
36	Mobile Registration Reply	[Simpson]
37	Domain Name Request	[Simpson]
38	Domain Name Reply	[Simpson]
41-255	Reserved	[JBP]

```
                                 IPv6

Type    Name/Codes                                             Reference
========================================================================
HOST GENERATED:
  1     Destination Unreachable                                [RFC 1885]
         4  Port Unreachable

Type    Name/Codes                                             Reference
========================================================================
ROUTER GENERATED:
  1     Destination Unreachable                                [RFC1885]
         0  No Route to Destination
         1  Comm. w/Destination is Administratively Prohibited
         2  Not a Neighbor
         3  Address Unreachable
  2     Packet Too Big                                         [RFC1885]
         0
  3     Time Exceeded                                          [RFC1885]
         0  Hop Limit Exceeded in Transit
         1  Fragment reassembly time exceeded


Type    Name/Codes                                             Reference
========================================================================
BOTH ROUTER AND HOST GENERATED:
  4     Parameter Problem                                      [RFC1885]
         0  Erroneous Header Field Encountered
         1  Unrecognized Next Header Type Encountered
         2  Unrecognized IPv6 Option Encountered
```

References

[BL73] Bell, D.E. & LaPadula, L.J., "Secure Computer Systems: Mathematical Foundations and Model", Technical Report M74-244, The MITRE Corporation, Bedford, MA, May 1973.

[Bra97] Bradner, S., "Key words for use in RFCs to Indicate Requirement Level", BCP 14, RFC 2119, March 1997.

[DoD85] US National Computer Security Center, "Department of Defense Trusted Computer System Evaluation Criteria", DoD 5200.28-STD, US Department of Defense, Ft. Meade, MD., December 1985.

[DoD87] US National Computer Security Center, "Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria", NCSC-TG-005, Version 1, US Department of Defense, Ft. Meade, MD., 31 July 1987.

[HA94] Haller, N., and R. Atkinson, "On Internet Authentication", RFC 1704, October 1994.

[HC98] Harkins, D., and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, November 1998.

[HM97] Harney, H., and C. Muckenhirn, "Group Key Management Protocol (GKMP) Architecture", RFC 2094, July 1997.

[ISO] ISO/IEC JTC1/SC6, Network Layer Security Protocol, ISO-IEC DIS 11577, International Standards Organisation, Geneva, Switzerland, 29 November 1992.

[IB93] John Ioannidis and Matt Blaze, "Architecture and Implementation of Network-layer Security Under Unix", Proceedings of USENIX Security Symposium, Santa Clara, CA, October 1993.

[IBK93] John Ioannidis, Matt Blaze, & Phil Karn, "swIPe: Network-Layer Security for IP", presentation at the Spring 1993 IETF Meeting, Columbus, Ohio

[KA98a] Kent, S., and R. Atkinson, "IP Authentication Header", RFC 2402, November 1998.

[KA98b] Kent, S., and R. Atkinson, "IP Encapsulating Security Payload (ESP)", RFC 2406, November 1998.

[Ken91] Kent, S., "US DoD Security Options for the Internet Protocol", RFC 1108, November 1991.

[MSST97] Maughan, D., Schertler, M., Schneider, M., and J. Turner, "Internet Security Association and Key Management Protocol (ISAKMP)", RFC 2408, November 1998.

[Orm97] Orman, H., "The OAKLEY Key Determination Protocol", RFC 2412, November 1998.

[Pip98] Piper, D., "The Internet IP Security Domain of Interpretation for ISAKMP", RFC 2407, November 1998.

[Sch94] Bruce Schneier, Applied Cryptography, Section 8.6, John Wiley & Sons, New York, NY, 1994.

[SDNS] SDNS Secure Data Network System, Security Protocol 3, SP3, Document SDN.301, Revision 1.5, 15 May 1989, published in NIST Publication NIST-IR-90-4250, February 1990.

[SMPT98] Shacham, A., Monsour, R., Pereira, R., and M. Thomas, "IP Payload Compression Protocol (IPComp)", RFC 2393, August 1998.

[TDG97] Thayer, R., Doraswamy, N., and R. Glenn, "IP Security Document Roadmap", RFC 2411, November 1998.

[VK83] V.L. Voydock & S.T. Kent, "Security Mechanisms in High-level Networks", ACM Computing Surveys, Vol. 15, No. 2, June 1983.

Disclaimer

Author Information

Stephen Kent
BBN Corporation
70 Fawcett Street
Cambridge, MA 02140
USA

Phone: +1 (617) 873-3988
EMail: kent@bbn.com

Randall Atkinson
@Home Network
425 Broadway
Redwood City, CA 94063
USA

Phone: +1 (415) 569-5000
EMail: rja@corp.home.net

Network Working Group
Request for Comments: 2408
Category: Standards Track

D. Maughan
National Security Agency
M. Schertler
Securify, Inc.
M. Schneider
National Security Agency
J. Turner
RABA Technologies, Inc.

November 1998

Internet Security Association and Key Management Protocol (ISAKMP)

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Abstract

This memo describes a protocol utilizing security concepts necessary for establishing Security Associations (SA) and cryptographic keys in an Internet environment. A Security Association protocol that negotiates, establishes, modifies and deletes Security Associations and their attributes is required for an evolving Internet, where there will be numerous security mechanisms and several options for each security mechanism. The key management protocol must be robust in order to handle public key generation for the Internet community at large and private key requirements for those private networks with that requirement. The Internet Security Association and Key Management Protocol (ISAKMP) defines the procedures for authenticating a communicating peer, creation and management of Security Associations, key generation techniques, and threat mitigation (e.g. denial of service and replay attacks). All of these are necessary to establish and maintain secure communications (via IP Security Service or any other security protocol) in an Internet environment.

Table of Contents

List of Figures

1 Introduction

This document describes an Internet Security Association and Key Management Protocol (ISAKMP). ISAKMP combines the security concepts of authentication, key management, and security associations to establish the required security for government, commercial, and private communications on the Internet.

The Internet Security Association and Key Management Protocol (ISAKMP) defines procedures and packet formats to establish, negotiate, modify and delete Security Associations (SA). SAs contain all the information required for execution of various network security services, such as the IP layer services (such as header authentication and payload encapsulation), transport or application layer services, or self-protection of negotiation traffic. ISAKMP defines payloads for exchanging key generation and authentication data. These formats provide a consistent framework for transferring key and authentication data which is independent of the key generation technique, encryption algorithm and authentication mechanism.

ISAKMP is distinct from key exchange protocols in order to cleanly separate the details of security association management (and key management) from the details of key exchange. There may be many different key exchange protocols, each with different security properties. However, a common framework is required for agreeing to the format of SA attributes, and for negotiating, modifying, and deleting SAs. ISAKMP serves as this common framework.

Separating the functionality into three parts adds complexity to the security analysis of a complete ISAKMP implementation. However, the separation is critical for interoperability between systems with differing security requirements, and should also simplify the analysis of further evolution of a ISAKMP server.

ISAKMP is intended to support the negotiation of SAs for security protocols at all layers of the network stack (e.g., IPSEC, TLS, TLSP, OSPF, etc.). By centralizing the management of the security associations, ISAKMP reduces the amount of duplicated functionality within each security protocol. ISAKMP can also reduce connection setup time, by negotiating a whole stack of services at once.

The remainder of section 1 establishes the motivation for security negotiation and outlines the major components of ISAKMP, i.e. Security Associations and Management, Authentication, Public Key Cryptography, and Miscellaneous items. Section 2 presents the terminology and concepts associated with ISAKMP. Section 3 describes the different ISAKMP payload formats. Section 4 describes how the payloads of ISAKMP are composed together as exchange types to establish security associations and perform key exchanges in an authenticated manner. Additionally, security association modification, deletion, and error notification are discussed. Section 5 describes the processing of each payload within the context of ISAKMP exchanges, including error handling and associated actions. The appendices provide the attribute values necessary for ISAKMP and requirement for defining a new Domain of Interpretation (DOI) within ISAKMP.

1.1 Requirements Terminology

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as described in [RFC-2119].

1.2 The Need for Negotiation

ISAKMP extends the assertion in [DOW92] that authentication and key exchanges must be combined for better security to include security association exchanges. The security services required for communications depends on the individual network configurations and environments. Organizations are setting up Virtual Private Networks (VPN), also known as Intranets, that will require one set of security functions for communications within the VPN and possibly many different security functions for communications outside the VPN to support geographically separate organizational components, customers, suppliers, sub-contractors (with their own VPNs), government, and others. Departments within large organizations may require a number of security associations to separate and protect data (e.g. personnel data, company proprietary data, medical) on internal networks and other security associations to communicate within the same department. Nomadic users wanting to "phone home" represent another set of security requirements. These requirements must be tempered with bandwidth challenges. Smaller groups of people may meet their security requirements by setting up "Webs of Trust". ISAKMP exchanges provide these assorted networking communities the ability to present peers with the security functionality that the user supports in an authenticated and protected manner for agreement upon a common set of security attributes, i.e. an interoperable security association.

1.3 What can be Negotiated?

Security associations must support different encryption algorithms, authentication mechanisms, and key establishment algorithms for other security protocols, as well as IP Security. Security associations must also support host-oriented certificates for lower layer protocols and user- oriented certificates for higher level protocols. Algorithm and mechanism independence is required in applications such as e-mail, remote login, and file transfer, as well as in session oriented protocols, routing protocols, and link layer protocols. ISAKMP provides a common security association and key establishment protocol for this wide range of security protocols, applications, security requirements, and network environments.

ISAKMP is not bound to any specific cryptographic algorithm, key generation technique, or security mechanism. This flexibility is beneficial for a number of reasons. First, it supports the dynamic communications environment described above. Second, the independence from specific security mechanisms and algorithms provides a forward migration path to better mechanisms and algorithms. When improved security mechanisms are developed or new attacks against current encryption algorithms, authentication mechanisms and key exchanges are discovered, ISAKMP will allow the updating of the algorithms and mechanisms without having to develop a completely new KMP or patch the current one.

ISAKMP has basic requirements for its authentication and key exchange components. These requirements guard against denial of service, replay / reflection, man-in-the-middle, and connection hijacking attacks. This is important because these are the types of attacks that are targeted against protocols. Complete Security Association (SA) support, which provides mechanism and algorithm independence, and protection from protocol threats are the strengths of ISAKMP.

1.4 Security Associations and Management

A Security Association (SA) is a relationship between two or more entities that describes how the entities will utilize security services to communicate securely. This relationship is represented by a set of information that can be considered a contract between the entities. The information must be agreed upon and shared between all the entities. Sometimes the information alone is referred to as an SA, but this is just a physical instantiation of the existing relationship. The existence of this relationship, represented by the information, is what provides the agreed upon security information needed by entities to securely interoperate. All entities must adhere to the SA for secure communications to be possible. When accessing SA attributes, entities use a pointer or identifier refered to as the Security Parameter Index (SPI). [SEC-ARCH] provides details on IP Security Associations (SA) and Security Parameter Index (SPI) definitions.

1.4.1 Security Associations and Registration

The SA attributes required and recommended for the IP Security (AH, ESP) are defined in [SEC-ARCH]. The attributes specified for an IP Security SA include, but are not limited to, authentication mechanism, cryptographic algorithm, algorithm mode, key length, and Initialization Vector (IV). Other protocols that provide algorithm and mechanism independent security MUST define their requirements for SA attributes. The separation of ISAKMP from a specific SA definition is important to ensure ISAKMP can es tablish SAs for all possible security protocols and applications.

NOTE: See [IPDOI] for a discussion of SA attributes that should be considered when defining a security protocol or application.

In order to facilitate easy identification of specific attributes (e.g. a specific encryption algorithm) among different network entites the attributes must be assigned identifiers and these identifiers must be registered by a central authority. The Internet Assigned Numbers Authority (IANA) provides this function for the Internet.

1.4.2 ISAKMP Requirements

Security Association (SA) establishment MUST be part of the key management protocol defined for IP based networks. The SA concept is required to support security protocols in a diverse and dynamic networking environment. Just as authentication and key exchange must be linked to provide assurance that the key is established with the authenticated party [DOW92], SA establishment must be linked with the authentication and the key exchange protocol.

ISAKMP provides the protocol exchanges to establish a security association between negotiating entities followed by the establishment of a security association by these negotiating entities in behalf of some protocol (e.g. ESP/AH). First, an initial protocol exchange allows a basic set of security attributes to be agreed upon. This basic set provides protection for subsequent ISAKMP exchanges. It also indicates the authentication method and key exchange that will be performed as part of the ISAKMP protocol. If a basic set of security attributes is already in place between the negotiating server entities, the initial ISAKMP exchange may be skipped and the establishment of a security association can be done directly. After the basic set of security attributes has been agreed upon, initial identity authenticated, and required keys generated, the established SA can be used for subsequent communications by the entity that invoked ISAKMP. The basic set of SA attributes that MUST be implemented to provide ISAKMP interoperability are defined in Appendix A.

1.5 Authentication

A very important step in establishing secure network communications is authentication of the entity at the other end of the communication. Many authentication mechanisms are available. Authentication mechanisms fall into two catagories of strength - weak and strong. Sending cleartext keys or other unprotected authenticating information over a network is weak, due to the threat of reading them with a network sniffer. Additionally, sending one-way hashed poorly-chosen keys with low entropy is also weak, due to the threat of brute-force guessing attacks on the sniffed messages. While passwords can be used for establishing identity, they are not considered in this context because of recent statements from the Internet Architecture Board [IAB]. Digital signatures, such as the Digital Signature Standard (DSS) and the Rivest-Shamir-Adleman (RSA) signature, are public key based strong authentication mechanisms. When using public key digital signatures each entity requires a public key and a private key. Certificates are an essential part of a digital signature authentication mechanism. Certificates bind a specific entity's identity (be it host, network, user, or application) to its public keys and possibly other security-related information such as privileges, clearances, and compartments. Authentication based on digital signatures requires a trusted third party or certificate authority to create, sign and properly distribute certificates. For more detailed information on digital signatures, such as DSS and RSA, and certificates see [Schneier].

1.5.1 Certificate Authorities

Certificates require an infrastructure for generation, verification, revocation, management and distribution. The Internet Policy Registration Authority (IPRA) [RFC-1422] has been established to direct this infrastructure for the IETF. The IPRA certifies Policy Certification Authorities (PCA). PCAs control Certificate Authorities (CA) which certify users and subordinate entities. Current certificate related work includes the Domain Name System (DNS) Security Extensions [DNSSEC] which will provide signed entity keys in the DNS. The Public Key Infrastucture (PKIX) working group is specifying an Internet profile for X.509 certificates. There is also work going on in industry to develop X.500 Directory Services which would provide X.509 certificates to users. The U.S. Post Office is developing a (CA) hierarchy. The NIST Public Key Infrastructure Working Group has also been doing work in this area. The DOD Multi Level Information System Security Initiative (MISSI) program has begun deploying a certificate infrastructure for the U.S. Government. Alternatively, if no infrastructure exists, the PGP Web of Trust certificates can be used to provide user authentication and privacy in a community of users who know and trust each other.

1.5.2 Entity Naming

An entity's name is its identity and is bound to its public keys in certificates. The CA MUST define the naming semantics for the certificates it issues. See the UNINETT PCA Policy Statements [Berge] for an example of how a CA defines its naming policy. When the certificate is verified, the name is verified and that name will have meaning within the realm of that CA. An example is the DNS security extensions which make DNS servers CAs for the zones and nodes they serve. Resource records are provided for public keys and signatures on those keys. The names associated with the keys are IP addresses and domain names which have meaning to entities accessing the DNS for this information. A Web of Trust is another example. When webs of trust are set up, names are bound with the public keys. In PGP the name is usually the entity's e-mail address which has meaning to those, and only those, who understand e-mail. Another web of trust could use an entirely different naming scheme.

1.5.3 ISAKMP Requirements

Strong authentication MUST be provided on ISAKMP exchanges. Without being able to authenticate the entity at the other end, the Security Association (SA) and session key established are suspect. Without authentication you are unable to trust an entity's identification, which makes access control questionable. While encryption (e.g. ESP) and integrity (e.g. AH) will protect subsequent communications from passive eavesdroppers, without authentication it is possible that the SA and key may have been established with an adversary who performed an active man-in-the-middle attack and is now stealing all your personal data.

A digital signature algorithm MUST be used within ISAKMP's authentication component. However, ISAKMP does not mandate a specific signature algorithm or certificate authority (CA). ISAKMP allows an entity initiating communications to indicate which CAs it supports. After selection of a CA, the protocol provides the messages required to support the actual authentication exchange. The protocol provides a facility for identification of different certificate authorities, certificate types (e.g. X.509, PKCS #7, PGP, DNS SIG and KEY records), and the exchange of the certificates identified.

ISAKMP utilizes digital signatures, based on public key cryptography, for authentication. There are other strong authentication systems available, which could be specified as additional optional authentication mechanisms for ISAKMP. Some of these authentication systems rely on a trusted third party called a key distribution center (KDC) to distribute secret session keys. An example is Kerberos, where the trusted third party is the Kerberos server, which holds secret keys for all clients and servers within its network domain. A client's proof that it holds its secret key provides authenticaton to a server.

The ISAKMP specification does not specify the protocol for communicating with the trusted third parties (TTP) or certificate directory services. These protocols are defined by the TTP and directory service themselves and are outside the scope of this specification. The use of these additional services and protocols will be described in a Key Exchange specific document.

1.6 Public Key Cryptography

Public key cryptography is the most flexible, scalable, and efficient way for users to obtain the shared secrets and session keys needed to support the large number of ways Internet users will interoperate. Many key generation algorithms, that have different properties, are available to users (see [DOW92], [ANSI], and [Oakley]). Properties of key exchange protocols include the key establishment method, authentication, symmetry, perfect forward secrecy, and back traffic protection.

NOTE: Cryptographic keys can protect information for a considerable length of time. However, this is based on the assumption that keys used for protection of communications are destroyed after use and not kept for any reason.

1.6.1 Key Exchange Properties

Key Establishment (Key Generation / Key Transport): The two common methods of using public key cryptography for key establishment are key transport and key generation. An example of key transport is the use of the RSA algorithm to encrypt a randomly generated session key (for encrypting subsequent communications) with the recipient's public key. The encrypted random key is then sent to the recipient, who decrypts it using his private key. At this point both sides have the same session key, however it was created based on input from only one side of the communications. The benefit of the key transport method is that it has less computational overhead than the following method. The Diffie-Hellman (D-H) algorithm illustrates key generation using public key cryptography. The D-H algorithm is begun by two users exchanging public information. Each user then mathematically combines the other's public information along with their own secret information to compute a shared secret value. This secret value can be used as a session key or as a key encryption key for encrypting a randomly generated session key. This method generates a session key based on public and secret information held by both users. The benefit of the D-H algorithm is that the key used for encrypting messages is based on information held by both users and the independence of keys from one key exchange to another provides perfect forward secrecy. Detailed descriptions of these algorithms can be found in [Schneier]. There are a number of variations on these two key generation schemes and these variations do not necessarily interoperate.

Key Exchange Authentication: Key exchanges may be authenticated during the protocol or after protocol completion. Authentication of the key exchange during the protocol is provided when each party provides proof it has the secret session key before the end of the protocol. Proof can be provided by encrypting known data in the secret session key during the protocol echange. Authentication after the protocol must occur in subsequent commu nications. Authentication during the protocol is preferred so subsequent communications are not initiated if the secret session key is not established with the desired party.

Key Exchange Symmetry: A key exchange provides symmetry if either party can initiate the exchange and exchanged messages can cross in transit without affecting the key that is generated. This is desirable so that computation of the keys does not require either party to know who initated the exchange. While key exchange symmetry is desirable, symmetry in the entire key management protocol may provide a vulnerablity to reflection attacks.

Perfect Forward Secrecy: As described in [DOW92], an authenticated key exchange protocol provides perfect forward secrecy if disclosure of longterm secret keying material does not compromise the secrecy of the exchanged keys from previous communications. The property of perfect forward secrecy does not apply to key exchange without authentication.

1.6.2 ISAKMP Requirements

An authenticated key exchange MUST be supported by ISAKMP. Users SHOULD choose additional key establishment algorithms based on their requirements. ISAKMP does not specify a specific key exchange. However, [IKE] describes a proposal for using the Oakley key exchange [Oakley] in conjunction with ISAKMP. Requirements that should be evaluated when choosing a key establishment algorithm include establishment method (generation vs. transport), perfect forward secrecy, computational overhead, key escrow, and key strength. Based on user requirements, ISAKMP allows an entity initiating communications to indicate which key exchanges it supports. After selection of a key exchange, the protocol provides the messages required to support the actual key establishment.

1.7 ISAKMP Protection

1.7.1 Anti-Clogging (Denial of Service)

Of the numerous security services available, protection against denial of service always seems to be one of the most difficult to address. A "cookie" or anti-clogging token (ACT) is aimed at protecting the computing resources from attack without spending excessive CPU resources to determine its authenticity. An exchange prior to CPU-intensive public key operations can thwart some denial of service attempts (e.g. simple flooding with bogus IP source addresses). Absolute protection against denial of service is impossible, but this anti-clogging token provides a technique for making it easier to handle. The use of an anti-clogging token was introduced by Karn and Simpson in [Karn].

It should be noted that in the exchanges shown in section 4, the anticlogging mechanism should be used in conjuction with a garbage-state collection mechanism; an attacker can still flood a server using packets with bogus IP addresses and cause state to be created. Such aggressive memory management techniques SHOULD be employed by protocols using ISAKMP that do not go through an initial, anti-clogging only phase, as was done in [Karn].

1.7.2 Connection Hijacking

ISAKMP prevents connection hijacking by linking the authentication, key exchange and security association exchanges. This linking prevents an attacker from allowing the authentication to complete and then jumping in and impersonating one entity to the other during the key and security association exchanges.

1.7.3 Man-in-the-Middle Attacks

Man-in-the-Middle attacks include interception, insertion, deletion, and modification of messages, reflecting messages back at the sender, replaying old messages and redirecting messages. ISAKMP features prevent these types of attacks from being successful. The linking of the ISAKMP exchanges prevents the insertion of messages in the protocol exchange. The ISAKMP protocol state machine is defined so deleted messages will not cause a partial SA to be created, the state machine will clear all state and return to idle. The state machine also prevents reflection of a message from causing harm. The requirement for a new cookie with time variant material for each new SA establishment prevents attacks that involve replaying old messages. The ISAKMP strong authentication requirement prevents an SA from being established with anyone other than the intended party. Messages may be redirected to a different destination or modified but this will be detected and an SA will not be established. The ISAKMP specification defines where abnormal processing has occurred and recommends notifying the appropriate party of this abnormality.

1.8 Multicast Communications

It is expected that multicast communications will require the same security services as unicast communications and may introduce the need for additional security services. The issues of distributing SPIs for multicast traffic are presented in [SEC-ARCH]. Multicast security issues are also discussed in [RFC-1949] and [BC]. A future extension to ISAKMP will support multicast key distribution. For an introduction to the issues related to multicast security, consult the Internet Drafts, [RFC-2094] and [RFC-2093], describing Sparta's research in this area.

2 Terminology and Concepts

2.1 ISAKMP Terminology

Security Protocol: A Security Protocol consists of an entity at a single point in the network stack, performing a security service for network communication. For example, IPSEC ESP and IPSEC AH are two different security protocols. TLS is another example. Security Protocols may perform more than one service, for example providing integrity and confidentiality in one module.

Protection Suite: A protection suite is a list of the security services that must be applied by various security protocols. For example, a protection suite may consist of DES encryption in IP ESP, and keyed MD5 in IP AH. All of the protections in a suite must be treated as a single unit. This is necessary because security services in different security protocols can have subtle interactions, and the effects of a suite must be analyzed and verified as a whole.

Security Association (SA): A Security Association is a security-protocol- specific set of parameters that completely defines the services and mechanisms necessary to protect traffic at that security protocol location. These parameters can include algorithm identifiers, modes, cryptographic keys, etc. The SA is referred to by its associated security protocol (for example, "ISAKMP SA", "ESP SA", "TLS SA").

ISAKMP SA: An SA used by the ISAKMP servers to protect their own traffic. Sections 2.3 and 2.4 provide more details about ISAKMP SAs.

Security Parameter Index (SPI): An identifier for a Security Assocation, relative to some security protocol. Each security protocol has its own "SPI-space". A (security protocol, SPI) pair may uniquely identify an SA. The uniqueness of the SPI is implementation dependent, but could be based per system, per protocol, or other options. Depending on the DOI, additional information (e.g. host address) may be necessary to identify an SA. The DOI will also determine which SPIs (i.e. initiator's or responder's) are sent during communication.

Domain of Interpretation: A Domain of Interpretation (DOI) defines payload formats, exchange types, and conventions for naming security-relevant information such as security policies or cryptographic algorithms and modes. A Domain of Interpretation (DOI) identifier is used to interpret the payloads of ISAKMP payloads. A system SHOULD support multiple Domains of Interpretation simultaneously. The concept of a DOI is based on previous work by the TSIG CIPSO Working Group, but extends beyond security label interpretation to include naming and interpretation of security services. A DOI defines:

- o A "situation": the set of information that will be used to determine the required security services.
- o The set of security policies that must, and may, be supported.
- o A syntax for the specification of proposed security services.
- o A scheme for naming security-relevant information, including encryption algorithms, key exchange algorithms, security policy attributes, and certificate authorities.
- o The specific formats of the various payload contents.
- o Additional exchange types, if required.

The rules for the IETF IP Security DOI are presented in [IPDOI]. Specifications of the rules for customized DOIs will be presented in separate documents.

Situation: A situation contains all of the security-relevant information that a system considers necessary to decide the security services required to protect the session being negotiated. The situation may include addresses, security classifications, modes of operation (normal vs. emergency), etc.

Proposal: A proposal is a list, in decreasing order of preference, of the protection suites that a system considers acceptable to protect traffic under a given situation.

Payload: ISAKMP defines several types of payloads, which are used to transfer information such as security association data, or key exchange data, in DOI-defined formats. A payload consists of a generic payload header and a string of octects that is opaque to ISAKMP. ISAKMP uses DOI- specific functionality to synthesize and interpret these payloads. Multiple payloads can be sent in a single ISAKMP message. See section 3 for more details on the payload types, and [IPDOI] for the formats of the IETF IP Security DOI payloads.

Exchange Type: An exchange type is a specification of the number of messages in an ISAKMP exchange, and the payload types that are contained in each of those messages. Each exchange type is designed to provide a particular set of security services, such as anonymity of the participants, perfect forward secrecy of the keying material, authentication of the participants, etc. Section 4.1 defines the default set of ISAKMP exchange types. Other exchange types can be added to support additional key exchanges, if required.

2.2 ISAKMP Placement

Figure 1 is a high level view of the placement of ISAKMP within a system context in a network architecture. An important part of negotiating security services is to consider the entire "stack" of individual SAs as a unit. This is referred to as a "protection suite".

```
     +------------+        +--------+                +--------------+
     !     DOI    !        !        !                !  Application !
     ! Definition ! <----> ! ISAKMP !                !    Process   !
     +------------+    --> !        !                !--------------!
    +--------------+   !   +--------+                ! Appl Protocol!
    ! Key Exchange !   !     ^  ^                    +--------------+
    !  Definition  !<--      !  !                           ^
    +--------------+         !  !                           !
                             !  !                           !
            !----------------!  !                           !
            v                   !                           !
        +-------+               v                           v
        !  API  !        +---------------------------------------------+
        +-------+        !                Socket Layer                 !
            !            !---------------------------------------------!
            v            !        Transport Protocol (TCP / UDP)       !
     +----------+        !---------------------------------------------!
     ! Security ! <----> !                     IP                      !
     ! Protocol !        !---------------------------------------------!
     +----------+        !             Link Layer Protocol             !
                         +---------------------------------------------+
```

Figure 1: ISAKMP Relationships

2.3 Negotiation Phases

ISAKMP offers two "phases" of negotiation. In the first phase, two entities (e.g. ISAKMP servers) agree on how to protect further negotiation traffic between themselves, establishing an ISAKMP SA. This ISAKMP SA is then used to protect the negotiations for the Protocol SA being requested. Two entities (e.g. ISAKMP servers) can negotiate (and have active) multiple ISAKMP SAs.

The second phase of negotiation is used to establish security associations for other security protocols. This second phase can be used to establish many security associations. The security associations established by ISAKMP during this phase can be used by a security protocol to protect many message/data exchanges.

While the two-phased approach has a higher start-up cost for most simple scenarios, there are several reasons that it is beneficial for most cases.

First, entities (e.g. ISAKMP servers) can amortize the cost of the first phase across several second phase negotiations. This allows multiple SAs to be established between peers over time without having to start over for each communication.

Second, security services negotiated during the first phase provide security properties for the second phase. For example, after the first phase of negotiation, the encryption provided by the ISAKMP SA can provide identity protection, potentially allowing the use of simpler second-phase exchanges. On the other hand, if the channel established during the first phase is not adequate to protect identities, then the second phase must negotiate adequate security mechanisms.

Third, having an ISAKMP SA in place considerably reduces the cost of ISAKMP management activity - without the "trusted path" that an ISAKMP SA gives you, the entities (e.g. ISAKMP servers) would have to go through a complete re-authentication for each error notification or deletion of an SA.

Negotiation during each phase is accomplished using ISAKMP-defined exchanges (see section 4) or exchanges defined for a key exchange within a DOI.

Note that security services may be applied differently in each negotiation phase. For example, different parties are being authenticated during each of the phases of negotiation. During the first phase, the parties being authenticated may be the ISAKMP servers/hosts, while during the second phase, users or application level programs are being authenticated.

2.4 Identifying Security Associations

While bootstrapping secure channels between systems, ISAKMP cannot assume the existence of security services, and must provide some protections for itself. Therefore, ISAKMP considers an ISAKMP Security Association to be different than other types, and manages ISAKMP SAs itself, in their own name space. ISAKMP uses the two cookie fields in the ISAKMP header to identify ISAKMP SAs. The Message ID in the ISAKMP Header and the SPI field in the Proposal payload are used during SA establishment to identify the SA for other security protocols. The interpretation of these four fields is dependent on the operation taking place.

The following table shows the presence or absence of several fields during SA establishment. The following fields are necessary for various operations associated with SA establishment: cookies in the ISAKMP header, the ISAKMP Header Message ID field, and the SPI field in the Proposal payload. An 'X' in the column means the value MUST be present. An 'NA' in the column means a value in the column is Not Applicable to the operation.

#	Operation	I-Cookie	R-Cookie	Message ID	SPI
(1)	Start ISAKMP SA negotiation	X	0	0	0
(2)	Respond ISAKMP SA negotiation	X	X	0	0
(3)	Init other SA negotiation	X	X	X	X
(4)	Respond other SA negotiation	X	X	X	X
(5)	Other (KE, ID, etc.)	X	X	X/0	NA
(6)	Security Protocol (ESP, AH)	NA	NA	NA	X

In the first line (1) of the table, the initiator includes the Initiator Cookie field in the ISAKMP Header, using the procedures outlined in sections 2.5.3 and 3.1.

In the second line (2) of the table, the responder includes the Initiator and Responder Cookie fields in the ISAKMP Header, using the procedures outlined in sections 2.5.3 and 3.1. Additional messages may be exchanged between ISAKMP peers, depending on the ISAKMP exchange type used during the phase 1 negotiation. Once the phase 1 exchange is completed, the Initiator and Responder cookies are included in the ISAKMP Header of all subsequent communications between the ISAKMP peers.

During phase 1 negotiations, the initiator and responder cookies determine the ISAKMP SA. Therefore, the SPI field in the Proposal payload is redundant and MAY be set to 0 or it MAY contain the transmitting entity's cookie.

In the third line (3) of the table, the initiator associates a Message ID with the Protocols contained in the SA Proposal. This Message ID and the initiator's SPI(s) to be associated with each protocol in the Proposal are sent to the responder. The SPI(s) will be used by the security protocols once the phase 2 negotiation is completed.

In the fourth line (4) of the table, the responder includes the same Message ID and the responder's SPI(s) to be associated with each protocol in the accepted Proposal. This information is returned to the initiator.

In the fifth line (5) of the table, the initiator and responder use the Message ID field in the ISAKMP Header to keep track of the in-progress protocol negotiation. This is only applicable for a phase 2 exchange and the value MUST be 0 for a phase 1 exchange because the combined cookies identify the ISAKMP SA. The SPI field in the Proposal payload is not applicable because the Proposal payload is only used during the SA negotiation message exchange (steps 3 and 4).

In the sixth line (6) of the table, the phase 2 negotiation is complete. The security protocols use the SPI(s) to determine which security services and mechanisms to apply to the communication between them. The SPI value shown in the sixth line (6) is not the SPI field in the Proposal payload, but the SPI field contained within the security protocol header.

During the SA establishment, a SPI MUST be generated. ISAKMP is designed to handle variable sized SPIs. This is accomplished by using the SPI Size field within the Proposal payload during SA establishment. Handling of SPIs will be outlined by the DOI specification (e.g. [IPDOI]).

When a security association (SA) is initially established, one side assumes the role of initiator and the other the role of responder. Once the SA is established, both the original initiator and responder can initiate a phase 2 negotiation with the peer entity. Thus, ISAKMP SAs are bidirectional in nature.

Additionally, ISAKMP allows both initiator and responder to have some control during the negotiation process. While ISAKMP is designed to allow an SA negotiation that includes multiple proposals, the initiator can maintain some control by only making one proposal in accordance with the initiator's local security policy. Once the initiator sends a proposal containing more than one proposal (which are sent in decreasing preference order), the initiator relinquishes control to the responder. Once the responder is controlling the SA establishment, the responder can make its policy take precedence over the initiator within the context of the multiple options offered by the initiator. This is accomplished by selecting the proposal best suited for the responder's local security policy and returning this selection to the initiator.

2.5 Miscellaneous

2.5.1 Transport Protocol

ISAKMP can be implemented over any transport protocol or over IP itself. Implementations MUST include send and receive capability for ISAKMP using the User Datagram Protocol (UDP) on port 500. UDP Port 500 has been assigned to ISAKMP by the Internet Assigned Numbers Authority (IANA). Implementations MAY additionally support ISAKMP over other transport protocols or over IP itself.

2.5.2 RESERVED Fields

The existence of RESERVED fields within ISAKMP payloads are used strictly to preserve byte alignment. All RESERVED fields in the ISAKMP protocol MUST be set to zero (0) when a packet is issued. The receiver SHOULD check the RESERVED fields for a zero (0) value and discard the packet if other values are found.

2.5.3 Anti-Clogging Token ("Cookie") Creation

The details of cookie generation are implementation dependent, but MUST satisfy these basic requirements (originally stated by Phil Karn in [Karn]):

1. The cookie must depend on the specific parties. This prevents an attacker from obtaining a cookie using a real IP address and UDP port, and then using it to swamp the victim with Diffie-Hellman requests from randomly chosen IP addresses or ports.

2. It must not be possible for anyone other than the issuing entity to generate cookies that will be accepted by that entity. This implies that the issuing entity must use local secret information in the generation and subsequent verification of a cookie. It must not be possible to deduce this secret information from any particular cookie.

3. The cookie generation function must be fast to thwart attacks intended to sabotage CPU resources.

Karn's suggested method for creating the cookie is to perform a fast hash (e.g. MD5) over the IP Source and Destination Address, the UDP Source and Destination Ports and a locally generated secret random value. ISAKMP requires that the cookie be unique for each SA establishment to help prevent replay attacks, therefore, the date and time MUST be added to the information hashed. The generated cookies are placed in the ISAKMP Header (described in section 3.1) Initiator and Responder cookie fields. These fields are 8 octets in length, thus, requiring a generated cookie to be 8 octets. Notify and Delete messages (see sections 3.14, 3.15, and 4.8) are uni-directional transmissions and are done under the protection of an existing ISAKMP SA, thus, not requiring the generation of a new cookie. One exception to this is the transmission of a Notify message during a Phase 1 exchange, prior to completing the establishment of an SA. Sections 3.14 and 4.8 provide additional details.

3 ISAKMP Payloads

ISAKMP payloads provide modular building blocks for constructing ISAKMP messages. The presence and ordering of payloads in ISAKMP is defined by and dependent upon the Exchange Type Field located in the ISAKMP Header (see Figure 2). The ISAKMP payload types are discussed in sections 3.4 through 3.15. The descriptions of the ISAKMP payloads, messages, and exchanges (see Section 4) are shown using network octet ordering.

3.1 ISAKMP Header Format

An ISAKMP message has a fixed header format, shown in Figure 2, followed by a variable number of payloads. A fixed header simplifies parsing, providing the benefit of protocol parsing software that is less complex and easier to implement. The fixed header contains the information required by the protocol to maintain state, process payloads and possibly prevent denial of service or replay attacks.

The ISAKMP Header fields are defined as follows:

- o Initiator Cookie (8 octets) - Cookie of entity that initiated SA establishment, SA notification, or SA deletion.

- o Responder Cookie (8 octets) - Cookie of entity that is responding to an SA establishment request, SA notification, or SA deletion.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                          Initiator                            !
!                            Cookie                             !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                          Responder                            !
!                            Cookie                             !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!  Next Payload ! MjVer ! MnVer ! Exchange Type !     Flags     !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                          Message ID                           !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                            Length                             !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: ISAKMP Header Format

- o Next Payload (1 octet) - Indicates the type of the first payload in the message. The format for each payload is defined in sections 3.4 through 3.16. The processing for the payloads is defined in section 5.

Next Payload Type	Value
NONE	0
Security Association (SA)	1
Proposal (P)	2
Transform (T)	3
Key Exchange (KE)	4
Identification (ID)	5
Certificate (CERT)	6
Certificate Request (CR)	7
Hash (HASH)	8
Signature (SIG)	9
Nonce (NONCE)	10
Notification (N)	11
Delete (D)	12
Vendor ID (VID)	13
RESERVED	14 - 127
Private USE	128 - 255

- o Major Version (4 bits) - indicates the major version of the ISAKMP protocol in use. Implementations based on this version of the ISAKMP Internet-Draft MUST set the Major Version to 1. Implementations based on previous versions of ISAKMP Internet-Drafts MUST set the Major Version to 0. Implementations SHOULD never accept packets with a major version number larger than its own.

- o Minor Version (4 bits) - indicates the minor version of the ISAKMP protocol in use. Implementations based on this version of the ISAKMP Internet-Draft MUST set the Minor Version to 0. Implementations based on previous versions of ISAKMP Internet-Drafts MUST set the Minor Version to 1. Implementations SHOULD never accept packets with a minor version number larger than its own, given the major version numbers are identical.

- o Exchange Type (1 octet) - indicates the type of exchange being used. This dictates the message and payload orderings in the ISAKMP exchanges.

Exchange Type	Value
NONE	0
Base	1
Identity Protection	2
Authentication Only	3
Aggressive	4
Informational	5
ISAKMP Future Use	6 - 31
DOI Specific Use	32 - 239
Private Use	240 - 255

- o Flags (1 octet) - indicates specific options that are set for the ISAKMP exchange. The flags listed below are specified in the Flags field beginning with the least significant bit, i.e the Encryption bit is bit 0 of the Flags field, the Commit bit is bit 1 of the Flags field, and the Authentication Only bit is bit 2 of the Flags field. The remaining bits of the Flags field MUST be set to 0 prior to transmission.

  - -- E(ncryption Bit) (1 bit) - If set (1), all payloads following the header are encrypted using the encryption algorithm identified in the ISAKMP SA. The ISAKMP SA Identifier is the combination of the initiator and responder cookie. It is RECOMMENDED that encryption of communications be done as soon as possible between the peers. For all ISAKMP exchanges described in section 4.1, the encryption SHOULD begin after both parties have exchanged Key Exchange payloads. If the E(ncryption Bit) is not set (0), the payloads are not encrypted.

-- C(ommit Bit) (1 bit) - This bit is used to signal key exchange synchronization. It is used to ensure that encrypted material is not received prior to completion of the SA establishment. The Commit Bit can be set (at anytime) by either party participating in the SA establishment, and can be used during both phases of an ISAKMP SA establishment. However, the value MUST be reset after the Phase 1 negotiation. If set(1), the entity which did not set the Commit Bit MUST wait for an Informational Exchange containing a Notify payload (with the CONNECTED Notify Message) from the entity which set the Commit Bit. In this instance, the Message ID field of the Informational Exchange MUST contain the Message ID of the original ISAKMP Phase 2 SA negotiation. This is done to ensure that the Informational Exchange with the CONNECTED Notify Message can be associated with the correct Phase 2 SA. The receipt and processing of the Informational Exchange indicates that the SA establishment was successful and either entity can now proceed with encrypted traffic communication. In addition to synchronizing key exchange, the Commit Bit can be used to protect against loss of transmissions over unreliable networks and guard against the need for multiple re-transmissions.

NOTE: It is always possible that the final message of an exchange can be lost. In this case, the entity expecting to receive the final message of an exchange would receive the Phase 2 SA negotiation message following a Phase 1 exchange or encrypted traffic following a Phase 2 exchange. Handling of this situation is not standardized, but we propose the following possibilities. If the entity awaiting the Informational Exchange can verify the received message (i.e. Phase 2 SA negotiation message or encrypted traffic), then they MAY consider the SA was established and continue processing. The other option is to retransmit the last ISAKMP message to force the other entity to retransmit the final message. This suggests that implementations may consider retaining the last message (locally) until they are sure the SA is established.

-- A(uthentication Only Bit) (1 bit) - This bit is intended for use with the Informational Exchange with a Notify payload and will allow the transmission of information with integrity checking, but no encryption (e.g. "emergency mode"). Section 4.8 states that a Phase 2 Informational Exchange MUST be sent under the protection of an ISAKMP SA. This is the only exception to that policy. If the Authentication Only bit is set (1), only authentication security services will be applied to the entire Notify payload of the Informational Exchange and the payload will not be encrypted.

o Message ID (4 octets) - Unique Message Identifier used to identify protocol state during Phase 2 negotiations. This value is randomly generated by the initiator of the Phase 2 negotiation. In the event of simultaneous SA establishments (i.e. collisions), the value of this field will likely be different because they are independently generated and, thus, two security associations will progress toward establishment. However, it is unlikely there will be absolute simultaneous establishments. During Phase 1 negotiations, the value MUST be set to 0.

o Length (4 octets) - Length of total message (header + payloads) in octets. Encryption can expand the size of an ISAKMP message.

3.2 Generic Payload Header

Each ISAKMP payload defined in sections 3.4 through 3.16 begins with a generic header, shown in Figure 3, which provides a payload "chaining" capability and clearly defines the boundaries of a payload.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !   RESERVED    !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: Generic Payload Header

The Generic Payload Header fields are defined as follows:

o Next Payload (1 octet) - Identifier for the payload type of the next payload in the message. If the current payload is the last in the message, then this field will be 0. This field provides the "chaining" capability.

o RESERVED (1 octet) - Unused, set to 0.

o Payload Length (2 octets) - Length in octets of the current payload, including the generic payload header.

3.3 Data Attributes

There are several instances within ISAKMP where it is necessary to represent Data Attributes. An example of this is the Security Association (SA) Attributes contained in the Transform payload (described in section 3.6). These Data Attributes are not an ISAKMP payload, but are contained within ISAKMP payloads. The format of the Data Attributes provides the flexibility for representation of many different types of information. There can be multiple Data Attributes within a payload. The length of the Data Attributes will either be 4 octets or defined by the Attribute Length field. This is done using the Attribute Format bit described below. Specific information about the attributes for each domain will be described in a DOI document, e.g. IPSEC DOI [IPDOI].

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!A!       Attribute Type        !    AF=0  Attribute Length     !
!F!                             !    AF=1  Attribute Value      !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                   AF=0  Attribute Value                       .
.                   AF=1  Not Transmitted                       .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: Data Attributes

The Data Attributes fields are defined as follows:

o Attribute Type (2 octets) - Unique identifier for each type of attribute. These attributes are defined as part of the DOI-specific information.

  The most significant bit, or Attribute Format (AF), indicates whether the data attributes follow the Type/Length/Value (TLV) format or a shortened Type/Value (TV) format. If the AF bit is a zero (0), then the Data Attributes are of the Type/Length/Value (TLV) form. If the AF bit is a one (1), then the Data Attributes are of the Type/Value form.

o Attribute Length (2 octets) - Length in octets of the Attribute Value. When the AF bit is a one (1), the Attribute Value is only 2 octets and the Attribute Length field is not present.

o Attribute Value (variable length) - Value of the attribute associated with the DOI-specific Attribute Type. If the AF bit is a zero (0), this field has a variable length defined by the Attribute Length field. If the AF bit is a one (1), the Attribute Value has a length of 2 octets.

3.4 Security Association Payload

The Security Association Payload is used to negotiate security attributes and to indicate the Domain of Interpretation (DOI) and Situation under which the negotiation is taking place. Figure 5 shows the format of the Security Association payload.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !   RESERVED    !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!              Domain of Interpretation  (DOI)                  !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                           Situation                           ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: Security Association Payload

- o Next Payload (1 octet) - Identifier for the payload type of the next payload in the message. If the current payload is the last in the message, then this field will be 0. This field MUST NOT contain the values for the Proposal or Transform payloads as they are considered part of the security association negotiation. For example, this field would contain the value "10" (Nonce payload) in the first message of a Base Exchange (see Section 4.4) and the value "0" in the first message of an Identity Protect Exchange (see Section 4.5).

- o RESERVED (1 octet) - Unused, set to 0.

- o Payload Length (2 octets) - Length in octets of the entire Security Association payload, including the SA payload, all Proposal payloads, and all Transform payloads associated with the proposed Security Association.

- o Domain of Interpretation (4 octets) - Identifies the DOI (as described in Section 2.1) under which this negotiation is taking place. The DOI is a 32-bit unsigned integer. A DOI value of 0 during a Phase 1 exchange specifies a Generic ISAKMP SA which can be used for any protocol during the Phase 2 exchange. The necessary SA Attributes are defined in A.4. A DOI value of 1 is assigned to the IPsec DOI [IPDOI]. All other DOI values are reserved to IANA for future use. IANA will not normally assign a DOI value without referencing some public specification, such as an Internet RFC. Other DOI's can be defined using the description in appendix B. This field MUST be present within the Security Association payload.

- o Situation (variable length) - A DOI-specific field that identifies the situation under which this negotiation is taking place. The Situation is used to make policy decisions regarding the security attributes being negotiated. Specifics for the IETF IP Security DOI Situation are detailed in [IPDOI]. This field MUST be present within the Security Association payload.

3.5 Proposal Payload

The Proposal Payload contains information used during Security Association negotiation. The proposal consists of security mechanisms, or transforms, to be used to secure the communications channel. Figure 6 shows the format of the Proposal Payload. A description of its use can be found in section 4.2.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !   RESERVED    !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!  Proposal #   !  Protocol-Id  !    SPI Size   !# of Transforms!
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                        SPI (variable)                         !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 6: Proposal Payload Format

The Proposal Payload fields are defined as follows:

- o Next Payload (1 octet) - Identifier for the payload type of the next payload in the message. This field MUST only contain the value "2" or "0". If there are additional Proposal payloads in the message, then this field will be 2. If the current Proposal payload is the last within the security association proposal, then this field will be 0.

- o RESERVED (1 octet) - Unused, set to 0.

- o Payload Length (2 octets) - Length in octets of the entire Proposal payload, including generic payload header, the Proposal payload, and all Transform payloads associated with this proposal. In the event there are multiple proposals with the same proposal number (see section 4.2), the Payload Length field only applies to the current Proposal payload and not to all Proposal payloads.

o Proposal # (1 octet) - Identifies the Proposal number for the current payload. A description of the use of this field is found in section 4.2.

o Protocol-Id (1 octet) - Specifies the protocol identifier for the current negotiation. Examples might include IPSEC ESP, IPSEC AH, OSPF, TLS, etc.

o SPI Size (1 octet) - Length in octets of the SPI as defined by the Protocol-Id. In the case of ISAKMP, the Initiator and Responder cookie pair from the ISAKMP Header is the ISAKMP SPI, therefore, the SPI Size is irrelevant and MAY be from zero (0) to sixteen (16). If the SPI Size is non-zero, the content of the SPI field MUST be ignored. If the SPI Size is not a multiple of 4 octets it will have some impact on the SPI field and the alignment of all payloads in the message. The Domain of Interpretation (DOI) will dictate the SPI Size for other protocols.

o # of Transforms (1 octet) - Specifies the number of transforms for the Proposal. Each of these is contained in a Transform payload.

o SPI (variable) - The sending entity's SPI. In the event the SPI Size is not a multiple of 4 octets, there is no padding applied to the payload, however, it can be applied at the end of the message.

The payload type for the Proposal Payload is two (2).

3.6 Transform Payload

The Transform Payload contains information used during Security Association negotiation. The Transform payload consists of a specific security mechanism, or transforms, to be used to secure the communications channel. The Transform payload also contains the security association attributes associated with the specific transform. These SA attributes are DOI-specific. Figure 7 shows the format of the Transform Payload. A description of its use can be found in section 4.2.

```
                      1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 ! Next Payload  !   RESERVED    !         Payload Length        !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 !  Transform #  !  Transform-Id !           RESERVED2           !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 !                                                               !
 ~                        SA Attributes                          ~
 !                                                               !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: Transform Payload Format

The Transform Payload fields are defined as follows:

- o Next Payload (1 octet) - Identifier for the payload type of the next payload in the message. This field MUST only contain the value "3" or "0". If there are additional Transform payloads in the proposal, then this field will be 3. If the current Transform payload is the last within the proposal, then this field will be 0.

- o RESERVED (1 octet) - Unused, set to 0.

- o Payload Length (2 octets) - Length in octets of the current payload, including the generic payload header, Transform values, and all SA Attributes.

- o Transform # (1 octet) - Identifies the Transform number for the current payload. If there is more than one transform proposed for a specific protocol within the Proposal payload, then each Transform payload has a unique Transform number. A description of the use of this field is found in section 4.2.

- o Transform-Id (1 octet) - Specifies the Transform identifier for the protocol within the current proposal. These transforms are defined by the DOI and are dependent on the protocol being negotiated.

- o RESERVED2 (2 octets) - Unused, set to 0.

- o SA Attributes (variable length) - This field contains the security association attributes as defined for the transform given in the Transform-Id field. The SA Attributes SHOULD be represented using the Data Attributes format described in section 3.3. If the SA Attributes are not aligned on 4-byte boundaries, then subsequent payloads will not be aligned and any padding will be added at the end of the message to make the message 4-octet aligned.

The payload type for the Transform Payload is three (3).

3.7 Key Exchange Payload

The Key Exchange Payload supports a variety of key exchange techniques. Example key exchanges are Oakley [Oakley], Diffie-Hellman, the enhanced Diffie-Hellman key exchange described in X9.42 [ANSI], and the RSA-based key exchange used by PGP. Figure 8 shows the format of the Key Exchange payload.

The Key Exchange Payload fields are defined as follows:

- o Next Payload (1 octet) - Identifier for the payload type of the nextpayload in the message. If the current payload is the last in the message, then this field will be 0.

```
                      1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 ! Next Payload  !   RESERVED    !         Payload Length        !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 !                                                               !
 ~                       Key Exchange Data                       ~
 !                                                               !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8: Key Exchange Payload Format

- o RESERVED (1 octet) - Unused, set to 0.
- o Payload Length (2 octets) - Length in octets of the current payload, including the generic payload header.
- o Key Exchange Data (variable length) - Data required to generate a session key. The interpretation of this data is specified by the DOI and the associated Key Exchange algorithm. This field may also contain pre-placed key indicators.

The payload type for the Key Exchange Payload is four (4).

3.8 Identification Payload

The Identification Payload contains DOI-specific data used to exchange identification information. This information is used for determining the identities of communicating peers and may be used for determining authenticity of information. Figure 9 shows the format of the Identification Payload.

The Identification Payload fields are defined as follows:

o Next Payload (1 octet) - Identifier for the payload type of the next payload in the message. If the current payload is the last in the message, then this field will be 0.

o RESERVED (1 octet) - Unused, set to 0.

o Payload Length (2 octets) - Length in octets of the current payload, including the generic payload header.

o ID Type (1 octet) - Specifies the type of Identification being used.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload !   RESERVED    !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!   ID Type     !             DOI Specific ID Data              !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                   Identification Data                         ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 9: Identification Payload Format

This field is DOI-dependent.

o DOI Specific ID Data (3 octets) - Contains DOI specific Identification data. If unused, then this field MUST be set to 0.

o Identification Data (variable length) - Contains identity information. The values for this field are DOI-specific and the format is specified by the ID Type field. Specific details for the IETF IP Security DOI Identification Data are detailed in [IPDOI].

The payload type for the Identification Payload is five (5).

3.9 Certificate Payload

The Certificate Payload provides a means to transport certificates or other certificate-related information via ISAKMP and can appear in any ISAKMP message. Certificate payloads SHOULD be included in an exchange whenever an appropriate directory service (e.g. Secure DNS [DNSSEC]) is not available to distribute certificates. The Certificate payload MUST be accepted at any point during an exchange. Figure 10 shows the format of the Certificate Payload.

NOTE: Certificate types and formats are not generally bound to a DOI - it is expected that there will only be a few certificate types, and that most DOIs will accept all of these types.

The Certificate Payload fields are defined as follows:

- o Next Payload (1 octet) - Identifier for the payload type of the next payload in the message. If the current payload is the last in the message, then this field will be 0.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !   RESERVED    !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Cert Encoding !                                               !
+-+-+-+-+-+-+-+-+                                               !
~                       Certificate Data                        ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 10: Certificate Payload Format

- o RESERVED (1 octet) - Unused, set to 0.

- o Payload Length (2 octets) - Length in octets of the current payload, including the generic payload header.

- o Certificate Encoding (1 octet) - This field indicates the type of certificate or certificate-related information contained in the Certificate Data field.

Certificate Type	Value
NONE	0
PKCS #7 wrapped X.509 certificate	1
PGP Certificate	2
DNS Signed Key	3
X.509 Certificate - Signature	4
X.509 Certificate - Key Exchange	5
Kerberos Tokens	6
Certificate Revocation List (CRL)	7
Authority Revocation List (ARL)	8
SPKI Certificate	9
X.509 Certificate - Attribute	10
RESERVED	11 - 255

- o Certificate Data (variable length) - Actual encoding of certificate data. The type of certificate is indicated by the Certificate Encoding field.

The payload type for the Certificate Payload is six (6).

3.10 Certificate Request Payload

The Certificate Request Payload provides a means to request certificates via ISAKMP and can appear in any message. Certificate Request payloads SHOULD be included in an exchange whenever an appropriate directory service (e.g. Secure DNS [DNSSEC]) is not available to distribute certificates. The Certificate Request payload MUST be accepted at any point during the exchange. The responder to the Certificate Request payload MUST send its certificate, if certificates are supported, based on the values contained in the payload. If multiple certificates are required, then multiple Certificate Request payloads SHOULD be transmitted. Figure 11 shows the format of the Certificate Request Payload.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !   RESERVED    !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!  Cert. Type   !                                               !
+-+-+-+-+-+-+-+-+                                               !
~                    Certificate Authority                      ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 11: Certificate Request Payload Format

The Certificate Payload fields are defined as follows:

- o Next Payload (1 octet) - Identifier for the payload type of the next payload in the message. If the current payload is the last in the message, then this field will be 0.

- o RESERVED (1 octet) - Unused, set to 0.

- o Payload Length (2 octets) - Length in octets of the current payload, including the generic payload header.

- o Certificate Type (1 octet) - Contains an encoding of the type of certificate requested. Acceptable values are listed in section 3.9.

- o Certificate Authority (variable length) - Contains an encoding of an acceptable certificate authority for the type of certificate requested. As an example, for an X.509 certificate this field would contain the Distinguished Name encoding of the Issuer Name of an X.509 certificate authority acceptable to the sender of this payload. This would be included to assist the responder in determining how much of the certificate chain would need to be sent in response to this request. If there is no specific certificate authority requested, this field SHOULD not be included.

The payload type for the Certificate Request Payload is seven (7).

3.11 Hash Payload

The Hash Payload contains data generated by the hash function (selected during the SA establishment exchange), over some part of the message and/or ISAKMP state. This payload may be used to verify the integrity of the data in an ISAKMP message or for authentication of the negotiating entities. Figure 12 shows the format of the Hash Payload.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !   RESERVED    !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                           Hash Data                           ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 12: Hash Payload Format

The Hash Payload fields are defined as follows:

- o Next Payload (1 octet) - Identifier for the payload type of the next payload in the message. If the current payload is the last in the message, then this field will be 0.
- o RESERVED (1 octet) - Unused, set to 0.
- o Payload Length (2 octets) - Length in octets of the current payload, including the generic payload header.
- o Hash Data (variable length) - Data that results from applying the hash routine to the ISAKMP message and/or state.

3.12 Signature Payload

The Signature Payload contains data generated by the digital signature function (selected during the SA establishment exchange), over some part of the message and/or ISAKMP state. This payload is used to verify the integrity of the data in the ISAKMP message, and may be of use for non-repudiation services. Figure 13 shows the format of the Signature Payload.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !   RESERVED    !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                       Signature Data                          ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 13: Signature Payload Format

The Signature Payload fields are defined as follows:

- o Next Payload (1 octet) - Identifier for the payload type of the next payload in the message. If the current payload is the last in the message, then this field will be 0.

- o RESERVED (1 octet) - Unused, set to 0.

- o Payload Length (2 octets) - Length in octets of the current payload, including the generic payload header.

- o Signature Data (variable length) - Data that results from applying the digital signature function to the ISAKMP message and/or state.

The payload type for the Signature Payload is nine (9).

3.13 Nonce Payload

The Nonce Payload contains random data used to guarantee liveness during an exchange and protect against replay attacks. Figure 14 shows the format of the Nonce Payload. If nonces are used by a particular key exchange, the use of the Nonce payload will be dictated by the key exchange. The nonces may be transmitted as part of the key exchange data, or as a separate payload. However, this is defined by the key exchange, not by ISAKMP.

```
                         1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    ! Next Payload  !   RESERVED    !         Payload Length        !
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    !                                                               !
    ~                            Nonce Data                         ~
    !                                                               !
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 14: Nonce Payload Format

The Nonce Payload fields are defined as follows:

- o Next Payload (1 octet) - Identifier for the payload type of the next payload in the message. If the current payload is the last in the message, then this field will be 0.

- o RESERVED (1 octet) - Unused, set to 0.

- o Payload Length (2 octets) - Length in octets of the current payload, including the generic payload header.

- o Nonce Data (variable length) - Contains the random data generated by the transmitting entity.

The payload type for the Nonce Payload is ten (10).

3.14 Notification Payload

The Notification Payload can contain both ISAKMP and DOI-specific data and is used to transmit informational data, such as error conditions, to an ISAKMP peer. It is possible to send multiple Notification payloads in a single ISAKMP message. Figure 15 shows the format of the Notification Payload.

Notification which occurs during, or is concerned with, a Phase 1 negotiation is identified by the Initiator and Responder cookie pair in the ISAKMP Header. The Protocol Identifier, in this case, is ISAKMP and the SPI value is 0 because the cookie pair in the ISAKMP Header identifies the ISAKMP SA. If the notification takes place prior to the completed exchange of keying information, then the notification will be unprotected.

Notification which occurs during, or is concerned with, a Phase 2 negotiation is identified by the Initiator and Responder cookie pair in the ISAKMP Header and the Message ID and SPI associated with the current negotiation. One example for this type of notification is to indicate why a proposal was rejected.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !   RESERVED    !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!              Domain of Interpretation  (DOI)                  !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!  Protocol-ID  !   SPI Size    !      Notify Message Type      !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                Security Parameter Index (SPI)                 ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                       Notification Data                       ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 15: Notification Payload Format

The Notification Payload fields are defined as follows:

- o Next Payload (1 octet) - Identifier for the payload type of the next payload in the message. If the current payload is the last in the message, then this field will be 0.
- o RESERVED (1 octet) - Unused, set to 0.
- o Payload Length (2 octets) - Length in octets of the current payload, including the generic payload header.
- o Domain of Interpretation (4 octets) - Identifies the DOI (as described in Section 2.1) under which this notification is taking place. For ISAKMP this value is zero (0) and for the IPSEC DOI it is one (1). Other DOI's can be defined using the description in appendix B.
- o Protocol-Id (1 octet) - Specifies the protocol identifier for the current notification. Examples might include ISAKMP, IPSEC ESP, IPSEC AH, OSPF, TLS, etc.

- o SPI Size (1 octet) - Length in octets of the SPI as defined by the Protocol-Id. In the case of ISAKMP, the Initiator and Responder cookie pair from the ISAKMP Header is the ISAKMP SPI, therefore, the SPI Size is irrelevant and MAY be from zero (0) to sixteen (16). If the SPI Size is non-zero, the content of the SPI field MUST be ignored. The Domain of Interpretation (DOI) will dictate the SPI Size for other protocols.

- o Notify Message Type (2 octets) - Specifies the type of notification message (see section 3.14.1). Additional text, if specified by the DOI, is placed in the Notification Data field.

- o SPI (variable length) - Security Parameter Index. The receiving entity's SPI. The use of the SPI field is described in section 2.4. The length of this field is determined by the SPI Size field and is not necessarily aligned to a 4 octet boundary.

- o Notification Data (variable length) - Informational or error data transmitted in addition to the Notify Message Type. Values for this field are DOI-specific.

The payload type for the Notification Payload is eleven (11).

3.14.1 Notify Message Types

Notification information can be error messages specifying why an SA could not be established. It can also be status data that a process managing an SA database wishes to communicate with a peer process. For example, a secure front end or security gateway may use the Notify message to synchronize SA communication. The table below lists the Nofitication messages and their corresponding values. Values in the Private Use range are expected to be DOI-specific values.

NOTIFY MESSAGES - ERROR TYPES

Errors	Value
INVALID-PAYLOAD-TYPE	1
DOI-NOT-SUPPORTED	2
SITUATION-NOT-SUPPORTED	3
INVALID-COOKIE	4
INVALID-MAJOR-VERSION	5
INVALID-MINOR-VERSION	6
INVALID-EXCHANGE-TYPE	7
INVALID-FLAGS	8
INVALID-MESSAGE-ID	9
INVALID-PROTOCOL-ID	10
INVALID-SPI	11

INVALID-TRANSFORM-ID	12
ATTRIBUTES-NOT-SUPPORTED	13
NO-PROPOSAL-CHOSEN	14
BAD-PROPOSAL-SYNTAX	15
PAYLOAD-MALFORMED	16
INVALID-KEY-INFORMATION	17
INVALID-ID-INFORMATION	18
INVALID-CERT-ENCODING	19
INVALID-CERTIFICATE	20
CERT-TYPE-UNSUPPORTED	21
INVALID-CERT-AUTHORITY	22
INVALID-HASH-INFORMATION	23
AUTHENTICATION-FAILED	24
INVALID-SIGNATURE	25
ADDRESS-NOTIFICATION	26
NOTIFY-SA-LIFETIME	27
CERTIFICATE-UNAVAILABLE	28
UNSUPPORTED-EXCHANGE-TYPE	29
UNEQUAL-PAYLOAD-LENGTHS	30
RESERVED (Future Use)	31 - 8191
Private Use	8192 - 16383

NOTIFY MESSAGES - STATUS TYPES

Status	Value
CONNECTED	16384
RESERVED (Future Use)	16385 - 24575
DOI-specific codes	24576 - 32767
Private Use	32768 - 40959
RESERVED (Future Use)	40960 - 65535

3.15 Delete Payload

The Delete Payload contains a protocol-specific security association identifier that the sender has removed from its security association database and is, therefore, no longer valid. Figure 16 shows the format of the Delete Payload. It is possible to send multiple SPIs in a Delete payload, however, each SPI MUST be for the same protocol. Mixing of Protocol Identifiers MUST NOT be performed with the Delete payload.

Deletion which is concerned with an ISAKMP SA will contain a Protocol-Id of ISAKMP and the SPIs are the initiator and responder cookies from the ISAKMP Header. Deletion which is concerned with a Protocol SA, such as ESP or AH, will contain the Protocol-Id of that protocol (e.g. ESP, AH) and the SPI is the sending entity's SPI(s).

NOTE: The Delete Payload is not a request for the responder to delete an SA, but an advisory from the initiator to the responder. If the responder chooses to ignore the message, the next communication from the responder to the initiator, using that security association, will fail. A responder is not expected to acknowledge receipt of a Delete payload.

```
                      1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 ! Next Payload  !   RESERVED    !         Payload Length        !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 !              Domain of Interpretation  (DOI)                  !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 !  Protocol-Id  !   SPI Size    !           # of SPIs           !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 !                                                               !
 ~               Security Parameter Index(es) (SPI)              ~
 !                                                               !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 16: Delete Payload Format

The Delete Payload fields are defined as follows:

- o Next Payload (1 octet) - Identifier for the payload type of the next payload in the message. If the current payload is the last in the message, then this field will be 0.

- o RESERVED (1 octet) - Unused, set to 0.

- o Payload Length (2 octets) - Length in octets of the current payload, including the generic payload header.

- o Domain of Interpretation (4 octets) - Identifies the DOI (as described in Section 2.1) under which this deletion is taking place. For ISAKMP this value is zero (0) and for the IPSEC DOI it is one (1). Other DOI's can be defined using the description in appendix B.

- o Protocol-Id (1 octet) - ISAKMP can establish security associations for various protocols, including ISAKMP and IPSEC. This field identifies which security association database to apply the delete request.

- o SPI Size (1 octet) - Length in octets of the SPI as defined by the Protocol-Id. In the case of ISAKMP, the Initiator and Responder cookie pair is the ISAKMP SPI. In this case, the SPI Size would be 16 octets for each SPI being deleted.

- o # of SPIs (2 octets) - The number of SPIs contained in the Delete payload. The size of each SPI is defined by the SPI Size field.

- o Security Parameter Index(es) (variable length) - Identifies the specific security association(s) to delete. Values for this field are DOI and protocol specific. The length of this field is determined by the SPI Size and # of SPIs fields.

The payload type for the Delete Payload is twelve (12).

3.16 Vendor ID Payload

The Vendor ID Payload contains a vendor defined constant. The constant is used by vendors to identify and recognize remote instances of their implementations. This mechanism allows a vendor to experiment with new features while maintaining backwards compatibility. This is not a general extension facility of ISAKMP. Figure 17 shows the format of the Vendor ID Payload.

The Vendor ID payload is not an announcement from the sender that it will send private payload types. A vendor sending the Vendor ID MUST not make any assumptions about private payloads that it may send unless a Vendor ID is received as well. Multiple Vendor ID payloads MAY be sent. An implementation is NOT REQUIRED to understand any Vendor ID payloads. An implementation is NOT REQUIRED to send any Vendor ID payload at all. If a private payload was sent without prior agreement to send it, a compliant implementation may reject a proposal with a notify message of type INVALID-PAYLOAD-TYPE.

If a Vendor ID payload is sent, it MUST be sent during the Phase 1 negotiation. Reception of a familiar Vendor ID payload in the Phase 1 negotiation allows an implementation to make use of Private USE payload numbers (128-255), described in section 3.1 for vendor specific extensions during Phase 2 negotiations. The definition of "familiar" is left to implementations to determine. Some vendors may wish to implement another vendor's extension prior to standardization. However, this practice SHOULD not be widespread and vendors should work towards standardization instead.

The vendor defined constant MUST be unique. The choice of hash and text to hash is left to the vendor to decide. As an example, vendors could generate their vendor id by taking a plain (non-keyed) hash of a string containing the product name, and the version of the product.

A hash is used instead of a vendor registry to avoid local cryptographic policy problems with having a list of "approved" products, to keep away from maintaining a list of vendors, and to allow classified products to avoid having to appear on any list. For instance:

"Example Company IPsec. Version 97.1"

(not including the quotes) has MD5 hash: 48544f9b1fe662af98b9b39e50c01a5a, when using MD5file. Vendors may include all of the hash, or just a portion of it, as the payload length will bound the data. There are no security implications of this hash, so its choice is arbitrary.

```
                      1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 ! Next Payload  !   RESERVED    !         Payload Length        !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 !                                                               !
 ~                        Vendor ID (VID)                        ~
 !                                                               !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 17: Vendor ID Payload Format

The Vendor ID Payload fields are defined as follows:

- o Next Payload (1 octet) - Identifier for the payload type of the next payload in the message. If the current payload is the last in the message, then this field will be 0.
- o RESERVED (1 octet) - Unused, set to 0.
- o Payload Length (2 octets) - Length in octets of the current payload, including the generic payload header.
- o Vendor ID (variable length) - Hash of the vendor string plus version (as described above).

The payload type for the Vendor ID Payload is thirteen (13).

4 ISAKMP Exchanges

ISAKMP supplies the basic syntax of a message exchange. The basic building blocks for ISAKMP messages are the payload types described in section 3. This section describes the procedures for SA establishment and SA modification, followed by a default set of exchanges that MAY be used for initial interoperability. Other exchanges will be defined depending on the DOI and key exchange. [IPDOI] and [IKE] are examples of how this is achieved. Appendix B explains the procedures for accomplishing these additions.

4.1 ISAKMP Exchange Types

ISAKMP allows the creation of exchanges for the establishment of Security Associations and keying material. There are currently five default Exchange Types defined for ISAKMP. Sections 4.4 through 4.8 describe these exchanges. Exchanges define the content and ordering of ISAKMP messages during communications between peers. Most exchanges will include all the basic payload types - SA, KE, ID, SIG - and may include others. The primary difference between exchange types is the ordering of the messages and the payload ordering within each message. While the ordering of payloads within messages is not mandated, for processing efficiency it is RECOMMENDED that the Security Association payload be the first payload within an exchange. Processing of each payload within an exchange is described in section 5.

Sections 4.4 through 4.8 provide a default set of ISAKMP exchanges. These exchanges provide different security protection for the exchange itself and information exchanged. The diagrams in each of the following sections show the message ordering for each exchange type as well as the payloads included in each message, and provide basic notes describing what has happened after each message exchange. None of the examples include any "optional payloads", like certificate and certificate request. Additionally, none of the examples include an initial exchange of ISAKMP Headers (containing initiator and responder cookies) which would provide protection against clogging (see section 2.5.3).

The defined exchanges are not meant to satisfy all DOI and key exchange protocol requirements. If the defined exchanges meet the DOI requirements, then they can be used as outlined. If the defined exchanges do not meet the security requirements defined by the DOI, then the DOI MUST specify new exchange type(s) and the valid sequences of payloads that make up a successful exchange, and how to build and interpret those payloads. All ISAKMP implementations MUST implement the Informational Exchange and SHOULD implement the other four exchanges. However, this is dependent on the definition of the DOI and associated key exchange protocols.

As discussed above, these exchange types can be used in either phase of negotiation. However, they may provide different security properties in each of the phases. With each of these exchanges, the combination of cookies and SPI fields identifies whether this exchange is being used in the first or second phase of a negotiation.

4.1.1 Notation

The following notation is used to describe the ISAKMP exchange types, shown in the next section, with the message formats and associated payloads:

HDR is an ISAKMP header whose exchange type defines the payload orderings
SA is an SA negotiation payload with one or more Proposal and Transform payloads. An initiator MAY provide multiple proposals for negotiation; a responder MUST reply with only one.
KE is the key exchange payload.
IDx is the identity payload for "x". x can be: "ii" or "ir" for the ISAKMP initiator and responder, respectively, or x can be: "ui", "ur" (when the ISAKMP daemon is a proxy negotiator), for the user initiator and responder, respectively.
HASH is the hash payload.
SIG is the signature payload. The data to sign is exchange-specific.
AUTH is a generic authentication mechanism, such as HASH or SIG.
NONCE is the nonce payload.
'*' signifies payload encryption after the ISAKMP header. This encryption MUST begin immediately after the ISAKMP header and all payloads following the ISAKMP header MUST be encrypted.

=> signifies "initiator to responder" communication
<= signifies "responder to initiator" communication

4.2 Security Association Establishment

The Security Association, Proposal, and Transform payloads are used to build ISAKMP messages for the negotiation and establishment of SAs. An SA establishment message consists of a single SA payload followed by at least one, and possibly many, Proposal payloads and at least one, and possibly many, Transform payloads associated with each Proposal payload. Because these payloads are considered together, the SA payload will point to any following payloads and not to the Proposal payload included with the SA payload. The SA Payload contains the DOI and Situation for the proposed SA. Each Proposal payload contains a Security Parameter Index (SPI) and ensures that the SPI is associated with the Protocol-Id in accordance with the Internet Security Architecture [SEC-ARCH]. Proposal payloads may or may not have the same SPI, as this is implementation dependent. Each

Transform Payload contains the specific security mechanisms to be used for the designated protocol. It is expected that the Proposal and Transform payloads will be used only during SA establishment negotiation. The creation of payloads for security association negotiation and establishment described here in this section are applicable for all ISAKMP exchanges described later in sections 4.4 through 4.8. The examples shown in 4.2.1 contain only the SA, Proposal, and Transform payloads and do not contain other payloads that might exist for a given ISAKMP exchange.

The Proposal payload provides the initiating entity with the capability to present to the responding entity the security protocols and associated security mechanisms for use with the security association being negotiated. If the SA establishment negotiation is for a combined protection suite consisting of multiple protocols, then there MUST be multiple Proposal payloads each with the same Proposal number. These proposals MUST be considered as a unit and MUST NOT be separated by a proposal with a different proposal number. The use of the same Proposal number in multiple Proposal payloads provides a logical AND operation, i.e. Protocol 1 AND Protocol 2. The first example below shows an ESP AND AH protection suite. If the SA establishment negotiation is for different protection suites, then there MUST be multiple Proposal payloads each with a monotonically increasing Proposal number. The different proposals MUST be presented in the initiator's preference order. The use of different Proposal numbers in multiple Proposal payloads provides a logical OR operation, i.e. Proposal 1 OR Proposal 2, where each proposal may have more than one protocol. The second example below shows either an AH AND ESP protection suite OR just an ESP protection suite. Note that the Next Payload field of the Proposal payload points to another Proposal payload (if it exists). The existence of a Proposal payload implies the existence of one or more Transform payloads.

The Transform payload provides the initiating entity with the capability to present to the responding entity multiple mechanisms, or transforms, for a given protocol. The Proposal payload identifies a Protocol for which services and mechanisms are being negotiated. The Transform payload allows the initiating entity to present several possible supported transforms for that proposed protocol. There may be several transforms associated with a specific Proposal payload each identified in a separate Transform payload. The multiple transforms MUST be presented with monotonically increasing numbers in the initiator's preference order. The receiving entity MUST select a single transform for each protocol in a proposal or reject the entire proposal. The use of the Transform number in multiple Transform payloads provides a second level OR operation, i.e. Transform 1 OR Transform 2 OR Transform 3. Example 1 below shows two possible transforms for ESP and a single transform for AH. Example 2 below shows one transform for AH AND one transform for ESP OR two transforms for ESP alone. Note that the Next Payload field of the Transform payload points to another Transform payload or 0. The Proposal payload delineates the different proposals.

When responding to a Security Association payload, the responder MUST send a Security Association payload with the selected proposal, which may consist of multiple Proposal payloads and their associated Transform payloads. Each of the Proposal payloads MUST contain a single Transform payload associated with the Protocol. The responder SHOULD retain the Proposal # field in the Proposal payload and the Transform # field in each Transform payload of the selected Proposal. Retention of Proposal and Transform numbers should speed the initiator's protocol processing by negating the need to compare the respondor's selection with every offered option. These values enable the initiator to perform the comparison directly and quickly. The initiator MUST verify that the Security Association payload received from the responder matches one of the proposals sent initially.

4.2.1 Security Association Establishment Examples

This example shows a Proposal for a combined protection suite with two different protocols. The first protocol is presented with two transforms supported by the proposer. The second protocol is presented with a single transform. An example for this proposal might be: Protocol 1 is ESP with Transform 1 as 3DES and Transform 2 as DES AND Protocol 2 is AH with Transform 1 as SHA. The responder MUST select from the two transforms proposed for ESP. The resulting protection suite will be either (1) 3DES AND SHA OR (2) DES AND SHA, depending on which ESP transform was selected by the responder. Note this example is shown using the Base Exchange.

```
                      1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 /+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/ ! NP = Nonce    !   RESERVED    !         Payload Length        !
/  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
SA Pay !                 Domain of Interpretation (DOI)                 !
\  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 \ !                          Situation                            !
  >+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 / ! NP = Proposal !   RESERVED    !         Payload Length        !
/  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
Prop 1 ! Proposal # = 1!  Protocol-Id  !    SPI Size   !# of Trans. = 2!
Prot 1 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 \ !                        SPI (variable)                         !
  >+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 / ! NP = Transform!   RESERVED    !         Payload Length        !
```

```
      /  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
Tran 1 ! Transform # 1 ! Transform ID  !           RESERVED2           !
      \  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       \ !                        SA Attributes                        !
        >+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       / ! NP = 0        !   RESERVED    !         Payload Length        !
      /  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
Tran 2 ! Transform # 2 ! Transform ID  !           RESERVED2           !
      \  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       \ !                        SA Attributes                        !
        >+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       / ! NP = 0        !   RESERVED    !         Payload Length        !
      /  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
Prop 1 ! Proposal # = 1!  Protocol ID  !    SPI Size   !# of Trans. = 1!
Prot 2 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       \ !                        SPI (variable)                       !
        >+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       / ! NP = 0        !   RESERVED    !         Payload Length        !
      /  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
Tran 1 ! Transform # 1 ! Transform ID  !           RESERVED2           !
      \  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       \ !                        SA Attributes                        !
        \+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

This second example shows a Proposal for two different protection suites. The SA Payload was omitted for space reasons. The first protection suite is presented with one transform for the first protocol and one transform for the second protocol. The second protection suite is presented with two transforms for a single protocol. An example for this proposal might be: Proposal 1 with Protocol 1 as AH with Transform 1 as MD5 AND Protocol 2 as ESP with Transform 1 as 3DES. This is followed by Proposal 2 with Protocol 1 as ESP with Transform 1 as DES and Transform 2 as 3DES. The responder MUST select from the two different proposals. If the second Proposal is selected, the responder MUST select from the two transforms for ESP. The resulting protection suite will be either (1) MD5 AND 3DES OR the selection between (2) DES OR (3) 3DES.

```
                            1                   2                   3
          0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
        /+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       / ! NP = Proposal !   RESERVED    !         Payload Length        !
      /  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
Prop 1 ! Proposal # = 1!  Protocol ID  !    SPI Size   !# of Trans. = 1!
Prot 1 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       \ !                        SPI (variable)                       !
        >+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       / ! NP = 0        !   RESERVED    !         Payload Length        !
```

```
    /  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
Tran 1 ! Transform # 1 ! Transform ID  !           RESERVED2           !
    \  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     \ !                        SA Attributes                          !
      >+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     / ! NP = Proposal !   RESERVED    !         Payload Length        !
    /  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
Prop 1 ! Proposal # = 1! Protocol ID   !    SPI Size   !# of Trans. = 1!
Prot 2 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     \ !                        SPI (variable)                         !
      >+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     / ! NP = 0        !   RESERVED    !         Payload Length        !
    /  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
Tran 1 ! Transform # 1 ! Transform ID  !           RESERVED2           !
    \  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     \ !                        SA Attributes                          !
      >+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     / ! NP = 0        !   RESERVED    !         Payload Length        !
    /  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
Prop 2 ! Proposal # = 2! Protocol ID   !    SPI Size   !# of Trans. = 2!
Prot 1 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     \ !                        SPI (variable)                         !
      >+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     / ! NP = Transform!   RESERVED    !         Payload Length        !
    /  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
Tran 1 ! Transform # 1 ! Transform ID  !           RESERVED2           !
    \  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     \ !                        SA Attributes                          !
      >+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     / ! NP = 0        !   RESERVED    !         Payload Length        !
    /  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
Tran 2 ! Transform # 2 ! Transform ID  !           RESERVED2           !
    \  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     \ !                        SA Attributes                          !
      \+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

4.3 Security Association Modification

Security Association modification within ISAKMP is accomplished by creating a new SA and initiating communications using that new SA. Deletion of the old SA can be done anytime after the new SA is established. Deletion of the old SA is dependent on local security policy. Modification of SAs by using a "Create New SA followed by Delete Old SA" method is done to avoid potential vulnerabilities in synchronizing modification of existing SA attributes. The procedure for creating new SAs is outlined in section 4.2. The procedure for deleting SAs is outlined in section 5.15.

Modification of an ISAKMP SA (phase 1 negotiation) follows the same procedure as creation of an ISAKMP SA. There is no relationship between the two SAs and the initiator and responder cookie pairs SHOULD be different, as outlined in section 2.5.3.

Modification of a Protocol SA (phase 2 negotiation) follows the same procedure as creation of a Protocol SA. The creation of a new SA is protected by the existing ISAKMP SA. There is no relationship between the two Protocol SAs. A protocol implementation SHOULD begin using the newly created SA for outbound traffic and SHOULD continue to support incoming traffic on the old SA until it is deleted or until traffic is received under the protection of the newly created SA. As stated previously in this section, deletion of an old SA is then dependent on local security policy.

4.4 Base Exchange

The Base Exchange is designed to allow the Key Exchange and Authentication related information to be transmitted together. Combining the Key Exchange and Authentication-related information into one message reduces the number of round-trips at the expense of not providing identity protection. Identity protection is not provided because identities are exchanged before a common shared secret has been established and, therefore, encryption of the identities is not possible. The following diagram shows the messages with the possible payloads sent in each message and notes for an example of the Base Exchange.

```
                         BASE EXCHANGE

 #  Initiator Direction  Responder           NOTE
(1)  HDR; SA; NONCE  =>           Begin ISAKMP-SA or Proxy negotiation

(2)                  <=  HDR; SA; NONCE
                                  Basic SA agreed upon
(3)  HDR; KE;        =>
     IDii; AUTH                   Key Generated (by responder)
                                  Initiator Identity Verified by
                                  Responder
(4)                  <=  HDR; KE;
                         IDir; AUTH
                                  Responder Identity Verified by
                                  Initiator Key Generated (by
                                  initiator) SA established
```

In the first message (1), the initiator generates a proposal it considers adequate to protect traffic for the given situation. The Security Association, Proposal, and Transform payloads are included in the Security Association payload (for notation purposes). Random information which is used to guarantee liveness and protect against replay attacks is also transmitted. Random information provided by both parties SHOULD be used by the authentication mechanism to provide shared proof of participation in the exchange.

In the second message (2), the responder indicates the protection suite it has accepted with the Security Association, Proposal, and Transform payloads. Again, random information which is used to guarantee liveness and protect against replay attacks is also transmitted. Random information provided by both parties SHOULD be used by the authentication mechanism to provide shared proof of participation in the exchange. Local security policy dictates the action of the responder if no proposed protection suite is accepted. One possible action is the transmission of a Notify payload as part of an Informational Exchange.

In the third (3) and fourth (4) messages, the initiator and responder, respectively, exchange keying material used to arrive at a common shared secret and identification information. This information is transmitted under the protection of the agreed upon authentication function. Local security policy dictates the action if an error occurs during these messages. One possible action is the transmission of a Notify payload as part of an Informational Exchange.

4.5 Identity Protection Exchange

The Identity Protection Exchange is designed to separate the Key Exchange information from the Identity and Authentication related information. Separating the Key Exchange from the Identity and Authentication related information provides protection of the communicating identities at the expense of two additional messages. Identities are exchanged under the protection of a previously established common shared secret. The following diagram shows the messages with the possible payloads sent in each message and notes for an example of the Identity Protection Exchange.

IDENTITY PROTECTION EXCHANGE

#	Initiator	Direction	Responder	NOTE
(1)	HDR; SA	=>		Begin ISAKMP-SA or Proxy negotiation
(2)		<=	HDR; SA	
				Basic SA agreed upon
(3)	HDR; KE; NONCE	=>		
(4)		<=	HDR; KE; NONCE	
				Key Generated (by Initiator and Responder)
(5)	HDR*; IDii; AUTH	=>		
				Initiator Identity Verified by Responder
(6)		<=	HDR*; IDir; AUTH	
				Responder Identity Verified by Initiator SA established

In the first message (1), the initiator generates a proposal it considers adequate to protect traffic for the given situation. The Security Association, Proposal, and Transform payloads are included in the Security Association payload (for notation purposes).

In the second message (2), the responder indicates the protection suite it has accepted with the Security Association, Proposal, and Transform payloads. Local security policy dictates the action of the responder if no proposed protection suite is accepted. One possible action is the transmission of a Notify payload as part of an Informational Exchange.

In the third (3) and fourth (4) messages, the initiator and responder, respectively, exchange keying material used to arrive at a common shared secret and random information which is used to guarantee liveness and protect against replay attacks. Random information provided by both parties SHOULD be used by the authentication mechanism to provide shared proof of participation in the exchange. Local security policy dictates the action if an error occurs during these messages. One possible action is the transmission of a Notify payload as part of an Informational Exchange.

In the fifth (5) and sixth (6) messages, the initiator and responder, respectively, exchange identification information and the results of the agreed upon authentication function. This information is transmitted under the protection of the common shared secret. Local security policy dictates the action if an error occurs during these messages. One possible action is the transmission of a Notify payload as part of an Informational Exchange.

4.6 Authentication Only Exchange

The Authentication Only Exchange is designed to allow only Authentication related information to be transmitted. The benefit of this exchange is the ability to perform only authentication without the computational expense of computing keys. Using this exchange during negotiation, none of the transmitted information will be encrypted. However, the information may be encrypted in other places. For example, if encryption is negotiated during the first phase of a negotiation and the authentication only exchange is used in the second phase of a negotiation, then the authentication only exchange will be encrypted by the ISAKMP SAs negotiated in the first phase. The following diagram shows the messages with possible payloads sent in each message and notes for an example of the Authentication Only Exchange.

```
                        AUTHENTICATION ONLY EXCHANGE

 #      Initiator     Direction     Responder     NOTE
(1)  HDR; SA; NONCE      =>                       Begin ISAKMP-SA or
                                                  Proxy negotiation
(2)                       <=     HDR; SA; NONCE;
                                 IDir; AUTH
                                                  Basic SA agreed upon
                                                  Responder Identity
                                                  Verified by Initiator
(3)  HDR; IDii; AUTH      =>
                                                  Initiator Identity
                                                  Verified by Responder
                                                  SA established
```

In the first message (1), the initiator generates a proposal it considers adequate to protect traffic for the given situation. The Security Association, Proposal, and Transform payloads are included in the Security Association payload (for notation purposes). Random information which is used to guarantee liveness and protect against replay attacks is also transmitted. Random information provided by both parties SHOULD be used by the authentication mechanism to provide shared proof of participation in the exchange.

In the second message (2), the responder indicates the protection suite it has accepted with the Security Association, Proposal, and Transform payloads. Again, random information which is used to guarantee liveness and protect against replay attacks is also transmitted. Random information provided by both parties SHOULD be used by the authentication mechanism to provide shared proof of participation in the exchange. Additionally, the responder transmits identification information. All of this information is transmitted under the protection of the agreed upon authentication function. Local security policy dictates the action of the responder if no proposed protection suite is accepted. One possible action is the transmission of a Notify payload as part of an Informational Exchange.

In the third message (3), the initiator transmits identification information. This information is transmitted under the protection of the agreed upon authentication function. Local security policy dictates the action if an error occurs during these messages. One possible action is the transmission of a Notify payload as part of an Informational Exchange.

4.7 Aggressive Exchange

The Aggressive Exchange is designed to allow the Security Association, Key Exchange and Authentication related payloads to be transmitted together. Combining the Security Association, Key Exchange, and Authentication-related information into one message reduces the number of round-trips at the expense of not providing identity protection. Identity protection is not provided because identities are exchanged before a common shared secret has been established and, therefore, encryption of the identities is not possible. Additionally, the Aggressive Exchange is attempting to establish all security relevant information in a single exchange. The following diagram shows the messages with possible payloads sent in each message and notes for an example of the Aggressive Exchange.

AGGRESSIVE EXCHANGE

```
 #     Initiator   Direction      Responder      NOTE
(1)  HDR; SA; KE;      =>                        Begin ISAKMP-SA or
                                                 Proxy negotiation
     NONCE; IDii                                 and Key Exchange

(2)                    <=     HDR; SA; KE;
                              NONCE; IDir; AUTH
                                                 Initiator Identity
                                                 Verified by Responder
                                                 Key Generated
                                                 Basic SA agreed upon
(3)  HDR*; AUTH        =>
                                                 Responder Identity
                                                 Verified by Initiator
                                                 SA established
```

In the first message (1), the initiator generates a proposal it considers adequate to protect traffic for the given situation. The Security Association, Proposal, and Transform payloads are included in the Security Association payload (for notation purposes). There can be only one Proposal and one Transform offered (i.e. no choices) in order for the aggressive exchange to work. Keying material used to arrive at a common shared secret and random information which is used to guarantee liveness and protect against replay attacks are also transmitted. Random information provided by both parties SHOULD be used by the authentication mechanism to provide shared proof of participation in the exchange. Additionally, the initiator transmits identification information.

In the second message (2), the responder indicates the protection suite it has accepted with the Security Association, Proposal, and Transform payloads. Keying material used to arrive at a common shared secret and random information which is used to guarantee liveness and protect against replay attacks is also transmitted. Random information provided by both parties SHOULD be used by the authentication mechanism to provide shared proof of participation in the exchange. Additionally, the responder transmits identification information. All of this information is transmitted under the protection of the agreed upon authentication function. Local security policy dictates the action of the responder if no proposed protection suite is accepted. One possible action is the transmission of a Notify payload as part of an Informational Exchange.

In the third (3) message, the initiator transmits the results of the agreed upon authentication function. This information is transmitted under the protection of the common shared secret. Local security policy dictates the action if an error occurs during these messages. One possible action is the transmission of a Notify payload as part of an Informational Exchange.

4.8 Informational Exchange

The Informational Exchange is designed as a one-way transmittal of information that can be used for security association management. The following diagram shows the messages with possible payloads sent in each message and notes for an example of the Informational Exchange.

```
                     INFORMATIONAL EXCHANGE

 #   Initiator  Direction Responder  NOTE
(1)  HDR*; N/D     =>                Error Notification or Deletion
```

In the first message (1), the initiator or responder transmits an ISAKMP Notify or Delete payload.

If the Informational Exchange occurs prior to the exchange of keying meterial during an ISAKMP Phase 1 negotiation, there will be no protection provided for the Informational Exchange. Once keying material has been exchanged or an ISAKMP SA has been established, the Informational Exchange MUST be transmitted under the protection provided by the keying material or the ISAKMP SA.

All exchanges are similar in that with the beginning of any exchange, cryptographic synchronization MUST occur. The Informational Exchange is an exchange and not an ISAKMP message. Thus, the generation of an Message ID (MID) for an Informational Exchange SHOULD be independent of IVs of other on-going communication. This will ensure cryptographic synchronization is maintained for existing communications and the Informational Exchange will be processed correctly. The only exception to this is when the Commit Bit of the ISAKMP Header is set. When the Commit Bit is set, the Message ID field of the Informational Exchange MUST contain the Message ID of the original ISAKMP Phase 2 SA negotiation, rather than a new Message ID (MID). This is done to ensure that the Informational Exchange with the CONNECTED Notify Message can be associated with the correct Phase 2 SA. For a description of the Commit Bit, see section 3.1.

5 ISAKMP Payload Processing

Section 3 describes the ISAKMP payloads. These payloads are used in the exchanges described in section 4 and can be used in exchanges defined for a specific DOI. This section describes the processing for each of the payloads. This section suggests the logging of events to a system audit file. This action is controlled by a system security policy and is, therefore, only a suggested action.

5.1 General Message Processing

Every ISAKMP message has basic processing applied to insure protocol reliability, and to minimize threats, such as denial of service and replay attacks. All processing SHOULD include packet length checks to insure the packet received is at least as long as the length given in the ISAKMP Header. If the ISAKMP message length and the value in the Payload Length field of the ISAKMP Header are not the same, then the ISAKMP message MUST be rejected. The receiving entity (initiator or responder) MUST do the following:

1. The event, UNEQUAL PAYLOAD LENGTHS, MAY be logged in the appropriate system audit file.

2. An Informational Exchange with a Notification payload containing the UNEQUAL-PAYLOAD-LENGTHS message type MAY be sent to the transmitting entity. This action is dictated by a system security policy.

When transmitting an ISAKMP message, the transmitting entity (initiator or responder) MUST do the following:

1. Set a timer and initialize a retry counter.

   NOTE: Implementations MUST NOT use a fixed timer. Instead, transmission timer values should be adjusted dynamically based on measured round trip times. In addition, successive retransmissions of the same packet should be separated by increasingly longer time intervals (e.g., exponential backoff).

2. If the timer expires, the ISAKMP message is resent and the retry counter is decremented.

3. If the retry counter reaches zero (0), the event, RETRY LIMIT REACHED, MAY be logged in the appropriate system audit file.

4. The ISAKMP protocol machine clears all states and returns to IDLE.

5.2 ISAKMP Header Processing

When creating an ISAKMP message, the transmitting entity (initiator or responder) MUST do the following:

1. Create the respective cookie. See section 2.5.3 for details.

2. Determine the relevant security characteristics of the session (i.e. DOI and situation).

3. Construct an ISAKMP Header with fields as described in section 3.1.

4. Construct other ISAKMP payloads, depending on the exchange type.

5. Transmit the message to the destination host as described in section5.1.

When an ISAKMP message is received, the receiving entity (initiator or responder) MUST do the following:

1. Verify the Initiator and Responder "cookies". If the cookie validation fails, the message is discarded and the following actions are taken:

   (a) The event, INVALID COOKIE, MAY be logged in the appropriate system audit file.

   (b) An Informational Exchange with a Notification payload containing the INVALID-COOKIE message type MAY be sent to the transmitting entity. This action is dictated by a system security policy.

2. Check the Next Payload field to confirm it is valid. If the Next Payload field validation fails, the message is discarded and the following actions are taken:

   (a) The event, INVALID NEXT PAYLOAD, MAY be logged in the appropriate system audit file.

   (b) An Informational Exchange with a Notification payload containing the INVALID-PAYLOAD-TYPE message type MAY be sent to the transmitting entity. This action is dictated by a system security policy.

3. Check the Major and Minor Version fields to confirm they are correct (see section 3.1). If the Version field validation fails, the message is discarded and the following actions are

taken:

(a) The event, INVALID ISAKMP VERSION, MAY be logged in the appropriate system audit file.

(b) An Informational Exchange with a Notification payload containing the INVALID-MAJOR-VERSION or INVALID-MINOR-VERSION message type MAY be sent to the transmitting entity. This action is dictated by a system security policy.

4. Check the Exchange Type field to confirm it is valid. If the Exchange Type field validation fails, the message is discarded and the following actions are taken:

   (a) The event, INVALID EXCHANGE TYPE, MAY be logged in the appropriate system audit file.

   (b) An Informational Exchange with a Notification payload containing the INVALID-EXCHANGE-TYPE message type MAY be sent to the transmitting entity. This action is dictated by a system security policy.

5. Check the Flags field to ensure it contains correct values. If the Flags field validation fails, the message is discarded and the following actions are taken:

   (a) The event, INVALID FLAGS, MAY be logged in the appropriate systemaudit file.

   (b) An Informational Exchange with a Notification payload containing the INVALID-FLAGS message type MAY be sent to the transmitting entity. This action is dictated by a system security policy.

6. Check the Message ID field to ensure it contains correct values. If the Message ID validation fails, the message is discarded and the following actions are taken:

   (a) The event, INVALID MESSAGE ID, MAY be logged in the appropriate system audit file.

   (b) An Informational Exchange with a Notification payload containing the INVALID-MESSAGE-ID message type MAY be sent to the transmitting entity. This action is dictated by a system security policy.

7. Processing of the ISAKMP message continues using the value in the Next Payload field.

5.3 Generic Payload Header Processing

When creating any of the ISAKMP Payloads described in sections 3.4 through 3.15 a Generic Payload Header is placed at the beginning of these payloads. When creating the Generic Payload Header, the transmitting entity (initiator or responder) MUST do the following:

1. Place the value of the Next Payload in the Next Payload field. These values are described in section 3.1.

2. Place the value zero (0) in the RESERVED field.

3. Place the length (in octets) of the payload in the Payload Length field.

4. Construct the payloads as defined in the remainder of this section.

When any of the ISAKMP Payloads are received, the receiving entity (initiator or responder) MUST do the following:

1. Check the Next Payload field to confirm it is valid. If the Next Payload field validation fails, the message is discarded and the following actions are taken:

   (a) The event, INVALID NEXT PAYLOAD, MAY be logged in the appropriate system audit file.

   (b) An Informational Exchange with a Notification payload containing the INVALID-PAYLOAD-TYPE message type MAY be sent to the transmitting entity. This action is dictated by a system security policy.

2. Verify the RESERVED field contains the value zero. If the value in the RESERVED field is not zero, the message is discarded and the following actions are taken:

   (a) The event, INVALID RESERVED FIELD, MAY be logged in the appropriate system audit file.

   (b) An Informational Exchange with a Notification payload containing the BAD-PROPOSAL-SYNTAX or PAYLOAD-MALFORMED message type MAY be sent to the transmitting entity. This action is dictated by a system security policy.

3. Process the remaining payloads as defined by the Next Payload field.

5.4 Security Association Payload Processing

When creating a Security Association Payload, the transmitting entity (initiator or responder) MUST do the following:

1. Determine the Domain of Interpretation for which this negotiation is being performed.

2. Determine the situation within the determined DOI for which this negotiation is being performed.

3. Determine the proposal(s) and transform(s) within the situation. These are described, respectively, in sections 3.5 and 3.6.

4. Construct a Security Association payload.

5. Transmit the message to the receiving entity as described in section 5.1.

When a Security Association payload is received, the receiving entity (initiator or responder) MUST do the following:

1. Determine if the Domain of Interpretation (DOI) is supported. If the DOI determination fails, the message is discarded and the following actions are taken:

   (a) The event, INVALID DOI, MAY be logged in the appropriate system audit file.

   (b) An Informational Exchange with a Notification payload containing the DOI-NOT-SUPPORTED message type MAY be sent to the transmitting entity. This action is dictated by a system security policy.

2. Determine if the given situation can be protected. If the Situation determination fails, the message is discarded and the following actions are taken:

   (a) The event, INVALID SITUATION, MAY be logged in the appropriate system audit file.

   (b) An Informational Exchange with a Notification payload containing the SITUATION-NOT-SUPPORTED message type MAY be sent to the transmitting entity. This action is dictated by a system security policy.

3. Process the remaining payloads (i.e. Proposal, Transform) of the Security Association Payload. If the Security Association

Proposal (as described in sections 5.5 and 5.6) is not accepted, then the following actions are taken:

(a) The event, INVALID PROPOSAL, MAY be logged in the appropriate system audit file.

(b) An Informational Exchange with a Notification payload containing the NO-PROPOSAL-CHOSEN message type MAY be sent to the transmitting entity. This action is dictated by a system security policy.

5.5 Proposal Payload Processing

When creating a Proposal Payload, the transmitting entity (initiator or responder) MUST do the following:

1. Determine the Protocol for this proposal.

2. Determine the number of proposals to be offered for this protocol and the number of transforms for each proposal. Transforms are described in section 3.6.

3. Generate a unique pseudo-random SPI.

4. Construct a Proposal payload.

When a Proposal payload is received, the receiving entity (initiator or responder) MUST do the following:

1. Determine if the Protocol is supported. If the Protocol-ID field is invalid, the payload is discarded and the following actions are taken:

   (a) The event, INVALID PROTOCOL, MAY be logged in the appropriate system audit file.

   (b) An Informational Exchange with a Notification payload containing the INVALID-PROTOCOL-ID message type MAY be sent to the transmitting entity. This action is dictated by a system security policy.

2. Determine if the SPI is valid. If the SPI is invalid, the payload is discarded and the following actions are taken:

   (a) The event, INVALID SPI, MAY be logged in the appropriate system audit file.

(b) An Informational Exchange with a Notification payload containing the INVALID-SPI message type MAY be sent to the transmitting entity. This action is dictated by a system security policy.

3. Ensure the Proposals are presented according to the details given in section 3.5 and 4.2. If the proposals are not formed correctly, the following actions are taken:

   (a) Possible events, BAD PROPOSAL SYNTAX, INVALID PROPOSAL, are logged in the appropriate system audit file.

   (b) An Informational Exchange with a Notification payload containing the BAD-PROPOSAL-SYNTAX or PAYLOAD-MALFORMED message type MAY be sent to the transmitting entity. This action is dictated by a system security policy.

4. Process the Proposal and Transform payloads as defined by the Next Payload field. Examples of processing these payloads are given in section 4.2.1.

5.6 Transform Payload Processing

When creating a Transform Payload, the transmitting entity (initiator or responder) MUST do the following:

1. Determine the Transform # for this transform.

2. Determine the number of transforms to be offered for this proposal. Transforms are described in sections 3.6.

3. Construct a Transform payload.

When a Transform payload is received, the receiving entity (initiator or responder) MUST do the following:

1. Determine if the Transform is supported. If the Transform-ID field contains an unknown or unsupported value, then that Transform payload MUST be ignored and MUST NOT cause the generation of an INVALID TRANSFORM event. If the Transform-ID field is invalid, the payload is discarded and the following actions are taken:

   (a) The event, INVALID TRANSFORM, MAY be logged in the appropriate system audit file.

   (b) An Informational Exchange with a Notification payload containing the INVALID-TRANSFORM-ID message type MAY be sent to the transmitting entity. This action is dictated by a system security policy.

2. Ensure the Transforms are presented according to the details given in section 3.6 and 4.2. If the transforms are not formed correctly, the following actions are taken:

   (a) Possible events, BAD PROPOSAL SYNTAX, INVALID TRANSFORM, INVALID ATTRIBUTES, are logged in the appropriate system audit file.

   (b) An Informational Exchange with a Notification payload containing the BAD-PROPOSAL-SYNTAX, PAYLOAD-MALFORMED or ATTRIBUTES-NOT-SUPPORTED message type MAY be sent to the transmitting entity. This action is dictated by a system security policy.

3. Process the subsequent Transform and Proposal payloads as defined by the Next Payload field. Examples of processing these payloads are given in section 4.2.1.

5.7 Key Exchange Payload Processing

When creating a Key Exchange Payload, the transmitting entity (initiator or responder) MUST do the following:

1. Determine the Key Exchange to be used as defined by the DOI.

2. Determine the usage of the Key Exchange Data field as defined by the DOI.

3. Construct a Key Exchange payload.

4. Transmit the message to the receiving entity as described in section 5.1.

When a Key Exchange payload is received, the receiving entity (initiator or responder) MUST do the following:

1. Determine if the Key Exchange is supported. If the Key Exchange determination fails, the message is discarded and the following actions are taken:

   (a) The event, INVALID KEY INFORMATION, MAY be logged in the appropriate system audit file.

   (b) An Informational Exchange with a Notification payload containing the INVALID-KEY-INFORMATION message type MAY be sent to the transmitting entity. This action is dictated by a system security policy.

5.8 Identification Payload Processing

When creating an Identification Payload, the transmitting entity (initiator or responder) MUST do the following:

1. Determine the Identification information to be used as defined by the DOI (and possibly the situation).

2. Determine the usage of the Identification Data field as defined by the DOI.

3. Construct an Identification payload.

4. Transmit the message to the receiving entity as described in section 5.1.

When an Identification payload is received, the receiving entity (initiator or responder) MUST do the following:

1. Determine if the Identification Type is supported. This may be based on the DOI and Situation. If the Identification determination fails, the message is discarded and the following actions are taken:

   (a) The event, INVALID ID INFORMATION, MAY be logged in the appropriate system audit file.

   (b) An Informational Exchange with a Notification payload containing the INVALID-ID-INFORMATION message type MAY be sent to the transmitting entity. This action is dictated by a system security policy.

5.9 Certificate Payload Processing

When creating a Certificate Payload, the transmitting entity (initiator or responder) MUST do the following:

1. Determine the Certificate Encoding to be used. This may be specified by the DOI.

2. Ensure the existence of a certificate formatted as defined by the Certificate Encoding.

3. Construct a Certificate payload.

4. Transmit the message to the receiving entity as described in section 5.1.

When a Certificate payload is received, the receiving entity (initiator or responder) MUST do the following:

1. Determine if the Certificate Encoding is supported. If the Certificate Encoding is not supported, the payload is discarded and the following actions are taken:

   (a) The event, INVALID CERTIFICATE TYPE, MAY be logged in the appropriate system audit file.

   (b) An Informational Exchange with a Notification payload containing the INVALID-CERT-ENCODING message type MAY be sent to the transmitting entity. This action is dictated by a system security policy.

2. Process the Certificate Data field. If the Certificate Data is invalid or improperly formatted, the payload is discarded and the following actions are taken:

   (a) The event, INVALID CERTIFICATE, MAY be logged in the appropriate system audit file.

   (b) An Informational Exchange with a Notification payload containing the INVALID-CERTIFICATE message type MAY be sent to the transmitting entity. This action is dictated by a system security policy.

5.10 Certificate Request Payload Processing

When creating a Certificate Request Payload, the transmitting entity (initiator or responder) MUST do the following:

1. Determine the type of Certificate Encoding to be requested. This may be specified by the DOI.

2. Determine the name of an acceptable Certificate Authority which is to be requested (if applicable).

3. Construct a Certificate Request payload.

4. Transmit the message to the receiving entity as described in section 5.1.

When a Certificate Request payload is received, the receiving entity (initiator or responder) MUST do the following:

1. Determine if the Certificate Encoding is supported. If the Certificate Encoding is invalid, the payload is discarded and the following actions are taken:

   (a) The event, INVALID CERTIFICATE TYPE, MAY be logged in the appropriate system audit file.

   (b) An Informational Exchange with a Notification payload containing the INVALID-CERT-ENCODING message type MAY be sent to the transmitting entity. This action is dictated by a system security policy.

   If the Certificate Encoding is not supported, the payload is discarded and the following actions are taken:

   (a) The event, CERTIFICATE TYPE UNSUPPORTED, MAY be logged in the appropriate system audit file.

   (b) An Informational Exchange with a Notification payload containing the CERT-TYPE-UNSUPPORTED message type MAY be sent to the transmitting entity. This action is dictated by a system security policy.

2. Determine if the Certificate Authority is supported for the specified Certificate Encoding. If the Certificate Authority is invalid or improperly formatted, the payload is discarded and the following actions are taken:

   (a) The event, INVALID CERTIFICATE AUTHORITY, MAY be logged in the appropriate system audit file.

   (b) An Informational Exchange with a Notification payload containing the INVALID-CERT-AUTHORITY message type MAY be sent to the transmitting entity. This action is dictated by a system security policy.

3. Process the Certificate Request. If a requested Certificate Type with the specified Certificate Authority is not available, then the payload is discarded and the following actions are taken:

   (a) The event, CERTIFICATE-UNAVAILABLE, MAY be logged in the appropriate system audit file.

   (b) An Informational Exchange with a Notification payload containing the CERTIFICATE-UNAVAILABLE message type MAY be sent to the transmitting entity. This action is dictated by a system security policy.

5.11 Hash Payload Processing

When creating a Hash Payload, the transmitting entity (initiator or responder) MUST do the following:

1. Determine the Hash function to be used as defined by the SA negotiation.

2. Determine the usage of the Hash Data field as defined by the DOI.

3. Construct a Hash payload.

4. Transmit the message to the receiving entity as described in section 5.1.

When a Hash payload is received, the receiving entity (initiator or responder) MUST do the following:

1. Determine if the Hash is supported. If the Hash determination fails, the message is discarded and the following actions are taken:

   (a) The event, INVALID HASH INFORMATION, MAY be logged in the appropriate system audit file.

   (b) An Informational Exchange with a Notification payload containing the INVALID-HASH-INFORMATION message type MAY be sent to the transmitting entity. This action is dictated by a system security policy.

2. Perform the Hash function as outlined in the DOI and/or Key Exchange protocol documents. If the Hash function fails, the message is discarded and the following actions are taken:

   (a) The event, INVALID HASH VALUE, MAY be logged in the appropriate system audit file.

   (b) An Informational Exchange with a Notification payload containing the AUTHENTICATION-FAILED message type MAY be sent to the transmitting entity. This action is dictated by a system security policy.

5.12 Signature Payload Processing

When creating a Signature Payload, the transmitting entity (initiator or responder) MUST do the following:

1. Determine the Signature function to be used as defined by the SA negotiation.

2. Determine the usage of the Signature Data field as defined by the DOI.

3. Construct a Signature payload.

4. Transmit the message to the receiving entity as described in section 5.1.

When a Signature payload is received, the receiving entity (initiator or responder) MUST do the following:

1. Determine if the Signature is supported. If the Signature determination fails, the message is discarded and the following actions are taken:

   (a) The event, INVALID SIGNATURE INFORMATION, MAY be logged in the appropriate system audit file.

   (b) An Informational Exchange with a Notification payload containing the INVALID-SIGNATURE message type MAY be sent to the transmitting entity. This action is dictated by a system security policy.

2. Perform the Signature function as outlined in the DOI and/or Key Exchange protocol documents. If the Signature function fails, the message is discarded and the following actions are taken:

   (a) The event, INVALID SIGNATURE VALUE, MAY be logged in the appropriate system audit file.

   (b) An Informational Exchange with a Notification payload containing the AUTHENTICATION-FAILED message type MAY be sent to the transmitting entity. This action is dictated by a system security policy.

5.13 Nonce Payload Processing

When creating a Nonce Payload, the transmitting entity (initiator or responder) MUST do the following:

1. Create a unique random value to be used as a nonce.

2. Construct a Nonce payload.

3. Transmit the message to the receiving entity as described in section 5.1.

When a Nonce payload is received, the receiving entity (initiator or responder) MUST do the following:

1. There are no specific procedures for handling Nonce payloads. The procedures are defined by the exchange types (and possibly the DOI and Key Exchange descriptions).

5.14 Notification Payload Processing

During communications it is possible that errors may occur. The Informational Exchange with a Notify Payload provides a controlled method of informing a peer entity that errors have occurred during protocol processing. It is RECOMMENDED that Notify Payloads be sent in a separate Informational Exchange rather than appending a Notify Payload to an existing exchange.

When creating a Notification Payload, the transmitting entity (initiator or responder) MUST do the following:

1. Determine the DOI for this Notification.

2. Determine the Protocol-ID for this Notification.

3. Determine the SPI size based on the Protocol-ID field. This field is necessary because different security protocols have different SPI sizes. For example, ISAKMP combines the Initiator and Responder cookie pair (16 octets) as a SPI, while ESP and AH have 4 octet SPIs.

4. Determine the Notify Message Type based on the error or status message desired.

5. Determine the SPI which is associated with this notification.

6. Determine if additional Notification Data is to be included. This is additional information specified by the DOI.

7. Construct a Notification payload.

8. Transmit the message to the receiving entity as described in section 5.1.

Because the Informational Exchange with a Notification payload is a unidirectional message a retransmission will not be performed. The local security policy will dictate the procedures for continuing.

However, we RECOMMEND that a NOTIFICATION PAYLOAD ERROR event be logged in the appropriate system audit file by the receiving entity.

If the Informational Exchange occurs prior to the exchange of keying material during an ISAKMP Phase 1 negotiation there will be no protection provided for the Informational Exchange. Once the keying material has been exchanged or the ISAKMP SA has been established, the Informational Exchange MUST be transmitted under the protection provided by the keying material or the ISAKMP SA.

When a Notification payload is received, the receiving entity (initiator or responder) MUST do the following:

1. Determine if the Informational Exchange has any protection applied to it by checking the Encryption Bit and the Authentication Only Bit in the ISAKMP Header. If the Encryption Bit is set, i.e. the Informational Exchange is encrypted, then the message MUST be decrypted using the (in-progress or completed) ISAKMP SA. Once the decryption is complete the processing can continue as described below. If the Authentication Only Bit is set, then the message MUST be authenticated using the (in-progress or completed) ISAKMP SA. Once the authentication is completed, the processing can continue as described below. If the Informational Exchange is not encrypted or authentication, the payload processing can continue as described below.

2. Determine if the Domain of Interpretation (DOI) is supported. If the DOI determination fails, the payload is discarded and the following action is taken:

   (a) The event, INVALID DOI, MAY be logged in the appropriate system audit file.

3. Determine if the Protocol-Id is supported. If the Protocol-Id determination fails, the payload is discarded and the following action is taken:

   (a) The event, INVALID PROTOCOL-ID, MAY be logged in the appropriate system audit file.

4. Determine if the SPI is valid. If the SPI is invalid, the payload is discarded and the following action is taken:

   (a) The event, INVALID SPI, MAY be logged in the appropriate system audit file.

5. Determine if the Notify Message Type is valid. If the Notify Message Type is invalid, the payload is discarded and the following action is taken:

   (a) The event, INVALID MESSAGE TYPE, MAY be logged in the appropriate system audit file.

6. Process the Notification payload, including additional Notification Data, and take appropriate action, according to local security policy.

5.15 Delete Payload Processing

During communications it is possible that hosts may be compromised or that information may be intercepted during transmission. Determining whether this has occurred is not an easy task and is outside the scope of this memo. However, if it is discovered that transmissions are being compromised, then it is necessary to establish a new SA and delete the current SA.

The Informational Exchange with a Delete Payload provides a controlled method of informing a peer entity that the transmitting entity has deleted the SA(s). Deletion of Security Associations MUST always be performed under the protection of an ISAKMP SA. The receiving entity SHOULD clean up its local SA database. However, upon receipt of a Delete message the SAs listed in the Security Parameter Index (SPI) field of the Delete payload cannot be used with the transmitting entity. The SA Establishment procedure must be invoked to re-establish secure communications.

When creating a Delete Payload, the transmitting entity (initiator or responder) MUST do the following:

1. Determine the DOI for this Deletion.

2. Determine the Protocol-ID for this Deletion.

3. Determine the SPI size based on the Protocol-ID field. This field is necessary because different security protocols have different SPI sizes. For example, ISAKMP combines the Initiator and Responder cookie pair (16 octets) as a SPI, while ESP and AH have 4 octet SPIs.

4. Determine the # of SPIs to be deleted for this protocol.

5. Determine the SPI(s) which is (are) associated with this deletion.

6. Construct a Delete payload.

7. Transmit the message to the receiving entity as described in section 5.1.

Because the Informational Exchange with a Delete payload is a unidirectional message a retransmission will not be performed. The local security policy will dictate the procedures for continuing. However, we RECOMMEND that a DELETE PAYLOAD ERROR event be logged in the appropriate system audit file by the receiving entity.

As described above, the Informational Exchange with a Delete payload MUST be transmitted under the protection provided by an ISAKMP SA.

When a Delete payload is received, the receiving entity (initiator or responder) MUST do the following:

1. Because the Informational Exchange is protected by some security service (e.g. authentication for an Auth-Only SA, encryption for other exchanges), the message MUST have these security services applied using the ISAKMP SA. Once the security service processing is complete the processing can continue as described below. Any errors that occur during the security service processing will be evident when checking information in the Delete payload. The local security policy SHOULD dictate any action to be taken as a result of security service processing errors.

2. Determine if the Domain of Interpretation (DOI) is supported. If the DOI determination fails, the payload is discarded and the following action is taken:

   (a) The event, INVALID DOI, MAY be logged in the appropriate system audit file.

3. Determine if the Protocol-Id is supported. If the Protocol-Id determination fails, the payload is discarded and the following action is taken:

   (a) The event, INVALID PROTOCOL-ID, MAY be logged in the appropriate system audit file.

4. Determine if the SPI is valid for each SPI included in the Delete payload. For each SPI that is invalid, the following action is taken:

   (a) The event, INVALID SPI, MAY be logged in the appropriate system audit file.

5. Process the Delete payload and take appropriate action, according to local security policy. As described above, one appropriate action SHOULD include cleaning up the local SA database.

6 Conclusions

The Internet Security Association and Key Management Protocol (ISAKMP) is a well designed protocol aimed at the Internet of the future. The massive growth of the Internet will lead to great diversity in network utilization, communications, security requirements, and security mechanisms. ISAKMP contains all the features that will be needed for this dynamic and expanding communications environment.

ISAKMP's Security Association (SA) feature coupled with authentication and key establishment provides the security and flexibility that will be needed for future growth and diversity. This security diversity of multiple key exchange techniques, encryption algorithms, authentication mechanisms, security services, and security attributes will allow users to select the appropriate security for their network, communications, and security needs. The SA feature allows users to specify and negotiate security requirements with other users. An additional benefit of supporting multiple techniques in a single protocol is that as new techniques are developed they can easily be added to the protocol. This provides a path for the growth of Internet security services. ISAKMP supports both publicly or privately defined SAs, making it ideal for government, commercial, and private communications.

ISAKMP provides the ability to establish SAs for multiple security protocols and applications. These protocols and applications may be session-oriented or sessionless. Having one SA establishment protocol that supports multiple security protocols eliminates the need for multiple, nearly identical authentication, key exchange and SA establishment protocols when more than one security protocol is in use or desired. Just as IP has provided the common networking layer for the Internet, a common security establishment protocol is needed if security is to become a reality on the Internet. ISAKMP provides the common base that allows all other security protocols to interoperate.

ISAKMP follows good security design principles. It is not coupled to other insecure transport protocols, therefore it is not vulnerable or weakened by attacks on other protocols. Also, when more secure transport protocols are developed, ISAKMP can be easily migrated to them. ISAKMP also provides protection against protocol related attacks. This protection provides the assurance that the SAs and keys established are with the desired party and not with an attacker.

ISAKMP also follows good protocol design principles. Protocol specific information only is in the protocol header, following the design principles of IPv6. The data transported by the protocol is separated into functional payloads. As the Internet grows and evolves, new payloads to support new security functionality can be added without modifying the entire protocol.

A ISAKMP Security Association Attributes

A.1 Background/Rationale

As detailed in previous sections, ISAKMP is designed to provide a flexible and extensible framework for establishing and managing Security Associations and cryptographic keys. The framework provided by ISAKMP consists of header and payload definitions, exchange types for guiding message and payload exchanges, and general processing guidelines. ISAKMP does not define the mechanisms that will be used to establish and manage Security Associations and cryptographic keys in an authenticated and confidential manner. The definition of mechanisms and their application is the purview of individual Domains of Interpretation (DOIs).

This section describes the ISAKMP values for the Internet IP Security DOI, supported security protocols, and identification values for ISAKMP Phase 1 negotiations. The Internet IP Security DOI is MANDATORY to implement for IP Security. [Oakley] and [IKE] describe, in detail, the mechanisms and their application for establishing and managing Security Associations and cryptographic keys for IP Security.

A.2 Internet IP Security DOI Assigned Value

As described in [IPDOI], the Internet IP Security DOI Assigned Number is one (1).

A.3 Supported Security Protocols

Values for supported security protocols are specified in the most recent "Assigned Numbers" RFC [STD-2]. Presented in the following table are the values for the security protocols supported by ISAKMP for the Internet IP Security DOI.

Protocol	Assigned Value
RESERVED	0
ISAKMP	1

All DOIs MUST reserve ISAKMP with a Protocol-ID of 1. All other security protocols within that DOI will be numbered accordingly.

Security protocol values 2-15359 are reserved to IANA for future use. Values 15360-16383 are permanently reserved for private use amongst mutually consenting implementations. Such private use values are unlikely to be interoperable across different implementations.

A.4 ISAKMP Identification Type Values

The following table lists the assigned values for the Identification Type field found in the Identification payload during a generic Phase 1 exchange, which is not for a specific protocol.

ID Type	Value
ID_IPV4_ADDR	0
ID_IPV4_ADDR_SUBNET	1
ID_IPV6_ADDR	2
ID_IPV6_ADDR_SUBNET	3

A.4.1 ID_IPV4_ADDR

The ID_IPV4_ADDR type specifies a single four (4) octet IPv4 address.

A.4.2 ID_IPV4_ADDR_SUBNET

The ID_IPV4_ADDR_SUBNET type specifies a range of IPv4 addresses, represented by two four (4) octet values. The first value is an IPv4 address. The second is an IPv4 network mask. Note that ones (1s) in the network mask indicate that the corresponding bit in the address is fixed, while zeros (0s) indicate a "wildcard" bit.

A.4.3 ID_IPV6_ADDR

The ID_IPV6_ADDR type specifies a single sixteen (16) octet IPv6 address.

A.4.4 ID_IPV6_ADDR_SUBNET

The ID_IPV6_ADDR_SUBNET type specifies a range of IPv6 addresses, represented by two sixteen (16) octet values. The first value is an IPv6 address. The second is an IPv6 network mask. Note that ones (1s) in the network mask indicate that the corresponding bit in the address is fixed, while zeros (0s) indicate a "wildcard" bit.

B Defining a new Domain of Interpretation

The Internet DOI may be sufficient to meet the security requirements of a large portion of the internet community. However, some groups may have a need to customize some aspect of a DOI, perhaps to add a different set of cryptographic algorithms, or perhaps because they want to make their security-relevant decisions based on something other than a host id or user id. Also, a particular group may have a need for a new exchange type, for example to support key management for multicast groups.

This section discusses guidelines for defining a new DOI. The full specification for the Internet DOI can be found in [IPDOI].

Defining a new DOI is likely to be a time-consuming process. If at all possible, it is recommended that the designer begin with an existing DOI and customize only the parts that are unacceptable.

If a designer chooses to start from scratch, the following MUST be defined:

- o A "situation": the set of information that will be used to determine the required security services.

- o The set of security policies that must be supported.

- o A scheme for naming security-relevant information, including encryption algorithms, key exchange algorithms, etc.

- o A syntax for the specification of proposed security services, attributes, and certificate authorities.

- o The specific formats of the various payload contents.

- o Additional exchange types, if required.

B.1 Situation

The situation is the basis for deciding how to protect a communications channel. It must contain all of the data that will be used to determine the types and strengths of protections applied in an SA. For example, a US Department of Defense DOI would probably use unpublished algorithms and have additional special attributes to negotiate. These additional security attributes would be included in the situation.

B.2 Security Policies

Security policies define how various types of information must be categorized and protected. The DOI must define the set of security policies supported, because both parties in a negotiation must trust that the other party understands a situation, and will protect information appropriately, both in transit and in storage. In a corporate setting, for example, both parties in a negotiation must agree to the meaning of the term "proprietary information" before they can negotiate how to protect it.

Note that including the required security policies in the DOI only specifies that the participating hosts understand and implement those policies in a full system context.

B.3 Naming Schemes

Any DOI must define a consistent way to name cryptographic algorithms, certificate authorities, etc. This can usually be done by using IANA naming conventions, perhaps with some private extensions.

B.4 Syntax for Specifying Security Services

In addition to simply specifying how to name entities, the DOI must also specify the format for complete proposals of how to protect traffic under a given situation.

B.5 Payload Specification

The DOI must specify the format of each of the payload types. For several of the payload types, ISAKMP has included fields that would have to be present across all DOI (such as a certificate authority in the certificate payload, or a key exchange identifier in the key exchange payload).

B.6 Defining new Exchange Types

If the basic exchange types are inadequate to meet the requirements within a DOI, a designer can define up to thirteen extra exchange types per DOI. The designer creates a new exchange type by choosing an unused exchange type value, and defining a sequence of messages composed of strings of the ISAKMP payload types.

Note that any new exchange types must be rigorously analyzed for vulnerabilities. Since this is an expensive and imprecise undertaking, a new exchange type should only be created when absolutely necessary.

Security Considerations

Cryptographic analysis techniques are improving at a steady pace. The continuing improvement in processing power makes once computationally prohibitive cryptographic attacks more realistic. New cryptographic algorithms and public key generation techniques are also being developed at a steady pace. New security services and mechanisms are being developed at an accelerated pace. A consistent method of choosing from a variety of security services and mechanisms and to exchange attributes required by the mechanisms is important to security in the complex structure of the Internet. However, a system that locks itself into a single cryptographic algorithm, key exchange technique, or security mechanism will become increasingly vulnerable as time passes.

UDP is an unreliable datagram protocol and therefore its use in ISAKMP introduces a number of security considerations. Since UDP is unreliable, but a key management protocol must be reliable, the reliability is built into ISAKMP. While ISAKMP utilizes UDP as its transport mechanism, it doesn't rely on any UDP information (e.g. checksum, length) for its processing.

Another issue that must be considered in the development of ISAKMP is the effect of firewalls on the protocol. Many firewalls filter out all UDP packets, making reliance on UDP questionable in certain environments.

A number of very important security considerations are presented in [SEC-ARCH]. One bears repeating. Once a private session key is created, it must be safely stored. Failure to properly protect the private key from access both internal and external to the system completely nullifies any protection provided by the IP Security services.

IANA Considerations

This document contains many "magic" numbers to be maintained by the IANA. This section explains the criteria to be used by the IANA to assign additional numbers in each of these lists.

Domain of Interpretation

The Domain of Interpretation (DOI) is a 32-bit field which identifies the domain under which the security association negotiation is taking place. Requests for assignments of new DOIs must be accompanied by a standards-track RFC which describes the specific domain.

Supported Security Protocols

ISAKMP is designed to provide security association negotiation and key management for many security protocols. Requests for identifiers for additional security protocols must be accompanied by a standards-track RFC which describes the security protocol and its relationship to ISAKMP.

Acknowledgements

Dan Harkins, Dave Carrel, and Derrell Piper of Cisco Systems provided design assistance with the protocol and coordination for the [IKE] and [IPDOI] documents.

Hilarie Orman, via the Oakley key exchange protocol, has significantly influenced the design of ISAKMP.

Marsha Gross, Bill Kutz, Mike Oehler, Pete Sell, and Ruth Taylor provided significant input and review to this document.

Scott Carlson ported the TIS DNSSEC prototype to FreeBSD for use with the ISAKMP prototype.

Jeff Turner and Steve Smalley contributed to the prototype development and integration with ESP and AH.

Mike Oehler and Pete Sell performed interoperability testing with other ISAKMP implementors.

Thanks to Carl Muckenhirn of SPARTA, Inc. for his assistance with LaTeX.

References

[ANSI] ANSI, X9.42: Public Key Cryptography for the Financial Services Industry -- Establishment of Symmetric Algorithm Keys Using Diffie-Hellman, Working Draft, April 19, 1996.

[BC] Ballardie, A., and J. Crowcroft, Multicast-specific Security Threats and Countermeasures, Proceedings of 1995 ISOC Symposium on Networks & Distributed Systems Security, pp. 17-30, Internet Society, San Diego, CA, February 1995.

[Berge] Berge, N., "UNINETT PCA Policy Statements", RFC 1875, December 1995.

[CW87] Clark, D.D. and D.R. Wilson, A Comparison of Commercial and Military Computer Security Policies, Proceedings of the IEEE Symposium on Security & Privacy, Oakland, CA, 1987, pp. 184-193.

[DNSSEC] D. Eastlake III, Domain Name System Protocol Security Extensions, Work in Progress.

[DOW92] Diffie, W., M.Wiener, P. Van Oorschot, Authentication and Authenticated Key Exchanges, Designs, Codes, and Cryptography, 2, 107-125, Kluwer Academic Publishers, 1992.

[IAB] Bellovin, S., "Report of the IAB Security Architecture Workshop", RFC 2316, April 1998.

[IKE] Harkins, D., and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, November 1998.

[IPDOI] Piper, D., "The Internet IP Security Domain of Interpretation for ISAKMP", RFC 2407, November 1998.

[Karn] Karn, P., and B. Simpson, Photuris: Session Key Management Protocol, Work in Progress.

[Kent94] Steve Kent, IPSEC SMIB, e-mail to ipsec@ans.net, August 10, 1994.

[Oakley] Orman, H., "The Oakley Key Determination Protocol", RFC 2412, November 1998.

[RFC-1422] Kent, S., "Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management", RFC 1422, February 1993.

[RFC-1949] Ballardie, A., "Scalable Multicast Key Distribution", RFC 1949, May 1996.

[RFC-2093] Harney, H., and C. Muckenhirn, "Group Key Management Protocol (GKMP) Specification", RFC 2093, July 1997.

[RFC-2094] Harney, H., and C. Muckenhirn, "Group Key Management Protocol (GKMP) Architecture", RFC 2094, July 1997.

[RFC-2119] Bradner, S., "Key Words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[Schneier] Bruce Schneier, Applied Cryptography - Protocols, Algorithms, and Source Code in C (Second Edition), John Wiley & Sons, Inc., 1996.

[SEC-ARCH] Atkinson, R., and S. Kent, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[STD-2] Reynolds, J., and J. Postel, "Assigned Numbers", STD 2, RFC 1700, October 1994. See also: http://www.iana.org/numbers.html

Authors' Addresses

Douglas Maughan
National Security Agency
ATTN: R23
9800 Savage Road
Ft. Meade, MD. 20755-6000

Phone: 301-688-0847
EMail:wdm@tycho.ncsc.mil

Mark Schneider
National Security Agency
ATTN: R23
9800 Savage Road
Ft. Meade, MD. 20755-6000

Phone: 301-688-0851
EMail:mss@tycho.ncsc.mil

Mark Schertler
Securify, Inc.
2415-B Charleston Road
Mountain View, CA 94043

Phone: 650-934-9303
EMail:mjs@securify.com

Jeff Turner
RABA Technologies, Inc.
10500 Little Patuxent Parkway
Columbia, MD. 21044

Phone: 410-715-9399
EMail:jeff.turner@raba.com

Full Copyright Statement

Network Working Group
Request for Comments: 3102
Category: Experimental

Editors:
M. Borella
CommWorks
J. Lo
Candlestick Networks
Contributors:
D. Grabelsky
CommWorks
G. Montenegro
Sun Microsystems
October 2001

Realm Specific IP: Framework

Status of this Memo

This memo defines an Experimental Protocol for the Internet community. It does not specify an Internet standard of any kind. Discussion and suggestions for improvement are requested. Distribution of this memo is unlimited.

Copyright Notice

IESG Note

The IESG notes that the set of documents describing the RSIP technology imply significant host and gateway changes for a complete implementation. In addition, the floating of port numbers can cause problems for some applications, preventing an RSIP-enabled host from interoperating transparently with existing applications in some cases (e.g., IPsec). Finally, there may be significant operational complexities associated with using RSIP. Some of these and other complications are outlined in section 6 of RFC 3102, as well as in the Appendices of RFC 3104. Accordingly, the costs and benefits of using RSIP should be carefully weighed against other means of relieving address shortage.

Abstract

This document examines the general framework of Realm Specific IP (RSIP). RSIP is intended as a alternative to NAT in which the end-to-end integrity of packets is maintained. We focus on implementation issues, deployment scenarios, and interaction with other layer-three protocols.

Table of Contents

1. Introduction

Network Address Translation (NAT) has become a popular mechanism of enabling the separation of addressing spaces. A NAT router must examine and change the network layer, and possibly the transport layer, header of each packet crossing the addressing domains that the NAT router is connecting. This causes the mechanism of NAT to violate the end-to-end nature of the Internet connectivity, and disrupts protocols requiring or enforcing end-to-end integrity of packets.

While NAT does not require a host to be aware of its presence, it requires the presence of an application layer gateway (ALG) within the NAT router for each application that embeds addressing information within the packet payload. For example, most NATs ship with an ALG for FTP, which transmits IP addresses and port numbers on its control channel. RSIP (Realm Specific IP) provides an alternative to remedy these limitations.

RSIP is based on the concept of granting a host from one addressing realm a presence in another addressing realm by allowing it to use resources (e.g., addresses and other routing parameters) from the second addressing realm. An RSIP gateway replaces the NAT router, and RSIP-aware hosts on the private network are referred to as RSIP hosts. RSIP requires ability of the RSIP gateway to grant such resources to RSIP hosts. ALGs are not required on the RSIP gateway for communications between an RSIP host and a host in a different addressing realm.

RSIP can be viewed as a "fix", of sorts, to NAT. It may ameliorate some IP address shortage problems in some scenarios without some of the limitations of NAT. However, it is not a long-term solution to the IP address shortage problem. RSIP allows a degree of address realm transparency to be achieve between two differently-scoped, or completely different addressing realms. This makes it a useful architecture for enabling end-to-end packet transparency between addressing realms. RSIP is expected to be deployed on privately addresses IPv4 networks and used to grant access to publically addressed IPv4 networks. However, in place of the private IPv4 network, there may be an IPv6 network, or a non-IP network. Thus, RSIP allows IP connectivity to a host with an IP stack and IP applications but no native IP access. As such, RSIP can be used, in conjunction with DNS and tunneling, to bridge IPv4 and IPv6 networks, such that dual-stack hosts can communicate with local or remote IPv4 or IPv6 hosts.

It is important to note that, as it is defined here, RSIP does NOT require modification of applications. All RSIP-related modifications to an RSIP host can occur at layers 3 and 4. However, while RSIP does allow end-to-end packet transparency, it may not be transparent to all applications. More details can be found in the section "RSIP complications", below.

1.1. Document Scope

This document provides a framework for RSIP by focusing on four particular areas:

- Requirements of an RSIP host and RSIP gateway.

- Likely initial deployment scenarios.

- Interaction with other layer-three protocols.

- Complications that RSIP may introduce.

The interaction sections will be at an overview level. Detailed modifications that would need to be made to RSIP and/or the interacting protocol are left for separate documents to discuss in detail.

Beyond the scope of this document is discussion of RSIP in large, multiple-gateway networks, or in environments where RSIP state would need to be distributed and maintained across multiple redundant entities.

Discussion of RSIP solutions that do not use some form of tunnel between the RSIP host and RSIP gateway are also not considered in this document.

This document focuses on scenarios that allow privately-addressed IPv4 hosts or IPv6 hosts access to publically-addressed IPv4 networks.

1.2. Terminology

Private Realm

A routing realm that uses private IP addresses from the ranges (10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16) specified in [RFC1918], or addresses that are non-routable from the Internet.

Public Realm

A routing realm with globally unique network addresses.

RSIP Host

A host within an addressing realm that uses RSIP to acquire addressing parameters from another addressing realm via an RSIP gateway.

RSIP Gateway

A router or gateway situated on the boundary between two addressing realms that is assigned one or more IP addresses in at least one of the realms. An RSIP gateway is responsible for parameter management and assignment from one realm to RSIP hosts in the other realm. An RSIP gateway may act as a normal NAT router for hosts within the a realm that are not RSIP enabled.

RSIP Client

An application program that performs the client portion of the RSIP client/server protocol. An RSIP client application MUST exist on all RSIP hosts, and MAY exist on RSIP gateways.

RSIP Server

An application program that performs the server portion of the RSIP client/server protocol. An RSIP server application MUST exist on all RSIP gateways.

RSA-IP: Realm Specific Address IP

An RSIP method in which each RSIP host is allocated a unique IP address from the public realm.

RSAP-IP: Realm Specific Address and Port IP

An RSIP method in which each RSIP host is allocated an IP address (possibly shared with other RSIP hosts) and some number of per-address unique ports from the public realm.

Demultiplexing Fields

Any set of packet header or payload fields that an RSIP gateway uses to route an incoming packet to an RSIP host.

All other terminology found in this document is consistent with that of [RFC2663].

1.3. Specification of Requirements

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this documents are to be interpreted as described in [RFC2119].

2. Architecture

In a typical scenario where RSIP is deployed, there are some number of hosts within one addressing realm connected to another addressing realm by an RSIP gateway. This model is diagrammatically represented as follows:

```
      RSIP Host             RSIP Gateway                    Host

         Xa                    Na   Nb                       Yb
      [X]------( Addr sp. A )----[N]-----( Addr sp. B )-------[Y]
               (  Network   )            (  Network   )
```

Hosts X and Y belong to different addressing realms A and B, respectively, and N is an RSIP gateway (which may also perform NAT functions). N has two interfaces: Na on address space A, and Nb on address space B. N may have a pool of addresses in address space B which it can assign to or lend to X and other hosts in address space A. These addresses are not shown above, but they can be denoted as Nb1, Nb2, Nb3 and so on.

As is often the case, the hosts within address space A are likely to use private addresses while the RSIP gateway is multi-homed with one or more private addresses from address space A in addition to its public addresses from address space B. Thus, we typically refer to the realm in which the RSIP host resides as "private" and the realm from which the RSIP host borrows addressing parameters as the "public" realm. However, these realms may both be public or private - our notation is for convenience. In fact, address space A may be an IPv6 realm or a non-IP address space.

Host X, wishing to establish an end-to-end connection to a network entity Y situated within address space B, first negotiates and obtains assignment of the resources (e.g., addresses and other routing parameters of address space B) from the RSIP gateway. Upon assignment of these parameters, the RSIP gateway creates a mapping, referred as a "bind", of X's addressing information and the assigned resources. This binding enables the RSIP gateway to correctly de-multiplex and forward inbound traffic generated by Y for X. If permitted by the RSIP gateway, X may create multiple such bindings on the same RSIP gateway, or across several RSIP gateways. A lease time SHOULD be associated with each bind.

Using the public parameters assigned by the RSIP gateway, RSIP hosts tunnel data packets across address space A to the RSIP gateway. The RSIP gateway acts as the end point of such tunnels, stripping off the outer headers and routing the inner packets onto the public realm. As mentioned above, an RSIP gateway maintains a mapping of the assigned public parameters as demultiplexing fields for uniquely mapping them to RSIP host private addresses. When a packet from the public realm arrives at the RSIP gateway and it matches a given set of demultiplexing fields, then the RSIP gateway will tunnel it to the appropriate RSIP host. The tunnel headers of outbound packets from X to Y, given that X has been assigned Nb, are as follows:

```
+---------+---------+---------+
| X -> Na | Nb -> Y | payload |
+---------+---------+---------+
```

There are two basic flavors of RSIP: RSA-IP and RSAP-IP. RSIP hosts and gateways MAY support RSA-IP, RSAP-IP, or both.

When using RSA-IP, an RSIP gateway maintains a pool of IP addresses to be leased by RSIP hosts. Upon host request, the RSIP gateway allocates an IP address to the host. Once an address is allocated to a particular host, only that host may use the address until the address is returned to the pool. Hosts MAY NOT use addresses that have not been specifically assigned to them. The hosts may use any TCP/UDP port in combination with their assigned address. Hosts may also run gateway applications at any port and these applications will be available to the public network without assistance from the RSIP gateway. A host MAY lease more than one address from the same or different RSIP gateways. The demultiplexing fields of an RSA-IP session MUST include the IP address leased to the host.

When using RSAP-IP, an RSIP gateway maintains a pool of IP addresses as well as pools of port numbers per address. RSIP hosts lease an IP address and one or more ports to use with it. Once an address / port tuple has been allocated to a particular host, only that host may use the tuple until it is returned to the pool(s). Hosts MAY NOT use address / port combinations that have not been specifically assigned to them. Hosts may run gateway applications bound to an allocated tuple, but their applications will not be available to the public network unless the RSIP gateway has agreed to route all traffic destined to the tuple to the host. A host MAY lease more than one tuple from the same or different RSIP gateways. The demultiplexing fields of an RSAP-IP session MUST include the tuple(s) leased to the host.

3. Requirements

3.1. Host and Gateway Requirements

An RSIP host MUST be able to maintain one or more virtual interfaces for the IP address(es) that it leases from an RSIP gateway. The host MUST also support tunneling and be able to serve as an end-point for one or more tunnels to RSIP gateways. An RSIP host MUST NOT respond to ARPs for a public realm address that it leases.

An RSIP host supporting RSAP-IP MUST be able to maintain a set of one or more ports assigned by an RSIP gateway from which choose ephemeral source ports. If the host's pool does not have any free ports and the host needs to open a new communication session with a public host, it MUST be able to dynamically request one or more additional ports via its RSIP mechanism.

An RSIP gateway is a multi-homed host that routes packets between two or more realms. Often, an RSIP gateway is a boundary router between two or more administrative domains. It MUST also support tunneling and be able to serve as an end-point for tunnels to RSIP hosts. The RSIP gateway MAY be a policy enforcement point, which in turn may require it to perform firewall and packet filtering duties in addition to RSIP. The RSIP gateway MUST reassemble all incoming packet fragments from the public network in order to be able to route and tunnel them to the proper host. As is necessary for fragment reassembly, an RSIP gateway MUST timeout fragments that are never fully reassembled.

An RSIP gateway MAY include NAT functionality so that hosts on the private network that are not RSIP-enabled can still communicate with the public network. An RSIP gateway MUST manage all resources that are assigned to RSIP hosts. This management MAY be done according to local policy.

3.2. Processing of Demultiplexing Fields

Each active RSIP host must have a unique set of demultiplexing fields assigned to it so that an RSIP gateway can route incoming packets appropriately. Depending on the type of mapping used by the RSIP gateway, demultiplexing fields have been defined to be one or more of the following:

- destination IP address
- IP protocol
- destination TCP or UDP port
- IPSEC SPI present in ESP or AH header (see [RFC3104])
- others

Note that these fields may be augmented by source IP address and source TCP or UDP port.

Demultiplexing of incoming traffic can be based on a decision tree. The process begins with the examination of the IP header of the incoming packet, and proceeds to subsequent headers and then the payload.

- In the case where a public IP address is assigned for each host, a unique public IP address is mapped to each RSIP host.

- If the same IP address is used for more than one RSIP host, then subsequent headers must have at least one field that will be assigned a unique value per host so that it is usable as a demultiplexing field. The IP protocol field SHOULD be used to determine what in the subsequent headers these demultiplexing fields ought to be.

- If the subsequent header is TCP or UDP, then destination port number can be used. However, if the TCP/UDP port number is the same for more than one RSIP host, the payload section of the packet must contain a demultiplexing field that is guaranteed to be different for each RSIP host. Typically this requires negotiation of said fields between the RSIP host and gateway so that the RSIP gateway can guarantee that the fields are unique per-host

- If the subsequent header is anything other than TCP or UDP, there must exist other fields within the IP payload usable as demultiplexing fields. In other words, these fields must be able to be set such that they are guaranteed to be unique per-host. Typically this requires negotiation of said fields between the RSIP host and gateway so that the RSIP gateway can guarantee that the fields are unique per-host.

It is desirable for all demultiplexing fields to occur in well-known fixed locations so that an RSIP gateway can mask out and examine the appropriate fields on incoming packets. Demultiplexing fields that are encrypted MUST NOT be used for routing.

3.3. RSIP Protocol Requirements and Recommendations

RSIP gateways and hosts MUST be able to negotiate IP addresses when using RSA-IP, IP address / port tuples when using RSAP-IP, and possibly other demultiplexing fields for use in other modes.

In this section we discuss the requirements and implementation issues of an RSIP negotiation protocol.

For each required demultiplexing field, an RSIP protocol MUST, at the very least, allow for:

- RSIP hosts to request assignments of demultiplexing fields

- RSIP gateways to assign demultiplexing fields with an associated lease time

- RSIP gateways to reclaim assigned demultiplexing fields

Additionally, it is desirable, though not mandatory, for an RSIP protocol to negotiate an RSIP method (RSA-IP or RSAP-IP) and the type of tunnel to be used across the private network. The protocol SHOULD be extensible and facilitate vendor-specific extensions.

If an RSIP negotiation protocol is implemented at the application layer, a choice of transport protocol MUST be made. RSIP hosts and gateways may communicate via TCP or UDP. TCP support is required in all RSIP gateways, while UDP support is optional. In RSIP hosts, TCP, UDP, or both may be supported. However, once an RSIP host and gateway have begun communicating using either TCP or UDP, they MAY NOT switch to the other transport protocol. For RSIP implementations and deployments considered in this document, TCP is the recommended transport protocol, because TCP is known to be robust across a wide range of physical media types and traffic loads.

It is recommended that all communication between an RSIP host and gateway be authenticated. Authentication, in the form of a message hash appended to the end of each RSIP protocol packet, can serve to authenticate the RSIP host and gateway to one another, provide message integrity, and (with an anti-replay counter) avoid replay attacks. In order for authentication to be supported, each RSIP host and the RSIP gateway MUST either share a secret key (distributed, for example, by Kerberos) or have a private/public key pair. In the latter case, an entity's public key can be computed over each message and a hash function applied to the result to form the message hash.

3.4. Interaction with DNS

An RSIP-enabled network has three uses for DNS: (1) public DNS services to map its static public IP addresses (i.e., the public address of the RSIP gateway) and for lookups of public hosts, (2) private DNS services for use only on the private network, and (3) dynamic DNS services for RSIP hosts.

With respect to (1), public DNS information MUST be propagated onto the private network. With respect to (2), private DNS information MUST NOT be propagated into the public network.

With respect to (3), an RSIP-enabled network MAY allow for RSIP hosts with FQDNs to have their A and PTR records updated in the public DNS. These updates are based on address assignment facilitated by RSIP, and should be performed in a fashion similar to DHCP updates to dynamic DNS [DHCP-DNS]. In particular, RSIP hosts should be allowed to update their A records but not PTR records, while RSIP gateways can update both. In order for the RSIP gateway to update DNS records on behalf on an RSIP host, the host must provide the gateway with its FQDN.

Note that when using RSA-IP, the interaction with DNS is completely analogous to that of DHCP because the RSIP host "owns" an IP address for a period of time. In the case of RSAP-IP, the claim that an RSIP host has to an address is only with respect to the port(s) that it has leased along with an address. Thus, two or more RSIP hosts' FQDNs may map to the same IP address. However, a public host may expect that all of the applications running at a particular address are owned by the same logical host, which would not be the case. It is recommended that RSAP-IP and dynamic DNS be integrated with some caution, if at all.

3.5. Locating RSIP Gateways

When an RSIP host initializes, it requires (among other things) two critical pieces of information. One is a local (private) IP address to use as its own, and the other is the private IP address of an RSIP gateway. This information can be statically configured or dynamically assigned.

In the dynamic case, the host's private address is typically supplied by DHCP. A DHCP option could provide the IP address of an RSIP gateway in DHCPOFFER messages. Thus, the host's startup procedure would be as follows: (1) perform DHCP, (2) if an RSIP gateway option is present in the DHCPOFFER, record the IP address therein as the RSIP gateway.

Alternatively, the RSIP gateway can be discovered via SLP (Service Location Protocol) as specified in [SLP-RSIP]. The SLP template defined allows for RSIP service provisioning and load balancing.

3.6. Implementation Considerations

RSIP can be accomplished by any one of a wide range of implementation schemes. For example, it can be built into an existing configuration protocol such as DHCP or SOCKS, or it can exist as a separate protocol. This section discusses implementation issues of RSIP in general, regardless of how the RSIP mechanism is implemented.

Note that on a host, RSIP is associated with a TCP/IP stack implementation. Modifications to IP tunneling and routing code, as well as driver interfaces may need to be made to support RSA-IP. Support for RSAP-IP requires modifications to ephemeral port selection code as well. If a host has multiple TCP/IP stacks or TCP/IP stacks and other communication stacks, RSIP will only operate on the packets / sessions that are associated with the TCP/IP stack(s) that use RSIP. RSIP is not application specific, and if it is implemented in a stack, it will operate beneath all applications that use the stack.

4. Deployment

When RSIP is deployed in certain scenarios, the network characteristics of these scenarios will determine the scope of the RSIP solution, and therefore impact the requirements of RSIP. In this section, we examine deployment scenarios, and the impact that RSIP may have on existing networks.

4.1. Possible Deployment Scenarios

In this section we discuss a number of potential RSIP deployment scenarios. The selection below are not comprehensive and other scenarios may emerge.

4.1.1. Small / Medium Enterprise

Up to several hundred hosts will reside behind an RSIP-enabled router. It is likely that there will be only one gateway to the public network and therefore only one RSIP gateway. This RSIP gateway may control only one, or perhaps several, public IP addresses. The RSIP gateway may also perform firewall functions, as well as routing inbound traffic to particular destination ports on to a small number of dedicated gateways on the private network.

4.1.2. Residential Networks

This category includes both networking within just one residence, as well as within multiple-dwelling units. At most several hundred hosts will share the gateway’s resources. In particular, many of these devices may be thin hosts or so-called "network appliances" and therefore not require access to the public Internet frequently. The RSIP gateway is likely to be implemented as part of a residential firewall, and it may be called upon to route traffic to particular destination ports on to a small number of dedicated gateways on the private network. It is likely that only one gateway to the public network will be present and that this gateway's RSIP gateway will control only one IP address. Support for secure end-to-end VPN access to corporate intranets will be important.

4.1.3. Hospitality Networks

A hospitality network is a general type of "hosting" network that a traveler will use for a short period of time (a few minutes or a few hours). Examples scenarios include hotels, conference centers and airports and train stations. At most several hundred hosts will share the gateway's resources. The RSIP gateway may be implemented as part of a firewall, and it will probably not be used to route traffic to particular destination ports on to dedicated gateways on the private network. It is likely that only one gateway to the public network will be present and that this gateway's RSIP gateway will control only one IP address. Support for secure end-to-end VPN access to corporate intranets will be important.

4.1.4. Dialup Remote Access

RSIP gateways may be placed in dialup remote access concentrators in order to multiplex IP addresses across dialup users. At most several hundred hosts will share the gateway's resources. The RSIP gateway may or may not be implemented as part of a firewall, and it will probably not be used to route traffic to particular destination ports on to dedicated gateways on the private network. Only one gateway to the public network will be present (the remote access concentrator itself) and that this gateway's RSIP gateway will control a small number of IP addresses. Support for secure end-to-end VPN access to corporate intranets will be important.

4.1.5. Wireless Remote Access Networks

Wireless remote access will become very prevalent as more PDA and IP / cellular devices are deployed. In these scenarios, hosts may be changing physical location very rapidly - therefore Mobile IP will play a role. Hosts typically will register with an RSIP gateway for a short period of time. At most several hundred hosts will share the gateway's resources. The RSIP gateway may be implemented as part of a firewall, and it will probably not be used to route traffic to particular destination ports on to dedicated gateways on the private network. It is likely that only one gateway to the public network will be present and that this gateway's RSIP gateway will control a small number of IP addresses. Support for secure end-to-end VPN access to corporate intranets will be important.

4.2. Cascaded RSIP and NAT

It is possible for RSIP to allow for cascading of RSIP gateways as well as cascading of RSIP gateways with NAT boxes. For example, consider an ISP that uses RSIP for address sharing amongst its customers. It might assign resources (e.g., IP addresses and ports) to a particular customer. This customer may use RSIP to further subdivide the port ranges and address(es) amongst individual end hosts. No matter how many levels of RSIP assignment exists, RSIP MUST only assign public IP addresses.

Note that some of the architectures discussed below may not be useful or desirable. The goal of this section is to explore the interactions between NAT and RSIP as RSIP is incrementally deployed on systems that already support NAT.

4.2.1. RSIP Behind RSIP

A reference architecture is depicted below.

```
                              +-----------+
                              |           |
                              |   RSIP    |
                              |  gateway  +---- 10.0.0.0/8
                              |     B     |
                              |           |
                              +-----+-----+
                                    |
                                    | 10.0.1.0/24
                     +-----------+  | (149.112.240.0/25)
                     |           |  |
      149.112.240.0/24|   RSIP    +--+
      ----------------+  gateway  |
                     |     A     +--+
                     |           |  |
                     +-----------+  | 10.0.2.0/24
                                    | (149.112.240.128/25)
                                    |
                              +-----+-----+
                              |           |
                              |   RSIP    |
                              |  gateway  +---- 10.0.0.0/8
                              |     C     |
                              |           |
                              +-----------+
```

RSIP gateway A is in charge of the IP addresses of subnet 149.112.240.0/24. It distributes these addresses to RSIP hosts and RSIP gateways. In the given configuration, it distributes addresses 149.112.240.0 - 149.112.240.127 to RSIP gateway B, and addresses 149.112.240.128 - 149.112.240.254 to RSIP gateway C. Note that the subnet broadcast address, 149.112.240.255, must remain unclaimed, so that broadcast packets can be distributed to arbitrary hosts behind RSIP gateway A. Also, the subnets between RSIP gateway A and RSIP gateways B and C will use private addresses.

Due to the tree-like fashion in which addresses will be cascaded, we will refer to RSIP gateways A as the 'parent' of RSIP gateways B and C, and RSIP gateways B and C as 'children' of RSIP gateways A. An arbitrary number of levels of children may exist under a parent RSIP gateway.

A parent RSIP gateway will not necessarily be aware that the address(es) and port blocks that it distributes to a child RSIP gateway will be further distributed. Thus, the RSIP hosts MUST tunnel their outgoing packets to the nearest RSIP gateway. This gateway will then verify that the sending host has used the proper address and port block, and then tunnel the packet on to its parent RSIP gateway.

For example, in the context of the diagram above, host 10.0.0.1, behind RSIP gateway C will use its assigned external IP address (say, 149.112.240.130) and tunnel its packets over the 10.0.0.0/8 subnet to RSIP gateway C. RSIP gateway C strips off the outer IP header. After verifying that the source public IP address and source port number is valid, RSIP gateway C will tunnel the packets over the 10.0.2.0/8 subnet to RSIP gateway A. RSIP gateway A strips off the outer IP header. After verifying that the source public IP address and source port number is valid, RSIP gateway A transmits the packet on the public network.

While it may be more efficient in terms of computation to have a RSIP host tunnel directly to the overall parent of an RSIP gateway tree, this would introduce significant state and administrative difficulties.

A RSIP gateway that is a child MUST take into consideration the parameter assignment constraints that it inherits from its parent when it assigns parameters to its children. For example, if a child RSIP gateway is given a lease time of 3600 seconds on an IP address, it MUST compare the current time to the lease time and the time that the lease was assigned to compute the maximum allowable lease time on the address if it is to assign the address to a RSIP host or child RSIP gateway.

4.2.2. NAT Behind RSIP

```
             +--------+      +--------+
             | NAT w/ |      |  RSIP  |
hosts ------+ RSIP   +------+ gate-  +----- public network
             | host   |      |  way   |
             +--------+      +--------+
```

In this architecture, an RSIP gateway is between a NAT box and the public network. The NAT is also equipped with an RSIP host. The NAT dynamically requests resources from the RSIP gateway as the hosts establish sessions to the public network. The hosts are not aware of the RSIP manipulation. This configuration does not enable the hosts to have end-to-end transparency and thus the NAT still requires ALGs and the architecture cannot support IPSEC.

4.2.3. RSIP Behind NAT

```
             +--------+      +--------+
RSIP         |  RSIP  |      |        |
hosts ------+ gate-  +------+   NAT  +----- public network
             |  way   |      |        |
             +--------+      +--------+
```

In this architecture, the RSIP hosts and gateway reside behind a NAT. This configuration does not enable the hosts to have end-to-end transparency and thus the NAT still requires ALGs and the architecture cannot support IPSEC. The hosts may have transparency if there is another gateway to the public network besides the NAT box, and this gateway supports cascaded RSIP behind RSIP.

4.2.4. RSIP Through NAT

```
             +--------+      +--------+
RSIP         |        |      |  RSIP  |
hosts ------+   NAT  +------+ gate-  +----- public network
             |        |      |  way   |
             +--------+      +--------+
```

In this architecture, the RSIP hosts are separated from the RSIP gateway by a NAT. RSIP signaling may be able to pass through the NAT if an RSIP ALG is installed. The RSIP data flow, however, will have its outer IP address translated by the NAT. The NAT must not translate the port numbers in order for RSIP to work properly. Therefore, only traditional NAT will make sense in this context.

5. Interaction with Layer-Three Protocols

Since RSIP affects layer-three objects, it has an impact on other layer three protocols. In this section, we outline the impact of RSIP on these protocols, and in each case, how RSIP, the protocol, or both, can be extended to support interaction.

Each of these sections is an overview and not a complete technical specification. If a full technical specification of how RSIP interacts with a layer-three protocol is necessary, a separate document will contain it.

5.1. IPSEC

RSIP is a mechanism for allowing end-to-end IPSEC with sharing of IP addresses. Full specification of RSIP/IPSEC details are in [RSIP-IPSEC]. This section provides a brief summary. Since IPSEC may encrypt TCP/UDP port numbers, these objects cannot be used as demultiplexing fields. However, IPSEC inserts an AH or ESP header following the IP header in all IPSEC-protected packets (packets that are transmitted on an IPSEC Security Association (SA)). These headers contain a 32-bit Security Parameter Index (SPI) field, the value of which is determined by the receiving side. The SPI field is always in the clear. Thus, during SA negotiation, an RSIP host can instruct their public peer to use a particular SPI value. This SPI value, along with the assigned IP address, can be used by an RSIP gateway to uniquely identify and route packets to an RSIP host. In order to guarantee that RSIP hosts use SPIs that are unique per address, it is necessary for the RSIP gateway to allocate unique SPIs to hosts along with their address/port tuple.

IPSEC SA negotiation takes place using the Internet Key Exchange (IKE) protocol. IKE is designated to use port 500 on at least the destination side. Some host IKE implementations will use source port 500 as well, but this behavior is not mandatory. If two or more RSIP hosts are running IKE at source port 500, they MUST use different initiator cookies (the first eight bytes of the IKE payload) per assigned IP address. The RSIP gateway will be able to route incoming IKE packets to the proper host based on initiator cookie value. Initiator cookies can be negotiated, like ports and SPIs. However, since the likelihood of two hosts assigned the same IP address attempting to simultaneously use the same initiator cookie is very small, the RSIP gateway can guarantee cookie uniqueness by dropping IKE packets with a cookie value that is already in use.

5.2. Mobile IP

Mobile IP allows a mobile host to maintain an IP address as it moves from network to network. For Mobile IP foreign networks that use private IP addresses, RSIP may be applicable. In particular, RSIP would allow a mobile host to bind to a local private address, while maintaining a global home address and a global care-of address. The global care-of address could, in principle, be shared with other mobile nodes.

The exact behavior of Mobile IP with respect to private IP addresses has not be settled. Until it is, a proposal to adapt RSIP to such a scenario is premature. Also, such an adaptation may be considerably complex. Thus, integration of RSIP and Mobile IP is a topic of ongoing consideration.

5.3. Differentiated and Integrated Services

To attain the capability of providing quality of service between two communicating hosts in different realms, it is important to consider the interaction of RSIP with different quality of service provisioning models and mechanisms. In the section, RSIP interaction with the integrated service and differentiated service frameworks is discussed.

5.3.1. Differentiated Services

The differentiated services architecture defined in [RFC2475] allows networks to support multiple levels of best-effort service through the use of "markings" of the IP Type-of-Service (now DS) byte. Each value of the DS byte is termed a differentiated services code point (DSCP) and represents a particular per-hop behavior. This behavior may not be the same in all administrative domains. No explicit signaling is necessary to support differentiated services.

For outbound packets from an edge network, DSCP marking is typically performed and/or enforced on a boundary router. The marked packet is then forwarded onto the public network. In an RSIP-enabled network, a natural place for DSCP marking is the RSIP gateway. In the case of RSAP-IP, the RSIP gateway can apply its micro-flow (address/port tuple) knowledge of RSIP assignments in order to provide different service levels to different RSIP host. For RSA-IP, the RSIP gateway will not necessarily have knowledge of micro-flows, so it must rely on markings made by the RSIP hosts (if any) or apply a default policy to the packets.

When differentiated services is to be performed between RSIP hosts and gateways, it must be done over the tunnel between these entities. Differentiated services over a tunnel is considered in detail in [DS-TUNN], the key points that need to be addressed here are the behaviors of tunnel ingress and egress for both incoming and going packets.

For incoming packets arriving at an RSIP gateway tunnel ingress, the RSIP gateway may either copy the DSCP from the inner header to the outer header, leave the inner header DSCP untouched, but place a different DSCP in the outer header, or change the inner header DSCP while applying either the same or a different DSCP to the outer header.

For incoming packets arriving at an RSIP host tunnel egress, behavior with respect to the DSCP is not necessarily important if the RSIP host not only terminates the tunnel, but consumes the packet as well. If this is not the case, as per some cascaded RSIP scenarios, the RSIP host must apply local policy to determine whether to leave the inner header DSCP as is, overwrite it with the outer header DSCP, or overwrite it with a different value.

For outgoing packets arriving at an RSIP host tunnel ingress, the host may either copy the DSCP from the inner header to the outer header, leave the inner header DSCP untouched, but place a different DSCP in the outer header, or change the inner header DSCP while applying either the same or a different DSCP to the outer header.

For outgoing packets arriving at an RSIP gateway tunnel egress, the RSIP gateway must apply local policy to determine whether to leave the inner header DSCP as is, overwrite it with the outer header DSCP, or overwrite it with a different value.

It is reasonable to assume that in most cases, the diffserv policy applicable on a site will be the same for RSIP and non-RSIP hosts. For this reason, a likely policy is that the DSCP will always be copied between the outer and inner headers in all of the above cases. However, implementations should allow for the more general case.

5.3.2. Integrated Services

The integrated services model as defined by [RFC2205] requires signalling using RSVP to setup a resource reservation in intermediate nodes between the communicating endpoints. In the most common scenario in which RSIP is deployed, receivers located within the private realm initiate communication sessions with senders located within the public realm. In this section, we discuss the interaction of RSIP architecture and RSVP in such a scenario. The less common case of having senders within the private realm and receivers within the public realm is not discussed although concepts mentioned here may be applicable.

With senders in the public realm, RSVP PATH messages flow downstream from sender to receiver, inbound with respect to the RSIP gateway, while RSVP RESV messages flow in the opposite direction. Since RSIP uses tunneling between the RSIP host and gateway within the private realm, how the RSVP messages are handled within the RSIP tunnel depends on situations elaborated in [RFC2746].

Following the terminology of [RFC2476], if Type 1 tunnels exist between the RSIP host and gateway, all intermediate nodes inclusive of the RSIP gateway will be treated as a non-RSVP aware cloud without QoS reserved on these nodes. The tunnel will be viewed as a single (logical) link on the path between the source and destination. End-to-end RSVP messages will be forwarded through the tunnel encapsulated in the same way as normal IP packets. We see this as the most common and applicable deployment scenario.

However, should Type 2 or 3 tunnels be deployed between the tunneling endpoints , end-to-end RSVP session has to be statically mapped (Type 2) or dynamically mapped (Type 3) into the tunnel sessions. While the end-to-end RSVP messages will be forwarded through the tunnel encapsulated in the same way as normal IP packets, a tunnel session is established between the tunnel endpoints to ensure QoS reservation within the tunnel for the end-to-end session. Data traffic needing special QoS assurance will be encapsulated in a UDP/IP header while normal traffic will be encapsulated using the normal IP-IP encapsulation. In the type 2 deployment scenario where all data traffic flowing to the RSIP host receiver are given QoS treatment, UDP/IP encapsulation will be rendered in the RSIP gateway for all data flows. The tunnel between the RSIP host and gateway could be seen as a "hard pipe". Traffic exceeding the QoS guarantee of the "hard pipe" would fall back to the best effort IP-IP tunneling.

In the type 2 deployment scenario where data traffic could be selectively channeled into the UDP/IP or normal IP-IP tunnel, or for type 3 deployment where end-to-end sessions could be dynamically mapped into tunnel sessions, integration with the RSIP model could be complicated and tricky. (Note that these are the cases where the tunnel link could be seen as a expandable soft pipe.) Two main issues are worth considering.

- For RSIP gateway implementations that does encapsulation of the incoming stream before passing to the IP layer for forwarding, the RSVP daemon has to be explicitly signaled upon reception of incoming RSVP PATH messages. The RSIP implementation has to recognize RSVP PATH messages and pass them to the RSVP daemon instead of doing the default tunneling. Handling of other RSVP messages would be as described in [RFC2746].

- RSIP enables an RSIP host to have a temporary presence at the RSIP gateway by assuming one of the RSIP gateway's global interfaces. As a result, the RSVP PATH messages would be addressed to the RSIP gateway. Also, the RSVP SESSION object within an incoming RSVP PATH would carry the global destination address, destination port (and protocol) tuples that were leased by the RSIP gateway to the RSIP host. Hence the realm unaware RSVP daemon running on the RSIP gateway has to be presented with a translated version of the RSVP messages. Other approaches are possible, for example making the RSVP daemon realm aware.

A simple mechanism would be to have the RSIP module handle the necessary RSVP message translation. For an incoming RSVP signalling flow, the RSIP module does a packet translation of the IP header and RSVP SESSION object before handling the packet over to RSVP. The global address leased to the host is translated to the true private address of the host. (Note that this mechanism works with both RSA-IP and RSAP-IP.) The RSIP module also has to do an opposite translation from private to global parameter (plus tunneling) for end-to-end PATH messages generated by the RSVP daemon towards the RSIP host receiver. A translation on the SESSION object also has to be done for RSVP outbound control messages. Once the RSVP daemon gets the message, it maps them to an appropriate tunnel sessions.

Encapsulation of the inbound data traffic needing QoS treatment would be done using UDP-IP encapsulation designated by the tunnel session. For this reason, the RSIP module has to be aware of the UDP-IP encapsulation to use for a particular end-to-end session. Classification and scheduling of the QoS guaranteed end-to-end flow on the output interface of the RSIP gateway would be based on the UDP/IP encapsulation. Mapping between the tunnel session and end-to-end session could continue to use the mechanisms proposed in [RFC2746]. Although [RFC2746] proposes a number of approaches for this purpose, we propose using the SESSION_ASSOC object introduced because of its simplicity.

5.4. IP Multicast

The amount of specific RSIP/multicast support that is required in RSIP hosts and gateways is dependent on the scope of multicasting in the RSIP-enabled network, and the roles that the RSIP hosts will play. In this section, we discuss RSIP and multicast interactions in a number of scenarios.

Note that in all cases, the RSIP gateway MUST be multicast aware because it is on an administrative boundary between two domains that will not be sharing their all of their routing information. The RSIP gateway MUST NOT allow private IP addresses to be propagated on the public network as part of any multicast message or as part of a routing table.

5.4.1. Receiving-Only Private Hosts, No Multicast Routing on Private Network

In this scenario, private hosts will not source multicast traffic, but they may join multicast groups as recipients. In the private network, there are no multicast-aware routers, except for the RSIP gateway.

Private hosts may join and leave multicast groups by sending the appropriate IGMP messages to an RSIP gateway (there may be IGMP proxy routers between RSIP hosts and gateways). The RSIP gateway will coalesce these requests and perform the appropriate actions, whether they be to perform a multicast WAN routing protocol, such as PIM, or to proxy the IGMP messages to a WAN multicast router. In other words, if one or more private hosts request to join a multicast group, the RSIP gateway MUST join in their stead, using one of its own public IP addresses.

Note that private hosts do not need to acquire demultiplexing fields and use RSIP to receive multicasts. They may receive all multicasts using their private addresses, and by private address is how the RSIP gateway will keep track of their group membership.

5.4.2. Sending and Receiving Private Hosts, No Multicast Routing on Private Network

This scenarios operates identically to the previous scenario, except that when a private host becomes a multicast source, it MUST use RSIP and acquire a public IP address (note that it will still receive on its private address). A private host sending a multicast will use a public source address and tunnel the packets to the RSIP gateway. The RSIP gateway will then perform typical RSIP functionality, and route the resulting packets onto the public network, as well as back to the private network, if there are any listeners on the private network.

If there is more than one sender on the private network, then, to the public network it will seem as if all of these senders share the same IP address. If a downstream multicasting protocol identifies sources based on IP address alone and not port numbers, then it is possible that these protocols will not be able to distinguish between the senders.

6. RSIP Complications

In this section we document the know complications that RSIP may cause. While none of these complications should be considered "show stoppers" for the majority of applications, they may cause unexpected or undefined behavior. Where it is appropriate, we discuss potential remedial procedures that may reduce or eliminate the deleterious impact of a complication.

6.1. Unnecessary TCP TIME_WAIT

When TCP disconnects a socket, it enters the TCP TIME_WAIT state for a period of time. While it is in this state it will refuse to accept new connections using the same socket (i.e., the same source address/port and destination address/port). Consider the case in which an RSIP host (using RSAP-IP) is leased an address/port tuple and uses this tuple to contact a public address/port tuple. Suppose that the host terminates the session with the public tuple and immediately returns its leased tuple to the RSIP gateway. If the RSIP gateway immediately allocates this tuple to another RSIP host (or to the same host), and this second host uses the tuple to contact the same public tuple while the socket is still in the TIME_WAIT phase, then the host's connection may be rejected by the public host.

In order to mitigate this problem, it is recommended that RSIP gateways hold recently deallocated tuples for at least two minutes, which is the greatest duration of TIME_WAIT that is commonly implemented. In situations where port space is scarce, the RSIP gateway MAY choose to allocate ports in a FIFO fashion from the pool of recently deallocated ports.

6.2. ICMP State in RSIP Gateway

Like NAT, RSIP gateways providing RSAP-IP must process ICMP responses from the public network in order to determine the RSIP host (if any) that is the proper recipient. We distinguish between ICMP error packets, which are transmitted in response to an error with an associated IP packet, and ICMP response packets, which are transmitted in response to an ICMP request packet.

ICMP request packets originating on the private network will typically consist of echo request, timestamp request and address mask request. These packets and their responses can be identified by the tuple of source IP address, ICMP identifier, ICMP sequence number, and destination IP address. An RSIP host sending an ICMP request packet tunnels it to the RSIP gateway, just as it does TCP and UDP packets. The RSIP gateway must use this tuple to map incoming ICMP responses to the private address of the appropriate RSIP host. Once it has done so, it will tunnel the ICMP response to the host. Note that it is possible for two RSIP hosts to use the same values for the tuples listed above, and thus create an ambiguity. However, this occurrence is likely to be quite rare, and is not addressed further in this document.

Incoming ICMP error response messages can be forwarded to the appropriate RSIP host by examining the IP header and port numbers embedded within the ICMP packet. If these fields are not present, the packet should be silently discarded.

Occasionally, an RSIP host will have to send an ICMP response (e.g., port unreachable). These responses are tunneled to the RSIP gateway, as is done for TCP and UDP packets. All ICMP requests (e.g., echo request) arriving at the RSIP gateway MUST be processed by the RSIP gateway and MUST NOT be forwarded to an RSIP host.

6.3. Fragmentation and IP Identification Field Collision

If two or more RSIP hosts on the same private network transmit outbound packets that get fragmented to the same public gateway, the public gateway may experience a reassembly ambiguity if the IP header ID fields of these packets are identical.

For TCP packets, a reasonably small MTU can be set so that fragmentation is guaranteed not to happen, or the likelihood or fragmentation is extremely small. If path MTU discovery works properly, the problem is mitigated. For UDP, applications control the size of packets, and the RSIP host stack may have to fragment UDP packets that exceed the local MTU. These packets may be fragmented by an intermediate router as well.

The only completely robust solution to this problem is to assign all RSIP hosts that are sharing the same public IP address disjoint blocks of numbers to use in their IP identification fields. However, whether this modification is worth the effort of implementing is currently unknown.

6.4. Application Servers on RSAP-IP Hosts

RSAP-IP hosts are limited by the same constraints as NAT with respect to hosting servers that use a well-known port. Since destination port numbers are used as routing information to uniquely identify an RSAP-IP host, typically no two RSAP-IP hosts sharing the same public

IP address can simultaneously operate publically-available gateways on the same port. For protocols that operate on well-known ports, this implies that only one public gateway per RSAP-IP IP address / port tuple is used simultaneously. However, more than one gateway per RSAP-IP IP address / port tuple may be used simultaneously if and only if there is a demultiplexing field within the payload of all packets that will uniquely determine the identity of the RSAP-IP host, and this field is known by the RSIP gateway.

In order for an RSAP-IP host to operate a publically-available gateway, the host must inform the RSIP gateway that it wishes to receive all traffic destined to that port number, per its IP address. Such a request MUST be denied if the port in question is already in use by another host.

In general, contacting devices behind an RSIP gateway may be difficult. A potential solution to the general problem would be an architecture that allows an application on an RSIP host to register a public IP address / port pair in a public database. Simultaneously, the RSIP gateway would initiate a mapping from this address / port tuple to the RSIP host. A peer application would then be required to contact the database to determine the proper address / port at which to contact the RSIP host's application.

6.5. Determining Locality of Destinations from an RSIP Host

In general, an RSIP host must know, for a particular IP address, whether it should address the packet for local delivery on the private network, or if it has to use an RSIP interface to tunnel to an RSIP gateway (assuming that it has such an interface available).

If the RSIP hosts are all on a single subnet, one hop from an RSIP gateway, then examination of the local network and subnet mask will provide the appropriate information. However, this is not always the case.

An alternative that will work in general for statically addressed private networks is to store a list of the network and subnet masks of every private subnet at the RSIP gateway. RSIP hosts may query the gateway with a particular target IP address, or for the entire list.

If the subnets on the local side of the network are changing more rapidly than the lifetime of a typical RSIP session, the RSIP host may have to query the location of every destination that it tries to communicate with.

If an RSIP host transmits a packet addressed to a public host without using RSIP, then the RSIP gateway will apply NAT to the packet (if it supports NAT) or it may discard the packet and respond with and appropriate ICMP message.

A robust solution to this problem has proven difficult to develop. Currently, it is not known how severe this problem is. It is likely that it will be more severe on networks where the routing information is changing rapidly that on networks with relatively static routes.

6.6. Implementing RSIP Host Deallocation

An RSIP host MAY free resources that it has determined it no longer requires. For example, on an RSAP-IP subnet with a limited number of public IP addresses, port numbers may become scarce. Thus, if RSIP hosts are able to dynamically deallocate ports that they no longer need, more hosts can be supported.

However, this functionality may require significant modifications to a vanilla TCP/IP stack in order to implement properly. The RSIP host must be able to determine which TCP or UDP sessions are using RSIP resources. If those resources are unused for a period of time, then the RSIP host may deallocate them. When an open socket's resources are deallocated, it will cause some associated applications to fail. An analogous case would be TCP and UDP sessions that must terminate when an interface that they are using loses connectivity.

On the other hand, this issue can be considered a resource allocation problem. It is not recommended that a large number (hundreds) of hosts share the same IP address, for performance purposes. Even if, say, 100 hosts each are allocated 100 ports, the total number of ports in use by RSIP would be still less than one-sixth the total port space for an IP address. If more hosts or more ports are needed, more IP addresses should be used. Thus, it is reasonable, that in many cases, RSIP hosts will not have to deallocate ports for the lifetime of their activity.

Since RSIP demultiplexing fields are leased to hosts, an appropriately chosen lease time can alleviate some port space scarcity issues.

6.7. Multi-Party Applications

Multi-party applications are defined to have at least one of the following characteristics:

- A third party sets up sessions or connections between two hosts.

- Computation is distributed over a number of hosts such that the individual hosts may communicate with each other directly.

RSIP has a fundamental problem with multi-party applications. If some of the parties are within the private addressing realm and others are within the public addressing realm, an RSIP host may not know when to use private addresses versus public addresses. In particular, IP addresses may be passed from party to party under the assumption that they are global endpoint identifiers. This may cause multi-party applications to fail.

There is currently no known solution to this general problem. Remedial measures are available, such as forcing all RSIP hosts to always use public IP addresses, even when communicating only on to other RSIP hosts. However, this can result in a socket set up between two RSIP hosts having the same source and destination IP addresses, which most TCP/IP stacks will consider as intra-host communication.

6.8. Scalability

The scalability of RSIP is currently not well understood. While it is conceivable that a single RSIP gateway could support hundreds of RSIP hosts, scalability depends on the specific deployment scenario and applications used. In particular, three major constraints on scalability will be (1) RSIP gateway processing requirements, (2) RSIP gateway memory requirements, and (3) RSIP negotiation protocol traffic requirements. It is advisable that all RSIP negotiation protocol implementations attempt to minimize these requirements.

7. Security Considerations

RSIP, in and of itself, does not provide security. It may provide the illusion of security or privacy by hiding a private address space, but security can only be ensured by the proper use of security protocols and cryptographic techniques.

8. Acknowledgements

The authors would like to thank Pyda Srisuresh, Dan Nessett, Gary Jaszewski, Ajay Bakre, Cyndi Jung, and Rick Cobb for their input. The IETF NAT working group as a whole has been extremely helpful in the ongoing development of RSIP.

9. References

[DHCP-DNS] Stapp, M. and Y. Rekhter, "Interaction Between DHCP and DNS", Work in Progress.

[RFC2983] Black, D., "Differentiated Services and Tunnels", RFC 2983, October 2000.

[RFC3104] Montenegro, G. and M. Borella, "RSIP Support for End-to-End IPSEC", RFC 3104, October 2001.

[RFC3103] Borella, M., Grabelsky, D., Lo, J. and K. Taniguchi, "Realm Specific IP: Protocol Specification", RFC 3103, October 2001.

[RFC2746] Terzis, A., Krawczyk, J., Wroclawski, J. and L. Zhang, "RSVP Operation Over IP Tunnels", RFC 2746, January 2000.

[RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G.J. and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.

[RFC2002] Perkins, C., "IP Mobility Support", RFC 2002, October 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to indicate requirement levels", BCP 14, RFC 2119, March 1997.

[RFC2663] Srisuresh, P. and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, August 1999.

[RFC2205] Braden, R., Zhang, L., Berson, S., Herzog, S. and S. Jamin, "Resource Reservation Protocol (RSVP)", RFC 2205, September 1997.

[RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z. and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.

[RFC3105] Kempf, J. and G. Montenegro, "Finding an RSIP Server with SLP", RFC 3105, October 2001.

10. Authors' Addresses

Michael Borella
CommWorks
3800 Golf Rd.
Rolling Meadows IL 60008

Phone: (847) 262-3083
EMail: mike_borella@commworks.com

Jeffrey Lo
Candlestick Networks, Inc
70 Las Colinas Lane,
San Jose, CA 95119

Phone: (408) 284 4132
EMail: yidarlo@yahoo.com

David Grabelsky
CommWorks
3800 Golf Rd.
Rolling Meadows IL 60008

Phone: (847) 222-2483
EMail: david_grabelsky@commworks.com

Gabriel E. Montenegro
Sun Microsystems
Laboratories, Europe
29, chemin du Vieux Chene
38240 Meylan
FRANCE

Phone: +33 476 18 80 45
EMail: gab@sun.com

11. Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.

Network Working Group
Request for Comments: 3103
Category: Experimental

M. Borella
D. Grabelsky
CommWorks
J. Lo
Candlestick Networks
K. Taniguchi
NEC USA
October 2001

Realm Specific IP: Protocol Specification

Status of this Memo

This memo defines an Experimental Protocol for the Internet community. It does not specify an Internet standard of any kind. Discussion and suggestions for improvement are requested. Distribution of this memo is unlimited.

Copyright Notice

IESG Note

The IESG notes that the set of documents describing the RSIP technology imply significant host and gateway changes for a complete implementation. In addition, the floating of port numbers can cause problems for some applications, preventing an RSIP-enabled host from interoperating transparently with existing applications in some cases (e.g., IPsec). Finally, there may be significant operational complexities associated with using RSIP. Some of these and other complications are outlined in section 6 of the RFC 3102, as well as in the Appendices of RFC 3104. Accordingly, the costs and benefits of using RSIP should be carefully weighed against other means of relieving address shortage.

Abstract

This document presents a protocol with which to implement Realm Specific IP (RSIP). The protocol defined herein allows negotiation of resources between an RSIP host and gateway, so that the host can lease some of the gateway's addressing parameters in order to establish a global network presence. This protocol is designed to operate on the application layer and to use its own TCP or UDP port. In particular, the protocol allows a gateway to allocate addressing and control parameters to a host such that a flow policy can be enforced at the gateway.

Table of Contents

1. Introduction

Network Address Translation (NAT) has gained popularity as a method of separating public and private address spaces, and alleviating network address shortages. A NAT translates the addresses of packets leaving a first routing realm to an address from a second routing realm, and performs the reverse function for packets entering the first routing realm from the second routing realm. This translation is performed transparently to the hosts in either space, and may include modification of TCP/UDP port numbers and IP addresses in packets that traverse the NAT.

While a NAT does not require hosts to be aware of the translation, it will require an application layer gateway (ALG) for any protocol that transmits IP addresses or port numbers in packet payloads (such as FTP). Additionally, a NAT will not work with protocols that require IP addresses and ports to remain unmodified between the source and destination hosts, or protocols that prevent such modifications from occurring (such as some IPsec modes, or application-layer end-to-end encryption).

An alternative to a NAT is an architecture that allows the hosts within the first (e.g., private) routing realm to directly use addresses and other routing parameters from the second (e.g., public) routing realm. Thus, RSIP [RSIP-FRAME] has been defined as a method for address sharing that exhibits more transparency than NAT. In particular, RSIP requires that an RSIP gateway (a router or gateway between the two realms) assign at least one address from the second routing realm, and perhaps some other resources, to each RSIP host. An RSIP host is a host in the first routing realm that needs to establish end-to-end connectivity to a host, entity or device in the second routing realm. Thus, the second routing realm is not directly accessible from the RSIP host, but this system allows packets to maintain their integrity from the RSIP host to their destination. ALGs are not required in the RSIP gateway.

RSIP requires that hosts be modified so that they place some number of layer three, layer four or other values from those assigned by the RSIP gateway in each packet bound for the second routing realm.

This document discusses a method for assigning parameters to an RSIP host from an RSIP gateway. The requirements, scope, and applicability of RSIP, as well as its interaction with other layer 3 protocols, are discussed in a companion framework document [RSIP-FRAME]. Extensions to this protocol that enable end-to-end IPsec are discussed in [RSIP-IPSEC].

2. Specification of Requirements

The keywords "MUST", "MUST NOT", "REQUIRED", "SHOULD", "SHOULD NOT", "SHALL", "SHALL NOT", "MAY" and "MAY NOT" that appear in this document are to be interpreted as described in [RFC2119].

3. Terminology

Private Realm

A routing realm that uses private IP addresses from the ranges (10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16) specified in [RFC1918], or addresses that are non-routable from the Internet.

Public Realm

A routing realm with unique network addresses assigned by the Internet Assigned Number Authority (IANA) or an equivalent address registry.

RSIP Host

A host within the private realm that acquires publicly unique parameters from an RSIP gateway through the use of the RSIP client/server protocol.

RSIP Gateway

A router situated on the boundary between a private realm and a public realm and owns one or more public IP addresses. An RSIP gateway is responsible for public parameter management and assignment to RSIP hosts. An RSIP gateway may act as a NAT router for hosts within the private realm that are not RSIP enabled.

RSIP Client

An application program that performs the client portion of the RSIP client/server protocol. An RSIP client application MUST exist on all RSIP hosts, and MAY exist on RSIP gateways.

RSIP Server

An application program that performs the server portion of the RSIP client/server protocol. An RSIP server application MUST exist on all RSIP gateways.

RSA-IP: Realm Specific Address IP

An RSIP method in which each RSIP host is allocated a unique IP address from the public realm. Discussed in detail in [RSIP-FRAME]

RSAP-IP: Realm Specific Address and Port IP

An RSIP method in which each RSIP host is allocated an IP address (possibly shared with other RSIP hosts) and some number of per-address unique ports from the public realm. Discussed in detail in [RSIP-FRAME]

Binding

An association of some combination of a local address, one or more local ports, a remote address, and a remote port with an RSIP host.

Resource

A general way to refer to an item that an RSIP host leases from an RSIP gateway; e.g., an address or port.

All other terminology found in this document is consistent with that of [RFC2663] and [RSIP-FRAME].

4. Architecture

For simplicity, in the remainder of this document we will assume that the RSIP hosts in the first routing realm (network) use private (e.g., see [RFC1918]) IP addresses, and that the second routing realm (network) uses public IP addresses. (This assumption is made without loss of generality and the ensuing discussion applies to more general cases.) The RSIP gateway connects the public and private realms and contains interfaces to both. Other NAT terminology found in this document is defined in [RFC2663].

The diagram below describes an exemplary reference architecture for RSIP.

```
   RSIP Host              RSIP Gateway                    Host

      Xa                     Na   Nb                       Yb
   [X]------( Addr sp. A )----[N]-----( Addr sp. B )-------[Y]
            (  Network   )            (  Network   )
```

Hosts X and Y belong to different addressing realms A and B, respectively, and N is an RSIP gateway (which may also perform NAT functions). N has two interfaces: Na on address space A, and Nb on address space B. N may have a pool of addresses in address space B which it can assign to or lend to X and other hosts in address space

A. These addresses are not shown above, but they can be denoted as Nb1, Nb2, Nb3 and so on.

Host X, needing to establish an end-to-end connection to a network entity Y situated within address space B, first negotiates and obtains assignment of the resources from the RSIP gateway. Upon assignment of these parameters, the RSIP gateway creates a mapping, of X's addressing information and the assigned resources. This binding enables the RSIP gateway to correctly de-multiplex and forward inbound traffic generated by Y for X. A lease time is associated with each bind.

Using the public parameters assigned by the RSIP gateway, RSIP hosts tunnel data packets across address space A to the RSIP gateway. The RSIP gateway acts as the end point of such tunnels, stripping off the outer headers and routing the inner packets onto the public realm. As mentioned above, an RSIP gateway maintains a mapping of the assigned public parameters as demultiplexing fields for uniquely mapping them to RSIP host private addresses. When a packet from the public realm arrives at the RSIP gateway and it matches a given set of demultiplexing fields, then the RSIP gateway will tunnel it to the appropriate RSIP host. The tunnel headers of outbound packets from X to Y, given that X has been assigned Nb, are as follows:

```
+---------+---------+---------+
| X -> Na | Nb -> Y | payload |
+---------+---------+---------+
```

There are two basic flavors of RSIP: RSA-IP and RSAP-IP. RSIP hosts and gateways MUST support RSAP-IP and MAY support RSA-IP. Details of RSA-IP and RSAP-IP are found in [RSIP-FRAME].

5. Transport Protocol

RSIP is an application layer protocol that requires the use of a transport layer protocol for end-to-end delivery of packets.

RSIP gateways MUST support TCP, and SHOULD support UDP. Due to the fact that RSIP may be deployed across a wide variety of network links, RSIP hosts SHOULD support TCP, because of TCP's robustness across said variety of links. However, RSIP hosts MAY support UDP instead of TCP, or both UDP and TCP.

For RSIP hosts and gateways using UDP, timeout and retransmissions MUST occur. We recommend a binary exponential backoff scheme with an initial duration of 12.5 ms, and a maximum of six retries (seven total attempts before failure). However, these parameters MAY be adjusted or tuned for specific link types or scenarios.

Once a host and gateway have established a registration using either TCP or UDP, they may not switch between the two protocols for the duration of the registration. The decision of whether to use TCP or UDP is made by the client, and is determined by the transport protocol of the first packet sent by a client in a successful registration procedure.

6. Host / Gateway Relationships

An RSIP host can be in exactly one of three fundamental relationships with respect to an RSIP gateway:

Unregistered: The RSIP gateway does not know of the RSIP host's existence, and it will not forward or deliver globally addressed packets on behalf of the host. The only valid RSIP-related action for an RSIP host to perform in this state is to request registration with an RSIP gateway.

Registered: The RSIP gateway knows of the RSIP host and has assigned it a client ID and has specified the flow policies that it requires of the host. However, no resources, such as addresses or ports, have been allocated to the host, and the gateway will not forward or deliver globally addressed packets on behalf of the host. All registrations have an associated lease time. If this lease time expires, the RSIP host automatically reverts to the unregistered state.

Assigned: The RSIP gateway has granted one or more bindings of resources to the host. The gateway will forward and deliver globally addressed packets on behalf of the host. Each binding has an associated lease time. If this lease time expires, the binding is automatically revoked.

Architectures in which an RSIP host is simultaneously registered with more than one RSIP gateway are possible. In such cases, an RSIP host may be in different relationships with different RSIP gateways at the same time.

An RSIP gateway MAY redirect an RSIP host to use a tunnel endpoint for data traffic that is not the RSIP gateway itself, or perhaps is a different interface on the RSIP gateway. This is done by specifying the tunnel endpoint's address as part of an assignment. In such an architecture, it is desirable (though not necessary) for the RSIP gateway to have a method with which to notify the tunnel endpoint of assignments, and the expiration status of these assignments.

Lease times for bindings and registrations are managed as follows. All lease times are given in units of seconds from the current time, indicating a time in the future at which the lease will expire. These expiration times are used in the ensuing discussion.

An initial expiration time (R) is given to a registration. Under this registration, multiple bindings may be established, each with their own expiration times (B1, B2, ...). When each binding is established or extended, the registration expiration time is adjusted so that the registration will last at least as long as the longest lease. In other words, when binding Bi is established or extended, the following calculation is performed: R = max(R, Bi).

Under this scheme, a registration will never expire while any binding's lease is still valid. However, a registration may expire when the last binding's lease expires, or at some point thereafter.

7. Gateway Flow Policy and State

Since an RSIP gateway is likely to reside on the boundary between two or more different administrative domains, it is desirable to enable an RSIP gateway to be able to enforce flow-based policy. In other words, an RSIP gateway should have the ability to explicitly control which local addresses and ports are used to communicate with remote addresses and ports.

In the following, macro-flow policy refers to controlling flow policy at the granularity level of IP addresses, while micro-flow policy refers to controlling flow policy at the granularity of IP address and port tuples. Of course there may be no policy at all, which indicates that the RSIP gateway does not care about the flow parameters used by RSIP hosts. We consider two levels of local flow policy and three levels of remote flow policy.

7.1. Local Flow Policy

Local flow policy determines the granularity of control that an RSIP gateway has over the local addressing parameters that an RSIP host uses for particular sessions.

Since an RSIP host must use at least an IP address allocated by the gateway, the loosest level of local flow policy is macro-flow based. Under local macro-flow policy, an RSIP host is allocated an IP address (RSA-IP) or an IP address and one or more ports to use with it (RSAP-IP). However, the host may use the ports as it desires for establishing sessions with public hosts.

Under micro-flow policy, a host is allocated exactly one port at a time. The host may request more ports, also one at a time. This policy gives the gateway very tight control over local port use, although it affords the host less flexibility.

Note that only local macro-flow policy can be used with RSA-IP, while either local macro-flow or local micro-flow policy may be used with RSAP-IP.

Examples of how RSIP flow policy operates are given in Appendix C.

7.2. Remote Flow Policy

Remote flow policy determines the granularity of control that an RSIP gateway has over the remote (public) hosts with which an RSIP host communicates. In particular, remote flow policy dictates what level of detail that a host must specify addressing parameters of a remote host or application before the RSIP gateway allows the host to communicate with that host or application.

The simplest and loosest form of flow policy is no policy at all. In other words, the RSIP gateway allocates addressing parameters to the host, and the host may use these parameters to communicate with any remote host, without explicitly notifying the gateway.

Macro-flow policy requires that the host identify the remote address of the host that it wishes to communicate with as part of its request for local addressing parameters. If the request is granted, the host MUST use the specified local parameters only with the remote address specified, and MUST NOT communicate with the remote address using any local parameters but the ones allocated. However, the host may contact any port number at the remote host without explicitly notifying the gateway.

Micro-flow policy requires that the host identify the remote address and port of the host that it wishes to communicate with as part of its request for local addressing parameters. If the request is granted, the host MUST use the specified local parameters only with the remote address and port specified, and MUST NOT communicate with the remote address and port using any local parameters but the ones allocated.

Remote flow policy is implemented in both the ingress and egress directions, with respect to the location of the RSIP gateway.

7.3. Gateway State

An RSIP gateway must maintain state for all RSIP hosts and their assigned resources. The amount and type of state maintained depends on the local and remote flow policy. The required RSIP gateway state will vary based on the RSIP method, but will always include the chosen method's demultiplexing parameters.

7.3.1. RSA-IP State

An RSIP gateway serving an RSIP host using the RSA-IP method MUST maintain the following minimum state to ensure proper mapping of incoming packets to RSIP hosts:

- Host's private address
- Host's assigned public address(es)

7.3.2. RSAP-IP State

An RSIP gateway serving an RSIP host using the RSAP-IP method MUST maintain the following minimum state to ensure proper mapping of incoming packets to RSIP hosts:

- Host's private address
- Host's assigned public address(es)
- Host's assigned port(s) per address

7.3.3. Flow State

Regardless of whether the gateway is using RSA-IP or RSAP-IP, additional state is necessary if either micro-flow based or macro-flow based remote policy is used.

If the gateway is using macro-flow based remote policy, the following state must be maintained:

- Remote host's address

If the gateway is using micro-flow based remote policy, the following state must be maintained:

- Remote host's address
- Remote host's port

More state MAY be used by an RSIP gateway if desired. For example, ToS/DS bytes may be recorded in order to facilitate quality of service support.

8. Parameter Specification and Formats

In this section we define the formats for RSIP parameters. Each RSIP message contains one or more parameters that encode the information passed between the host and gateway. The general format of all parameters is TLV (type-length-value) consisting of a 1-byte type followed by a 2-byte length followed by a 'length' byte value as shown below.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |             Length            |     Value     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Value ...
+-+-+-+-+-+-+-+-+-+-+-+-+
```

The value field may be divided into a number of other fields as per the type of the parameter. Note that the length field encodes the number of bytes in the value field, NOT the overall number of bytes in the parameter.

8.1. Address

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Type = 1   |             Length            |    Addrtype   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Address...
+-+-+-+-+-+-+-+-+-+-+-+-+
```

The address parameter contains addressing information, either an IPv4 address or netmask, an IPv6 address or netmask, or a fully qualified domain name (FQDN). The Addrtype field is 1 byte in length, indicating the type of address.

	Addrtype	Length of address field (in bytes)
0	Reserved	0
1	IPv4	4
2	IPv4 netmask	4
3	IPv6	16
4	FQDN	varies

For FQDN (Fully qualified domain name), the length of the address field will be one less than the value of the length field, and the name will be represented as an ASCII string (no terminating character).

In some cases, it is necessary to specify a "don't care" value for an address. This is signified by a setting the length field to 1 and omitting the value field.

It is not valid for a host to request an address with an FQDN type as its local address (See specification of ASSIGN_REQUEST_RSA-IP and ASSIGN_REQUEST_RSAP-IP, below).

8.2. Ports

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Type = 2   |             Length            |     Number    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Port number         |  ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The ports parameter encodes zero or more TCP or UDP ports. When a single port is specified, the value of the number field is 1 and there is one port field following the number field. When more than one port is specified, the value of the number field will indicate the total number of ports contained, and the parameter may take one of two forms. If there is one port field, the ports specified are considered to be contiguous starting at the port number specified in the port field. Alternatively, there may be a number of port fields equal to the value of the number field. The number of port fields can be extrapolated from the length field.

In some cases, it is necessary to specify a don't care value for one or more ports (e.g., when a client application is using ephemeral source ports). This is accomplished by setting the length field to 1, setting the number field to the number of ports necessary, and omitting all port fields. The value of the number field MUST be greater than or equal to one.

If micro-flow based policy applies to a given ports parameter, it MUST contain exactly one port field.

8.3. Lease Time

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |    Type = 3   |           Length = 4          |  Lease time   |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                   Lease time                  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The lease time parameter specifies the length, in seconds, of an RSIP host registration or parameter binding.

8.4. Client ID

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |    Type = 4   |           Length = 4          |   Client ID   |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                    Client ID                  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The client ID parameter specifies an RSIP client ID. Client ID's by an RSIP gateway to differentiate RSIP hosts.

8.5. Bind ID

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |    Type = 5   |           Length = 4          |    Bind ID    |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                     Bind ID                   |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The bind ID parameter specifies an RSIP bind ID. Bind ID's are used by RSIP hosts and gateways to differentiate an RSIP host's bindings.

8.6. Tunnel Type

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Type = 6   |           Length = 1          |  Tunnel type  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The tunnel type parameter specifies the type of tunnel used between an RSIP host and an RSIP gateway. Defined tunnel types are:

```
          Tunnel Type
          -----------
   0      Reserved
   1      IP-IP
   2      GRE
   3      L2TP
```

8.7. RSIP Method

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Type = 7   |           Length = 1          |  RSIP method  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The RSIP method parameter specifies an RSIP method. Defined RSIP methods are:

```
          RSIP method
          -----------
   0      Reserved
   1      RSA-IP
   2      RSAP-IP
```

8.8. Error

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Type = 8   |           Length = 2          |     Error     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Error     |
+-+-+-+-+-+-+-+-+
```

The error parameter specifies an error. The currently defined error values are presented in Appendix A.

8.9. Flow Policy

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Type = 9   |           Length = 2          |     Local     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Remote     |
+-+-+-+-+-+-+-+-+
```

The flow policy parameter specifies both the local and remote flow policy.

Defined local flow policies are:

```
          Local Flow Policy
          -----------------
   0      Reserved
   1      Macro flows
   2      Micro flows
```

Defined remote flow policies are:

```
          Remote Flow Policy
          ------------------
   0      Reserved
   1      Macro flows
   2      Micro flows
   3      No policy
```

8.10. Indicator

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Type = 10  |           Length = 2          |     Value     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Value     |
+-+-+-+-+-+-+-+-+
```

An indicator parameter is a general-purpose parameter, the use of which is defined by the message that it appears in. An RSIP message that uses an indicator parameter MUST define the meaning and interpretation of all of the indicator's possible values.

8.11. Message Counter

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Type = 11  |             Length = 4        |    Counter    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Counter                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

A message counter parameter is used to mark RSIP messages with sequentially-increasing values. Message counters MUST be used with UDP, in order to facilitate reliability.

8.12. Vendor Specific Parameter

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Type = 12  |               Length          |   Vendor ID   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Vendor ID  |               Subtype         |   Value...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The vendor specific parameter is used to encode parameters that are defined by a particular vendor. The vendor ID field is the vendor-specific ID assigned by IANA. Subtypes are defined and used by each vendor as necessary. An RSIP host or gateway SHOULD silently ignore vendor-specific messages that it does not understand.

9. Message Types

RSIP messages consist of three mandatory fields, version, message type, and overall length, followed by one or more required parameters, followed in turn by zero or more optional parameters. In an RSIP message, all required parameters MUST appear in the exact order specified below. Optional parameters MAY appear in any order. Message format is shown below:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Version    | Message type  |         Overall length        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Parameters...
+-+-+-+-+-+-+-+-+-+-+-+
```

The version number field is a single byte and specifies the RSIP version number that is being used. The current RSIP version number is 1.

The message type field is a single byte and specifies the message contained in the current packet. There may be only one message per packet. Message types are given numerical assignments in Appendix B.

The overall length field is two bytes and contains the number of bytes in the RSIP message, including the three mandatory fields.

Most parameters are only allowed to appear once in each message. The exceptions are as follows:

- Multiple address parameters MUST appear in ASSIGN_REQUEST_RSA-IP, ASSIGN_RESPONSE_RSA-IP, ASSIGN_REQUEST_RSAP-IP, ASSIGN_RESPONSE_RSAP-IP, LISTEN_REQUEST and LISTEN_RESPONSE.

- Multiple ports parameters MUST appear in ASSIGN_REQUEST_RSAP-IP, ASSIGN_RESPONSE_RSAP-IP, LISTEN_REQUEST and LISTEN_RESPONSE.

- Multiple RSIP method and tunnel type parameters MAY appear in RESISTER_RESPONSE.

- Multiple address parameters and multiple indicator parameters MAY appear in QUERY_REQUEST and QUERY_RESPONSE.

The following message types are defined in BNF. Required parameters are enclosed in <> and MUST appear. Optional parameters are enclosed in [] and MAY appear. Not all message types need to be implemented in order to be RSIP compliant. For example, an RSIP host and/or gateway may not support LISTEN_REQUEST and LISTEN_RESPONSE, or may only support RSAP-IP and not RSA-IP.

9.1. ERROR_RESPONSE

9.1.1. Description

An ERROR_RESPONSE is used to provide error messages from an RSIP gateway to an RSIP host. Usually, errors indicate that the RSIP gateway cannot or will not perform an action or allocate resources on behalf of the host. If the error is related to a particular client ID or bind ID, these associated parameters MUST be included. Multiple errors MAY NOT be reported in the same ERROR_RESPONSE. In situations where more than one error has occurred, the RSIP gateway MUST choose only one error to report.

9.1.2. Format

```
<ERROR_RESPONSE> ::= <Version>
                     <Message Type>
                     <Overall Length>
                     <Error>
                     [Message Counter]
                     [Client ID]
                     [Bind ID]
```

9.1.3. Behavior

An ERROR_RESPONSE message MUST only be transmitted by an RSIP gateway. An RSIP host that detects an error in a message received from an RSIP gateway MUST silently discard the message. There are no error conditions that can be caused by an ERROR_RESPONSE. An ERROR_RESPONSE is typically transmitted in response to a request from an RSIP host, but also may be transmitted asynchronously by an RSIP gateway.

9.2. REGISTER_REQUEST

9.2.1. Description

The REGISTER_REQUEST message is used by an RSIP host to establish registration with an RSIP gateway. An RSIP host MUST register before it requests resources or services from an RSIP gateway. Once an RSIP host has registered with an RSIP gateway, it may not register again until it has de-registered from that gateway.

9.2.2. Format

```
<REGISTER_REQUEST> ::= <Version>
                       <Message Type>
                       <Overall Length>
                       [Message Counter]
```

9.2.3. Behavior

The following message-specific error conditions exist:

- If the host is already registered with the gateway, the gateway MUST respond with an ERROR_RESPONSE containing the ALREADY_REGISTERED error and the RSIP host's client ID.

- If the gateway's policy will not allow the host to register, the gateway MUST respond with an ERROR_RESPONSE containing the REGISTRATION_DENIED error.

9.3. REGISTER_RESPONSE

9.3.1. Description

The REGISTER_RESPONSE message is used by an RSIP gateway to confirm the registration of an RSIP host, and to provide a client ID, flow policy, and possibly a message counter and one or more RSIP methods and/or tunnel types.

9.3.2. Format

```
<REGISTER_RESPONSE> ::= <Version>
                        <Message Type>
                        <Overall Length>
                        <Client ID>
                        <Lease time>
                        <Flow Policy>
                        [Message Counter]
                        [RSIP Method]...
                        [Tunnel Type]...
```

9.3.3. Behavior

An RSIP gateway MUST assign a different client ID to each host that is simultaneously registered with it. The RSIP gateway MAY respond with one or more RSIP methods and tunnel types that it supports. If an RSIP method is not specified, RSAP-IP MUST be assumed. If a tunnel type is not specified, IP-IP MUST be assumed.

9.4. DE-REGISTER_REQUEST

9.4.1. Description

The DE-REGISTER_REQUEST message is used by an RSIP host to de-register with an RSIP gateway. If a host de-registers from the assigned state, all of the host's bindings are revoked. The host SHOULD NOT de-register from the unregistered state.

9.4.2. Format

```
<DE-REGISTER_REQUEST> ::= <Version>
                          <Message Type>
                          <Overall Length>
                          <Client ID>
                          [Message Counter]
```

9.4.3. Behavior

The following message-specific error conditions exist:

- If the host is not registered with the gateway, the gateway MUST respond with an ERROR_RESPONSE containing the REGISTER_FIRST error.

- If the message contains an incorrect client ID, the gateway MUST respond with an ERROR_RESPONSE containing the BAD_CLIENT_ID error.

If there are no errors that result from this message, the gateway MUST respond with an appropriate DE-REGISTER_RESPONSE. Upon de-registering a host, an RSIP gateway must delete all binds associated with that host and return their resources to the pool of free resources. Once a host has de-registered, it may not use any of the RSIP gateway's resources without registering again.

9.5. DE-REGISTER_RESPONSE

9.5.1. Description

The DE-REGISTER_RESPONSE message is used by an RSIP gateway to confirm the de-registration of an RSIP host or to force an RSIP host to relinquish all of its bindings and terminate its relationship with the RSIP gateway. Upon receiving a DE-REGISTER_RESPONSE message, an RSIP host MUST stop all use of the resources that have been allocated to it by the gateway.

9.5.2. Format

```
<DE-REGISTER_RESPONSE> ::= <Version>
                           <Message Type>
                           <Overall Length>
                           <Client ID>
                           [Message Counter]
```

9.5.3. Behavior

An RSIP gateway MUST send a DE-REGISTER_RESPONSE in response to a valid DE-REGISTER_REQUEST. An RSIP gateway MUST send a DE-REGISTER_RESPONSE to an RSIP host when that host's registration lease time times out. An RSIP gateway SHOULD send a DE-REGISTER_RESPONSE if it detects that it will no longer be able to perform RSIP functionality for a given host. An RSIP host MUST be ready to accept a DE-REGISTER_RESPONSE at any moment.

9.6. ASSIGN_REQUEST_RSA-IP

9.6.1. Description

The ASSIGN_REQUEST_RSA-IP message is used by an RSIP host to request resources to use with RSA-IP. Note that RSA-IP cannot be used in combination with micro-flow based local policy.

9.6.2. Format

```
<ASSIGN_REQUEST_RSA-IP> ::= <Version>
                            <Message Type>
                            <Overall Length>
                            <Client ID>
                            <Address (local)>
                            <Address (remote)>
                            <Ports (remote)>
                            [Message Counter]
                            [Lease Time]
                            [Tunnel Type]
```

9.6.3. Behavior

The RSIP host specifies two address parameters. The RSIP host may request a particular local address by placing that address in the first address parameter. To indicate that it has no preference for local address, the RSIP host may place a "don't care" value in the address parameter.

If macro-flow based remote policy is used, the host MUST specify the remote address that it will use this binding (if granted) to contact; however, the remote port number MAY remain unspecified. If micro-flow based remote policy is used, the host MUST specify the remote address and port number that it will use this binding (if granted) to contact. If no flow policy is used, the RSIP host may place a "don't care" value in the value fields of the respective address and ports parameters.

The following message-specific error conditions exist:

- If the host is not registered with the gateway, the gateway MUST respond with an ERROR_RESPONSE containing the REGISTER_FIRST error.

- If the message contains an incorrect client ID, the gateway MUST respond with an ERROR_RESPONSE containing the BAD_CLIENT_ID error.

- If the local address parameter is a don't care value and the RSIP gateway cannot allocate ANY addresses, the RSIP gateway MUST respond with an ERROR_RESPONSE containing the LOCAL_ADDR_UNAVAILABLE error.

- If the local address parameter is not a don't care value there are three possible error conditions:

  o If the RSIP gateway cannot allocate ANY addresses, it MUST respond with an ERROR_RESPONSE containing the LOCAL_ADDR_UNAVAILABLE error.

  o If the RSIP gateway cannot allocate the requested address because it is in use, the RSIP gateway MUST respond with an ERROR_RESPONSE containing the LOCAL_ADDR_INUSE error.

  o If the RSIP gateway cannot allocate the requested address because it is not allowed by policy, the RSIP gateway MUST respond with an ERROR_RESPONSE containing the LOCAL_ADDR_UNALLOWED error.

- If macro-flow based remote policy is used and the requested remote address is not allowed by the RSIP gateway's policy, the RSIP gateway MUST respond with an ERROR_RESPONSE containing the REMOTE_ADDR_UNALLOWED error.

- If micro-flow based remote policy is used and the requested remote address / port pair is not allowed by the RSIP gateway's policy, the RSIP gateway MUST respond with an ERROR_RESPONSE containing the REMOTE_ADDRPORT_UNALLOWED error.

- If an unsupported or unallowed tunnel type is specified, the RSIP gateway MUST respond with an ERROR_RESPONSE containing the BAD_TUNNEL_TYPE error.

- If the host has not specified local or remote address or port information in enough detail, the RSIP gateway MUST respond with an ERROR_RESPONSE containing the FLOW_POLICY_VIOLATION error.

9.7. ASSIGN_RESPONSE_RSA-IP

9.7.1. Description

The ASSIGN_RESPONSE_RSA-IP message is used by an RSIP gateway to deliver parameter assignments to an RSIP host using RSA-IP. A host-wise unique bind ID, lease time, and tunnel type must be provided for every assignment.

9.7.2. Format

```
<ASSIGN_RESPONSE_RSA-IP> ::= <Version>
                             <Message Type>
                             <Overall Length>
                             <Client ID>
                             <Bind ID>
                             <Address (local)>
                             <Address (remote)>
                             <Ports (remote)>
                             <Lease Time>
                             <Tunnel Type>
                             [Address (tunnel endpoint)]
                             [Message Counter]
```

9.7.3. Behavior

If no remote flow policy is used, the RSIP gateway MUST use "don't care" values for the remote address and ports parameters. If macro-flow based remote policy is used, the remote address parameter MUST contain the address specified in the associated request, and the remote ports parameter MUST contain a "don't care" value. If micro-flow based remote policy is used, the remote address and remote ports parameters MUST contain the address and port information specified in the associated request.

If the host detects an error or otherwise does not "understand" the gateway's response, it SHOULD send a FREE_REQUEST with the bind ID from the said ASSIGN_RESPONSE_RSA-IP. This will serve to help synchronize the states of the host and gateway.

The address of a tunnel endpoint that is not the RSIP gateway MAY be specified. If this parameter is not specified, the RSIP gateway MUST be assumed to be the tunnel endpoint.

9.8. ASSIGN_REQUEST_RSAP-IP

9.8.1. Description

The ASSIGN_REQUEST_RSAP-IP message is used by an RSIP host to request resources to use with RSAP-IP. The RSIP host specifies two address and two port parameters, the first of each, respectively, refer to the local address and port(s) that will be used, and the second of each, respectively, refer to the remote address and port(s) that will be contacted.

9.8.2. Format

```
<ASSIGN_REQUEST_RSAP-IP> ::= <Version>
                             <Message Type>
                             <Overall Length>
                             <Client ID>
                             <Address (local)>
                             <Ports (local)>
                             <Address (remote)>
                             <Ports (remote)>
                             [Message Counter]
                             [Lease Time]
                             [Tunnel Type]
```

9.8.3. Behavior

An RSIP host may request a particular local address by placing that address in the value field of the first address parameter. The RSIP host may request particular local ports by placing them in the first port parameter. To indicate that it has no preference for local address or ports, the RSIP host may place a "don't care" value in the respective address or ports parameters.

If macro-flow based remote policy is used, the host MUST specify the remote address that it will use this binding (if granted) to contact; however, the remote port number(s) MAY remain unspecified. If micro-flow based remote policy is used, the host MUST specify the remote address and port number(s) that it will use this binding (if granted) to contact. If no flow policy is used, the RSIP host may place a value of all 0's in the value fields of the respective address or port parameters.

The following message-specific error conditions exist:

- If the host is not registered with the gateway, the gateway MUST respond with an ERROR_RESPONSE containing the REGISTER_FIRST error.

- If the message contains an incorrect client ID, the gateway MUST respond with an ERROR_RESPONSE containing the BAD_CLIENT_ID error.

- If the local address parameter is a don't care value and the RSIP gateway cannot allocate ANY addresses, the RSIP gateway MUST respond with an ERROR_RESPONSE containing the LOCAL_ADDR_UNAVAILABLE error.

- If the local address parameter is not a don't care value there are five possible error conditions:

  o If the RSIP gateway cannot allocate ANY addresses, it MUST respond with an ERROR_RESPONSE containing the LOCAL_ADDR_UNAVAILABLE error.

  o If the RSIP gateway cannot allocate the requested address because it is in use, the RSIP gateway MUST respond with an ERROR_RESPONSE containing the LOCAL_ADDR_INUSE error.

  o If the RSIP gateway cannot allocate the requested address because it is not allowed by policy, the RSIP gateway MUST respond with an ERROR_RESPONSE containing the LOCAL_ADDR_UNALLOWED error.

  o If the RSIP gateway cannot allocate a requested address / port tuple because it is in use, the RSIP gateway MUST respond with an ERROR_RESPONSE containing the LOCAL_ADDRPORT_INUSE error.

  o If the RSIP gateway cannot allocate a requested address / port tuple because it is not allowed by policy, the RSIP gateway MUST respond with an ERROR_RESPONSE containing the LOCAL_ADDRPORT_UNALLOWED error.

- If the RSIP host requests a number of ports (greater that one), but does not specify particular port numbers (i.e., uses "don't care" values) the RSIP gateway cannot grant the entire request, the RSIP gateway MUST return an ERROR_RESPONSE containing the LOCAL_ADDRPORT_UNAVAILABLE error.

- If macro-flow based remote policy is used and the requested remote address is not allowed by the RSIP gateway's policy, the RSIP gateway MUST respond with an ERROR_RESPONSE containing the REMOTE_ADDR_UNALLOWED error.

- If micro-flow based remote policy is used and the requested remote address / port pair is not allowed by the RSIP gateway's policy, the RSIP gateway MUST respond with an ERROR_RESPONSE containing the REMOTE_ADDRPORT_UNALLOWED error.

- If an unsupported or unallowed tunnel type is specified, the RSIP gateway MUST respond with an ERROR_RESPONSE containing the BAD_TUNNEL_TYPE error.

- If the host has not specified local or remote address or port information in enough detail, the RSIP gateway MUST respond with an ERROR_RESPONSE containing the FLOW_POLICY_VIOLATION error.

9.9. ASSIGN_RESPONSE_RSAP-IP

9.9.1. Description

The ASSIGN_RESPONSE_RSAP-IP message is used by an RSIP gateway to deliver parameter assignments to an RSIP host. A host-wise unique bind ID, lease time, and tunnel type must be provided for every assignment.

9.9.2. Format

```
<ASSIGN_RESPONSE_RSAP-IP> ::= <Version>
                              <Message Type>
                              <Overall Length>
                              <Client ID>
                              <Bind ID>
                              <Address (local)>
                              <Ports (local)>
                              <Address (remote)>
                              <Ports (remote)>
                              <Lease Time>
                              <Tunnel Type>
                              [Address (tunnel endpoint)]
                              [Message Counter]
```

9.9.3. Behavior

Regardless of local flow policy, a local address and port(s) MUST be assigned to the host. If macro-flow based local policy is used, the host is assigned an address and one or more ports. If micro-flow based local policy is used, the host is assigned an address and exactly one port.

If no remote flow policy is used, the RSIP gateway MUST use "don't care" values for the remote address and ports parameters. If macro-flow based remote policy is used, the remote address parameter MUST contain the address specified in the associated request, and the remote ports parameter must contain a "don't care" value. If micro-flow based remote policy is used, the remote address and remote ports parameters MUST contain the address and port information specified in the associated request.

If the host detects an error or otherwise does not "understand" the gateway's response, it SHOULD send a FREE_REQUEST with the bind ID from the said ASSIGN_RESPONSE_RSAP-IP. This will serve to help synchronize the states of the host and gateway.

The address of a tunnel endpoint that is not the RSIP gateway MAY be specified. If this parameter is not specified, the RSIP gateway MUST be assumed to be the tunnel endpoint.

9.10. EXTEND_REQUEST

9.10.1. Description

The EXTEND_REQUEST message is used to request a lease extension to a current bind. It may be used with both RSA-IP and RSAP-IP. The host MUST specify its client ID and the bind ID in question, and it MAY suggest a lease time to the gateway.

9.10.2. Format

```
<EXTEND_REQUEST> ::= <Version>
                     <Message Type>
                     <Overall Length>
                     <Client ID>
                     <Bind ID>
                     [Lease Time]
                     [Message Counter]
```

9.10.3. Behavior

The following message-specific error conditions exist:

- If the host is not registered with the gateway, the gateway MUST respond with an ERROR_RESPONSE containing the REGISTER_FIRST error.

- If the message contains an incorrect client ID, the gateway MUST respond with an ERROR_RESPONSE containing the BAD_CLIENT_ID error.

- If the message contains an incorrect bind ID, the gateway MUST respond with an ERROR_RESPONSE containing the BAD_BIND_ID error.

If the RSIP gateway grants an extension to the host's lease, it MUST RESPOND with an appropriate EXTEND_RESPONSE message. If the lease is not renewed, the RSIP gateway MAY let it implicitly expire by doing nothing or make it explicitly expire by sending an appropriate FREE_RESPONSE message.

9.11. EXTEND_RESPONSE

9.11.1. Description

The EXTEND_RESPONSE message is used by an RSIP gateway to grant a requested lease extension. The gateway MUST specify the client ID of the host, the bind ID in question, and the new assigned lease time.

9.11.2. Format

```
<EXTEND_RESPONSE> ::= <Version>
                      <Message Type>
                      <Overall Length>
                      <Client ID>
                      <Bind ID>
                      <Lease Time>
                      [Message Counter]
```

9.11.3. Behavior

The RSIP gateway will determine lease time as per its local policy. The returned time is to be interpreted as the number of seconds before the lease expires, counting from the time at which the message is sent/received.

9.12. FREE_REQUEST

9.12.1. Description

The FREE_REQUEST message is used by an RSIP host to free a binding. The given bind ID identifies the bind to be freed. Resources may only be freed using the granularity of a bind ID.

9.12.2. Format

```
<FREE_REQUEST> ::= <Version>
                   <Message Type>
                   <Overall Length>
                   <Client ID>
                   <Bind ID>
                   [Message Counter]
```

9.12.3. Behavior

The following message-specific error conditions exist:

- If the host is not registered with the gateway, the gateway MUST respond with an ERROR_RESPONSE containing the REGISTER_FIRST error.

- If the message contains an incorrect client ID, the gateway MUST respond with an ERROR_RESPONSE containing the BAD_CLIENT_ID error.

- If the message contains an incorrect bind ID, the gateway MUST respond with an ERROR_RESPONSE containing the BAD_BIND_ID error.

If a host receives an error in response to a FREE_REQUEST, this may indicate that the host and gateway's states have become unsynchronized. Therefore, the host SHOULD make an effort to resynchronize, such as freeing resources then re-requesting them, or de-registering then re-registering.

9.13. FREE_RESPONSE

9.13.1. Description

The FREE_RESPONSE message is used by an RSIP gateway to acknowledge a FREE_REQUEST sent by an RSIP host, and to asynchronously deallocate resources granted to an RSIP host.

9.13.2. Format

```
<FREE_RESPONSE> ::= <Version>
                    <Message Type>
                    <Overall Length>
                    <Client ID>
                    <Bind ID>
                    [Message Counter]
```

9.13.3. Behavior

An RSIP host must always be ready to accept a FREE_RESPONSE, even if its lease on the specified bind ID is not yet expired.

9.14. QUERY_REQUEST

9.14.1. Description

A QUERY_REQUEST message is used by an RSIP host to ask an RSIP gateway whether or not a particular address or network is local or remote. The host uses this information to determine whether to contact the host(s) directly (in the local case), or via RSIP (in the remote case).

This message defines an indicator parameter with a 1-byte value field and 2 defined values:

- 1 address
- 2 network

9.14.2. Format

```
<QUERY_REQUEST> ::= <Version>
                    <Message Type>
                    <Overall Length>
                    <Client ID>
                    [Message Counter]
                    [Address Tuple]...
                    [Network Tuple]...
where

<Address Tuple> ::= <Indicator (address)>
                    <Address>

<Network Tuple> ::= <Indicator (network)>
                    <Address (network)>
                    <Address (netmask)>
```

9.14.3. Behavior

One or more address or network tuples may be specified. Each tuple encodes a request regarding the locality (local or remote) of the encoded address or network. If no tuple is specified, the RSIP gateway should interpret the message as a request for all tuples that it is willing to provide. Note that the FQDN form of the address parameter cannot be used to specify the address of a network, and only the netmask form of the address parameter can be used to specify the netmask of a network.

If an RSIP gateway cannot determine whether a queried host or network is local or remote, it SHOULD transmit a QUERY_RESPONSE with no response specified for the said host or network.

The following message-specific error conditions exist:

- If the host is not registered with the gateway, the gateway MUST respond with an ERROR_RESPONSE containing the REGISTER_FIRST error.

- If the message contains an incorrect client ID, the gateway MUST respond with an ERROR_RESPONSE containing the BAD_CLIENT_ID error.

9.15. QUERY_RESPONSE

9.15.1. Description

A QUERY_RESPONSE message is used by an RSIP gateway to answer a QUERY_REQUEST from an RSIP host.

This message defines an indicator parameter with a 1-byte value field and 4 defined values:

- 1 local address
- 2 local network
- 3 remote address
- 4 remote network

9.15.2. Format

```
<QUERY_RESPONSE> ::= <Version>
                     <Message Type>
                     <Overall Length>
                     <Client ID>
                     [Message Counter]
                     [Local Address Tuple]...
                     [Local Network Tuple]...
                     [Remote Address Tuple]...
                     [Remote Network Tuple]...
```

where

```
<Local Address Tuple> ::= <Indicator (local address)>
                          <Address>

<Local Network Tuple> ::= <Indicator (local network)>
                          <Address (network)>
                          <Address (netmask)>

<Remote Address Tuple> ::= <Indicator (remote address)>
                           <Address>
```

```
<Remote Network Tuple> ::= <Indicator (remote network)>
                           <Address (network)>
                           <Address (netmask)>
```

9.15.3. Behavior

An RSIP gateway has some leeway in how it responds to a QUERY_REQUEST. It may just provide the information requested, if it can provide such information. It may provide its complete list of address and networks, in order to minimize the number of requests that the host needs to perform in the future. How an RSIP gateway responds may depend on network traffic considerations as well.

If an RSIP gateway sends a QUERY_RESPONSE that does not contain any tuples, or a QUERY_RESPONSE that does not contain a tuple that applies to an associated tuple in the associated QUERY_REQUEST, this should be interpreted that the RSIP gateway does not know whether the queried host or network is local or remote. Appropriate host behavior upon receipt of such a message is to assume that the queried host or network is remote.

Note that an RSIP gateway is not expected to maintain a complete list of all remote hosts and networks. In fact, a typical RSIP gateway will only maintain a list of the networks and hosts that it knows are local (private with respect to the RSIP host).

9.16. LISTEN_REQUEST

9.16.1. Description

A LISTEN_REQUEST message is sent by an RSIP host that wants to register a service on a particular address and port number. The host must include its client ID, local address parameter and ports parameters, and remote address and ports parameters. The client MAY suggest a lease time and one or more tunnel types.

9.16.2. Format

```
<LISTEN_REQUEST> ::= <Version>
                     <Message Type>
                     <Overall Length>
                     <Client ID>
                     <Address (local)>
                     <Ports (local)>
                     <Address (remote)>
                     <Ports (remote)>
                     [Message Counter]
                     [Lease Time]
                     [Tunnel Type]...
```

9.16.3. Behavior

If the host wants to listen on a particular address or port, it may specify these in the address and ports parameters. Otherwise it may leave one or both of these parameters with "don’t care" values.

If no remote flow policy is being used, the host MUST fill both the remote address and ports parameters with "don’t care" values. If macro-flow based remote policy is used, the host MUST specify the remote address, but MAY or MAY NOT specify the remote port(s). If micro-flow based remote policy is used, the host MUST specify the remote address and ports parameter.

Once a LISTEN_REQUEST has been granted, the RSIP gateway MUST forward all packets destined to the address and port in question to the host, even if the remote host address and port tuple has not been previously contacted by the host.

LISTEN_REQUEST is not necessary for RSA-IP.

The following message-specific error conditions exist:

- If the host is not registered with the gateway, the gateway MUST respond with an ERROR_RESPONSE containing the REGISTER_FIRST error.

- If the message contains an incorrect client ID, the gateway MUST respond with an ERROR_RESPONSE containing the BAD_CLIENT_ID error.

- If the local address parameter is a don’t care value and the RSIP gateway cannot allocate ANY addresses, the RSIP gateway MUST respond with an ERROR_RESPONSE containing the LOCAL_ADDR_UNAVAILABLE error.

- If the local address parameter is not a don't care value there are five possible error conditions:

  o If the RSIP gateway cannot allocate ANY addresses, it MUST respond with an ERROR_RESPONSE containing the LOCAL_ADDR_UNAVAILABLE error.

  o If the RSIP gateway cannot allocate the requested address because it is in use, the RSIP gateway MUST respond with an ERROR_RESPONSE containing the LOCAL_ADDR_INUSE error.

  o If the RSIP gateway cannot allocate the requested address because it is not allowed by policy, the RSIP gateway MUST respond with an ERROR_RESPONSE containing the LOCAL_ADDR_UNALLOWED error.

  o If the RSIP gateway cannot allocate the requested address / port tuple because it is in use, the RSIP gateway MUST respond with an ERROR_RESPONSE containing the LOCAL_ADDRPORT_INUSE error.

  o If the RSIP gateway cannot allocate the requested address / port tuple because it is not allowed by policy, the RSIP gateway MUST respond with an ERROR_RESPONSE containing the LOCAL_ADDRPORT_UNALLOWED error.

- If macro-flow based remote policy is used and the requested remote address is not allowed by the RSIP gateway's policy, the RSIP gateway MUST respond with an ERROR_RESPONSE containing the REMOTE_ADDR_UNALLOWED error.

- If micro-flow based remote policy is used and the requested remote address / port pair is not allowed by the RSIP gateway's policy, the RSIP gateway MUST respond with an ERROR_RESPONSE containing the REMOTE_ADDRPORT_UNALLOWED error.

- If an unsupported or unallowed tunnel type is specified, the RSIP gateway MUST respond with an ERROR_RESPONSE containing the BAD_TUNNEL_TYPE error.

- If the host has not specified local or remote address or port information in enough detail, the RSIP gateway MUST respond with an ERROR_RESPONSE containing the FLOW_POLICY_VIOLATION error.

9.17. LISTEN_RESPONSE

9.17.1. Description

A LISTEN_RESPONSE message is used by an RSIP gateway to respond to a LISTEN_REQUEST message from an RSIP host. The RSIP gateway MUST issue a bind ID, and specify the address and port which have been granted to the host. The gateway must also specify a tunnel type and lease time.

If no remote flow policy is being used, the gateway MUST fill both the remote address and ports parameters with "don't care" values. If macro-flow based remote policy is used, the gateway MUST specify the remote address, but MAY or MAY NOT specify the remote port(s). If micro-flow based remote policy is used, the gateway MUST specify the remote address and ports parameter.

9.17.2. Format

```
<LISTEN_RESPONSE> ::= <Version>
                      <Message Type>
                      <Overall Length>
                      <Client ID>
                      <Bind ID>
                      <Address (local)>
                      <Ports (local)>
                      <Address (remote)>
                      <Ports (remote)>
                      <Tunnel Type>
                      <Lease Time>
                      [Address (tunnel endpoint)]
                      [Message Counter]
```

9.17.3. Behavior

If no remote flow policy is being used, the gateway MUST fill both the remote address and ports parameters with "don't care" values. If macro-flow based remote policy is used, the gateway MUST specify the remote address, but MAY or MAY NOT specify the remote port(s). If micro-flow based remote policy is used, the gateway MUST specify the remote address and ports parameter.

The address of a tunnel endpoint that is not the RSIP gateway MAY be specified. If this parameter is not specified, the RSIP gateway MUST be assumed to be the tunnel endpoint.

10. Discussion

10.1. Use of Message Counters, Timeouts, and Retransmissions

Message counters are conceptually similar to sequence numbers. They are necessary to facilitate reliability when UDP is the transport protocol. Each UDP message is marked with a message counter. When such a message is transmitted, the message is stored in a "last message" buffer. For RSIP hosts, a timer is set to expire at the appropriate timeout value.

General rules:

- When an RSIP host transmits a message with a message counter value of n, the RSIP gateway's response will contain a message counter value of n.

- An RSIP host will not increment its message counter value to n+1 until it receives a message from the RSIP gateway with a message counter value of n.

- An RSIP gateway begins all sessions with a message counter value of 1.

- If the message counter value reaches the maximum possible 32-bit value, it will wrap around to 1, not 0.

- If a message with a message counter value of n is transmitted by an RSIP host, but a timer expires before a response to that message is received, the copy of the message (from the "last message" buffer) is retransmitted.

- When an RSIP gateway receives a duplicate copy of a message with a message counter value of n, it transmits the contents of its "last message" buffer.

- When the RSIP gateway transmits an asynchronous RSIP message (an RSIP message for which there was no request by the RSIP host), a message counter value of 0 MUST be used. Note that only three RSIP messages can be transmitted asynchronously: ERROR_RESPONSE, DE-REGISTER_RESPONSE, and FREE_RESPONSE. These messages may also be transmitted in response to an RSIP host request, so their message counter values MAY be non-zero.

- If a message counter is not present in a message from an RSIP host, but is required, the RSIP gateway MUST respond with an ERROR_RESPONSE containing the MESSAGE_COUNTER_REQUIRED error.

10.2. RSIP Host and Gateway Failure Scenarios

When either the RSIP host or gateway suffers from an unrecoverable failure, such as a crash, all RSIP-related state will be lost. In this section, we describe the sequence of events that will occur in both host and gateway failures, and how the host and gateway re-synchronize.

10.2.1. Host Failure

After a host failure, the host will reboot and be unaware of any RSIP state held on its behalf at the gateway.

If the host does not immediately attempt to re-establish a session, it may receive RSIP packets on the RSIP client application port that it was using before it rebooted. If an RSIP client application is not active on this port, these packets will be responded to with ICMP port unreachable messages. If TCP is the transport protocol, it is likely that the connection will be terminated with a TCP RST. If an RSIP client is active on this port, it will not recognize the session that these packets belong to, and it SHOULD silently ignore them.

The RSIP host may also receive packets from a remote host with which it was communicating before it rebooted. These packets will be destined to the RSIP tunnel interface, which should not exist. Thus they SHOULD be silently discarded by the RSIP host's stack, or the RSIP host will transmit appropriate ICMP messages to the tunnel endpoint (e.g., the RSIP gateway). The behavior of the system with respect to sessions that were active before the reboot should be similar to that of a publically addressable non-RSIP host that reboots.

Upon rebooting, an RSIP host may attempt to establish a new RSIP session with the RSIP gateway. Upon receiving the REGISTER_REQUEST message, the RSIP gateway will be able to determine that, as far as it is concerned, the RSIP host is already registered. Thus, it will transmit an ERROR_RESPONSE with the ALREADY_REGISTERED message. Upon receipt of this message, the RSIP host will know the client ID of its old registration, and SHOULD immediately transmit a DE-REGISTER_REQUEST using this client ID. After this is accomplished, the states of the RSIP host and gateway have been synchronized, and a new RSIP session may be established.

If the RSIP host does not de-register itself from the RSIP gateway, it will eventually receive a DE-REGISTER_RESPONSE from the gateway, when the gateway times out the host's session. Since the DE-REGISTER_RESPONSE will refer to a client ID that has no meaning to the host, the host SHOULD silently ignore such a message. At this point, the states of the RSIP host and gateway have been synchronized, and a new RSIP session may be established.

10.2.2. Gateway Failure

After a gateway failure, the gateway will reboot and be unaware of any RSIP state held by an RSIP host.

Since the gateway will not attempt to contact any of its RSIP hosts, a problem will first be detected when either an RSIP host sends an RSIP message to the gateway, an RSIP host sends tunneled data to the gateway, or data from a remote host intended for an RSIP host arrives.

In the first case, the RSIP gateway SHOULD immediately response to all messages (except for a REGISTER_REQUEST) with an ERROR_RESPONSE with a REGISTER_FIRST error. Upon receipt of such a message, an RSIP host MUST interpret the message as an indication of a loss of synchronization between itself and the RSIP gateway. The RSIP host SHOULD immediately transmit a DE-REGISTRATION_REQUEST with its old client ID (which will generate another error, but this error SHOULD be ignored by the host). At this point, the states of the RSIP host and gateway have been synchronized, and a new RSIP session may be established.

In the second case, all data that an RSIP host sends to the tunneled interface of an RSIP server will either (1) be discarded silently, (2) responded to with an ICMP Destination Unreachable message, such as "Communication Administratively Prohibited", or (3) blindly routed to the intended destination. In all of the above cases, the RSIP gateway will not have an explicit method to notify the RSIP host of the problem. To prevent a long term communications outage, small lease times of several minutes can be set by the RSIP gateway.

In the third case, the RSIP gateway SHOULD discard all incoming packets and/or respond with ICMP Port Unreachable messages.

10.3. General Gateway Policy

There is a significant amount of RSIP gateway policy that may be implemented, but is beyond the scope of this document. We expect that most of this policy will be site-specific or implementation-specific and therefore do not make any recommendations. Examples of general gateway policy include:

- How ports are allocated to RSIP hosts.
- Preferred length of lease times.

- How flow policy is applied to which hosts.
- How an RSIP gateway with multiple public IP addresses that may be leased by RSIP clients determines how to partition and/or lease these addresses.

10.4. Errors Not From the RSIP Protocol

Once an RSIP host and gateway have established a relationship and the host is assigned resources to use, error may occur due to the host's misuse of the resources or its attempting to use unassigned resources. The following error behavior is defined:

- If a host attempts to use a local address which it has not been allocated, the RSIP gateway MUST drop the associated packet(s) and send the host an ERROR_RESPONSE containing the LOCAL_ADDR_UNALLOWED error.

- If a host attempts to use a local address / port tuple which it has not been allocated, the RSIP gateway MUST drop the associated packet(s) and send the host an ERROR_RESPONSE containing the LOCAL_ADDRPORT_UNALLOWED error.

- If a host attempts to contact a remote address which has not been properly specified or otherwise approved (e.g., via an ASSIGN_RESPONSE_RSAP-IP and macro or micro based remote flow policy), the RSIP gateway MUST drop the associated packet(s) and send the host an ERROR_RESPONSE containing the REMOTE_ADDR_UNALLOWED error.

- If a host attempts to contact a remote address / port tuple which has not been properly specified or otherwise approved (e.g., via an ASSIGN_RESPONSE_RSAP-IP and micro based remote flow policy), the RSIP gateway MUST drop the associated packet(s) and send the host an ERROR_RESPONSE containing the REMOTE_ADDRPORT_UNALLOWED error.

- If a host attempts to establish or use an improper tunnel type, the RSIP gateway MUST respond with an ERROR_RESPONSE containing the BAD_TUNNEL_TYPE error.

- If the RSIP gateway's detects a local fault which prevents its RSIP server module from continuing operation, the RSIP gateway MUST respond with an ERROR_RESPONSE containing the INTERNAL_SERVER_ERROR error.

10.5. Address and Port Requests and Allocation

Regardless of local flow policy, an RSIP host may "suggest" that it would like to use a particular local address and/or port number in a particular binding. An RSIP gateway that cannot grant such a request, because the specified resources are already in use, MUST respond with an ERROR_RESPONSE containing the LOCAL_ADDR_INUSE or LOCAL_ADDRPORT_INUSE values.

10.6. Local Gateways and Flow Policy Interaction

An RSIP host may initialize a publically accessible gateway (such as an FTP or HTTP gateway) by transmitting a LISTEN_REQUEST message to an RSIP gateway and receiving a LISTEN_RESPONSE. However, unless no remote flow policy is used, the gateway will have to specify the address or address and port of a single remote host that will be allowed to contact it. Obviously, such as restriction is not very useful for hosts that require their gateways to be accessible by any remote host.

This indicates that there is a conflict between flow-based policy and support for gateways. The main purpose of enforcing flow-based policy for LISTEN_REQUESTs is that it allows an RSIP gateway tight control over how an RSIP host uses ports and the associated accounting. For example, an RSIP host, operating under remote micro-flow based policy and using a protocol such as FTP, will have to specify the address and port that it will receive FTP data on, as well as the address and port that the gateway will transmit data from, in a LISTEN_REQUEST.

In general, an RSIP gateway may not allow arbitrary hosts to start public gateways because of the traffic and security concerns. Thus, we recommend that if remote micro-flow based policy is used, that an RSIP gateway only allow public gateways on RSIP hosts via administrative override.

Currently, RSIP hosts can only be identified by their local IP address or MAC address.

11. Security Considerations

RSIP, in and of itself, does not provide security. It may provide the illusion of security or privacy by hiding a private address space, but security can only be ensured by the proper use of security protocols and cryptographic techniques.

An RSIP gateway should take all measures deemed necessary to prevent its hosts from performing intentional or unintentional denial-of-service attacks by request large sets of resources.

Currently, RSIP hosts can only be identified by their local IP address or, in some cases, MAC address. It is desirable to allow RSIP messages sent between a host and gateway to be authenticated. Further discussion of such authentication can be found in [RSIP-FRAME].

Discussion of RSIP support for end-to-end IPsec can be found in [RSIP-IPSEC].

12. IANA Considerations

All of the designations below have been registered by the IANA.

- RSIP port number: 4555
- RSIP error codes (see Appendix A).
- RSIP message type codes (see Appendix B).
- RSIP tunnel types, methods, and flow policies.

RSIP parameter values are designated as follows:

- 0 Reserved
- 1-240 Assigned by IANA
- 241-255 Reserved for private use

New registrations for the above namespaces are recommended to be allocated via the Specification Required method documented in [RFC2434].

13. Acknowledgements

The authors would like to specifically thank Gabriel Montenegro, Pyda Srisuresh, Brian Carpenter, Eliot Lear, Dan Nessett, Gary Jaszewski, Naveen Rajanikantha, Sudhakar Ramakrishna, Jim March, and Rick Cobb for their input. The IETF NAT working group as a whole has been extremely helpful in the ongoing development of RSIP.

14. Appendix A: RSIP Error Numbers

This section provides descriptions for the error values in the RSIP error parameter.

All errors are grouped into the following categories:

100's: General errors.

101: UNKNOWN_ERROR. An error that cannot be identified has occurred. This error should be used when all other error messages are inappropriate.

102: USE_TCP. A host has attempted to use UDP on a server that only supports TCP.

103: FLOW_POLICY_VIOLATION: A host has not specified address or port information in enough detail for its assigned flow policy.

104: INTERNAL_SERVER_ERROR: An RSIP server application has detected an unrecoverable error within itself or the RSIP gateway.

105: MESSAGE_COUNTER_REQUIRED: An RSIP host did not use a message counter parameter in a situation in which it should have.

106: UNSUPPORTED_RSIP_VERSION: An RSIP host sent a message with a version number that is not supported by the RSIP gateway.

200's: Parameter and message errors. The gateway uses these errors when it detects that a parameter or message is malformed, as well as when it does not understand a parameter or message.

201: MISSING_PARAM. The request does not contain a required parameter.

202: DUPLICATE_PARAM. The request contains an illegal duplicate parameter.

203: EXTRA_PARAM. The request contains a parameter that it should not.

204: ILLEGAL_PARAM. The gateway does not understand a parameter type.

205: BAD_PARAM. A parameter is malformed.

206: ILLEGAL_MESSAGE. The gateway does not understand the message type. The message type is neither mandatory nor optional.

207: BAD_MESSAGE. A message is malformed and gateway parsing failed.

208: UNSUPPORTED_MESSAGE: The host has transmitted an optional message that the gateway does not support.

300's: Permission, resource, and policy errors. The gateway uses these errors when a host has attempted to do something that it is not permitted to do, or something that violated gateway policy.

301: REGISTER_FIRST. The RSIP host has attempted to request or use resources without registering.

302: ALREADY_REGISTERED. The host has attempted to register again without first de-registering.

303: ALREADY_UNREGISTERED. The host has attempted to de-register but it is already in the unregistered state.

304: REGISTRATION_DENIED. The gateway will not allow the host to register.

305: BAD_CLIENT_ID. The host has referred to itself with the wrong client ID.

306: BAD_BIND_ID. The request refers to a bind ID that is not valid for the host.

307: BAD_TUNNEL_TYPE. The request refers to a tunnel type that is not valid for the host.

308: LOCAL_ADDR_UNAVAILABLE. The gateway is currently not able to allocate ANY local address, but the host may try again later.

309: LOCAL_ADDRPORT_UNAVAILABLE. The gateway is currently not able to allocate ANY local IP address / port tuple of the requested magnitude (i.e., number of ports), but the host may try again later.

310: LOCAL_ADDR_INUSE. The gateway was not able to allocate the requested local address because it is currently used by another entity.

311: LOCAL_ADDRPORT_INUSE. The gateway was not able to allocate the requested local address / port tuple because it is currently used by another entity.

312: LOCAL_ADDR_UNALLOWED. The gateway will not let the host use the specified local IP address due to policy.

313: LOCAL_ADDRPORT_UNALLOWED. The gateway will not let the host use the specified local address / port pair due to policy.

314: REMOTE_ADDR_UNALLOWED. The gateway will not allow the host to establish a session to the specified remote address.

315: REMOTE_ADDRPORT_UNALLOWED. The gateway will not allow the host to establish a session to the specified remote address / port tuple.

400's: IPsec errors. All errors specific to RSIP / IPsec operation. See [RSIP-IPSEC].

15. Appendix B: Message Types

This section defines the values assigned to RSIP message types. We also indicate which RSIP entity, host or gateway, produces each messages, and whether it is mandatory or optional. All *_REQUEST messages are only to be implemented on hosts, while all *_RESPONSE messages are only to be implemented on gateways. RSIP implementations (both host and gateway) MUST support all mandatory messages in order to be considered "RSIP compliant".

Value	Message	Implementation	Status
1	ERROR_RESPONSE	gateway	mandatory
2	REGISTER_REQUEST	host	mandatory
3	REGISTER_RESPONSE	gateway	mandatory
4	DE-REGISTER_REQUEST	host	mandatory
5	DE-REGISTER_RESPONSE	gateway	mandatory
6	ASSIGN_REQUEST_RSA-IP	host	optional
7	ASSIGN_RESPONSE_RSA-IP	gateway	optional
8	ASSIGN_REQUEST_RSAP-IP	host	mandatory
9	ASSIGN_RESPONSE_RSAP-IP	gateway	mandatory
10	EXTEND_REQUEST	host	mandatory
11	EXTEND_RESPONSE	gateway	mandatory
12	FREE_REQUEST	host	mandatory
13	FREE_RESPONSE	gateway	mandatory
14	QUERY_REQUEST	host	optional
15	QUERY_RESPONSE	gateway	mandatory
16	LISTEN_REQUEST	host	optional
17	LISTEN_RESPONSE	gateway	optional

16. Appendix C: Example RSIP host/gateway transactions

In this appendix, we present an exemplary series of annotated transactions between an RSIP host and an RSIP gateway. All host to gateway traffic is denote by ‘C --> S’ and all gateway to host traffic is denoted by ‘S --> C’. Parameter values are denoted inside of parentheses. Versions, message types, and overall lengths are not included in order to save space. "Don’t care" values are indicated by 0’s.

A ports parameter is represented by the number of ports followed by the port numbers, separated by dashes. For example, 2-1012-1013 indicates two ports, namely 1012 and 1013, while 16-10000 indicates 16 ports, namely 10000-10015, and 4-0 indicates four ports, but the sender doesn’t care where they are.

IPv4 addresses are assumed.

16.1. RSAP-IP with Local Macro-flow Based Policy and No Remote Flow Policy

This example exhibits the loosest policy framework for RSAP-IP.

C --> S: REGISTER_REQUEST ()

The host attempts to register with the gateway.

```
   S --> C: REGISTER_RESPONSE (Client ID = 1, Local Flow Policy =
      Macro, Remote Flow policy = None, Lease Time = 600)

      The gateway responds, assigning a Client ID of 1, local macro-
      flow based policy and no remote flow policy.  No RSIP method is
      indicated, so RSAP-IP is assumed.  No tunnel type is indicated,
      so IP-IP is assumed.  A lease time of 600 seconds is assigned.

   C --> S: ASSIGN_REQUEST_RSAP-IP: (Client ID = 1, Address (local) =
      0, Ports (local) = 4-0, Address (remote) = 0, Ports (remote) =
      0, Lease Time = 3600)

      The host requests an address and four ports to use with it, but
      doesn't care which address or ports are assigned.  The host
      does not specify the remote address or ports either.  The host
      suggests a lease time of 3600 seconds.

   S --> C: ASSIGN_RESPONSE_RSAP-IP: (Client ID = 1, Bind ID = 1,
      Address (local) = 149.112.240.156, Ports (local) = 4-1234,
      Address (remote) = 0, Ports (remote) = 0, Lease Time = 1800,
      Tunnel Type = IP-IP)

      The gateway responds by indicating that a bind ID of 1 has been
      assigned to IP address 149.112.240.156 with ports 1234-1237.
      Any remote host may be communicated with, using any remote port
      number.  The lease time has been assigned to be 1800 seconds,
      and the tunnel type is confirmed to be IP-IP.

      The host is now able to communicate with any host on the public
      network using these resources.

   C --> S: QUERY_REQUEST: (Client ID = 1, Indicator = network,
      Address (network) = 10.20.60.0, Address (netmask)
      255.255.255.0)

      The host asks the gateway if the network 10.20.60.0/24 is
      local.

   S --> C: QUERY_RESPONSE: (Client ID = 1, Indicator = network,
      Address (network) = 10.20.60.0, Address (netmask) =
      255.255.255.0)

      The gateway responds indicating that the network in question is
      local.

   C --> S: ASSIGN_REQUEST_RSAP-IP: (Client ID = 1, Address (local) =
      149.112.240.156, Ports (local) = 8-1238, Address (remote) = 0,
      Ports (remote) = 0, Lease Time = 1800)
```

The host requests eight more particular ports for use with RSAP-IP with the same address. A lease of 1800 seconds is requested. IP-IP tunneling is implied by default.

S --> C: ASSIGN_RESPONSE_RSAP-IP: (Client ID = 1, Bind ID = 2, Address (local) = 149.112.240.156, Ports (local) = 8-1305, Address (remote) = 0, Ports (remote) = 0, Lease Time = 1800)

The gateway grants the request with the same address, but with a different set of ports. IP-IP tunneling is implied by default.

C --> S: FREE_REQUEST (Client ID = 1, Bind ID = 1)

The host frees bind ID 1; i.e., ports 1234-1237 from IP address 149.112.240.156. Note that the address itself is still assigned to the host because the host is still assigned ports 1305-1314.

S --> C: FREE_RESPONSE (Client ID = 1, Bind ID = 1)

The gateway acknowledges that Bind ID 1 has been freed.

C --> S: EXTEND_REQUEST (Client ID = 1, Bind ID = 2, Lease Time = 1800)

The host request that the lease on bind ID 1 be extended for 1800 seconds.

S --> C: EXTEND_RESPONSE (Client ID = 1, Bind ID = 2, Lease Time = 1800)

The gateway confirms the request.

S --> C: FREE_RESPONSE (Client ID = 1, Bind ID = 2)

The gateway forces the host to free the resources of bind ID 2.

C --> S: DE-REGISTER_REQUEST (Client ID = 1)

The host de-registers with the sever.

S --> C: DE-REGISTER_RESPONSE (Client ID = 1)

The gateway acknowledges that the host has de-registered.

16.2. RSAP-IP with Local Micro-flow Based Policy and Remote Micro-flow Based Policy

This example exhibits the strictest policy framework for RSAP-IP.

```
C --> S: REGISTER_REQUEST ()
```

The host attempts to register with the gateway.

```
S --> C: REGISTER_RESPONSE (Client ID = 5, Local Flow Policy =
   Micro, Remote Flow policy = Micro, RSIP Method = RSAP-IP, RSIP
   Method = RSA-IP, Tunnel Type = IP-IP, Tunnel Type = GRE, Lease
   Time = 600)
```

The gateway responds, assigning a Client ID of 5, local micro-flow based policy and remote micro-flow based policy. Both RSAP-IP and RSA-IP are supported. Both IP-IP and GRE tunnel types are supported. A lease time of 600 seconds is assigned.

```
C --> S: ASSIGN_REQUEST_RSAP-IP: (Client ID = 5, Address (local) =
   0, Ports (local) = 0, Address (remote) = 38.196.73.6, Ports
   (remote) = 21, Lease Time = 600, Tunnel Type = IP-IP)
```

The host requests a local address and a port assignment to use with it. The host indicates that it wants to contact host 38.196.73.6 at port 21 (FTP control). The host requests a lease time of 600 seconds and a tunnel type of IP-IP.

```
S --> C: ASSIGN_RESPONSE_RSAP-IP: (Client ID = 5, Bind ID = 1,
   Address (local) = 149.112.240.156, Ports (local) = 2049,
   Address (remote) = 38.196.73.6, Ports (remote) = 21, Lease Time
   = 600, Tunnel Type = IP-IP)
```

The gateway responds by indicating that a bind ID of 1 has been assigned to IP address 149.112.240.156 with port 2049. Only host 38.196.73.6 at port 21 may be contacted. The lease time has been assigned to be 600 seconds, and the tunnel type is confirmed to be IP-IP.

```
C --> S: LISTEN_REQUEST: (Client ID = 5, Address (local) =
   149.112.240.156, Ports (local) = 2050, Address (remote) =
   38.196.73.6, Ports (remote) = 20)
```

The host requests a listen port 2050 at the same address that it has been assigned. Only host 38.196.73.6 from ports 20 (FTP data) will be able to contact it.

S --> C: LISTEN_RESPONSE: (Client ID = 5, Address (local) = 149.112.240.156, Ports (local) = 2050, Address (remote) = 38.196.73.6, Ports (remote) = 20, Lease Time = 600, Tunnel Type = IP-IP)

The gateway confirms the request and assigns a lease time of 600 seconds and a tunnel type of IP-IP.

C --> S: DE-REGISTER_REQUEST (Client ID = 5)

The host de-registers with the sever.

S --> C: DE-REGISTER_RESPONSE (Client ID = 5)

The gateway acknowledges that the host has de-registered. All of the host's bindings have been implicitly revoked.

16.3. RSA-IP with Local Macro-flow Based Policy and Remote Macro-flow based Policy

This example exhibits a medium level of control for RSA-IP.

C --> S: REGISTER_REQUEST ()

The host attempts to register with the gateway.

S --> C: REGISTER_RESPONSE (Client ID = 3, Local Flow Policy = Macro, Remote Flow policy = Macro, RSIP Method = RSAP-IP, RSIP Method = RSA-IP, Tunnel Type = IP-IP, Tunnel Type = L2TP, Lease Time = 600)

The gateway responds, assigning a Client ID of 3, local macro-flow based policy and remote macro-flow based policy. Both RSAP-IP and RSA-IP are supported. Both IP-IP and L2TP tunnel types are supported. A lease time of 600 seconds is assigned.

C --> S: ASSIGN_REQUEST_RSA-IP: (Client ID = 3, Address (local) = 0, Address (remote) = www.foo.com, Ports (remote) = 0, Lease Time = 3600, Tunnel Type = IP-IP)

The host requests a local address and indicates that it wants to contact host www.foo.com.

S --> C: ERROR_RESPONSE: (Error = REMOTE_ADDR_UNALLOWED, Client ID = 3)

The gateway indicates that the host is not permitted to establish communication with www.foo.com.

C --> S: ASSIGN_REQUEST_RSA-IP: (Client ID = 3, Address (local) = 0, Address (remote) = www.bar.com, Ports (remote) = 0, Lease Time = 3600, Tunnel Type = IP-IP)

The host requests a local address and indicates that it wants to contact host www.bar.com.

S --> C: ASSIGN_RESPONSE_RSA-IP: (Client ID = 3, Bind ID = 1, Address (local) = 149.112.240.17, Address (remote) = www.bar.com, Ports (remote) = 0, Lease Time = 3600, Tunnel Type = IP-IP)

The gateway responds by granting local IP address 149.112.240.17 to the host, and permitting it to communicate with www.bar.com, at any port. Requested lease time and tunnel type are also granted.

C --> S: DE-REGISTER_REQUEST (Client ID = 3)

The host de-registers with the sever.

S --> C: DE-REGISTER_RESPONSE (Client ID = 3)

The gateway acknowledges that the host has de-registered. All of the host's bindings have been implicitly revoked.

17. Appendix D: Example RSIP host state diagram

This appendix provides an exemplary diagram of RSIP host state. The host begins in the unregistered state. We assume that for UDP, if a message is lost, the host will timeout and retransmit another copy of it. We recommend a 7-fold binary exponential backoff timer for retransmissions, with the first timeout occurring after 12.5 ms. This diagram does not include transitions for the LISTEN_REQUEST message.

```
                        send
                  REGISTER_REQUEST
     +------------+              +------------+
     |            |------------->|Registration|<-- timeout/send
+--->|Unregistered|<-------------|  Pending   |--- REGISTER_REQUEST
|    |            |              +------------+
|    +------------+ 7th timeout/recv    |
|          ^         ERROR_RESPONSE     |
|          |                            |
|          |                            |
|          |7th timeout/recv            |recv                  timeout/send
|          |DE-REGISTER_RESPONSE        |REGISTER_RESPONSE QUERY_REQUEST
|          |                            |                        ^  |
|          |                            |                        |  |
|          |                            |           send         |  |
|          |            send DE-        v        QUERY_REQUEST   |  |
| +----------------+ REGISTER_REQUEST+----------+          +----------+
| |   Registered   |<----------------|          |--------->|Registered|
| | De-registration|                 |Registered|          |  Query   |
| |    Pending     |---------------->|          |<---------|  Pending |
| +----------------+      recv       +----------+          +----------+
|         | ^        ERROR_RESPONSE        ^  |   7th timeout/recv
|         | |                              |  |  QUERY_RESPONSE or
|    timeout/send                          |  |    ERROR_RESPONSE
| DE-REGISTER_REQUEST      7th timeout/recv|  |
|                           ERROR_RESPONSE |  |
|                                          |  |
| +----------------+                       |  |
| |Go to Registered|                       |  |send
| +----------------+                       |  |ASSIGN_REQUEST
|         ^                   timeout/send |  |
|         |Yes                FREE_REQUEST |  |
|         +                       |  |     |  |
|       +   +                     v  |     |  v
|     +       +   7th timeout/ +--------+ +----------+
|   +  Are all  +      recv     |  Free  | |Assignment|<--timeout/send
| +   resources   +<-----------|Pending | |  Pending |---ASSIGN_REQUEST
|   +   freed?  + FREE_RESPONSE+--------+ +----------+
|     +       +                    ^ |         |
|       +   +                      | |         |
|         +                        | |         |recv
|         |No                 send | |recv     |ASSIGN_RESPONSE
|         v           ERROR_REQUEST| |ERROR_   |
| +---------------+                | |RESPONSE |
| | Go to Assigned|                | |         | 7th timeout/recv
| +---------------+                | |         | QUERY_RESPONSE or
|                       recv       | |         | ERROR_RESPONSE
| +---------------+ERROR_RESPONSE  | v         v          +-----------+
```

```
| |    Assigned    |--------------->+-------------+--------->|  Assigned |
+>|De-registration|                |   Assigned  |          |   Query   |
  |    Pending     |<---------------+-------------+<---------|  Pending  |
  +---------------+       send          ^   |               +-----------+
        ^  |       DE-REGISTER_REQUEST  |   |        send        ^  |
        |  |                            |   |    QUERY_REQUEST   |  |
        |  |                            |   |                    |  |
   timeout/send        7th/timeout/recv |   |send                |  |
   DE-REGISTER_         ASSIGN_RESPONSE |   |ASSIGN_REQUEST timeout/send
     REQUEST         or ERROR_RESPONSE|   |             QUERY_REQUEST
                                        |   |
                                        |   v
                                   +----------+
                                   | Assigned |
                                   |Assignment|
                                   | Pending  |
                                   +----------+
                                        ^   |
                                        |   |
                                   timeout/send
                                  ASSIGN_REQUEST
```

18. References

[RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G.J. and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to indicate requirement levels", BCP 14, RFC 2119, March 1997.

[RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

[RFC2663] Srisuresh, P. and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, August 1999.

[RSIP-FRAME] Borella, M. Lo, J., Grabelsky, D. and G. Montenegro, "Realm Specific IP: Framework", RFC 3102, October 2001.

[RSIP-IPSEC] Montenegro, G. and M. Borella, "RSIP Support for End-to-end IPSEC", RFC 3104, October 2001.

19. Authors' Addresses

Michael Borella
CommWorks
3800 Golf Rd.
Rolling Meadows IL 60008

Phone: (847) 262-3083
EMail: mike_borella@commworks.com

David Grabelsky
CommWorks
3800 Golf Rd.
Rolling Meadows IL 60008

Phone: (847) 222-2483
EMail: david_grabelsky@commworks.com

Jeffrey Lo
Candlestick Networks, Inc
70 Las Colinas Lane,
San Jose, CA 95119

Phone: (408) 284 4132
EMail: yidarlo@yahoo.com

Kunihiro Taniguchi
NEC USA
C&C Research Labs.
110 Rio Robles
San Jose, CA 95134

Phone: (408) 943-3031
EMail: taniguti@ccrl.sj.nec.com

20. Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.

Network Working Group
Request for Comments: 3104
Category: Experimental

G. Montenegro
Sun Microsystems, Inc.
M. Borella
CommWorks

October 2001

RSIP Support for End-to-end IPsec

Status of this Memo

This memo defines an Experimental Protocol for the Internet community. It does not specify an Internet standard of any kind. Discussion and suggestions for improvement are requested. Distribution of this memo is unlimited.

Copyright Notice

IESG Note

The IESG notes that the set of documents describing the RSIP technology imply significant host and gateway changes for a complete implementation. In addition, the floating of port numbers can cause problems for some applications, preventing an RSIP-enabled host from interoperating transparently with existing applications in some cases (e.g., IPsec). Finally, there may be significant operational complexities associated with using RSIP. Some of these and other complications are outlined in section 6 of the RFC 3102, as well as in the Appendices of RFC 3104. Accordingly, the costs and benefits of using RSIP should be carefully weighed against other means of relieving address shortage.

Abstract

This document proposes mechanisms that enable Realm Specific IP (RSIP) to handle end-to-end IPsec (IP Security).

Table of Contents

1. Introduction

This document specifies RSIP extensions to enable end-to-end IPsec. It assumes the RSIP framework as presented in [RSIP-FW], and specifies extensions to the RSIP protocol defined in [RSIP-P]. Other terminology follows [NAT-TERMS].

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

2. Model

For clarity, the discussion below assumes this model:

```
RSIP client              RSIP server                 Host

   Xa                    Na   Nb                     Yb
         +------------+       Nb1  +------------+
[X]------| Addr space |----[N]-----| Addr space |-------[Y]
         |  A         |       Nb2  |  B         |
         +------------+       ...  +------------+
```

Hosts X and Y belong to different address spaces A and B, respectively, and N is an RSIP server. N has two addresses: Na on address space A, and Nb on address space B. For example, A could be a private address space, and B the public address space of the general Internet. Additionally, N may have a pool of addresses in address space B which it can assign to or lend to X.

This document proposes RSIP extensions and mechanisms to enable an RSIP client X to initiate IKE and IPsec sessions to a legacy IKE and IPsec node Y. In order to do so, X exchanges RSIP protocol messages with the RSIP server N. This document does not yet address IKE/IPsec session initiation from Y to an RSIP client X. For some thoughts on this matter see Appendix G.

The discussion below assumes that the RSIP server N is examining a packet sent by Y, destined for X. This implies that "source" refers to Y and "destination" refers to Y's peer, namely, X's presence at N.

This document assumes the use of the RSAP-IP flavor of RSIP (except that port number assignments are optional), on top of which SPI values are used for demultiplexing. Because of this, more than one RSIP client may share the same global IP address.

3. Implementation Notes

The RSIP server N is not required to have more than one address on address space B. RSIP allows X (and any other hosts on address space A) to reuse Nb. Because of this, Y's SPD SHOULD NOT be configured to support address-based keying. Address-based keying implies that only one RSIP client may, at any given point in time, use address Nb when exchanging IPsec packets with Y. Instead, Y's SPD SHOULD be configured to support session-oriented keying, or user-oriented keying [Kent98c]. In addition to user-oriented keying, other types of identifications within the IKE Identification Payload are equally effective at disambiguating who is the real client behind the single address Nb [Piper98].

Because it cannot rely on address-based keying, RSIP support for IPsec is similar to the application of IPsec for remote access using dynamically assigned addresses. Both cases impose additional requirements which are not met by minimally compliant IPsec implementations [Gupta]:

> Note that a minimally-compliant IKE implementation (which only implements Main mode with Pre-shared keys for Phase I authentication) cannot be used on a remote host with a dynamically assigned address. The IKE responder (gateway) needs to look up the initiator's (mobile node's) pre-shared key before it can

decrypt the latter’s third main mode message (fifth overall in Phase I). Since the initiator’s identity is contained in the encrypted message, only its IP address is available for lookup and must be predictable. Other options, such as Main mode with digital signatures/RSA encryption and Aggressive mode, can accommodate IKE peers with dynamically assigned addresses.

IKE packets are typically carried on UDP port 500 for both source and destination, although the use of ephemeral source ports is not precluded [ISAKMP]. IKE implementations for use with RSIP SHOULD employ ephemeral ports, and should handle them as follows [IPSEC-MSG]:

> IKE implementations MUST support UDP port 500 for both source and destination, but other port numbers are also allowed. If an implementation allows other-than-port-500 for IKE, it sets the value of the port numbers as reported in the ID payload to 0 (meaning "any port"), instead of 500. UDP port numbers (500 or not) are handled by the common "swap src/dst port and reply" method.

It is important to note that IPsec implementations MUST be aware of RSIP, at least in some peripheral sense, in order to receive assigned SPIs and perhaps other parameters from an RSIP client. Therefore, bump-in-the-stack (BITS) implementations of IPsec are not expected to work "out of the box" with RSIP.

4. IKE Handling and Demultiplexing

If an RSIP client requires the use of port 500 as its IKE source, this prevents that field being used for demultiplexing. Instead, the "Initiator Cookie" field in the IKE header fields must be used for this purpose. This field is appropriate as it is guaranteed to be present in every IKE exchange (Phase 1 and Phase 2), and is guaranteed to be in the clear (even if subsequent IKE payloads are encrypted). However, it is protected by the Hash payload in IKE [IKE]. Because of this, an RSIP client and server must agree upon a valid value for the Initiator Cookie.

Once X and N arrive at a mutually agreeable value for the Initiator Cookie, X uses it to create an IKE packet and tunnels it the RSIP server N. N decapsulates the IKE packet and sends it on address space B.

The minimum tuple negotiated via RSIP, and used for demultiplexing incoming IKE responses from Y at the RSIP server N, is:

- IKE destination port number
- Initiator Cookie
- Destination IP address

One problem still remains: how does Y know that it is supposed to send packets to X via Nb? Y is not RSIP-aware, but it is definitely IKE-aware. Y sees IKE packets coming from address Nb. To prevent Y from mistakenly deriving the identity of its IKE peer based on the source address of the packets (Nb), X MUST exchange client identifiers with Y:

- IDii, IDir if in Phase 1, and
- IDci, IDcr if in Phase 2.

The proper use of identifiers allows the clear separation between those identities and the source IP address of the packets.

5. IPsec Handling and Demultiplexing

The RSIP client X and server N must arrive at an SPI value to denote the incoming IPsec security association from Y to X. Once N and X make sure that the SPI is unique within both of their SPI spaces, X communicates its value to Y as part of the IPsec security association establishment process, namely, Quick Mode in IKE [IKE] or manual assignment.

This ensures that Y sends IPsec packets (protocols 51 and 50 for AH and ESP, respectively) [Kent98a,Kent98b] to X via address Nb using the negotiated SPI.

IPsec packets from Y destined for X arrive at RSIP server N. They are demultiplexed based on the following minimum tuple of demultiplexing fields:

- protocol (50 or 51)
- SPI
- destination IP address

If N is able to find a matching mapping, it tunnels the packet to X according to the tunneling mode in effect. If N cannot find an appropriate mapping, it MUST discard the packet.

6. RSIP Protocol Extensions

The next two sections specify how the RSIP protocol [RSIP-P] is extended to support both IKE (a UDP application) and the IPsec-defined AH and ESP headers (layered directly over IP with their own protocol numbers).

If a server implements RSIP support for IKE and IPsec as defined in this document, it MAY include the RSIP Method parameter for RSIP with IPsec in the REGISTER_RESPONSE method sent to the client. This method is assigned a value of 3:

3 RSIP with IPsec (RSIPSEC)

Unless otherwise specified, requirements of micro and macro flow-based policy are handled according to [RSIP-P].

6.1 IKE Support in RSIP

As discussed above, if X's IPsec implementation allows use of an ephemeral source port for IKE, then incoming IKE traffic can be demultiplexed by N based on the destination address and port tuple. This is the simplest and most desirable way of supporting IKE, and IPsec implementations that interact with RSIP SHOULD allow it.

However, if X must use source port 500 for IKE, there are two techniques with which X and N can arrive at a mutually unique Initiator Cookie.

- Trial and error.

- Negotiation via an extension of the RSIP protocol.

The trial and error technique consists of X first obtaining resources with which to use IPsec (via ASSIGN_REQUEST_RSIPSEC, defined below), and then randomly choosing an Initiator Cookie and transmitting the first packet to Y. Upon arrival at N, the RSIP server examines the Initiator Cookie for uniqueness per X's assigned address (Nb). If the cookie is unique, N allows the use of this cookie for this an all subsequent packets between X and Y on this RSIP binding. If the cookie is not unique, N drops the packet.

When an IKE packet is determined to be lost, the IKE client will attempt to retransmit at least three times [IKE]. An RSIP-aware IKE client SHOULD use different Initiator Cookies for each of these retransmissions.

The probability of an Initiator Cookie collision at N and subsequent retransmissions by X, is infinitesimal given the 64-bit cookie space. According to the birthday paradox, in a population of 640 million RSIP clients going through the same RSIP server, the chances of a first collision is just 1%. Thus, it is desirable to use the trial and error method over negotiation, for these reasons:

- Simpler implementation requirements

- It is highly unlikely that more than one round trip between X and N will be necessary.

6.2 IPsec Support in RSIP

This section defines the protocol extensions required for RSIP to support AH and ESP. The required message types are ASSIGN_REQUEST_RSIPSEC and ASSIGN_RESPONSE_RSIPSEC:

ASSIGN_REQUEST_RSIPSEC

The ASSIGN_REQUEST_RSIPSEC message is used by an RSIP client to request IPsec parameter assignments. An RSIP client MUST request an IP address and SPIs in one message.

If the RSIP client wishes to use IPsec to protect a TCP or UDP application, it MUST use the port range parameter (see Appendix A). Otherwise, it MUST set the port parameters to the "don't need" value. This is accomplished by setting the length field to 0, and by omitting both the number field and the port field. This informs the server that the client does not actually need any port assignments.

The client may initialize the SPI parameter to the "don't care" value (see below). In this case, it is requesting the server to assign it a valid SPI value to use.

Alternatively, the client may initialize the SPI parameter to a value it considers valid. In this case, it is suggesting that value to the server. Of course, the server may choose to reject that suggestion and return an appropriate error message.

The format of this message is:

```
<ASSIGN_REQUEST_RSIPSEC> ::= <Version>
                             <Message Type>
                             <Overall Length>
                             <Client ID>
                             <Address (local)>
                             <Ports (local)>
                             <Address (remote)>
                             <Ports (remote)>
                             <SPI>
                             [Message Counter]
                             [Lease Time]
                             [Tunnel Type]
```

The following message-specific error conditions exist. The error behavior of ASSIGN_REQUEST_RSIP_IPSEC follows that of ASSIGN_REQUEST_RSAP-IP for all non-IPsec errors.

- If the client is not allowed to use IPsec through the server, the server MUST respond with an ERROR_RESPONSE containing the IPSEC_UNALLOWED parameter.

- If the SPI parameter is a "don't care" value and the RSIP server cannot allocate ANY SPIs, the RSIP server MUST respond with an ERROR_RESPONSE containing the IPSEC_SPI_UNAVAILABLE error.

- If an SPI parameter is not a "don't care" value and the RSIP server cannot allocate it because the requested address and SPI tuple is in use, the RSIP server MUST respond with an ERROR_RESPONSE containing the IPSEC_SPI_INUSE error.

ASSIGN_RESPONSE_RSIPSEC

The ASSIGN_RESPONSE_RSIPSEC message is used by an RSIP server to assign parameters to an IPsec-enabled RSIP client.

The format of this message is:

```
<ASSIGN_RESPONSE_RSIPSEC> ::= <Version>
                              <Message Type>
                              <Overall Length>
                              <Client ID>
                              <Bind ID>
                              <Address (local)>
                              <Ports (local)>
                              <Address (remote)>
                              <Ports (remote)>
                              <SPI>
                              <Lease Time>
                              <Tunnel Type>
                              [Address (tunnel endpoint)]
                              [Message Counter]
```

If the port parameters were set to the "don’t need" value in the request (see above), the RSIP server must do the same in the response.

Additionally, RSIP support for IPsec requires the following new parameter:

SPI

```
  Code   Length    Number    SPI           SPI
+------+--------+---------+---------+     +---------+
|  22  |    2   | 2 bytes | 4 bytes | ... | 4 bytes |
+------+--------+---------+---------+     +---------+
```

Sent by the RSIP client in ASSIGN_REQUEST_RSIPSEC messages to ask for a particular number of SPIs to be assigned. Also sent by the RSIP server to the client in ASSIGN_RESPONSE_RSIPSEC messages.

The "SPI" fields encode one or more SPIs. When a single SPI is specified, the value of the number field is 1 and there is one SPI field following the number field. When more than one SPI is specified, the value of the number field will indicate the total number of SPIs contained, and the parameter may take one of two forms. If there is one SPI field, the SPIs specified are considered to be contiguous starting at the SPI number specified in the SPI field. Alternatively, there may be a number of SPI fields equal to the value of the number field. The number of SPI fields can be extrapolated from the value of the length field.

In some cases, it is necessary to specify a "don't care" value for one or more SPIs. This is accomplished by setting the length field to 2 (to account for the 2 bytes in the Number field), setting the number field to the number of SPIs necessary, and omitting all SPI fields. The value of the number field MUST be greater than or equal to one.

7. IANA Considerations

All of the designations below are tentative.

- RSIP IPsec error codes (see below).

- ASSIGN_REQUEST_RSIP_IPSEC message type code.

- SPI parameter code.

8. Security Considerations

This document does not add any security issues to those already posed by NAT, or normal routing operations. Current routing decisions typically are based on a tuple with only one element: destination IP address. This document just adds more elements to the tuple.

Furthermore, by allowing an end-to-end mode of operation and by introducing a negotiation phase to address reuse, the mechanisms described here are more secure and less arbitrary than NAT.

A word of caution is in order: SPI values are meant to be semi-random, and, thus serve also as anti-clogging tokens to reduce off-the-path denial-of-service attacks. However, RSIP support for IPsec, renders SPI's a negotiated item: in addition to being unique values at the receiver X, they must also be unique at the RSIP server, N. Limiting the range of the SPI values available to the RSIP clients reduces their entropy slightly.

9. Acknowledgements

Many thanks to Bernard Aboba, Vipul Gupta, Jeffrey Lo, Dan Nessett, Gary Jaszewski and Prakash Iyer for helpful discussions.

References

[Gupta] Gupta, V., "Secure Remote Access over the Internet using IPSec", Work in Progress.

[IKE] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, November 1998.

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

[IPSEC-MSG] Ted Ts'o, message to the IETF's IPsec mailing list, Message-Id:<199911232216.RAA01932@trampoline.thunk.org>, November 23, 1999.

[Jenkins] Jenkins, T., "IPsec Rekeying Issues", Work in Progress.

[Kent98a] Kent, S. and R. Atkinson, "IP Encapsulating Payload", RFC 2406, November 1998.

[Kent98b] Kent, S. and R. Atkinson, "IP Authentication Header", RFC 2402, November 1998.

[Kent98c] Kent, S. and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[Piper98] Piper, D., "The Internet IP Security Domain of Interpretation for ISAKMP", RFC 2407, November 1998.

[NAPT] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.

[NAT-TERMS] Srisuresh, P. and M. Holdredge, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, August 1999.

[RSIP-FW] Borella, M., Lo, J., Grabelsky, D. and G. Montenegro, "Realm Specific IP: A Framework", RFC 3102, October 2001.

[RSIP-P] Borella, M., Grabelsky, D., Lo, J. and K. Taniguchi, "Realm Specific IP: Protocol Specification", RFC 3103, October 2001.

Authors' Addresses

Gabriel E. Montenegro
Sun Microsystems
Laboratories, Europe
29, chemin du Vieux Chene
38240 Meylan
FRANCE

Phone: +33 476 18 80 45
EMail: gab@sun.com

Michael Borella
CommWorks
3800 Golf Rd.
Rolling Meadows IL 60008

Phone: (847) 262-3083
EMail: mike_borella@commworks.com

Appendix A: On Optional Port Allocation to RSIP Clients

Despite the fact that SPIs rather than ports are used to demultiplex packets at the RSIP server, the RSIP server may still allocate mutually exclusive port numbers to the RSIP clients. If this does not happen, there is the possibility that two RSIP clients using the same IP address attempt an IPsec session with the same server using the same source port numbers.

```
+-------------+
| RSIP client |
|      X1     +--+
|             |  |        +-------------+
+-------------+  |        |             |Nb
                 +---------+ RSIP server +----------------
+-------------+  |        |      N      |
| RSIP client |  |        +-------------+
|      X2     +--+ private                     public
|             |  | network                     network
+-------------+  |
                 |
                 |
```

For example, consider hosts X1 and X2 depicted above. Assume that they both are using public address Nb, and both are contacting an external server Y at port 80. If they are using IPsec but are not allocated mutually exclusive port numbers, they may both choose the same ephemeral port number to use when contacting Y at port 80. Assume client X1 does so first, and after engaging in an IKE negotiation begins communicating with the public server using IPsec.

When Client X2 starts its IKE session, it sends its identification to the public server. The latter's SPD requires that different identities use different flows (port numbers). Because of this, the IKE negotiation will fail. Client X2 will be forced to try another ephemeral port until it succeeds in obtaining one which is currently not in use by any other security association between the public server and any of the RSIP clients in the private network.

Each such iteration is costly in terms of round-trip times and CPU usage. Hence --and as a convenience to its RSIP clients--, an RSIP server may also assign mutually exclusive port numbers to its IPsec RSIP clients.

Despite proper allocation of port numbers, an RSIP server cannot prevent their misuse because it cannot examine the port fields in packets that have been encrypted by the RSIP clients. Presumably, if the RSIP clients have gone through the trouble of negotiating ports numbers, it is in their best interest to adhere to these assignments.

Appendix B: RSIP Error Numbers for IKE and IPsec Support

This section provides descriptions for the error values in the RSIP error parameter beyond those defined in [RSIP-P].

401: IPSEC_UNALLOWED. The server will not allow the client to use end-to-end IPsec.

402: IPSEC_SPI_UNAVAILABLE. The server does not have an SPI available for client use.

403: IPSEC_SPI_INUSE. The client has requested an SPI that another client is currently using.

Appendix C: Message Type Values for IPsec Support

This section defines the values assigned to RSIP message types beyond those defined in [RSIP-P].

22 ASSIGN_REQUEST_RSIPSEC

23 ASSIGN_RESPONSE_RSIPSEC

Appendix D: A Note on Flow Policy Enforcement

An RSIP server may not be able to enforce local or remote micro-flow policy when a client uses ESP for end-to-end encryption, since all TCP/UDP port numbers will be encrypted. However, if AH without ESP is used, micro-flow policy is enforceable. Macro-flow policy will always be enforceable.

Appendix E: Remote Host Rekeying

Occasionally, a remote host with which an RSIP client has established an IPsec security association (SA) will rekey [Jenkins]. SA rekeying is only an issue for RSIP when IKE port 500 is used by the client and the rekey is of ISAKMP phase 1 (the ISAKMP SA). The problem is that the remote host will transmit IKE packets to port 500 with a new initiator cookie. The RSIP server will not have a mapping for the cookie, and SHOULD drop the the packets. This will cause the ISAKMP

SA between the RSIP client and remote host to be deleted, and may lead to undefined behavior given that current implementations handle rekeying in a number of different ways.

If the RSIP client uses an ephemeral source port, rekeying will not be an issue for RSIP. If this cannot be done, there are a number of RSIP client behaviors that may reduce the number of occurrences of this problem, but are not guaranteed to eliminate it.

- The RSIP client's IKE implementation is given a smaller ISAKMP SA lifetime than is typically implemented. This would likely cause the RSIP client to rekey the ISAKMP SA before the remote host. Since the RSIP client chooses the Initiator Cookie, there will be no problem routing incoming traffic at the RSIP server.

- The RSIP client terminates the ISAKMP SA as soon as the first IPsec SA is established. This may alleviate the situation to some degree if the SA is coarse-grained. On the other hand, this exacerbates the problem if the SA is fine-grained (such that it cannot be reused by other application-level connections), and the remote host needs to initialize sockets back to the RSIP client.

Note that the unreliability of UDP essentially makes the ephemeral source approach the only robust solution.

Appendix F: Example Application Scenarios

This section briefly describes some examples of how RSIP may be used to enable applications of IPsec that are otherwise not possible.

The SOHO (small office, home office) scenario

```
+----------+
|RSIP      |
|client X1 +--+
|          |  |  +-------------+          +-------+
+----------+  |  |NAPT gateway |          |public |
              +--+ and         +--.......---+IPsec  |
+----------+  |  |RSIP server  |          |peer Y |
|RSIP      |  |  +-------------+          +-------+
|client X2 +--+ private            public
|          |  | "home"             Internet
+----------+  | network
              |
              |
```

Suppose the private "home" network is a small installation in somebody's home, and that the RSIP clients X1 and X2 must use the RSIP server N as a gateway to the outside world. N is connected via an ISP and obtains a single address which must be shared by its clients. Because of this, N has NAPT, functionality. Now, X1 wishes to establish an IPsec SA with peer Y. This is possible because N is also an RSIP server augmented with the IPsec support defined in this document. Y is IPsec-capable, but is not RSIP aware. This is perhaps the most typical application scenario.

The above is equally applicable in the ROBO (remote office, branch office) scenario.

The Roadwarrior scenario

```
+---------+              +------------+   +----------+
|RSIP     |              |Corporate   |   | IPsec    |
|client X +--..........--+Firewall    +---+ peer Y   |
|         |   public     | and        |   | (user's  |
+---------+   Internet   |RSIP server |   | desktop) |
                         | N          |   |          |
                         +------------+   +----------+
                               private corporate
                               network
```

In this example, a remote user with a laptop gains access to the Internet, perhaps by using PPP or DHCP. The user wants to access its corporation private network. Using mechanisms not specified in this document, the RSIP client in the laptop engages in an RSIP authentication and authorization phase with the RSIP server at the firewall. After that phase is completed, the IPsec extensions to RSIP defined here are used to establish an IPsec session with a peer, Y, that resides within the corporation's network. Y could be, for example, the remote user's usual desktop when at the office. The corporate firewall complex would use RSIP to selectively enable IPsec traffic between internal and external systems.

Note that this scenario could also be reversed in order to allow an internal system (Y) to initiate and establish an IPsec session with an external IPsec peer (X).

Appendix G: Thoughts on Supporting Incoming Connections

Incoming IKE connections are much easier to support if the peer Y can initiate IKE exchanges to a port other than 500. In this case, the RSIP client would allocate that port at the RSIP server via ASSIGN_REQUEST_RSAP-IP. Alternatively, if the RSIP client is able to allocate an IP address at the RSIP server via ASSIGN_REQUEST_RSA-IP, Y could simply initiate the IKE exchange to port 500 at that address.

If there is only one address Nb that must be shared by the RSIP server and all its clients, and if Y can only send to port 500, the problem is much more difficult. At any given time, the combination of address Nb and UDP port 500 may be registered and used by only one RSIP system (including clients and server).

Solving this issue would require demultiplexing the incoming IKE connection request based on something other than the port and address combination. It may be possible to do so by first registering an identity with a new RSIP command of LISTEN_RSIP_IKE. Note that the identity could not be that of the IKE responder (the RSIP client), but that of the initiator (Y). The reason is that IKE Phase 1 only allows the sender to include its own identity, not that of the intended recipient (both, by the way, are allowed in Phase 2). Furthermore, the identity must be in the clear in the first incoming packet for the RSIP server to be able to use it as a demultiplexor. This rules out all variants of Main Mode and Aggressive Mode with Public Key Encryption (and Revised Mode of Public Key Encryption), since these encrypt the ID payload.

The only Phase 1 variants which enable incoming IKE sessions are Aggressive Mode with signatures or with pre-shared keys. Because this scheme involves the RSIP server demultiplexing based on the identity of the IKE initiator, it is conceivable that only one RSIP client at a time may register interest in fielding requests from any given peer Y. Furthermore, this precludes more than one RSIP client's being available to any unspecified peer Y.

Once the IKE session is in place, IPsec is set up as discussed in this document, namely, by the RSIP client and the RSIP server agreeing on an incoming SPI value, which is then communicated to the peer Y as part of Quick Mode.

The alternate address and port combination must be discovered by the remote peer using methods such as manual configuration, or the use of KX (RFC2230) or SRV (RFC2052) records. It may even be possible for the DNS query to trigger the above mechanisms to prepare for the incoming and impending IKE session initiation. Such a mechanism would allow more than one RSIP client to be available at any given time, and would also enable each of them to respond to IKE initiations from unspecified peers. Such a DNS query, however, is not guaranteed to occur. For example, the result of the query could be cached and reused after the RSIP server is no longer listening for a given IKE peer's identity.

Because of the limitations implied by having to rely on the identity of the IKE initiator, the only practical way of supporting incoming connections is for the peer Y to initiate the IKE session at a port other than 500.

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.