

Network Working Group
Internet Draft
[draft-ietf-ipsec-rfc2401bis-03.txt](#)
Obsoletes: RFC [2401](#)
Expires March 2005

S. Kent
K. Seo
BBN Technologies
September 2004

Security Architecture for the Internet Protocol

Dedicated to the memory of Charlie Lynn, a long time senior colleague at BBN, who made very significant contributions to the IPsec documents.

Status of this Memo

By submitting this Internet-Draft, I certify that any applicable patent or other IPR claims of which I am aware have been disclosed, and any of which I become aware will be disclosed, in accordance with [RFC 3668](#).

This document is an Internet Draft and is subject to all provisions of [Section 10 of RFC2026](#). Internet Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet Drafts. Internet Drafts are draft documents valid for a maximum of 6 months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet Drafts as reference material or to cite them other than as a "work in progress".

The list of current Internet Drafts can be accessed at <http://www.ietf.org/1id-abstracts.html>

The list of Internet Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>

Copyright (C) The Internet Society (2004). All Rights Reserved.

Abstract

This document describes an updated version of the "Security Architecture for IP", which is designed to provide security services for traffic at the IP layer. This document is based upon [RFC 2401](#) (November 1998).

Comments should be sent to Stephen Kent (kent@bbn.com).

Table of Contents

1.	Introduction.....	4
1.1	Summary of Contents of Document.....	4
1.2	Audience.....	4
1.3	Related Documents.....	5
2.	Design Objectives.....	5
2.1	Goals/Objectives/Requirements/Problem Description.....	5
2.2	Caveats and Assumptions.....	6
3.	System Overview	7
3.1	What IPsec Does.....	7
3.2	How IPsec Works.....	9
3.3	Where IPsec May Be Implemented.....	10
4.	Security Associations.....	11
4.1	Definition and Scope.....	11
4.2	Security Association Functionality.....	15
4.3	Combining Security Associations.....	16
4.4	Major IPsec Databases.....	16
4.4.1	The Security Policy Database (SPD).....	18
4.4.1.1	Selectors.....	24
4.4.1.2	Structure of an SPD entry.....	27
4.4.1.3	More re: Fields Associated with Next Layer Protocols.	29
4.4.2	Security Association Database (SAD).....	31
4.4.2.1	Data Items in the SAD.....	31
4.4.2.2	Relationship between SPD, PFP flag, packet, and SAD..	33
4.4.3	Peer Authorization Database (PAD).....	38
4.5	SA and Key Management.....	39
4.5.1	Manual Techniques.....	39
4.5.2	Automated SA and Key Management.....	39
4.5.3	Locating a Security Gateway.....	40
4.6	Security Associations and Multicast.....	41
5.	IP Traffic Processing.....	42
5.1	Outbound IP Traffic Processing (protected-to-unprotected).....	42
5.1.1	Handling an Outbound Packet That Must Be Discarded.....	45
5.1.2	Header Construction for Tunnel Mode.....	46
5.1.2.1	IPv4 -- Header Construction for Tunnel Mode.....	47
5.1.2.2	IPv6 -- Header Construction for Tunnel Mode.....	49
5.2	Processing Inbound IP Traffic (unprotected-to-protected).....	49
6.	ICMP Processing	52
6.1	Processing ICMP Error Messages Directed to an IPsec Implementation.....	53
6.1.1	ICMP Error Messages Received on the Unprotected Side of the Boundary.....	53
6.1.2	ICMP Error Messages Received on the Protected Side of the Boundary.....	53
6.2	Processing Protected, Transit ICMP Error Messages.....	54
7.	Handling Fragments (on the protected side of the IPsec boundary)....	55
7.1	Tunnel Mode SAs that Carry Initial and Non-Initial Fragments...	56

7.2	Separate Tunnel Mode SAs for Non-Initial Fragments.....	57
7.3	Stateful Fragment Checking.....	57
7.4	BYPASS/DISCARD traffic.....	58
8.	Path MTU/DF Processing.....	58
8.1	DF Bit.....	59
8.2	Path MTU (PMTU) Discovery.....	59
8.2.1	Propagation of PMTU.....	59
8.2.2	PMTU Aging.....	60
9.	Auditing.....	60
10.	Conformance Requirements.....	60
11.	Security Considerations.....	60
12.	IANA Considerations.....	61
13.	Differences from RFC 2401	61
	Acknowledgements.....	64
	Appendix A -- Glossary.....	65
	Appendix B -- Decorrelation.....	68
	Appendix C -- ASN.1 for an SPD Entry.....	71
	Appendix D -- Fragment Handling Rationale.....	77
D.1	Transport Mode and Fragments.....	77
D.2	Tunnel Mode and Fragments.....	78
D.3	The Problem of Non-Initial Fragments.....	79
D.4	BYPASS/DROP traffic.....	82
D.5	Just say no to ports?.....	82
D.6	Other Suggested Solutions.....	83
D.7	Consistency.....	84
D.8	Conclusions.....	84
	Appendix E -- Example of Supporting Nested SAs via SPD and Forwarding Table Entries.....	85
	References.....	87
	Author Information.....	89
	Notices.....	90

1. Introduction

1.1 Summary of Contents of Document

This document specifies the base architecture for IPsec compliant systems. It describes how to provide a set of security services for traffic at the IP layer, in both the IPv4 [[Pos81a](#)] and IPv6 [[DH98](#)] environments. This document describes the requirements for systems that implement IPsec, the fundamental elements of such systems, and how the elements fit together and fit 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 the IPsec architecture. Other documents address additional architectural details in specialized environments, e.g., use of IPsec in NAT environments and more comprehensive support for IP multicast. The 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 automated (The Internet Key Exchange (IKE))
- d. Cryptographic algorithms for authentication and encryption

This document is not a 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 spelling "IPsec" is preferred and used throughout this and all related IPsec standards. All other capitalizations of IPsec (e.g., IPSEC, IPSec, ipsec) are deprecated. However, any capitalization of the sequence of letters "IPsec" should be understood to refer to the IPsec protocols.

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 is primarily individuals who implement this IP security technology or who architect systems that will use this technology. Technically adept users of this technology

(end users or system administrators) also are part of the target audience. A glossary is provided in [Appendix A](#) to help fill in gaps in background/vocabulary. This document assumes that the reader is familiar with the Internet Protocol (IP), related networking technology, and general information system 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. security protocols -- RFCs describing the Authentication Header (AH) [[Ken04b](#)] and Encapsulating Security Payload (ESP) [[Ken04a](#)] protocols.
- b. cryptographic algorithms for integrity and encryption -- one RFC that defines the mandatory, default algorithms for use with AH and ESP [[Eas03](#)], a similar RFC that defines the mandatory algorithms for use with IKEv2 [[Sch03](#)] plus a separate RFC for each cryptographic algorithm.
- c. automatic key management -- RFCs on "The Internet Key Exchange (IKEv2) Protocol" [[Kau04](#)] and "Cryptographic Algorithms for use in the Internet Key Exchange Version 2" [[Sch03](#)]

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, detection and rejection of replays (a form of partial sequence integrity), confidentiality (via encryption), and limited traffic flow confidentiality. These services are provided at the IP layer, offering protection for all protocols that may be carried over IP in a standard fashion (including IP itself).

IPsec includes a specification for minimal firewall functionality, since that is an essential aspect of access control at the IP layer. Implementations are free to provide more sophisticated firewall mechanisms, and to implement the IPsec-mandated functionality using those more sophisticated mechanisms. (Note that interoperability may suffer if additional firewall constraints on traffic flows are imposed by an IPsec implementation but cannot be negotiated based on the traffic selector features defined in this document and negotiated

via IKEv2.) The IPsec firewall function makes use of the cryptographically-enforced authentication and integrity provided for all IPsec traffic to offer better access control than could be obtained through use of a firewall (one not privy to IPsec internal parameters) plus separate cryptographic protection.

Most of the security services are provided through 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 a context, and the ways in which they are employed, will be determined by the users/administrators in that context. It is the goal of the IPsec architecture to ensure that compliant implementations include the services and management interfaces needed to meet the security requirements of a broad user population.

When IPsec is correctly implemented and deployed, it ought not adversely affect users, hosts, and other Internet components that do not employ IPsec for traffic protection. IPsec security protocols (AH & ESP, and to a lesser extent, IKE) are designed to be cryptographic algorithm-independent. This modularity permits selection of different sets of cryptographic algorithms as appropriate, without affecting the other parts of the implementation. For example, different user communities may select different sets of cryptographic algorithms (creating cryptographically-enforced cliques) if required.

A set of default cryptographic algorithms for use with AH and ESP is specified [[Eas03](#)] to facilitate interoperability in the global Internet. The use of these cryptographic 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 cryptographic 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, as a security gateway, or as an independent device, affording protection to IP traffic. (A security gateway is an intermediate system implementing IPsec, e.g., a firewall or router that has been IPsec-enabled.) More detail on these classes of implementations is provided later, in [Section 3.3](#). The protection offered by IPsec 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 actions based on IP and next layer header information (Selectors, [Section 4.4.1.1](#)) matched against entries in the database (SPD). Each packet is either PROTECTED using IPsec security services, DISCARDED, or allowed to BYPASS IPsec protection, based on the applicable SPD policies identified by the Selectors.

[3.1](#) What IPsec Does

IPsec creates a boundary between unprotected and protected interfaces, for a host or a network (see Figure 1 below). Traffic traversing the boundary is subject to the access controls specified by the user or administrator responsible for the IPsec configuration. These controls indicate whether packets cross the boundary unimpeded, are afforded security services via AH or ESP, or are discarded. IPsec security services are offered at the IP layer through selection of appropriate security protocols, cryptographic algorithms, and cryptographic keys. IPsec can be used to protect one or more "paths" (a) between a pair of hosts, (b) between a pair of security gateways, or (c) between a security gateway and a host. A compliant host implementation MUST support (a) and (c) and a compliant security

gateway must support all three of these forms of connectivity, since under certain circumstances a security gateway acts as a host.

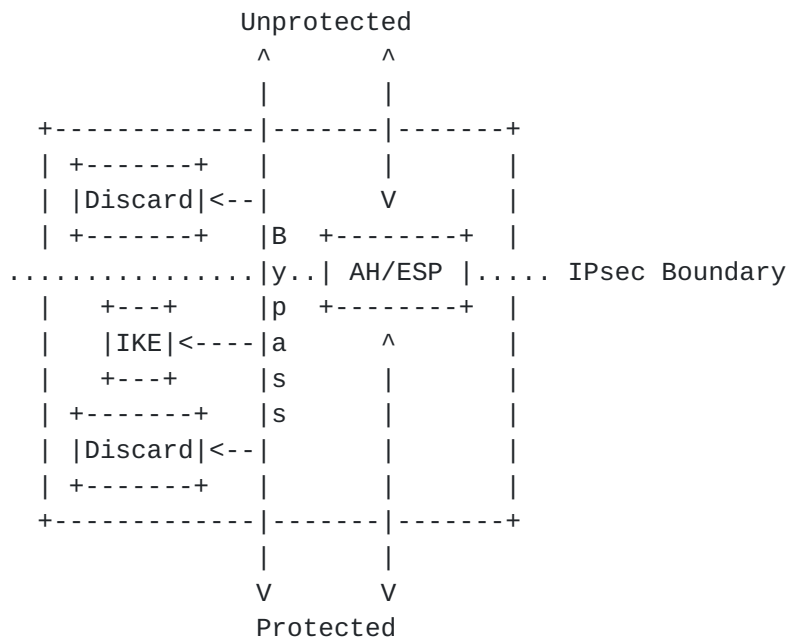


Figure 1. Top Level IPsec Processing Model

In this diagram, "unprotected" refers to an interface that might also be described as "black" or "ciphertext." Here, "protected" refers to an interface that might also be described as "red" or "plaintext." The protected interface noted above may be internal, e.g., in a host implementation of IPsec, the protected interface may link to a socket layer interface presented by the OS. In this document, the term "inbound" refers to traffic entering an IPsec implementation via the unprotected interface or emitted by the implementation on the unprotected side of the boundary and directed towards the unprotected interface. The term "outbound" refers to traffic entering the implementation via the protected interface, or emitted by the implementation on the protected side of the boundary and directed toward the unprotected interface. An IPsec implementation may support more than one interface on either or both sides of the boundary.

Note the facilities for discarding traffic on either side of the IPsec boundary, the BYPASS facility that allows traffic to transit the boundary without cryptographic protection, and the reference to IKE as a protected-side key and security management function.

IPsec optionally supports negotiation of IP compression [[SMPT01](#)], 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 services -- Authentication Header (AH) and Encapsulating Security Payload (ESP). Both protocols are described in detail in their respective RFCs [[Ken04b](#), [Ken04a](#)]. IPsec implementations MUST support ESP and MAY support AH. (Support for AH has been downgraded to MAY because experience has shown that there are very few contexts in which ESP cannot provide the requisite security services. Note that ESP can be used to provide only integrity, without confidentiality, making it comparable to AH in most contexts.)

- o The IP Authentication Header (AH) [[Ken04b](#)] offers integrity and data origin authentication, with optional (at the discretion of the receiver) anti-replay features.
- o The Encapsulating Security Payload (ESP) protocol [[Ken04a](#)] offers the same set of services, and also offers confidentiality. Use of ESP in a confidentiality-only mode is discouraged. When ESP is used with confidentiality enabled, there are provisions for limited traffic flow confidentiality, i.e., provisions for concealing packet length, and for facilitating efficient generation and discard of dummy packets. This capability is likely to be effective primarily in VPN and overlay network contexts.
- o Both AH and ESP offer access control, enforced through the distribution of cryptographic keys and the management of traffic flows as dictated by the Security Policy Database (SPD, [Section 4.4.1](#)).

These protocols may be applied individually or in combination with each other to provide security services in IPv4 and IPv6. However, most security requirements can be met through the use of ESP by itself. Each protocol supports two modes of use: transport mode and tunnel mode. In transport mode, AH and ESP provide protection primarily for next layer protocols; in tunnel mode, AH and ESP are applied to tunneled IP packets. The differences between the two modes are discussed in [Section 4.1](#).

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, through the SPD management paradigm,

incorporates facilities for specifying:

- o which security protocol (AH or ESP) to employ, the mode (transport or tunnel), security service options, what cryptographic algorithms to use, and in what combinations to use the specified protocols and services,
- o the granularity at which protection should be applied.

Because most of the security services provided by IPsec require the use of cryptographic keys, IPsec relies on a separate set of mechanisms for putting these keys in place. This document requires support for both manual and automated distribution of keys. It specifies a specific public-key based approach (IKEv2 [[Kau04](#)]) for automated key management, but other automated key distribution techniques MAY be used.

Note: This document mandates support for several features for which support is available in IKEv2 but not in IKEv1, e.g., negotiation of an SA representing ranges of local and remote ports or negotiation of multiple SAs with the same selectors. Therefore this document assumes use of IKEv2 or a key and security association management system with comparable features.

[3.3](#) Where IPsec Can Be Implemented

There are many ways in which IPsec may be implemented in a host, or in conjunction with a router or firewall to create a security gateway, or as an independent security device.

- a. IPsec may be integrated into the native IP stack. This requires access to the IP source code and is applicable to both hosts and security gateways, although native host implementations benefit the most from this strategy, as explained later ([Section 4.4.1](#), paragraph 6; [Section 4.4.1.1](#), last paragraph).
- b. In a "bump-in-the-stack" (BITS) implementation, 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 a dedicated, inline security protocol processor is a common design feature of 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. Usually the

BITW device is itself 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.

This document often talks in terms of use of IPsec by a host or a security gateway, without regard to whether the implementation is native, BITS or BITW. When the distinctions among these implementation options are significant, the document makes reference to specific implementation approaches.

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 AH and ESP. 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 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 are applied to a traffic stream, then two SAs must be created and coordinated to effect protection through iterated application of the security protocols. To secure typical, bi-directional communication between two IPsec-enabled systems, a pair of Security Associations (one in each direction) is required. IKE explicitly creates SA pairs in recognition of this common usage requirement.

For an SA used to carry unicast (or anycast) traffic, the SPI (Security Parameters Index - see [Appendix A](#) and AH [[Ken04b](#)] and ESP [[Ken04a](#)] specifications) by itself suffices to specify an SA. However, as a local matter, an implementation may choose to use the SPI in conjunction with the IPsec protocol type (AH or ESP) for SA identification. If an IPsec implementation supports multicast, then it MUST support multicast SAs using the algorithm below for mapping inbound IPsec datagrams to SAs. Implementations that support only unicast traffic need not implement this demultiplexing algorithm.

In many secure multicast architectures, e.g., [[RFC3740](#)], a central

Group Controller/Key Server unilaterally assigns the group security association's SPI. This SPI assignment is not negotiated or coordinated with the key management (e.g., IKE) subsystems that reside in the individual end systems that constitute the group. Consequently, it is possible that a group security association and a unicast security association can simultaneously use the same SPI. A multicast-capable IPsec implementation **MUST** correctly de-multiplex inbound traffic even in the context of SPI collisions.

Each entry in the Security Association Database (SAD) ([Section 4.4.2](#)) must indicate whether the SA lookup makes use of the destination IP address, or the destination and source IP addresses, in addition to the SPI. For multicast SAs, the protocol field is not employed for SA lookups. For each inbound, IPsec-protected packet, an implementation must conduct its search of the SAD such that it finds the entry that matches the "longest" SA identifier. In this context, if two or more SAD entries match based on the SPI value, then the entry that also matches based on destination address, or destination and source address (as indicated in the SAD entry) is the "longest" match. This implies a logical ordering of the SAD search as follows:

1. Search the SAD for a match on the combination of SPI, destination address, and source address}. If an SAD entry matches, then process the inbound ESP packet with that matching SAD entry. Otherwise, proceed to step 2.
2. Search the SAD for a match on both SPI and destination address. If the SAD entry matches then process the inbound ESP packet with that matching SAD entry. Otherwise, proceed to step 3.
3. Search the SAD for a match on only {SPI} if the receiver has chosen to maintain a single SPI space for AH and ESP, and on both SPI and protocol otherwise. If an SAD entry matches then process the inbound ESP packet with that matching SAD entry. Otherwise, discard the packet and log an auditable event.

In practice, an implementation may choose any method (or none at all) to accelerate this search, although its externally visible behavior **MUST** be functionally equivalent to having searched the SAD in the above order. For example, a software-based implementation could index into a hash table by the SPI. The SAD entries in each hash table bucket's linked list could be kept sorted to have those SAD entries with the longest SA identifiers first in that linked list. Those SAD entries having the shortest SA identifiers could be sorted so that they are the last entries in the linked list. A hardware-based implementation may be able to effect the longest match search

intrinsically, using commonly available TCAM features.

The indication of whether source and destination address matching is required to map inbound IPsec traffic to SAs MUST be set either as a side effect of manual SA configuration or via negotiation using an SA management protocol, e.g., IKE or GDOI [RFC3547]. Typically, Source-Specific Multicast (SSM) [HC03] groups use a 3-tuple SA identifier composed of an SPI, a destination multicast address, and source address. An Any-Source Multicast group SA requires only an SPI and a destination multicast address as an identifier.

If different classes of traffic (distinguished by DSCP bits [NiBlBaBL98], [Gro02]) are sent on the same SA, and if the receiver is employing the optional anti-replay feature available in both AH and ESP, this could result in inappropriate discarding of lower priority packets due to the windowing mechanism used by this feature. Therefore a sender SHOULD put traffic of different classes, but with the same selector values, on different SAs to support QoS appropriately. To permit this, the IPsec implementation MUST permit establishment and maintenance of multiple SAs between a given sender and receiver, with the same selectors. Distribution of traffic among these parallel SAs to support QoS is locally determined by the sender and is not negotiated by IKE. The receiver MUST process the packets from the different SAs without prejudice.

DISCUSSION: While the DSCP [NiBlBaBL98, Gro02] and ECN [RaFlBl01] fields are not "selectors", as that term is used in this architecture, the sender will need a mechanism to direct packets with a given (set of) DSCP values to the appropriate SA. This mechanism might be termed a "classifier".

As noted above, two types of SAs are defined: transport mode and tunnel mode. IKE creates pairs of SAs, so for simplicity, we choose to require that both SAs in a pair be of the same mode, transport or tunnel.

A transport mode SA is a security association typically employed between a pair of hosts to provide end-to-end security services. When security is desired between two intermediate systems along a path (vs. end-to-end use of IPsec), transport mode MAY be used between security gateways or between a security gateway and a host. In the latter case, transport mode may be used to support in-IP tunneling (e.g., IP-in-IP [Per96] or GRE tunneling [FaLiHaMeTr00]) over transport mode SAs. To clarify, the use of transport mode by an intermediate system (e.g., a security gateway) is permitted only when applied to packets whose source address (for outbound packets) or destination address (for inbound packets) is an address belonging to the intermediate system itself. The access control functions that are

an important part of IPsec are significantly limited in this context, as they cannot be applied to the end-to-end headers of the packets that traverse a transport mode SA used in this fashion. Thus this way of using transport mode should be evaluated carefully before being employed in a specific context.

In IPv4, a transport mode security protocol header appears immediately after the IP header and any options, and before any next layer protocols (e.g., TCP or UDP). In IPv6, the security protocol header appears after the base IP header and selected extension headers, but may appear before or after destination options; it **MUST** appear before next layer protocols (e.g., TCP, UDP, SCTP). In the case of ESP, a transport mode SA provides security services only for these next 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 preceding it, 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 [[Ken04b](#)].

A tunnel mode SA is essentially an SA applied to an IP tunnel, with the access controls applied to the headers of the traffic inside the tunnel. Two hosts **MAY** establish a tunnel mode SA between themselves. Aside from the two exceptions below, 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 typically a tunnel mode SA, as is an SA between a host and a security gateway. The two exceptions are as follows.

- o 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. In this case, the SA terminates at a host (management) function within a security gateway and thus merits different treatment.
- o As noted above, security gateways **MAY** support a transport mode SA to provide security for IP traffic between two intermediate systems along a path, e.g., between a host and a security gateway or between two security gateways.

Several concerns motivate the use of tunnel mode for an SA involving a security gateway. For example, if there are multiple paths (e.g., via different security gateways) to the same destination behind a security gateway, it is important that an IPsec packet be sent to the security gateway with which the SA was negotiated. Similarly, a packet that might be fragmented en-route must have all the fragments delivered to the same IPsec instance for reassembly prior to

cryptographic processing. Also, when a fragment is processed by IPsec and transmitted, then fragmented en-route, it is critical that there be inner and outer headers to retain the fragmentation state data for the pre- and post-IPsec packet formats. Hence there are several reasons for employing tunnel mode when either end of an SA is a security gateway.

Note: AH and ESP cannot be applied using transport mode to IPv4 packets that are fragments. Only tunnel mode can be employed in such cases. For IPv6, it would be feasible to carry a plaintext fragment on a transport mode SA; however, for simplicity, this restriction also applies to IPv6 packets. See [Section 7](#) for more details on handling plaintext fragments on the protected side of the IPsec barrier.

For a tunnel mode SA, there is an "outer" IP header that specifies the IPsec processing source and destination, plus an "inner" IP header that specifies the (apparently) ultimate source and 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 next 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 implementation of IPsec MUST support both transport and tunnel mode. This is true for native, BITS, and BITW implementations for hosts.
- b) A security gateway MUST support tunnel mode and MAY support transport 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, or to provide security between two intermediate systems along a path.

[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, both AH and ESP offer integrity and authentication services, but the coverage differs for each protocol and differs for transport vs. tunnel mode. 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 mutable (non-predictable) parts of the IP or extension headers. However, the same security may be achieved in some contexts by applying ESP to a tunnel carrying a packet.

The granularity of access control provided is determined by the choice of the selectors that define each security association. Moreover, the authentication means employed by IPsec peers, e.g., during creation of an IKE (vs. child) SA also effects the granularity of the access control afforded.

If confidentiality 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 that are carrying traffic from many subscribers.

NOTE: A compliant implementation MUST NOT allow instantiation of an ESP SA that employs both NULL encryption and no integrity algorithm. An attempt to negotiate such an SA is an auditable event by both initiator and responder. The audit log entry for this event SHOULD include the current date/time, local IKE IP address, and remote IKE IP address. The initiator SHOULD record the relevant SPD entry.

4.3 Combining Security Associations

This document does not require support for nested security associations or for what [RFC 2401](#) called "SA bundles." These features still can be effected by appropriate configuration of both the SPD and the local forwarding functions (for inbound and outbound traffic), but this capability is outside of the IPsec module and thus the scope of this specification. As a result, management of nested/bundled SAs is potentially more complex and less assured than under the model implied by [RFC 2401](#). An implementation that provides support for nested SAs SHOULD provide a management interface that enables a user or administrator to express the nesting requirement, and then create the appropriate SPD entries and forwarding table entries to effect the requisite processing. (See [Appendix E](#) for an example of how to configure nested SAs.)

4.4 Major IPsec 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 IPsec functionality, in support of these interoperability and functionality goals. The model described below is nominal; implementations need not match details of this model as presented, but the external behavior of implementations **MUST** correspond to the externally observable characteristics of this model in order to be deemed compliant.

There are three nominal databases in this model: the Security Policy Database (SPD), the Security Association Database (SAD), and the Peer Authorization Database (PAD). The first specifies the policies that determine the disposition of all IP traffic inbound or outbound from a host or security gateway ([Section 4.4.1](#)). The second database contains parameters that are associated with each established (keyed) security association ([Section 4.4.2](#)). The third database, the Peer Authorization Database (PAD) provides a link between an SA management protocol like IKE and the SPD ([Section 4.4.3](#)).

Multiple Separate IPsec Contexts

If an IPsec implementation acts as a security gateway for multiple subscribers, it **MAY** implement multiple separate IPsec contexts. Each context **MAY** have and **MAY** use completely independent identities, policies, key management SAs, and/or IPsec SAs. This is for the most part a local implementation matter. However, a means for associating inbound (SA) proposals with local contexts is required. To this end, if supported by the key management protocol in use, context identifiers **MAY** be conveyed from initiator to responder in the signaling messages, with the result that IPsec SAs are created with a binding to a particular context. For example, a security gateway that provides VPN service to multiple customers will be able to associate each customer's traffic with the correct VPN.

Forwarding vs Security Decisions

The IPsec model described here embodies a clear separation between forwarding (routing) and security decisions, to accommodate a wide range of contexts where IPsec may be employed. Forwarding may be trivial, in the case where there are only two interfaces, or it may be complex, e.g., if there are multiple protected or

unprotected interfaces or if the context in which IPsec is implemented employs a sophisticated forwarding function. IPsec assumes only that outbound and inbound traffic that has passed through IPsec processing is forwarded in a fashion consistent with the context in which IPsec is implemented. Support for nested SAs is optional; if required, it requires coordination between forwarding tables and SPD entries to cause a packet to traverse the IPsec boundary more than once.

"Local" vs "Remote"

In this document, with respect to IP addresses and ports, the terms "Local" and "Remote" are used for policy rules. "Local" refers to the entity being protected by an IPsec implementation, i.e., the "source" address/port of outbound packets or the "destination" address/port of inbound packets. "Remote" refers to a peer entity or peer entities. The terms "source" and "destination" are used for packet header fields.

"Non-initial" vs "Initial" Fragments

Throughout this document, the phrase "non-initial" fragments is used to mean fragments that do not contain all of the selector values that may be needed for access control (e.g., they might not contain Next Layer Protocol, source and destination ports, ICMP message type/code, Mobility Header type). And the phrase "initial" fragment is used to mean a fragment that contains all the selector values needed for access control. However, it should be noted that for IPv6, which fragment contains the Next Layer Protocol and ports (or ICMP message type/code or Mobility Header type) will depend on the kind and number of extension headers present. The "initial" fragment might not be the first fragment, in this context

4.4.1 The Security Policy Database (SPD)

A security association is a management construct used to enforce security policy for traffic crossing the IPsec boundary. 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 specifies minimum management functionality that must be provided, to allow a user or system administrator to control whether and how IPsec is applied to traffic transmitted or received by a host or transiting a security gateway. The SPD, or relevant caches, must be consulted during the processing of ALL traffic (inbound and outbound), including traffic not protected by IPsec, that traverses

the IPsec boundary. This includes IPsec management traffic such as IKE. An IPsec implementation **MUST** have at least one SPD, and it **MAY** support multiple SPDs, if appropriate for the context in which the IPsec implementation operates. There is no requirement to maintain SPDs on a per interface basis, as was specified in [RFC 2401](#). However, if an implementation supports multiple SPDs, then it **MUST** include an explicit SPD selection function, that is invoked to select the appropriate SPD for outbound traffic processing. The inputs to this function are the outbound packet and any local metadata (e.g., the interface via which the packet arrived) required to effect the SPD selection function. The output of the function is an SPD ID.

The SPD is an ordered database, consistent with the use of ACLs or packet filters in firewalls, routers, etc. The ordering requirement arises because entries often will overlap due to the presence of (non-trivial) ranges as values for selectors. Thus a user or administrator **MUST** be able to order the entries to express a desired access control policy. There is no way to impose a general, canonical order on SPD entries, because of the allowed use of wildcards for selector values and because the different types of selectors are not hierarchically related.

Processing Choices: DISCARD, BYPASS, PROTECT

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 PROTECT using IPsec. The first choice refers to traffic that is not allowed to traverse the IPsec boundary (in the specified direction). The second choice refers to traffic that is allowed to cross the IPsec boundary without IPsec protection. The third choice refers to traffic that is afforded IPsec protection, and for such traffic the SPD must specify the security protocols to be employed, their mode, security service options, and the cryptographic algorithms to be used.

SPD-S, SPD-I, SPD-O

An SPD is logically divided into three pieces. The SPD-S (secure traffic) contains entries for all traffic subject to IPsec protection. SPD-O (outbound) contains entries for all outbound traffic that is to be bypassed or discarded. SPD-I (inbound) is applied to inbound traffic that will be bypassed or discarded. All three of these can be decorrelated (with the exception noted above for native host implementations) to facilitate caching. If an

IPsec implementation supports only one SPD, then the SPD consists of all three parts. If multiple SPDs are supported, some of them may be partial, e.g., some SPDs might contain only SPD-I entries, to control inbound bypassed traffic on a per-interface basis. The split allows SPD-I to be consulted without having to consult SPD-S, for such traffic. Since the SPD-I is just a part of the SPD, the same rule applies here, i.e., if a packet that is looked up in the SPD-I cannot be matched to an entry there, then the packet MUST be discarded. Note that for outbound traffic, if a match is not found in SPD-S, then SPD-O must be checked to see if the traffic should be bypassed. Similarly, if SPD-O is checked first and no match is found, then SPD-S must be checked.

SPD entries

Each SPD entry specifies packet disposition as BYPASS, DISCARD, or PROTECT. The entry is keyed by a list of one or more selectors. The SPD contains an ordered list of these entries. The required selector types are defined in [Section 4.4.1.1](#). These selectors are used to define the granularity of the SAs that are created in response to an outbound packet or in response to a proposal from a peer. The detailed structure of an SPD entry is described in [Section 4.4.1.2](#). Every SPD SHOULD have a nominal, final entry that matches anything that is otherwise unmatched, and discards it.

The SPD MUST permit a user or administrator to specify policy entries as follows:

- SPD-I: For inbound traffic that is to be bypassed or discarded, the entry consists of the values of the selectors that apply to the traffic to be bypassed or discarded.
- SPD-O: For outbound traffic that is to be bypassed or discarded, the entry consists of the values of the selectors that apply to the traffic to be bypassed or discarded.
- SPD-S: For traffic that is to be protected using IPsec, the entry consists of the values of the selectors that apply to the traffic to be protected via AH or ESP, controls on how to create SAs based on these selectors, and the parameters needed to effect this protection (e.g., algorithms, modes, etc.). Note that an SPD-S entry also contains information such as "populate from packet" (PFP) flag (see paragraphs below on "How To Derive the Values for an SAD entry") and bits indicating whether the SA lookup makes use of the local and remote IP addresses in addition to the SPI (see AH [[Ken04b](#)] or ESP [[Ken04a](#)] specifications).

Representing directionality in an SPD entry

For traffic protected by IPsec, the Local and Remote address and ports in an SPD entry are swapped to represent directionality, consistent with IKE conventions. In general, the protocols that IPsec deals with have the property of requiring symmetric SAs with flipped Local/Remote IP addresses. However, for ICMP, there is often no such bi-directional authorization requirement. Nonetheless, for the sake of uniformity and simplicity, SPD entries for ICMP are specified in the same way as for other protocols. Note also that for ICMP, Mobility Header, and non-initial fragments, there are no port fields in these packets. ICMP has message type and code and Mobility Header has mobility header type. Thus SPD entries have provisions for expressing access controls appropriate for these protocols, in lieu of the normal port field controls. For bypassed or discarded traffic, separate inbound and outbound entries are supported, e.g., to permit unidirectional flows if required.

OPAQUE and ANY

For each selector in an SPD entry, in addition to the literal values that define a match, there are two special values: ANY and OPAQUE. ANY is a wildcard that matches any value in the corresponding field of the packet, or that matches packets where that field is not present or is obscured. OPAQUE indicates that the corresponding selector field is not available for examination because it may not be present in a fragment, does not exist for the given Next Layer Protocol, or because prior application of IPsec may have encrypted the value. The ANY value encompasses the OPAQUE value. Thus OPAQUE need be used only when it is necessary to distinguish between the case of any allowed value for a field, vs. the absence or unavailability (e.g., due to encryption) of the field.

How To Derive the Values for an SAD entry

For each selector in an SPD entry, the entry specifies how to derive the corresponding values for a new Security Association Database (SAD, see [Section 4.4.2](#)) entry from those in the SPD and the packet. The goal is to allow an SAD entry and an SPD cache entry to be created based on specific selector values from the packet, or from the matching SPD entry. If IPsec processing is specified for an entry, a "populate from packet" (PFP) flag may be asserted for one or more of the selectors in the SPD entry (Local IP address; Remote IP address; Next Layer Protocol; and, depending on Next Layer Protocol, Local port and Remote port, or ICMP type/code, or Mobility Header type). If asserted for a given

selector X, the flag indicates that the SA to be created should take its value for X from the value in the packet. Otherwise, the SA should take its value(s) for X from the value(s) in the SPD entry. Note: In the non-PFP case, the selector values negotiated by the SA management protocol (e.g., IKEv2) may be a subset of those in the SPD entry, depending on the SPD policy of the peer. Also, whether a single flag is used for, e.g., source port, ICMP type/code, and MH type, or a separate flag is used for each, is a local matter.

The following example illustrates the use of the PFP flag in the context of a security gateway or a BITS/BITW implementation. Consider an SPD entry where the allowed value for Remote address is a range of IPv4 addresses: 192.168.2.1 to 192.168.2.10. Suppose an outbound packet arrives with a destination address of 192.168.2.3, and there is no extant SA to carry this packet. The value used for the SA created to transmit this packet could be either of the two values shown below, depending on what the SPD entry for this selector says is the source of the selector value:

PFP flag value for the Remote addr. selector	example of new SAD dest. address selector value
-----	-----
a. PFP TRUE	192.168.2.3 (one host)
b. PFP FALSE	192.168.2.1 to 192.168.2.10 (range of hosts)

Note that if the SPD entry above had a value of ANY for the Remote address, then the SAD selector value would have to be ANY for case (b), but would still be as illustrated for case (a). Thus the PFP flag can be used to prohibit sharing of an SA, even among packets that match the same SPD entry.

Management Interface

For every IPsec implementation, there MUST be a management interface that allows a user or system administrator to manage the SPD. The interface must allow the user (or administrator) to specify the security processing to be applied to every packet that traverses the IPsec boundary. (In a native host IPsec implementation making use of a socket interface, the SPD may not need to be consulted on a per packet basis, as noted above.) The management interface for the SPD MUST allow creation of entries consistent with the selectors defined in [Section 4.4.1.1](#), and MUST support (total) ordering of these entries, as seen via this interface. The SPD entries' selectors are analogous to the ACL or packet filters commonly found in a stateless firewall or packet filtering router and which are currently managed this way.

In host systems, applications MAY be allowed to create SPD entries. (The means of signaling 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. 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.

Decorrelation

The processing model described in this document assumes the ability to decorrelate overlapping SPD entries to permit caching, which enables more efficient processing of outbound traffic in security gateways and BITS/BITW implementations. (Native host implementations have an implicit form of caching available, due to the use of, for example, socket interfaces for applications, and thus there is no requirement to be able to decorrelate SPD entries in these implementations.)

Note: Decorrelation is a means of improving performance and simplifying the processing description; it is not a requirement for a compliant implementation. In this section, unless otherwise noted, the use of "SPD" refers to the body of policy information in both ordered or decorrelated (unordered) state.

[Appendix B](#) provides an algorithm that can be used to decorrelate SPD entries, but any algorithm that produces equivalent output may be used. Note that when an SPD entry is decorrelated all the resulting entries MUST be linked together, so that all members of the group derived from an individual, SPD entry (prior to decorrelation) can all be placed into caches and into the SAD at the same time. For example, suppose one starts with an entry A (from an ordered SPD) that when decorrelated, yields entries A1, A2 and A3. When a packet comes along that matches, say A2, and triggers the creation of an SA, the SA management protocol, e.g., IKEv2, negotiates A. And all 3 decorrelated entries, A1, A2, and A3 are placed in the appropriate SPD-S cache and linked to the SA. The intent is that use of a decorrelated SPD ought not to create more SAs than would have resulted from use of a not-decorrelated SPD. Note also that if a decorrelated SPD is employed, the original entry from the (correlated) SPD should be retained and passed to the SA management protocol, e.g., IKE. Passing the correlated SPD entry to the SA management protocol keeps the use of a decorrelated SPD a local matter, not visible to peers. When acting as a responder, the peer uses a correlated SPD entry for

matching, and for issuing a "narrowed" response. Then the decorrelated entries are used to populate the SPD-S cache.

Handling Changes to the SPD while the System is Running

If a change is made to the SPD while the system is running, a check **SHOULD** be made of the effect of this change on extant SAs. An implementation **MAY** choose to check the impact of an SPD change on extant SAs and to provide a user/administrator with a mechanism for configuring what actions to take, e.g., delete an affected SA, allow an affected SA to continue unchanged, etc.

4.4.1.1 Selectors

An SA 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 Layer Protocol and related fields, e.g., ports), 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 by all IPsec implementations to facilitate control of SA granularity. Note that both Local and Remote addresses should either be IPv4 or IPv6, but not a mix of address types. Also, note that the Local/Remote port selectors (and ICMP message type and code, and Mobility Header type) may be labeled as OPAQUE to accommodate situations where these fields are inaccessible due to packet fragmentation.

- Remote IP Address(es) (IPv4 or IPv6): this is a list of ranges of IP addresses (unicast, anycast, broadcast (IPv4 only), or multicast group). This structure allows expression of a single IP address (via a trivial range), or a list of addresses (each a trivial ranges), or a range of addresses (low and high values, inclusive), as well as the most generic form of a list of ranges. Address ranges are used to support more than one destination system sharing the same SA, e.g., behind a security gateway.
- Local IP Address(es) (IPv4 or IPv6): this is a list of ranges of IP addresses (unicast, anycast, broadcast (IPv4 only), or multicast group). This structure allows expression of a single IP address (via a trivial range), or a list of addresses (each a trivial ranges), or a range of addresses (low and high values, inclusive), as well as the most generic form of a list of

ranges. Address ranges are used to support more than one source system sharing the same SA, e.g., behind a security gateway. Local refers to the address(es) being protected by this implementation (or policy entry).

- Next Layer Protocol: Obtained from the IPv4 "Protocol" or the IPv6 "Next Header" fields. This is an individual protocol number, ANY, or for IPv6 only, OPAQUE. The Next Layer Protocol is whatever comes after any IP extension headers that are present. To simplify locating the Next Layer Protocol, there SHOULD be a mechanism for configuring which IP extension headers to skip. The default configuration for which protocols to skip SHOULD include the following protocols: 0 (Hop-by-hop options), 43 (Routing Header), 44 (Fragmentation Header), and 60 (Destination Options). Note: The default list does NOT include 51 (AH), or 50 (ESP). From a selector lookup point of view, IPsec treats AH and ESP as Next Layer Protocols.

Several additional selectors depend on the Next Layer Protocol value:

- * If the Next Layer Protocol uses two ports (e.g., TCP, UDP, SCTP, these selectors has a list of ranges of values. Note that the Local and Remote ports may not be available in the case of receipt of a fragmented packet or if the port fields have been protected by IPsec (encrypted), thus a value of OPAQUE also MUST be supported. Note: In a non-initial fragment, port values will not be available. If a port selector specifies a value other than ANY or OPAQUE, it cannot match packets that are non-initial fragments. If the SA requires a port value other than ANY or OPAQUE, an arriving fragment without ports MUST be discarded. (See [Section 7](#) Handling Fragments.)
- * If the Next Layer Protocol is a Mobility Header, then there is a selector for IPv6 Mobility Header Message Type (MH type) [[Mobip](#)]. This is an 8-bit value that identifies a particular mobility message. Note that the MH type may not be available in the case of receipt of a fragmented packet. (See [Section 7](#) Handling Fragments.) The IPv6 mobility header message type (MH type) is placed in the most significant eight bits of the 16 bit local "port" selector.
- * If the Next Layer Protocol value is ICMP then there is a 16-bit selector for the ICMP message type and code. The message type is a single 8-bit value, which defines the type of an ICMP message, or ANY. The ICMP code is a single 8-bit value that defines a specific subtype for an ICMP message.

The message type is placed in the most significant 8 bits of the 16-bit selector and the code is placed in the least significant 8 bits. This 16-bit selector can contain a single type and a range of codes, a single type and ANY code, ANY type and ANY code. Given a policy entry with a range of Types (T-start to T-end) and a range of Codes (C-start to C-end), and an ICMP packet with Type t and Code c, an implementation MUST test for a match using

$$(T\text{-start} * 256) + C\text{-start} \leq (t * 256) + c \leq (T\text{-end} * 256) + C\text{-end}$$

Note that the ICMP message type and code may not be available in the case of receipt of a fragmented packet. (See [Section 7 Handling Fragments.](#))

- Name: A name may be used as a symbolic identifier for an IPsec Local or Remote address. Named SPD entries are used in two ways:
 1. A named SPD entry is used by a responder (not an initiator) in support of access control when an IP address would not be appropriate for the Remote IP address selector, e.g., for "road warriors." The name used to match this field is communicated during the IKE negotiation in the ID payload. In this context, the initiator's Source IP address (inner IP header in tunnel mode) is bound to the Remote IP address in the SAD entry created by the IKE negotiation. This address overrides the Remote IP address value in the SPD, when the SPD entry is selected in this fashion. All IPsec implementations MUST support this use of names.
 2. A named SPD entry may be used by an initiator to identify a user for whom an IPsec SA will be created (or for whom traffic may be bypassed). The initiator's IP source address (from inner IP header in tunnel mode) is used to replace the following if and when they are created:

- local address in the SPD cache entry
- local address in the outbound SAD entry
- remote address in the inbound SAD entry

Support for this use is optional for multi-user, native host implementations and not applicable to other implementations. Note that this name is used only locally; it is not communicated by the key management protocol.

An SPD entry can contain both a name (or a list of names) and also values for the Local or Remote IP address. The identifiers

employed in named SPD entries are one of the following four types:

- a. a fully qualified user name string (email), e.g.,
mozart@foo.example.com
(this corresponds to ID_RFC822_ADDR in IKEv2)
- b. a fully qualified DNS name, e.g.,
foo.example.com
(this corresponds to ID_FQDN in IKEv2)
- c. X.500 distinguished name, e.g.,
C = US, SP = MA,
O = BBN Technologies, CN = Stephen T. Kent
(this corresponds to ID_DER_ASN1_DN in IKEv2, after decoding)
- d. a byte string
(this corresponds to Key_ID in IKEv2)

The IPsec implementation context determines how selectors are used. For example, a native host implementation typically makes use of a socket interface. When a new connection is established the SPD can be consulted and an SA bound to the socket. Thus traffic sent via that socket need not result in additional lookups to the SPD (SPD-O and SPD-S) cache. In contrast, a BITS, BITW, or security gateway implementation needs to look at each packet and perform an SPD-O/SPD-S cache lookup based on the selectors.

4.4.1.2 Structure of an SPD entry

This section contains a prose description of an SPD entry. Also, an ASN.1 definition of an SPD entry is provided in [Appendix C](#).

This text describes the SPD in a fashion that maps directly into IKE payloads. One should not create SPD entries that cannot be mapped into something that IKE can negotiate. The management GUI can offer the user other forms of data entry and display, e.g., the option of using address prefixes as well as ranges, and symbolic names for protocols, ports, etc. (Do not confuse the use of symbolic names in a management interface with the SPD selector "Name".) Note that Remote/Local apply only to IP addresses and ports, not to ICMP message type/code or Mobility Header type. Also, if the reserved, symbolic selector value OPAQUE or ANY is employed for a given selector type, only that value may appear in the list for that selector, and it must appear only once in the list for that selector.

Note that ANY and OPAQUE are local syntax conventions -- IKEv2 negotiates these values via the ranges indicated below:

```
ANY:      start = 0          end = <max>
OPAQUE:   start = <max>      end = 0
```

An SPD is an ordered list of entries each of which contains the following fields.

- o Name -- a list of IDs. This selector is optional.
- o PFP flags -- one per traffic selector. A given flag, e.g., for Next Layer Protocol, applies to the relevant selector across all "selector sets" (see below) contained in an SPD entry. When creating an SA, each flag specifies for the corresponding traffic selector whether to instantiate the selector from the corresponding field in the packet that triggered the creation of the SA or from the value(s) in the corresponding SPD entry (see [Section 4.4.1](#), "How To Derive the Values for an SAD entry"). Whether a single flag is used for, e.g., source port, ICMP type/code, and MH type, or a separate flag is used for each, is a local matter. There are PFP flags for:
 - Local Address
 - Remote Address
 - Next Layer Protocol
 - Local Port, or ICMP message type/code or Mobility Header type (depending on the next layer protocol)
 - Remote Port, or ICMP message type/code or Mobility Header type (depending on the next layer protocol)
- o One to N selector sets that correspond to the "condition" for applying a particular IPsec action. Each selector set contains:
 - Local Address
 - Remote Address
 - Next Layer Protocol
 - Local Port, or ICMP message type/code or Mobility Header type (depending on the next layer protocol)
 - Remote Port, or ICMP message type/code or Mobility Header type (depending on the next layer protocol)

NOTE: The "next protocol" selector is an individual value (unlike the local and remote IP addresses) in a selector set entry. This is consistent with how IKE v2 negotiates the TS values for an SA. it also makes sense because one may need to associate different port fields with different protocols. It is possible to associate multiple protocols

(and ports) with a single SA by specifying multiple selector sets for that SA.

- o processing info -- which action is required -- PROTECT, BYPASS, or DISCARD. There is just one action that goes with all the selector sets, not a separate action for each set. If the required processing is PROTECT, the entry contains the following information.
 - IPsec mode -- tunnel or transport
 - extended sequence number -- Is this SA using extended sequence numbers?
 - stateful fragment checking -- Is this SA using stateful fragment checking (see [Section 7](#) for more details)
 - Bypass DF bit (T/F) - applicable to tunnel mode SAs
 - Bypass DSCP (T/F) or map to unprotected DSCP values (array) if needed to restrict bypass of DSCP values
 - IPsec protocol - AH or ESP
 - algorithms -- which ones to use for AH, which ones to use for ESP, which ones to use for combined mode, ordered by decreasing priority

It is a local matter as to what information is kept re: handling extant SAs when the SPD is changed.

4.4.1.3 More re: Fields Associated with Next Layer Protocols

Additional selectors are often associated with fields in the Next Layer Protocol header. A particular Next Layer Protocol can have zero, one, or two selectors. There may be situations where there aren't both local and remote selectors for the fields that are dependent on the Next Layer Protocol. The IPv6 Mobility Header has only a Mobility Header message type. AH and ESP have no further selector fields. A system may be willing to send an ICMP message type and code that it does not want to receive. In the descriptions below, "port" is used to mean a field that is dependent on the Next Layer Protocol.

- A. If a Next Layer Protocol has no "port" selectors, then the Local and Remote "port" selectors are set to OPAQUE, e.g.,

Local's

next layer protocol = AH
"port" selector = OPAQUE

Remote's

next layer protocol = AH
"port" selector = OPAQUE

- B. If a Next Layer Protocol has only one selector, e.g., Mobility Header type, then that field is placed in the Local "port" selector, and the Remote "port" selector is set to OPAQUE, e.g.,

Local's

next layer protocol = Mobility Header
"port" selector = Mobility Header message type

Remote's

next layer protocol = Mobility Header
"port" selector = OPAQUE

- C. If a system is willing to send traffic with a particular "port" value but NOT receive traffic with that kind of port value, the system's traffic selectors are set as follows in the relevant SPD entry:

Local's

next layer protocol = ICMP
"port" selector = <specific ICMP type & code>

Remote's

next layer protocol = ICMP
"port" selector = OPAQUE

- D. To indicate that a system is willing to receive traffic with a particular "port" value but NOT send that kind of traffic, the system's traffic selectors are set as follows in the relevant SPD entry:

Local's

next layer protocol = ICMP
"port" selector = OPAQUE

Remote's

next layer protocol = ICMP
"port" selector = <specific ICMP type & code>

For example, if a security gateway is willing to allow systems behind it to send ICMP traceroutes, but is not willing to let outside systems run ICMP traceroutes to systems behind it, then the security gateway's traffic selectors are set as follows in the relevant SPD entry:

Local's

next layer protocol = 1 (ICMPv4)


```
"port" selector      = 30 (traceroute)

Remote's
  next layer protocol = 1 (ICMPv4)
  "port" selector     = OPAQUE
```

4.4.2 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, each SAD entry is pointed to by entries in the SPD-S part of the SPD cache. For inbound processing, for unicast SAs, the SPI is used either alone to look up an SA, or the SPI may be used in conjunction with the IPsec protocol type. If an IPsec implementation supports multicast, the SPI plus destination address, or SPI plus destination and source addresses are used to look up the SA. (See [Section 4.1](#) for details on the algorithm that MUST be used for mapping inbound IPsec datagrams to SAs.) The following parameters are associated with each entry in the SAD. They should all be present except where otherwise noted, e.g., AH Authentication algorithm. This description does not purport to be a MIB, only a specification of the minimal data items required to support an SA in an IPsec implementation.

For each of the selectors defined in [Section 4.4.1.1](#), the entry for an inbound SA in the SAD MUST contain the value or values negotiated at the time the SA was created. For a receiver, these values are used to check that the header fields of an inbound packet (after IPsec processing) match the selector values negotiated for the SA. For the receiver, this is part of verifying that a packet arriving on an SA is consistent with the policy for the SA. (See [Section 6](#) for rules for ICMP messages.) These fields can have the form of specific values, ranges, ANY, or OPAQUE, as described in [section 4.4.1.1](#), "Selectors."

4.4.2.1 Data Items in the SAD

The following data items MUST be in the SAD:

- o Security Parameter Index (SPI): a 32-bit value selected by the receiving end of an SA to uniquely identify the SA. In an SAD entry for an outbound SA, the SPI is used to construct the packet's AH or ESP header. In an SAD entry for an inbound SA, the SPI is used to map traffic to the appropriate SA (see text on unicast/multicast in [Section 4.1](#)).
- o Sequence Number Counter: a 64-bit used to generate the Sequence Number field in AH or ESP headers. 64-bit sequence numbers are the

default, but 32-bit sequence numbers are also supported if negotiated.

- 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, or whether rollover is permitted. The audit log entry for this event SHOULD include the SPI value, current date/time, Local Address, Remote Address, and the selectors from the relevant SAD entry.
- o Anti-Replay Window: a 64-bit counter and a bit-map (or equivalent) used to determine whether an inbound AH or ESP packet is a replay.

NOTE: If anti-replay has been disabled by the receiver for an SA, e.g., in the case of a manually keyed SA, then the Anti-Replay Window is ignored for the SA in question. 64-bit sequence numbers are the default, but this counter size accommodates 32-bit sequence numbers as well.

- o AH Authentication algorithm, key, etc. This is required only if AH is supported.
- o ESP Encryption algorithm, key, mode, IV, etc. If a combined mode algorithm is used, these fields will not be applicable.
- o ESP integrity algorithm, keys, etc. If the integrity service is not selected, these fields will not be applicable. If a combined mode algorithm is used, these fields will not be applicable.
- o ESP combined mode algorithms, key(s), etc. This data is used when a combined mode (encryption and integrity) algorithm is used with ESP. If a combined mode algorithm is not used, these fields are not applicable.
- 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 with 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 the SA lifetime in this fashion. 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 cryptographic 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 MUST 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 that warns the implementation to initiate action such as setting up a replacement SA; and a hard lifetime when the current SA ends and is destroyed.
 - (c) If the entire packet does not get delivered during the SAs lifetime, the packet SHOULD be discarded.
- o IPsec protocol mode: tunnel or transport. Indicates which mode of AH or ESP is applied to traffic on this SA.
 - o Stateful fragment checking flag. Indicates whether or not stateful fragment checking applies to this SA.
 - o Path MTU: any observed path MTU and aging variables.
 - o Tunnel header IP source and destination address - both addresses must be either IPv4 or IPv6 addresses. The version implies the type of IP header to be used. Only used when the IPsec protocol mode is tunnel.

4.4.2.2 Relationship between SPD, PFP flag, packet, and SAD

For each selector, the following tables show the relationship between the value in the SPD, the PFP flag, the value in the triggering packet and the resulting value in the SAD. Note that the administrative interface for IPsec can use various syntactic options to make it easier for the administrator to enter rules. For example, although a list of ranges is what IKEv2 sends, it might be clearer and less error prone for the user to enter a single IP address or IP address prefix.

Selector	SPD Entry	PFP	Value in Triggering Packet	Resulting SAD Entry
loc addr	list of ranges	0	IP addr "S"	list of ranges
	ANY	0	IP addr "S"	ANY
	list of ranges	1	IP addr "S"	"S"
	ANY	1	IP addr "S"	"S"
rem addr	list of ranges	0	IP addr "D"	list of ranges
	ANY	0	IP addr "D"	ANY
	list of ranges	1	IP addr "D"	"D"
	ANY	1	IP addr "D"	"D"
protocol	list of prot's*	0	prot. "P"	list of prot's*
	ANY**	0	prot. "P"	ANY
	OPAQUE****	0	prot. "P"	OPAQUE
	list of prot's*	0	not avail.	discard packet
	ANY**	0	not avail.	ANY
	OPAQUE****	0	not avail.	OPAQUE
	list of prot's*	1	prot. "P"	"P"
	ANY**	1	prot. "P"	"P"
	OPAQUE****	1	prot. "P"	***
	list of prot's*	1	not avail.	discard packet
	ANY**	1	not avail.	discard packet
	OPAQUE****	1	not avail.	***

If the protocol is one that has two ports then there will be selectors for both Local and Remote ports.

Selector	SPD Entry	PFP	Value in Triggering Packet	Resulting SAD Entry
loc port	list of ranges	0	src port "s"	list of ranges
	ANY	0	src port "s"	ANY
	OPAQUE	0	src port "s"	OPAQUE
	list of ranges	0	not avail.	discard packet
	ANY	0	not avail.	ANY
	OPAQUE	0	not avail.	OPAQUE
	list of ranges	1	src port "s"	"s"
	ANY	1	src port "s"	"s"
	OPAQUE	1	src port "s"	***
	list of ranges	1	not avail.	discard packet
	ANY	1	not avail.	discard packet
	OPAQUE	1	not avail.	***
rem port	list of ranges	0	dst port "d"	list of ranges
	ANY	0	dst port "d"	ANY
	OPAQUE	0	dst port "d"	OPAQUE
	list of ranges	0	not avail	discard packet
	ANY	0	not avail	ANY
	OPAQUE	0	not avail.	OPAQUE
	list of ranges	1	dst port "d"	"d"
	ANY	1	dst port "d"	"d"
	OPAQUE	1	dst port "d"	***
	list of ranges	1	not avail.	discard packet
	ANY	1	not avail.	discard packet
	OPAQUE	1	not avail.	***

If the protocol is mobility header then there will be a selector for mh type.

Selector	SPD Entry	PFP	Value in Triggering Packet	Resulting SAD Entry
-----	-----	---	-----	-----
mh type	list of ranges	0	mh type "T"	list of ranges
	ANY	0	mh type "T"	ANY
	OPAQUE	0	mh type "T"	OPAQUE
	list of ranges	0	not avail.	discard packet
	ANY	0	not avail.	ANY
	OPAQUE	0	not avail.	OPAQUE
	list of ranges	1	mh type "T"	"T"
	ANY	1	mh type "T"	"T"
	OPAQUE	1	mh type "T"	***
	list of ranges	1	not avail.	discard packet
	ANY	1	not avail.	discard packet
	OPAQUE	1	not avail.	***

If the protocol is ICMP, then there will be a 16-bit selector for ICMP type and ICMP code. Note that the type and code are bound to each other, i.e., the codes apply to the particular type. This 16-bit selector can contain a single type and a range of codes, a single type and ANY code, and ANY type and ANY code.

Selector	SPD Entry	PFP	Value in Triggering Packet	Resulting SAD Entry
-----	-----	---	-----	-----
ICMP type and code	a single type & range of codes	0	type "t" & code "c"	single type & range of codes
	a single type & ANY code	0	type "t" & code "c"	single type & ANY code
	ANY type & ANY code	0	type "t" & code "c"	ANY type & ANY code
	OPAQUE	0	type "t" & code "c"	OPAQUE
	a single type & range of codes	0	not avail.	discard packet
	a single type & ANY code	0	not avail.	discard packet
	ANY type & ANY code	0	not avail.	ANY type & ANY code
	OPAQUE	0	not avail.	OPAQUE
	a single type & range of codes	1	type "t" & code "c"	"t" and "c"
	a single type & ANY code	1	type "t" & code "c"	"t" and "c"
	ANY type & ANY code	1	type "t" & code "c"	"t" and "c"
	OPAQUE	1	type "t" & code "c"	***
	a single type & range of codes	1	not avail.	discard packet
	a single type & ANY code	1	not avail.	discard packet
	ANY type & ANY code	1	not avail.	discard packet
	OPAQUE	1	not avail.	***

If the name selector is used...

Selector	SPD Entry	Value in Triggering PFP Packet	Resulting SAD Entry
-----	-----	---	-----
name	list of system- dependent user or sys. names	N/A packet from user or system	N/A

* "List of protocols" is the information, not the way that the SPD or SAD or IKv2 have to represent this information.

** 0 (zero) is used by IKE to indicate ANY for protocol.

*** Use of PFP=1 with an OPAQUE value is an error and SHOULD be prohibited by an IPsec implementation.

**** The protocol field cannot be OPAQUE in IPv4. This table entry applies only to IPv6.

4.4.3 Peer Authorization Database (PAD)

The Peer Authorization Database (PAD) provides a link between an SA management protocol like IKE and the SPD. The PAD indicates the range of identities that an IPv4 or IPv6 peer is authorized to represent when (child) SAs are negotiated with the peer. The identities may be a list of IPv4 or IPv6 address ranges or a set of symbolic names. The IP version of the identities does not have to be the same as that of the IP version of the peer representing them. The fundamental requirement associated with the PAD is that the traffic selectors passed by the SA management protocol for comparison against the SPD MUST be verified as authorized relative to the authenticated peer of the SA management protocol. (See also [Section 4.5.3](#), which levies requirements on the PAD in support of locating security gateways.)

The PAD also specifies how to authenticate each peer, e.g., via shared secret or use of a certificate. If a shared secret is used, the secret is stored here. If a certificate is to be used, a trust anchor for validating the certificate is available via the PAD. The PAD also MAY include data in support of certificate revocation status checking, if this information is not otherwise available from the trust anchor or the peer's certificate. Because the PAD might be incorporated into the SA management protocol implementation, it is not discussed extensively in this document.

4.5 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 service available for AH and ESP requires automated SA management. Moreover, the granularity of key distribution employed with IPsec determines the granularity of authentication provided. In general, data origin authentication in AH and ESP is limited by the extent to which secrets used with the integrity 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.

4.5.1 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 might 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.

4.5.2 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 IKEv2 [[Kau04](#)]. This document assumes the availability of certain functions from the key management protocol which are not supported by IKEv1. Other automated SA management protocols MAY be employed.

When an automated SA/key management protocol is employed, the output from this protocol is used to generate multiple keys for a single SA. This also occurs because distinct keys are used for each of the two SAs created by IKE. If both integrity and confidentiality are employed, then a minimum of four keys are required. Additionally, some cryptographic algorithms may require multiple keys, e.g., 3DES.

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 keys 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 keys MUST be taken from the first (left-most, high-order) bits and the integrity keys MUST be taken from the remaining bits. The number of bits for each key is defined in the relevant cryptographic algorithm specification RFC. In the case of multiple encryption keys or multiple integrity keys, the specification for the cryptographic algorithm must specify the order in which they are to be selected from a single string of bits provided to the cryptographic algorithm.

[4.5.3](#) 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, but the Peer Authorization Database described in [Section 4.4](#) is the most likely candidate. (Note: S* indicates a system that is running IPsec, e.g., SH1 and SG2 below.)

Consider a situation in which a remote host (SH1) 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 his home organization's firewall (SG2). This situation raises several issues:

1. How does SH1 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 SH1 and verify that SH1 is authorized to contact H2?
4. How does SH1 know/learn about any additional gateways that provide alternate paths to H2?

To address these problems, an IPsec-supporting host or security gateway MUST have an administrative interface that allows the user/administrator to configure the address of one or more security gateways for ranges of destination addresses that require its use. This includes the ability to configure information for locating and authenticating one or more security gateways and verifying the authorization of these gateways to represent the destination host. (The authorization function is implied in the PAD.) This document does not address the issue of how to automate the discovery/verification of security gateways. The IP Security Policy (IPSP) Working Group is a possible future source of guidance. One of its goals is to produce an Internet Draft on a "Security Gateway Discovery, Policy Exchange and Negotiation Protocol".

4.6 Security Associations and Multicast

The receiver-orientation of the Security Association implies that, in the case of unicast traffic, the destination system will 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 associated with a single SA. 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 integrity 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 approaches are deferred to the IETF's Multicast Security Working Group.

5. IP Traffic Processing

As mentioned in [Section 4.4.1](#) "The Security Policy Database (SPD)", the SPD (or associated caches) MUST be consulted during the processing of all traffic that crosses the IPsec protection boundary, including IPsec management traffic. If no policy is found in the SPD that matches a packet (for either inbound or outbound traffic), the packet MUST be discarded. To simplify processing, and to allow for very fast SA lookups (for SG/BITS/BITW), this document introduces the notion of an SPD cache for all outbound traffic (SPD-O plus SPD-S), and a cache for inbound, non-IPsec-protected traffic (SPD-I). There is nominally one cache per SPD. Since SPD entries may overlap, one cannot safely cache these entries in general. Simple caching might result in a match against a cache entry whereas an ordered search of the SPD would have resulted in a match against a different entry. But, if the SPD entries are first decorrelated, then the resulting entries can safely be cached, and each cached entry will map to exactly one SA, or indicate that matching traffic should be bypassed or discarded, appropriately. (Note: The original SPD entry might result in multiple SAs, e.g., because of PFP.) Unless otherwise noted, all references below to the "SPD" or "SPD cache" or "cache" are to a decorrelated SPD (SPD-I, SPD-O, SPD-S) or the SPD cache containing entries from the decorrelated SPD.

Note: 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. This provides an implicit caching mechanism and the portions of the preceding discussion that address caching can be ignored in such implementations.

Note: It is assumed that one starts with a correlated SPD because that is how users and administrators are accustomed to managing these sorts of access control lists or firewall filter rules. Then the decorrelation algorithm is applied to build a list of cache-able SPD entries. The decorrelation is invisible at the management interface.

For inbound IPsec traffic, the SAD entry selected by the SPI serves as the cache for the selectors to be matched against arriving IPsec packets, after AH or ESP processing has been performed.

5.1 Outbound IP Traffic Processing (protected-to-unprotected)

First consider the path for traffic entering the implementation via a protected interface and exiting via an unprotected interface.

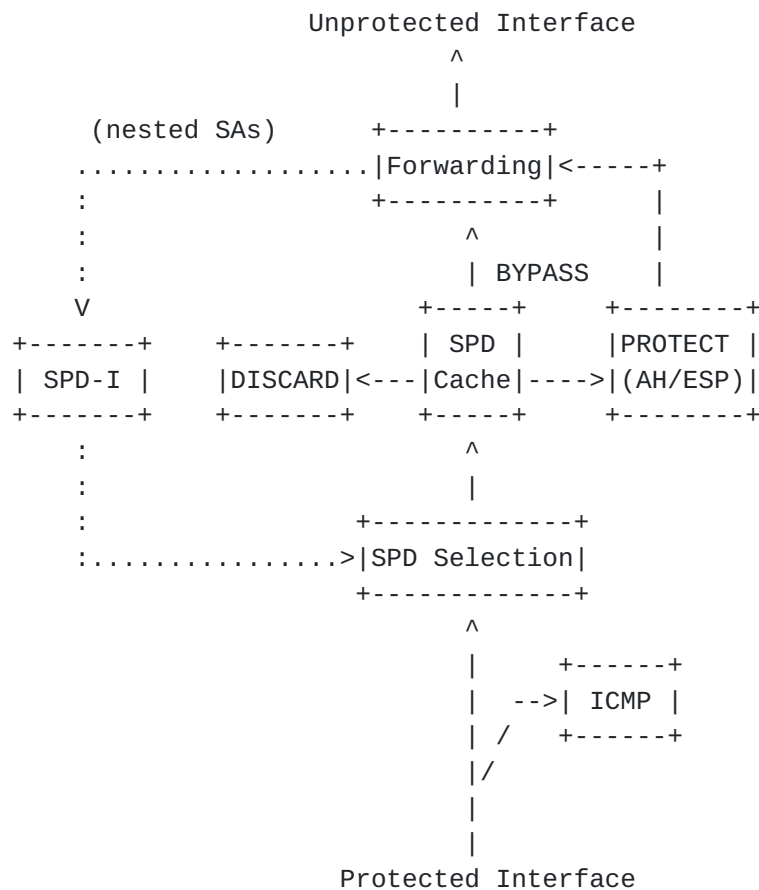


Figure 2. Processing Model for Outbound Traffic

IPsec MUST perform the following steps when processing outbound packets:

1. When a packet arrives from the subscriber (protected) interface, invoke the SPD selection function to obtain the SPD-ID needed to choose the appropriate SPD. (If the implementation uses only one SPD, this step is a no-op.)
2. Match the packet headers against the cache for the SPD specified by the SPD-ID from step 1. Note that this cache contains entries from SPD-0 and SPD-S.
- 3a. If there is a match, then process the packet as specified by the matching cache entry, i.e., BYPASS, DISCARD, or PROTECT using AH or ESP. If IPsec processing is applied, there is a link from the SPD cache entry to the relevant SAD entry (specifying the mode, cryptographic algorithms, keys, SPI, PMTU, etc.). IPsec processing is as previously defined, for tunnel or transport modes

and for AH or ESP, as specified in their respective RFCs [Ken04b and Ken04a]. Note that the SA PMTU value, plus the value of the stateful fragment checking flag (and the DF bit in the IP header of the outbound packet) determine whether the packet can (must) be fragmented prior to or after IPsec processing, or if it must be dropped and an ICMP PMTU message is sent.

- 3b. If no match is found in the cache, search the SPD (SPD-S and SPD-O parts) specified by SPD-ID. If the SPD entry calls for BYPASS or DISCARD, create one or more new outbound SPD cache entries and if BYPASS, create one or more new inbound SPD cache entries. (More than one cache entry may be created since a decorrelated SPD entry may be linked to other such entries that were created as a side effect of the decorrelation process.) If the SPD entry calls for PROTECT, i.e., creation of an SA, the key management mechanism (e.g., IKEv2) is invoked to create the SA. If SA creation succeeds, a new outbound (SPD-S) cache entry is created, along with outbound and inbound SAD entries, otherwise the packet is discarded. (A packet that triggers an SPD lookup MAY be discarded by the implementation, or it may be processed against the newly created cache entry, if one is created.) Since SAs are created in pairs, an SAD entry for the corresponding inbound SA also is created, and it contains the selector values derived from the SPD entry (and packet, if any PFP flags were "true") used to create the inbound SA, for use in checking inbound traffic delivered via the SA.
4. The packet is passed to the outbound forwarding function (operating outside of the IPsec implementation), to select the interface to which the packet will be directed. This function may cause the packet to be passed back across the IPsec boundary, for additional IPsec processing, e.g., in support of nested SAs. If so, there MUST be an entry in SPD-I database that permits inbound bypassing of the packet, otherwise the packet will be discarded.

NOTE: With the exception of IPv4 and IPv6 transport mode, an SG, BITS, or BITW implementation MAY fragment packets before applying IPsec. The device SHOULD have a configuration setting to disable this. The resulting fragments are evaluated against the SPD in the normal manner. Thus, fragments not containing port numbers (or ICMP message type and code, or Mobility Header type) will match rules only having port (or ICMP message type and code, or MH type) selectors of OPAQUE or ANY. (See [section 7](#) for more details.)

NOTE: With regard to determining and enforcing the PMTU of an SA, the IPsec system MUST follow the steps described in [Section 8.2](#).

5.1.1 Handling an Outbound Packet That Must Be Discarded

If an IPsec system receives an outbound packet that it finds it must discard, it SHOULD be capable of generating and sending an ICMP message to indicate to the sender of the outbound packet that the packet was discarded. The type and code of the ICMP message will depend on the reason for discarding the packet, as specified below. The reason SHOULD be recorded in the audit log. The audit log entry for this event SHOULD include the reason, current date/time, and the selector values from the packet.

- a. The selectors of the packet matched an SPD entry requiring the packet to be discarded.

IPv4 Type = 3 (destination unreachable) Code = 13
(Communication Administratively Prohibited)

IPv6 Type = 1 (destination unreachable) Code = 1
(Communication with destination administratively prohibited)

- b1. The IPsec system was unable to set up the SA required by the SPD entry matching the packet because the IPsec peer at the other end of the exchange is administratively prohibited from communicating with the initiator.

IPv4 Type = 3 (destination unreachable) Code = 13
(Communication Administratively Prohibited)

IPv6 Type = 1 (destination unreachable) Code = 1
(Communication with destination administratively prohibited)

- b2. The IPsec system was unable to set up the SA required by the SPD entry matching the packet because the IPsec peer at the other end of the exchange could not be contacted.

IPv4 Type = 3 (destination unreachable) Code = 1 (host unreachable)

IPv6 Type = 1 (destination unreachable) Code = 3 (address unreachable)

Note that an attacker behind a security gateway could send packets with a spoofed source address, W.X.Y.Z, to an IPsec entity causing it to send ICMP messages to W.X.Y.Z. This creates an opportunity for a DoS attack among hosts behind a security gateway. To address this, a security gateway SHOULD include a management control to allow an

administrator to configure an IPsec implementation to send or not send the ICMP messages under these circumstances, and if this facility is selected, to rate limit the transmission of such ICMP responses.

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, with regard to outbound traffic processing. This includes how to construct the encapsulating (outer) IP header, how to process fields in the inner IP header, and what other actions should be taken for outbound, tunnel mode traffic. The general processing described here is modeled after [RFC 2003](#), "IP Encapsulation with IP" [[Per96](#)]:

- 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 as noted below for TTL (or Hop Limit) and the DS/ECN Fields. The inner IP header otherwise 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.

Note: IPsec tunnel mode is different from IP-in-IP tunneling ([RFC 2003](#)) in several ways:

- o IPsec offers certain controls to a security administrator to manage covert channels (which would not normally be a concern for tunneling) and to ensure that the receiver examines the right portions of the received packet re: application of access controls. An IPsec implementation MAY be configurable with regard to how it processes the DS field for tunnel mode for transmitted packets. For outbound traffic, one configuration setting for the DS field will operate as described in the following sections on IPv4 and IPv6 header processing for IPsec tunnels. Another will allow the DS field to be mapped to a fixed value, which MAY be configured on a per SA basis. (The value might really be fixed for all traffic outbound from a device, but per SA granularity allows that as well.) This configuration option allows a local

administrator to decide whether the covert channel provided by copying these bits outweighs the benefits of copying.

- o IPsec describes how to handle ECN or DS.
- o IPsec allows the IP version of the encapsulating header to be different from that of the inner header.

The tables in the following sub-sections show the handling for the different header/option fields ("constructed" means that 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
DS Field	copied from inner hdr (5)	no change
ECN Field	copied from inner hdr	constructed (6)
total length	constructed	no change
ID	constructed	no change
flags (DF,MF)	constructed, DF (4)	no change
fragment offset	constructed	no change
TTL	constructed (2)	decrement (2)
protocol	AH, ESP	no change
checksum	constructed	constructed (2)(6)
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 IPv4 checksum changes when the TTL changes.)

Note: Decrementing the TTL value is a normal part of forwarding a packet. Thus, a packet originating from the same node as the encapsulator does not have its TTL decremented, since the sending node is originating the packet rather than forwarding it.

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

Note: For multicast traffic, the destination address, or source and destination addresses, may be required for demuxing. In that case, it is important to ensure consistency over the lifetime of the SA by ensuring that the source address that appears in the encapsulating tunnel header is the same as the one that was negotiated during the SA establishment process. There is an exception to this general rule, i.e., a mobile IPsec implementation will update its source address as it moves.

4. configuration determines whether to copy from the inner header (IPv4 only), clear, or set the DF.
5. If the packet will immediately enter a domain for which the DSCP value in the outer header is not appropriate, that value MUST be mapped to an appropriate value for the domain [[RFC 2474](#)]. See [RFC 2475](#)[BBCDWW98] for further information.
6. If the ECN field in the inner header is set to ECT(0) or ECT(1) and the ECN field in the outer header is set to CE, then set the ECN field in the inner header to CE, otherwise make no change to the ECN field in the inner header. (The IPv4 checksum changes when the ECN changes.)

Note: IPsec does not copy the options from the inner header into the outer header, nor does IPsec construct the options in the outer header. However, post-IPsec code MAY insert/construct options for the outer header.

5.1.2.2 IPv6 -- Header Construction for Tunnel Mode

See previous [section 5.1.2.1](#) for notes 1-6 indicated by (footnote number).

<-- How Outer Hdr Relates Inner Hdr --->		
	Outer Hdr at	Inner Hdr at
IPv6	Encapsulator	Decapsulator
Header fields:	-----	-----
version	6 (1)	no change
DS Field	copied from inner hdr (5)	no change
ECN Field	copied from inner hdr	constructed (6)
flow label	copied or configured	no change
payload length	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 (7)	no change

7. IPsec does not copy the extension headers from the inner packet into outer headers, nor does IPsec construct extension headers in the outer header. However, post-IPsec code MAY insert/construct extension headers for the outer header.

5.2 Processing Inbound IP Traffic (unprotected-to-protected)

Inbound processing is somewhat different from outbound processing, because of the use of SPIs to map IPsec protected traffic to SAs. The inbound SPD cache (SPD-I) is applied only to bypassed or discarded traffic. If an arriving packet appears to be an IPsec fragment from an unprotected interface, reassembly is performed prior to IPsec processing. The intent for any SPD cache is that a packet that fails to match any entry is then referred to the corresponding SPD. Every SPD SHOULD have a nominal, final entry that catches anything that is otherwise unmatched, and discards it. This ensures that non-IPsec protected traffic that arrives and does not match any SPD-I entry will be discarded.

IPsec MUST perform the following steps:

1. When a packet arrives, it may be tagged with the ID of the interface (physical or virtual) via which it arrived, if necessary to support multiple SPDs and associated SPD-I caches. (The interface ID is mapped to a corresponding SPD-ID.)
2. The packet is examined and demuxed into one of two categories:
 - If the packet appears to be IPsec protected and it is addressed to this device, an attempt is made to map it to an active SA via the SAD. Note that the device may have multiple IP addresses that may be used in the SAD lookup, e.g., in the case of protocols such as SCTP.
 - Traffic not addressed to this device, or addressed to this device and not AH or ESP, is directed to SPD-I lookup. (This implies that IKE traffic MUST have an explicit BYPASS entry in the SPD.) If multiple SPDs are employed, the tag assigned to the packet in step 1 is used to select the appropriate SPD-I (and cache) to search. SPD-I lookup determines whether the action is DISCARD or BYPASS.
- 3a. If the packet is addressed to the IPsec device and AH or ESP is specified as the protocol, the packet is looked up in the SAD. For unicast traffic, use only the SPI (or SPI plus protocol). For multicast traffic, use the SPI plus the destination or SPI plus destination and source addresses, as specified in [section 4.1](#) event. The audit log entry for this event SHOULD include the current date/time, SPI, source and destination of the packet, IPsec protocol, and any other selector values of the packet that are available. If the packet is found in the SAD, process it accordingly (see step 4).
- 3b. If the packet is not addressed to the device or is addressed to this device and is not AH or ESP, look up the packet header in the (appropriate) SPD-I cache. If there is a match and the packet is to be discarded or bypassed, do so. If there is no cache match, look up the packet in the corresponding SPD-I and create a cache entry as appropriate. (No SAs are created in response to receipt of a packet that requires IPsec protection; only BYPASS or DISCARD entries can be created this way.) If there is no match, discard the traffic. This is an auditable event. The audit log entry for this event SHOULD include the current date/time, SPI if available, IPsec protocol if available, source and destination of the packet, and any other selector values of the packet that are available.
- 3c. Processing of ICMP messages is assumed to take place on the unprotected side of the IPsec boundary. Unprotected ICMP messages are examined and local policy is applied to determine whether to accept or reject these messages and, if accepted, what action to take as a result. For example, if an ICMP unreachable message is

received, the implementation must decide whether to act on it, reject it, or act on it with constraints. (See [Section 6.](#))

4. Apply AH or ESP processing as specified, using the SAD entry selected in step 3a above. Then match the packet against the inbound selectors identified by the SAD entry to verify that the received packet is appropriate for the SA via which it was received.

If an IPsec system receives an inbound packet on an SA and the packet's header fields are not consistent with the selectors for the SA, it **MUST** discard the packet. This is an auditable event. The audit log entry for this event **SHOULD** include the current date/time, SPI, IPsec protocol(s), source and destination of the packet, and any other selector values of the packet that are available, and the selector values from the relevant SAD entry. The system **SHOULD** also be capable of generating and sending an IKE notification of **INVALID_SELECTORS** to the sender (IPsec peer), indicating that the received packet was discarded because of failure to pass selector checks.

To minimize the impact of a DoS attack, or a mis-configured peer, the IPsec system **SHOULD** include a management control to allow an administrator to configure the IPsec implementation to send or not send this IKE notification, and if this facility is selected, to rate limit the transmission of such notifications.

After traffic is bypassed or processed through IPsec, it is handed to the inbound forwarding function for disposition. This function may cause the packet to be sent (outbound) across the IPsec boundary for additional inbound IPsec processing, e.g., in support of nested SAs. If so, then as with ALL outbound traffic that is to be bypassed, the packet **MUST** be matched against an SPD-0 entry. Ultimately, the packet should be forwarded to the destination host or process for disposition.

6. ICMP Processing

This section describes IPsec handling of ICMP traffic. There are two categories of ICMP traffic: error messages (e.g., type = destination unreachable) and non-error messages (e.g., type = echo). This section applies exclusively to error messages. Disposition of non-error, ICMP messages (that are not addressed to the IPsec implementation itself) **MUST** be explicitly accounted for using SPD entries.

The discussion in this section applies to ICMPv6 as well as to ICMPv4. Also, a mechanism **SHOULD** be provided to allow an administrator to cause ICMP error messages (selected, all, or none)

to be logged as an aid to problem diagnosis.

6.1 Processing ICMP Error Messages Directed to an IPsec Implementation

6.1.1 ICMP Error Messages Received on the Unprotected Side of the Boundary

Figure 3 in [Section 5.2](#) shows a distinct ICMP processing module on the unprotected side of the IPsec boundary, for processing ICMP messages (error or otherwise) that are addressed to the IPsec device and that are not protected via AH or ESP. An ICMP message of this sort is unauthenticated and its processing may result in denial or degradation of service. This suggests that, in general, it would be desirable to ignore such messages. However, many ICMP messages will be received by hosts or security gateways from unauthenticated sources, e.g., routers in the public Internet. Ignoring these ICMP messages can degrade service, e.g., because of a failure to process PMTU message and redirection messages. Thus there is also a motivation for accepting and acting upon unauthenticated ICMP messages.

To accommodate both ends of this spectrum, a compliant IPsec implementation **MUST** permit a local administrator to configure an IPsec implementation to accept or reject unauthenticated ICMP traffic. This control **MUST** be at the granularity of ICMP type and **MAY** be at the granularity of ICMP type and code. Additionally, an implementation **SHOULD** incorporate mechanisms and parameters for dealing with such traffic. For example, there could be the ability to establish a minimum PMTU for traffic (on a per destination basis), to prevent receipt of an unauthenticated ICMP from setting the PMTU to a trivial size.

If an ICMP PMTU message passes the checks above and the system is configured to accept it, then it should be processed as described in [Section 8.2](#).

6.1.2 ICMP Error Messages Received on the Protected Side of the Boundary

These ICMP messages are not authenticated, but they do come from sources on the protected side of the IPsec boundary. Thus these messages generally are viewed as more "trustworthy" than their counterparts arriving from sources on the unprotected side of the boundary. The major security concern here is that a compromised host or router might emit erroneous ICMP error messages that could degrade service for other devices "behind" the security gateway, or that could even result in violations of confidentiality. For example, if a bogus ICMP redirect were consumed by a security gateway, it could cause the forwarding table on the protected side of the boundary to

be modified so as to deliver traffic to an inappropriate destination "behind" the gateway. Thus implementers MUST provide controls to allow local administrators to constrain the processing of ICMP error messages received on the protected side of the boundary, and directed to the IPsec implementation. These controls are of the same type as those employed on the unprotected side, described above in [Section 6.1.1](#).

6.2 Processing Protected, Transit ICMP Error Messages

When an ICMP error message is transmitted via an SA to a device "behind" an IPsec implementation, both the payload and the header of the ICMP message require checking from an access control perspective. If one of these messages is forwarded to a host behind a security gateway, the receiving host IP implementation will make decisions based on the payload, i.e., the header of the packet that purportedly triggered the error response. Thus an IPsec implementation MUST be configurable to check that this payload header information is consistent with the SA via which it arrives. (This means that the payload header, with source and destination address and port fields reversed, matches the traffic selectors for the SA.) If this sort of check is not performed, then for example, anyone with whom the receiving IPsec system (A) has an active SA could send an ICMP destination dead message that refers to any host/net with which A is currently communicating, and thus effect a highly efficient DoS attack re: communication with other peers of A. Normal IPsec receiver processing of traffic is not sufficient to protect against such attacks. However, not all contexts may require such checks, so it is also necessary to allow a local administrator to configure an implementation to NOT perform such checks.

To accommodate both policies, the following convention is adopted. If an administrator wants to allow ICMP error messages to be carried by an SA without inspection of the payload, then configure an SPD entry that explicitly allows for carriage of such traffic. If an administrator wants IPsec to check the payload of ICMP error messages for consistency, then do not create any SPD entries that accommodate carriage of such traffic based on the ICMP packet header. This convention motivates the following processing description.

IPsec senders and receivers MUST support the following processing for ICMP error messages that are sent and received via SAs.

If an SA exists that accommodates an outbound ICMP error message, then the message is mapped to the SA and only the ICMP header is checked upon receipt, just as would be the case for other traffic. If no SA exists that matches the traffic selectors associated with an ICMP error message, then the SPD is searched to determine if such an

SA can be created. If so, the SA is created and the ICMP error message is transmitted via that SA. Upon receipt, this message is subject to the usual traffic selector checks at the receiver. This processing is exactly what would happen for traffic in general, and thus does not represent any special processing for ICMP error messages.

If no SA exists that would carry the outbound ICMP message in question, and if no SPD entry would allow carriage of this outbound ICMP error message, then an IPsec implementation **MUST** map the message to the SA that would carry the return traffic associated with the packet that triggered the ICMP error message. This requires an IPsec implementation to detect outbound ICMP error messages that map to no extant SA or SPD entry, and treat them specially with regard to SA creation and lookup. The implementation extracts the header for the packet that triggered the error (from the ICMP message payload), reverses the source and destination IP address fields, extracts the protocol field, and reverses the port fields (if accessible). It then uses this extracted information to locate an appropriate, active outbound SA, and transmits the error message via this SA. If no such SA exists, no SA will be created, and this is an auditable event.

If an IPsec implementation receives an inbound ICMP error message on an SA, and the header of the message does not match the traffic selectors for the SA, the receiver **MUST** process the received message in a special fashion. Specifically, the receiver must extract the header of the triggering packet from the ICMP payload, and reverse fields as described above to determine if the packet is consistent with the selectors for the SA via which the ICMP error message was received. If the packet fails this check, the IPsec implementation **MUST NOT** forward the ICMP message to the destination. This is an auditable event.

7. Handling Fragments (on the protected side of the IPsec boundary)

Earlier sections of this document describe mechanisms for (a) fragmenting an outbound packet after IPsec processing has been applied and reassembling it at the receiver before IPsec processing and (b) handling inbound fragments received from the unprotected side of the IPsec boundary. This section describes how an implementation should handle the processing of outbound plaintext fragments on the protected side of the IPsec boundary. (See [Appendix D](#) for discussion of Fragment Handling Rationale.) In particular, it addresses:

- o mapping an outbound non-initial fragment to the right SA (or finding the right SPD entry)
- o verifying that a received non-initial fragment is authorized for the SA via which it was received

- o mapping outbound and inbound non-initial fragments to the right SPD-O/SPD-I entry or the relevant cache entry, for BYPASS/DISCARD traffic

Note: In [Section 4.1](#), transport mode SAs have been defined to not carry fragments (IPv4 or IPv6). Note also that in [Section 4.4.1](#), two special values, ANY and OPAQUE, were defined for selectors and that ANY includes OPAQUE.

Note: The term "non-initial fragment" is used here to indicate a fragment that does not contain all the selector values that may be needed for access control. As observed in [Section 4.4.1](#), depending on the Next Layer Protocol, in addition to Ports, the ICMP message type/code or Mobility Header type could be missing from non-initial fragments. Also, for IPv6, even an initial fragment might NOT contain the Next Layer Protocol or Ports (or ICMP message type/code, or Mobility Header type) depending on the kind and number of extension headers present. If a non-initial fragment contains the Port (or ICMP type and code or Mobility header type) but not the Next Layer Protocol, then unless there is an SPD entry for the relevant Local/Remote addresses with ANY for Next Layer Protocol and Port (or ICMP type and code or Mobility header type), the fragment would not contain all the selector information needed for access control.

To address the above issues, three approaches have been defined:

- o Tunnel mode SAs that carry initial and non-initial fragments (See [Section 7.1](#))
- o Separate tunnel mode SAs for non-initial fragments (See [Section 7.2](#))
- o Stateful fragment checking (See [Section 7.3](#))

[7.1](#) Tunnel Mode SAs that Carry Initial and Non-Initial Fragments

All implementations MUST support tunnel mode SAs that are configured to pass traffic without regard to port field (or ICMP type/code or Mobility Header type) values. If the SA will carry traffic for specified protocols, the selector set for the SA MUST specify the port fields (or ICMP type/code or Mobility Header type) as ANY. An SA defined in this fashion will carry all traffic including initial and non-initial fragments for the indicated Local/Remote addresses and specified Next Layer protocol(s). If the SA will carry traffic without regard to a specific protocol value (i.e., ANY is specified as the (Next Layer) protocol selector value), then the port field values are undefined and MUST be set to ANY as well. (As noted in 4.4.1, ANY includes OPAQUE as well as all specific values.)

7.2 Separate Tunnel Mode SAs for Non-Initial Fragments

All implementations MAY support tunnel mode SAs that will carry only non-initial fragments, separate from non-fragmented packets and initial fragments. The OPAQUE value will be used to specify port (or ICMP type/code or Mobility Header type) field selectors for an SA to carry such fragments. Receivers MUST perform a minimum offset check on IPv4 (non-initial) fragments to protect against overlapping fragment attacks when SAs of this type are employed. Because such checks cannot be performed on IPv6 non-initial fragments, users and administrators are advised that carriage of such fragments may be dangerous, and implementers may choose to NOT support such SAs for IPv6 traffic. Also, because an SA of this sort will carry ALL non-initial fragments that match a specified Local/Remote address pair and protocol value, users and administrators are advised to protect such traffic using ESP (with integrity) and the "strongest" integrity and encryption algorithms available at both peers. (Determination of the "strongest" algorithms requires imposing an ordering of the available algorithms, a local determination at the discretion of the initiator of the SA.)

Specific port (or ICMP type/code or Mobility header type) selector values will be used to define SAs to carry initial fragments and non-fragmented packets. This approach can be used if a user or administrator wants to create one or more tunnel mode SAs between the same Local/Remote addresses that discriminate based on port (or ICMP type/code or Mobility header type) fields. These SAs MUST have non-trivial protocol selector values, otherwise approach #1 above MUST be used.

Note: In general, for approach 2, one needs only a single SA between two implementations to carry all non-initial fragments. However, if one chooses to have multiple SAs between the two implementations for QoS differentiation, then one might also want multiple SAs to carry fragments-without-ports, one for each supported QoS class. Since support for QoS via distinct SAs is a local matter, not mandated by 2401bis, the choice to have multiple SAs to carry non-initial fragments should also be local.

7.3 Stateful Fragment Checking

An implementation MAY support some form of stateful fragment checking for a tunnel mode SA with non-trivial port (or ICMP type/code or MH type) field values (not ANY or OPAQUE). Implementations that will transmit non-initial fragments on a tunnel mode SA that makes use of non-trivial port (or ICMP type/code or MH type) selectors MUST notify a peer via the IKE NOTIFY NON_FIRST_FRAGMENTS_ALSO payload.

The peer MUST reject this proposal if it will not accept non-initial fragments in this context. If an implementation does not successfully negotiate transmission of non-initial fragments for such an SA, it MUST NOT send such fragments over the SA. This standard does not specify how peers will deal with such fragments, e.g., via reassembly or other means, at either sender or receiver. However, a receiver MUST discard non-initial fragments that arrive on an SA with non-trivial port (or ICMP type/code or MH type) selector values unless this feature has been negotiated. Also, the receiver MUST discard non-initial fragments that do not comply with the security policy applied to the overall packet. Discarding such packets is an auditable event. Note that in network configurations where fragments of a packet might be sent or received via different security gateways or BITW implementations, stateful strategies for tracking fragments may fail.

7.4 BYPASS/DISCARD traffic

An implementation MUST support DISCARD of fragments using the normal SPD packet classification mechanisms. An implementation MUST support stateful fragment checking to accommodate BYPASS traffic for which a non-trivial port range is specified. The concern is that BYPASS of a cleartext, non-initial fragment arriving at an IPsec implementation could undermine the security afforded IPsec-protected traffic directed to the same destination. For example, consider an IPsec implementation configured with an SPD entry that calls for IPsec-protection of traffic between a specific source/destination address pair, and for a specific protocol and destination port, e.g., TCP traffic on port 25 (Telnet). Assume that the implementation also allows BYPASS of traffic from the same source/destination address pair and protocol, but for a different destination port, e.g., port 119 (NNTP). An attacker could send a non-initial fragment (with a forged source address) that, if bypassed, could overlap with IPsec-protected traffic from the same source and thus violate the integrity of the IPsec-protected traffic. Requiring stateful fragment checking for BYPASS entries with non-trivial port ranges prevents attacks of this sort.

8. Path MTU/DF Processing

The application of AH or ESP to an outbound packet increases the size of a packet and thus may cause a packet to exceed the PMTU for the SA via which the packet will travel. An IPsec implementation also may receive an unprotected ICMP PMTU message and, if it choose to act upon it, the result will affect outbound traffic processing. This section describes the processing required of an IPsec implementation to deal with these two PMTU issues.

8.1 DF Bit

An IPsec implementation **MUST** support the option of copying the DF bit from an outbound packet to the tunnel mode header that it emits, when traffic is carried via a tunnel mode SA. This means that it **MUST** be possible to configure the system's treatment of the DF bit (set, clear, copy from inner header) for each SA.

8.2 Path MTU Discovery (PMTU)

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

IPv4 ([RFC 792](#) [[Pos81b](#)]):

- 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 (labeled "unused" in [RFC 792](#)), with high-order 16 bits set to zero)

IPv6 ([RFC 2463](#) [[CD98](#)]):

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

8.2.1 Propagation of PMTU

When an IPsec implementation receives an unauthenticated PMTU message, and it is configured to process (vs. ignore) such messages, it maps the message to the SA to which it corresponds. This mapping is effected by extracting the header information from the payload of the PMTU message and applying the procedure described in [Section 5.2](#). The PMTU determined by this message is used to update the SAD PMTU field, taking into account the size of the AH or ESP header that will be applied, any crypto synchronization data, and the overhead imposed by an additional IP header, in the case of a tunnel mode SA.

In a native host implementation it is possible to maintain PMTU data at the same granularity as for unprotected communication, so there is no loss of functionality. Signaling of the PMTU information is internal to the host. For all other IPsec implementation options, the PMTU data must be propagated via a synthesized ICMP PMTU. In these cases, the IPsec implementation **SHOULD** wait for outbound traffic to be mapped to the SAD entry. When such traffic arrives, if the traffic would exceed the updated PMTU value the traffic **MUST** be discarded and an appropriate ICMP PMTU message sent.

8.2.2 PMTU Aging

In all IPsec implementations the PMTU associated with an SA MUST be "aged" and some mechanism is required to update the PMTU in a timely manner, especially for discovering if the PMTU is smaller than it should be. A given PMTU has to remain in place long enough for a packet to get from the source of the security association to peer, and to propagate an ICMP error message if the current PMTU is too big.

Systems SHOULD use the approach described in the Path MTU Discovery document ([RFC 1191](#) [[MD90](#)], 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.

9. 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 this document 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.

10. Conformance Requirements

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

11. Security Considerations

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

If an IPsec implementation is configured to pass ICMP error messages over SAs based on the ICMP header values, without checking the header information from the ICMP message payload, serious vulnerabilities may arise. Consider a scenario in which several sites (A, B, and C) are connected to one another via ESP-protected tunnels: A-B, A-C, and B-C. Also assume that the traffic selectors for each tunnel specify ANY for protocol and port fields and IP source/destination address

ranges that encompass the address range for the systems behind the security gateways serving each site. This would allow a host at site B to send an ICMP destination dead message to any host at site A, that declares all hosts on the net at site C to be unreachable. This is a very efficient DoS attack that could have been prevented if the ICMP error messages were subjected to the checks that IPsec provides, if the SPD is suitably configured, as described in [Section 6.2](#).

12. IANA Considerations

This document has no actions for IANA.

13. Differences from [RFC 2401](#)

This architecture document differs substantially from [RFC 2401](#) in detail and in organization, but the fundamental notions are unchanged.

- o The processing model has been revised to address new IPsec scenarios, improve performance and simplify implementation. This includes a separation between forwarding (routing) and SPD selection, several SPD changes, and the addition of an outbound SPD cache and an inbound SPD cache for bypassed or discarded traffic. There is also a new database, the Peer Authorization Database (PAD). This provides a link between an SA management protocol like IKE and the SPD
- o There is no longer a requirement to support nested SAs or "SA bundles." Instead this functionality can be achieved through SPD and forwarding table configuration. An appendix has been added that provides an example of this.
- o SPD entries were redefined to provide more flexibility. Each SPD entry now consists of 1 to N sets of selectors, where each selector set contains one protocol and a "list of ranges" can now be specified for the Local IP address, Remote IP address, and whatever fields (if any) are associated with the Next Layer Protocol (Local Port, Remote Port, ICMP message type and code, and Mobility Header Type). An individual value for a selector is represented via a trivial range and ANY is represented via a range than spans all values for the selector. An ASN.1 description is included in [Appendix C](#).
- o TOS (IPv4) and Traffic Class (IPv6) have been replaced by DSCP and ECN. The tunnel section has been updated to explain how to handle DSCP and ECN bits.
- o For tunnel mode SAs, an SG, BITS, or BITW implementation is now

allowed to fragment packets before applying IPsec. This applies only to IPv4. For IPv6 packets, only the originator is allowed to fragment them.

- o When security is desired between two intermediate systems along a path or between an intermediate system and an end system, transport mode may now be used between security gateways and between a security gateway and a host.
- o 2401bis clarifies that for all traffic that crosses the IPsec boundary, including IPsec management traffic, the SPD or associated caches must be consulted.
- o 2401bis now defines how to handle the situation of a security gateway with multiple subscribers requiring separate IPsec contexts.
- o A definition of reserved SPIs has been added.
- o Text has been added explaining why ALL IP packets must be checked -- IPsec includes minimal firewall functionality to support access control at the IP layer.
- o The tunnel section has been updated to clarify how to handle the IP options field and IPv6 extension headers when constructing the outer header.
- o SA mapping for inbound traffic has been updated to be consistent with the changes made in AH and ESP for support of unicast, anycast, and multicast SAs.
- o Guidance has been added re: how to handle the covert channel created in tunnel mode by copying the DSCP value to outer header.
- o Support for AH in both IPv4 and IPv6 is now a MAY.
- o PMTU handling has been updated. The appendix on PMTU/DF/Fragmentation has been deleted.
- o Added text saying "The IP Security Policy (IPSP) Working Group is a possible future source of guidance. One of their goals is to produce a Internet Draft on a "Security Gateway Discovery, Policy Exchange and Negotiation Protocol."
- o Three approaches have been added for handling plaintext fragments on the protected side of the IPsec boundary. An appendix has been added documenting the rationale behind them.

- o Added revised text re: how to derive selector values for SAs (from the SPD entry or from the packet, etc.)
- o Added a new table describing the relationship between selector values in an SPD entry, the PFP flag, and resulting selector values in the corresponding SAD entry
- o Added an appendix on decorrelation.
- o Added text describing how to handle an outbound packet which must be discarded.
- o Added text describing how to handle a DROP'd inbound packet, i.e., one that does not match the SA upon which it arrived.
- o IPv6 mobility header has been added as a possible Next Layer Protocol. IPv6 mobility header message type has been added as a selector.
- o ICMP message type and code have been added as selectors.
- o The selector "data sensitivity level" has been removed to simplify things.
- o Updated text describing handling ICMP error messages. The appendix on "Categorization of ICMP messages" has been deleted.
- o The text for the selector name has been updated and clarified.
- o The "Next Layer Protocol" has been further explained and a default list of protocols to skip when looking for the Next Layer Protocol has been added.
- o The text has been amended to say that 2401bis assumes use of IKEv2 or an SA management protocol with comparable features.
- o Text has been added clarifying the algorithm for mapping inbound IPsec datagrams to SAs in the presence of multicast SAs
- o Text and an ASN.1 description have been added to clarify the structure of an SPD entry and its alignment with what can be negotiated in IKEv2.
- o The appendix "Sequence Space Window Code Example" has been removed.

Acknowledgements

The authors would like to acknowledge the contributions of Ran Atkinson, who played a critical role in initial IPsec activities, and who authored the first series of IPsec standards: RFCs 1825-1827. The authors also would like to thank the members of the IPsec and MSEC working groups who have contributed to the development of this protocol specification.

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., [[Shi00](#), [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.)

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.

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.

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.

Protected vs Unprotected

"Protected" refers to the systems or interfaces that are inside the IPsec protection boundary and "unprotected" refers to the systems or interfaces that are outside the IPsec protection boundary. IPsec provides a boundary through which traffic passes. There is an asymmetry to this barrier, which is reflected in the processing model. Outbound data, if not discarded or bypassed, is protected via the application of AH or ESP and the addition of the corresponding headers. Inbound data, if not discarded or bypassed, is processed via the removal of AH or ESP headers. In this document, inbound traffic enters an IPsec implementation from the "unprotected" interface. Outbound traffic enters the implementation via the "protected" interface, or is internally generated by the implementation on the "protected" side of the boundary and directed toward the "unprotected" interface. An IPsec implementation may support more than one interface on either or both sides of the boundary. The protected interface may be internal, e.g., in a host implementation of IPsec. The protected interface may link to a socket layer interface presented by the OS.

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. State data associated with an SA is represented in the Security Association Database (SAD).

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 termed unprotected (they are generally at least less protected than those "behind" the SG), while the networks and hosts on the internal side are viewed as protected. 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" (SPI). The SPI is an arbitrary 32-bit value that is used by a receiver to identify the SA to which an incoming packet should be bound. For a unicast SA, the SPI can be used by itself to specify an SA, or it may be used in conjunction with the IPsec protocol type. Additional IP address information is used to identify multicast SAs. 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](#)]

Appendix B - Decorrelation

This section is based on work done for caching of policies in the IP Security Policy Working Group by Luis Sanchez, Matt Condell, and John Zao.

Two SPD entries are correlated if there is a non-null intersection between the values of corresponding selectors in each entry. Caching correlated SPD entries can lead to incorrect policy enforcement. A solution to this problem, that still allows for caching, is to remove the ambiguities by decorrelating the entries. That is, the SPD entries must be rewritten so that for every pair of entries there exists a selector for which there is a null intersection between the values in both of the entries. Once the entries are decorrelated, there is no longer any ordering requirement on them, since only one entry will match any lookup. The next section describes decorrelation in more detail and presents an algorithm that may be used to implement decorrelation.

B.1 Decorrelation Algorithm

The basic decorrelation algorithm takes each entry in a correlated SPD and divides it up into a set of entries using a tree structure. The resulting entries that are decorrelated with the decorrelated set of entries are then added to that decorrelated set.

The basic algorithm does not guarantee an optimal set of decorrelated entries. That is, the entries may be broken up into smaller sets than is necessary, though they will still provide all the necessary policy information. Some extensions to the basic algorithm are described later to improve this and improve the performance of the algorithm.

C A set of ordered, correlated entries (a correlated SPD)

C_i The i th entry in C.

U The set of decorrelated entries being built from C

U_i The i th entry in U.

S_{ik} The k th selection for policy C_i

A_i The action for policy C_i

A policy (SPD entry) P may be expressed as a sequence of selector values and an action (BYPASS, DISCARD, or PROTECT):

$$C_i = S_{i1} \times S_{i2} \times \dots \times S_{ik} \rightarrow A_i$$

1) Put C_1 in set U as U_1

For each policy C_j ($j > 1$) in C

- 2) If C_j is decorrelated with every entry in U , then add it to U .
- 3) If C_j is correlated with one or more entries in U , create a tree rooted at the policy C_j that partitions C_j into a set of decorrelated entries. The algorithm starts with a root node where no selectors have yet been chosen.

- A) Choose a selector in C_j , S_{jn} , that has not yet been chosen when traversing the tree from the root to this node. If there are no selectors not yet used, continue to the next unfinished branch until all branches have been completed. When the tree is completed, go to step D.

T is the set of entries in U that are correlated with the entry at this node.

The entry at this node is the entry formed by the selector values of each of the branches between the root and this node. Any selector values that are not yet represented by branches assume the corresponding selector value in C_j , since the values in C_j represent the maximum value for each selector.

- B) Add a branch to the tree for each value of the selector S_{jn} that appears in any of the entries in T . (If the value is a superset of the value of S_{jn} in C_j , then use the value in C_j , since that value represents the universal set.) Also add a branch for the complement of the union of all the values of the selector S_{jn} in T . When taking the complement, remember that the universal set is the value of S_{jn} in C_j . A branch need not be created for the null set.
- C) Repeat A and B until the tree is completed.
- D) The entry to each leaf now represents an entry that is a subset of C_j . The entries at the leaves completely partition C_j in such a way that each entry is either completely overridden by an entry in U , or is decorrelated with the entries in U .

Add all the decorrelated entries at the leaves of the tree to U .

- 4) Get next C_j and go to 2.
- 5) When all entries in C have been processed, then U will contain an decorrelated version of C .

There are several optimizations that can be made to this algorithm. A few of them are presented here.

It is possible to optimize, or at least improve, the amount of branching that occurs by carefully choosing the order of the selectors used for the next branch. For example, if a selector S_{jn} can be chosen so that all the values for that selector in T are equal to or a superset of the value of S_{jn} in C_j , then only a single branch needs to be created (since the complement will be null).

Branches of the tree do not have to proceed with the entire decorrelation algorithm. For example, if a node represents an entry that is decorrelated with all the entries in U , then there is no reason to continue decorrelating that branch. Also, if a branch is completely overridden by an entry in U , then there is no reason to continue decorrelating the branch.

An additional optimization is to check to see if a branch is overridden by one of the CORRELATED entries in set C that has already been decorrelated. That is, if the branch is part of decorrelating C_j , then check to see if it was overridden by an entry C_m , $m < j$. This is a valid check, since all the entries C_m are already expressed in U .

Along with checking if an entry is already decorrelated in step 2, check if C_j is overridden by any entry in U . If it is, skip it since it is not relevant. An entry x is overridden by another entry y if every selector in x is equal to or a subset of the corresponding selector in entry y .

Appendix C -- ASN.1 for an SPD Entry

This appendix is included as an additional way to describe SPD entries, as defined in [Section 4.4.1](#). It uses ASN.1 syntax. Since it describes encodings to be used with IKEv2, to express SPD entries, using ASN.1 constraints, it will not compile as shown due to "duplicate" tags. However it has been successfully compiled when augmented with appropriate compiler directives. This syntax is merely illustrative and need not be employed in an implementation to achieve compliance. The SPD description in [Section 4.4.1](#) is normative.

-- An SPD is a list of policies in decreasing order of preference
SPD ::= SEQUENCE OF SPDEntry

DEFINITIONS IMPLICIT TAGS ::=

-- IMPORTS DistinguishedName RFC822Name FQDN

SPDEntry ::= CHOICE {
 iPsecEntry IPsecEntry, -- PROTECT traffic
 bypassOrDiscard [0] BypassOrDiscardEntry } -- DISCARD/BYPASS

IPsecEntry ::= SEQUENCE { -- Each entry consists of:
 name NameSets OPTIONAL,
 pFPs PacketFlags, -- Populate from packet flags
 -- Applies to ALL of the corresponding
 -- traffic selectors in the SelectorLists
 condition SelectorLists, -- Policy "condition"
 processing Processing, -- Policy "action"
 }

BypassOrDiscardEntry ::= SEQUENCE {
 bypass BOOLEAN, -- TRUE: BYPASS, FALSE: DISCARD
 condition InOutBound }

InOutBound ::= CHOICE {
 outbound [0] SelectorLists,
 inbound [1] SelectorLists,
 bothways [2] BothWays }

BothWays ::= SEQUENCE {
 inbound SelectorLists,
 outbound SelectorLists }

NameSets ::= SEQUENCE {
 passed SET OF Names, -- Matched to IKE ID
 local SET OF Names } -- Used internally


```
Names ::= CHOICE {
    dName      DistinguishedName,    -- IKEv2 IDs:
    fqdn       FQDN,                  -- ID_DER_ASN1_DN
    rfc822      [0] RFC822Name,         -- ID_FQDN
    keyID      OCTET STRING }         -- ID_RFC822_ADDR
                                      -- KEY_ID

PacketFlags ::= BIT STRING {
    -- if set, take selector value from packet establishing SA
    -- else use value in SPD entry

    localAddr  (0),
    remoteAddr (1),
    protocol   (2),
    localPort  (3),
    remotePort (4) }

SelectorLists ::= SET OF SelectorList

SelectorList ::= SEQUENCE {
    localAddr  AddrList,
    remoteAddr AddrList,
    protocol   ProtocolChoice }

Processing ::= SEQUENCE {
    extSeqNum   BOOLEAN,    -- TRUE: 64 bit counter, FALSE: 32 bit
    seqOverflow BOOLEAN,    -- TRUE: rekey, FALSE: terminate & audit
    fragCheck   BOOLEAN,    -- TRUE: stateful fragment checking,
                                -- FALSE: no stateful fragment checking
    lifetime    SALifetime,
    spi         ManualSPI,
    algorithms  ProcessingAlgs,
    tunnel      TunnelOptions OPTIONAL } -- if absent, use transport mode

SALifetime ::= SEQUENCE {
    seconds [0] INTEGER OPTIONAL,
    bytes   [1] INTEGER OPTIONAL }

ManualSPI ::= SEQUENCE {
    spi    INTEGER,
    keys   KeyIDs }

KeyIDs ::= SEQUENCE OF OCTET STRING

ProcessingAlgs ::= CHOICE {
    ah      [0] IntegrityAlgs, -- AH
    esp     [1] ESPAlgs }      -- ESP
```



```
ESPAlgs ::= CHOICE {
    integrity      [0] IntegrityAlgs,      -- ESP integrity only
    confidentiality [1] ConfidentialityAlgs, -- ESP confidentiality only
    both           [2] IntegrityConfidentialityAlgs,
    combined       [3] CombinedModeAlgs }

IntegrityConfidentialityAlgs ::= SEQUENCE { -- must have both
    integrity      IntegrityAlgs,
    confidentiality ConfidentialityAlgs }

-- Integrity Algorithms, ordered by decreasing preference
IntegrityAlgs ::= SEQUENCE OF IntegrityAlg

-- Confidentiality Algorithms, ordered by decreasing preference
ConfidentialityAlgs ::= SEQUENCE OF ConfidentialityAlg

-- Integrity Algorithms
IntegrityAlg ::= SEQUENCE {
    algorithm      IntegrityAlgType,
    parameters     ANY DEFINED BY algorithm OPTIONAL }

IntegrityAlgType ::= INTEGER {
    none           (0),
    auth_HMAC_MD5_96 (1),
    auth_HMAC_SHA1_96 (2),
    auth_DES_MAC    (3),
    auth_KPDK_MD5   (4),
    auth_AES_XCBC_96 (5)
-- tbd (6..65535)
}

-- Confidentiality Algorithms
ConfidentialityAlg ::= SEQUENCE {
    algorithm      ConfidentialityAlgType,
    parameters     ANY DEFINED BY algorithm OPTIONAL }

ConfidentialityAlgType ::= INTEGER {
    encr_DES_IV64   (1),
    encr_DES        (2),
    encr_3DES       (3),
    encr_RC5        (4),
    encr_IDEA       (5),
    encr_CAST       (6),
    encr_BLOWFISH   (7),
    encr_3IDEA      (8),
    encr_DES_IV32   (9),
    encr_RC4        (10),
    encr_NULL       (11),
```



```
    encr_AES_CBC    (12),
    encr_AES_CTR    (13)
--   tbd (14..65535)
}
```

CombinedModeAlgs ::= SEQUENCE OF CombinedModeAlg

CombinedModeAlg ::= SEQUENCE {
 algorithm CombinedModeType TABLE CombinedModeAlgTable,
 parameters ANY DEFINED BY algorithm }

CombinedModeType ::= INTEGER

TunnelOptions ::= SEQUENCE {
 dscp DSCP,
 ecn BOOLEAN, -- TRUE: Copy CE to inner header
 flowLabel BOOLEAN, -- TRUE: copy as is from ??? to ???
 df DF }

DSCP ::= SEQUENCE {
 copy BOOLEAN, -- TRUE: copy from inner header, FALSE: do not
 mapping OCTET STRING OPTIONAL} -- pointer to table if no copy

DF ::= INTEGER {
 clear (0),
 set (1),
 copy (2) }

FlowLabel ::= BOOLEAN -- TRUE copy as is, FALSE

ProtocolChoice ::= CHOICE {
 anyProt AnyProtocol, -- for ANY protocol
 noNext [0] NoNextLayerProtocol, -- has no next layer items
 oneNext [1] OneNextLayerProtocol, -- has one next layer item
 twoNext [2] TwoNextLayerProtocol, -- has two next layer items
 fragment FragmentNoNext } -- has no next layer info

AnyProtocol ::= SEQUENCE {
 id INTEGER (0), -- ANY protocol
 nextLayer AnyNextLayers }

AnyNextLayers ::= SEQUENCE { -- with either
 first AnyNextLayer, -- ANY next layer selector
 second AnyNextLayer } -- ANY next layer selector

NoNextLayerProtocol ::= INTEGER (2..254)

FragmentNoNext ::= INTEGER (44), -- Fragment identifier


```
OneNextLayerProtocol ::= SEQUENCE {
    id          INTEGER (1..254),    -- ICMP, MH, ICMPv6
    nextLayer   NextLayerChoice }   -- ICMP Type*256+Code
                                   -- MH   Type*256

TwoNextLayerProtocol ::= SEQUENCE {
    id          INTEGER (2..254),    -- Protocol
    local       NextLayerChoice,     -- Local and
    remote      NextLayerChoice }    -- Remote ports

NextLayerChoice ::= CHOICE {
    any         AnyNextLayer,
    opaque      [0] OpaqueNextLayer,
    range       [1] NextLayerRange }

    -- Representation of ANY in next layer field
AnyNextLayer ::= SEQUENCE {
    start       INTEGER (0),
    end         INTEGER (65535) }

    -- Representation of OPAQUE in next layer field.
    -- Matches IKE convention
OpaqueNextLayer ::= SEQUENCE {
    start       INTEGER (65535),
    end         INTEGER (0) }

    -- Range for a next layer field
NextLayerRange ::= SEQUENCE {
    start       INTEGER (0..65535),
    end         INTEGER (0..65535) }

    -- List of IP addresses
AddrList ::= SEQUENCE {
    v4Liist     IPv4List OPTIONAL,
    v6Liist     [0] IPv6List OPTIONAL }

    -- IPv4 address representations
IPv4List ::= SEQUENCE OF IPv4Range

IPv4Range ::= SEQUENCE {    -- close, but not quite right ...
    ipv4Start   OCTET STRING (SIZE (4)),
    ipv4End     OCTET STRING (SIZE (4)) }

    -- IPv6 address representations
IPv6List ::= SEQUENCE OF IPv6Range

IPv6Range ::= SEQUENCE {    -- close, but not quite right ...
    ipv6Start   OCTET STRING (SIZE (16)),
```



```
    ipv6End    OCTET STRING (SIZE (16)) }
```

Appendix D -- Fragment Handling Rationale

The Requirements

There are three issues that must be resolved re processing of (plaintext) fragments in IPsec:

- mapping a non-initial, outbound fragment to the right SA (or finding the right SPD entry)
- verifying that a received, non-initial fragment is authorized for the SA via which it is received
- mapping outbound and inbound non-initial fragments to the right SPD/cache entry, for bypass/drop traffic.

The first and third issues arise because we need a deterministic algorithm for mapping traffic to SAs (and SPD/cache entries). All three issues are important because we want to make sure that non-initial fragments that cross the IPsec boundary do not cause the access control policies in place at the receiver (or transmitter) to be violated.

D.1 Transport Mode and Fragments

First, we note that transport mode SAs have been defined (in 2401bis) to not carry fragments. This is a carryover from 2401, where transport mode SAs always terminated at end points. This is a fundamental requirement because, in the worst case, an IPv4 fragment to which IPsec was applied, might then be fragmented (as a ciphertext packet), en route to the destination. IP fragment reassembly procedures at the IPsec receiver would not be able to distinguish between pre-IPsec fragments and fragments created after IPsec processing.

For IPv6, only the sender is allowed to fragment a packet. As for IPv4, an IPsec implementation is allowed to fragment tunnel mode packets after IPsec processing, because it is the sender relative to the (outer) tunnel header. However, unlike IPv4, it would be feasible to carry a plaintext fragment on a transport mode SA, because the fragment header in IPv6 would appear after the AH or ESP header, and thus would not cause confusion at the receiver re reassembly. Specifically, the receiver would not attempt reassembly for the fragment until after IPsec processing. To keep things simple, this specification prohibits carriage of fragments on transport mode SAs for IPv6 traffic.

When only end systems used transport mode SAs, the prohibition on carriage of fragments was not a problem, since we assumed that the end system could be configured to not offer a fragment to IPsec. For

a native host implementation this seems reasonable, and, as someone already noted, 2401 warned that a BITS implementation might have to reassemble fragments before performing an SA lookup. (It would then apply AH or ESP and could re-fragment the packet after IPsec processing.) Because a BITS implementation is assumed to be able to have access to all traffic emanating from its host, even if the host has multiple interfaces, this was deemed a reasonable mandate.

In 2401bis, we have explicitly said that it is OK to use transport mode in cases where the IPsec implementation is not the ultimate destination, e.g., between two SGs. In principle, this creates a new opportunity for outbound, plaintext fragments to be mapped to a transport mode SA for IPsec processing. However, in these new contexts in which a transport mode SA is now approved for use, it seems likely that we can continue to prohibit transmission of fragments, as seen by IPsec. For example, in an IP overlay network, packets being sent over transport mode SAs are IP-in-IP tunneled and thus have the necessary inner header to accommodate fragmentation prior to IPsec processing. When carried via a transport mode SA, IPsec would not examine the inner IP header for such traffic, and thus would not consider the packet to be a fragment. Thus it seems reasonable to retain the prohibition on carrying IPv4 fragments on transport mode SAs, even when the source or destination is an SG.

D.2 Tunnel Mode and Fragments

For tunnel mode SAs, it has always been the case that outbound fragments might arrive for processing at an IPsec implementation. The need to accommodate fragmented outbound packets can pose a problem because a non-initial fragment generally will not contain the port fields associated with a next layer protocol such as TCP, UDP, or SCTP. Thus, depending on the SPD configuration for a given IPsec implementation, plaintext fragments might or might not pose a problem.

For example, if the SPD requires that all traffic between two address ranges is offered IPsec protection (no bypass or drop SPD entries apply to this address range), then it should be easy to carry non-initial fragments on the SA defined for this address range, since the SPD entry implies an intent to carry ALL traffic between the address ranges. But, if there are multiple SPD entries that could match a fragment, and if these entries reference different subsets of port fields (vs. ANY), then it is not possible to map an outbound non-initial fragment to the right entry, unambiguously. (If we choose to allow carriage of fragments on transport mode SAs for IPv6, the problems arises in that context as well.)

This problem largely, though not exclusively, motivated the

definition of OPAQUE as a selector value for port fields in [RFC 2401](#). The other motivation for OPAQUE is the observation that port fields might not be accessible due to the prior application of IPsec. For example, if a host applied IPsec to its traffic and that traffic arrived at an SG, these fields would be encrypted. The algorithm specified for locating the "next layer protocol" described in 2401 also motivated use of OPAQUE to accommodate an encrypted next layer protocol field in such circumstances. Nonetheless, the primary use of the OPAQUE value was to match traffic selector fields in packets that did not contain port fields (non-initial fragments), or packets in which the port fields were already encrypted (as a result of nested application of IPsec). 2401 was ambiguous in discussing the use of OPAQUE vs. ANY, suggesting in some places that ANY might be an alternative to OPAQUE.

We gain additional access control capability by defining both ANY and OPAQUE values. OPAQUE can be defined to match only fields that are not accessible. We could define ANY as the complement of OPAQUE, i.e., it would match all values but only for accessible port fields. If we simplify the procedure employed to locate the next layer protocol in 2401bis, so that we treat ESP and AH as next layer protocols, then the notion of an encrypted next layer protocol field has vanished, and there is also no need to worry about encrypted port fields either. In that case, OPAQUE will be applicable only to non-initial fragments.

If we adopt the definitions above for ANY and OPAQUE, we need to clarify how these values work when the specified protocol does not have port fields, and when ANY is used for the protocol selector. Accordingly, if a specific protocol value is used as a selector, and if that protocol has no port fields, then the port field selectors are to be ignored and ANY MUST be specified as the value for the port fields. (In this context, ICMP TYPE and CODE values are lumped together as a single port field (for IKEv2 negotiation), as is the IPv6 Mobility Header TYPE value.) If the protocol selector is ANY, then this should be treated as equivalent to specifying a protocol for which no port fields are defined, and thus the port selectors should be ignored, and MUST be set to ANY.

[D.3. The Problem of Non-Initial Fragments](#)

For an SG implementation, it is obvious that fragments might arrive from end systems behind the SG. A BITW implementation also may encounter fragments from a host or gateway behind it. (As noted earlier, native host implementations and BITS implementations probably can avoid the problems described below.) In the worst case, fragments from a packet might arrive at distinct BITW or SG instantiations and thus preclude reassembly as a solution option.

Hence, in 2401 we adopted a general requirement that fragments must be accommodated in tunnel mode for all implementations. However, 2401 did not provide a perfect solution. The use of OPAQUE as a selector value for port fields (a SHOULD in 2401) allowed an SA to carry non-initial fragments.

Using the features defined in 2401, if one defined an SA between two IPsec (SG or BITW) implementations using the OPAQUE value for both port fields, then all non-initial fragments matching the S/D address and protocol values for the SA would be mapped to that SA. Initial fragments would NOT map to this SA, if we adopt a strict definition of OPAQUE. However, 2401 did not provide detailed guidance on this and thus it may not have been apparent that use of this feature would essentially create a "non-initial fragment only" SA, precisely the solution that the WG rejected.

In the course of rejecting the "fragment-only" SA approach, it was noted that some subtle problems, problems not considered in 2401, would have to be avoided. For example, an SA of this sort must be configured to offer the "highest quality" security services for any traffic between the indicated S/D addresses (for the specified protocol). This is necessary to ensure that any traffic captured by the fragment-only SA is not offered degraded security relative to what it would have been offered if the packet were not fragmented. A possible problem here is that we may not be able to identify the "highest quality" security services defined for use between two IPsec implementations, since the choice of security protocols, options, and algorithms is a lattice, not a totally ordered set. (We might safely say that BYPASS < AH < ESP w/integrity, but it gets complicated if we have multiple ESP encryption or integrity algorithm options.) So, one has to impose a total ordering on these security parameters to make this work, but this can be done locally.

However, this conservative strategy has a possible performance downside; if most traffic traversing an IPsec implementation for a given S/D address pair (and specified protocol) is bypassed, then a fragment-only SA for that address pair might cause a dramatic increase in the volume of traffic afforded crypto processing. If the crypto implementation cannot support high traffic rates, this could cause problems. (An IPsec implementation that is capable of line rate or near line rate crypto performance would not be adversely affected by this SA configuration approach. Nonetheless, the performance impact is a potential concern, specific to implementation capabilities.)

Another concern is that non-initial fragments sent over a dedicated SA might be used to effect overlapping reassembly attacks, when combined with an apparently acceptable initial fragment. (This sort

of attack assumes creation of bogus fragments, and is not a side effect of normal fragmentation.) This concern is easily addressed in IPv4, by checking the fragment offset value to ensure that no non-initial fragments have a small enough offset to overlap port fields that should be contained in the initial fragment. Recall that the IPv4 MTU minimum is 576 bytes, and the max IP header length is 60 bytes, so any ports should be present in the initial fragment. If we require all non-initial fragments to have an offset of say 128 or greater, just to be on the safe side, this should prevent successful attacks of this sort. If the intent is only to protect against this sort of reassembly attack, this check need be implemented only by a receiver.

IPv6 also has a fragment offset, carried in the fragmentation extension header. However, IPv6 extension headers are variable in length and there is no analogous max header length value that we can use to check non-initial fragments, to reject ones that might be used for an attack of the sort noted above. A receiver would need to maintain state analogous to reassembly state, to provide equivalent protection. So, only for IPv4 it is feasible to impose a fragment offset check that would reject attacks designed to circumvent port field checks by IPsec (or firewalls) when passing non-initial fragments.

Another possible concern is that in some topologies and SPD configurations this approach might result in an access control surprise. The notion is that if we create an SA to carry ALL (non-initial) fragments then that SA would carry some traffic that might otherwise arrive as plaintext via a separate path, e.g., a path monitored by a proxy firewall. But, this concern arises only if the other path allows initial fragments to traverse it without requiring reassembly, presumably a bad idea for a proxy firewall. Nonetheless, this does represent a potential problem in some topologies and under certain assumptions re: SPD and (other) firewall rule sets, and administrators need to be warned of this possibility.

A less serious concern is that non-initial fragments sent over a non-initial fragment-only SA might represent a DoS opportunity, in that they could be sent when no valid, initial fragment will ever arrive. This might be used to attack hosts behind an SG or BITW device. However, the incremental risk posed by this sort of attack, which can be mounted only by hosts behind an SG or BITW device, seems small.

If we interpret the ANY selector value as encompassing OPAQUE, then a single SA with ANY values for both port fields would be able to accommodate all traffic matching the S/D address and protocol traffic selectors, an alternative to using the OPAQUE value. But, using ANY here precludes multiple, distinct SAs between the same IPsec

implementations for the same address pairs and protocol. So, it is not an exactly equivalent alternative.

Fundamentally, fragment handling problems arise only when more than one SA is defined with the same S/D address and protocol selector values, but with different port field selector values.

D.4 BYPASS/DROP Traffic

We also have to address the non-initial fragment processing issue for BYPASS/DROP entries, independent of SA processing. This is largely a local matter for two reasons:

- 1) We have no means for coordinating SPD entries for such traffic between IPsec implementations since IKE is not invoked.
- 2) Many of these entries refer to traffic that is NOT directed to or received from a location that is using IPsec. So there is no peer IPsec implementation with which to coordinate via any means.

However, 2401bis should provide guidance here, consistent with our goal of offering a well-defined, access control function for all traffic, relative to the IPsec boundary. To that end, this document says that implementations MUST support fragment reassembly for BYPASS/DROP traffic when port fields are specified. An implementation also MUST permit a user or administrator to accept such traffic or reject such traffic using the SPD conventions described in Section 4.4.1. The concern is that BYPASS of a cleartext, non-initial fragment arriving at an IPsec implementation could undermine the security afforded IPsec-protected traffic directed to the same destination. For example, consider an IPsec implementation configured with an SPD entry that calls for IPsec-protection of traffic between a specific source/destination address pair, and for a specific protocol and destination port, e.g., TCP traffic on port 25 (Telnet). Assume that the implementation also allows BYPASS of traffic from the same source/destination address pair and protocol, but for a different destination port, e.g., port 119 (NNTP). An attacker could send a non-initial fragment (with a forged source address) that, if bypassed, could overlap with IPsec-protected traffic from the same source and thus violate the integrity of the IPsec-protected traffic. Requiring stateful fragment checking for BYPASS entries with non-trivial port ranges prevents attacks of this sort.

D.5 Just say no to ports?

It has been suggested that we could avoid the problems described above by not allowing port field selectors to be used in tunnel mode. But the discussion above shows this to be an unnecessarily stringent

approach, i.e., since no problems arise for the native OS and BITS implementations. Moreover, some WG members have described scenarios where use of tunnel mode SAs with (non-trivial) port field selectors is appropriate. So the challenge is defining a strategy that can deal with this problem in BITW and SG contexts. Also note that bypass/drop entries in the SPD that make use of ports pose the same problems, irrespective of tunnel vs. transport mode notions.

Some folks have suggested that a firewall behind an SG or BITW should be left to enforce port level access controls, and the effects of fragmentation. However, this seems to be an incongruous suggestion in that elsewhere in IPsec (e.g., in IKE payloads) we are concerned about firewalls that always drop fragments. If many (most?) firewalls don't pass fragments in general, why should we expect them to deal with fragments in this case? So, this analysis rejects the suggestion of disallowing use of port field selectors with tunnel mode SAs.

D.6 Other Suggested Solutions

One suggestion is to reassemble fragments at the sending IPsec implementation, and thus avoid the problem entirely. This approach is invisible to a receiver and thus could be adopted as a purely local implementation option.

A more sophisticated version of this suggestion calls for establishing and maintaining minimal state from each initial fragment encountered, to allow non-initial fragments to be matched to the right SAs or SPD/cache entries. This implies an extension to the current processing model (and the old one). The IPsec implementation would intercept all fragments, capture S/D address, protocol, packet ID, and port fields from initial fragments and then use this data to map non-initial fragments to SAs that require port fields. If this approach is employed, the receiver needs to employ an equivalent scheme, as it too must verify that received fragments are consistent with SA selector values. A non-initial fragment that arrives prior to an initial fragment could be cached or discarded, awaiting arrival of the corresponding initial fragment.

A downside of both approaches noted above is that they will not always work. When a BITW device or SG is configured in a topology that might allow some fragments for a packet to be processed at different SGs or BITW devices, then there is no guarantee that all fragments will ever arrive at the same IPsec device. This approach also raises possible processing problems. If the sender caches non-initial fragments until the corresponding initial fragment arrives, buffering problems might arise, especially at high speeds. If the non-initial fragments are discarded rather than cached, there is no

guarantee that traffic will ever pass, e.g., retransmission will result in different packet IDs that cannot be matched with prior transmissions. In any case, housekeeping procedures will be needed to decide when to delete the fragment state data, adding some complexity to the system. Nonetheless, this is a viable solution in some topologies, and these are likely to be common topologies.

The Working Group rejected the convention of creating an SA to carry only non-initial fragments, something that was supported implicitly under the 2401 model via use of OPAQUE port fields, but never clearly articulated in the RFC. The (rejected) text called for each non-initial fragment to be treated as protocol 44 (the IPv6 fragment header protocol ID) by the sender and receiver. This approach has the potential to make IPv4 and IPv6 fragment handling more uniform, but it does not fundamentally change the problem, nor does it address the issue of fragment handling for bypass/drop traffic. Given the fragment overlap attack problem that IPv6 poses, it does not seem that it is worth the effort to adopt this strategy.

D.7 Consistency

Earlier the WG agreed to allow an IPsec BITS, BITW or SG to perform fragmentation prior to IPsec processing. If this fragmentation is performed after SA lookup at the sender, there is no "mapping to the right SA" problem. But, the receiver still needs to be able to verify that the non-initial fragments are consistent with the SA via which they are received. Since the initial fragment might be lost en route, the receiver encounters all of the potential problems noted above. Thus, if we are to be consistent in our decisions, we need to say how a receiver will deal with the non-initial fragments that arrive.

D.8 Conclusions

There is no simple, uniform way to handle fragments in all contexts. Different approaches work better in different contexts. Thus this document offers 3 choices -- one MUST and two MAYs. At some point in the future, if the community gains experience with the two MAYs, they may become SHOULDs or MUSTs or other approaches may be proposed.

Appendix E - Example of Supporting Nested SAs via SPD and Forwarding Table Entries

This appendix provides an example of how to configure the SPD and forwarding tables to support a nested pair of SAs, consistent with the new processing model.

The goal in this example is to support a transport mode SA from A to C, carried over a tunnel mode SA from A to B. For example, A might be a laptop connected to the public internet, B a firewall that protects a corporate network, and C a server on the corporate network that demands end-to-end authentication of A's traffic.

```

+---+      +---+  +---+
| A |=====| B |  | C |
|   |-----|   |  |   |
|   |=====|   |  |   |
+---+      +---+  +---+

```

A's SPD contains entries of the form:

Rule	Local	Remote	Next Layer Protocol	Action
1	C	A	ESP	BYPASS
2	A	C	ICMP, ESP	PROTECT(ESP, tunnel, integr+conf)
3	A	C	ANY	PROTECT(ESP, transport, integr-only)
4	A	B	ICMP, IKE	BYPASS

A's unprotected-side forwarding table is set so that outbound packets destined for C are looped back to the protected side. A's protected side forwarding table is set so that inbound ESP packets are looped back to the unprotected side. A's forwarding tables contain entries of the form:

Unprotected-side forwarding table

Rule	Local	Remote	Protocol	Action
1	A	C	ANY	loop back to protected side
2	A	B	ANY	forward to B

Protected-side forwarding table

Rule	Local	Remote	Protocol	Action
1	A	C	ESP	loop back to unprotected side

An outbound TCP packet from A to C would match SPD rule 3 and have transport mode ESP applied to it. The unprotected-side forwarding table would then loop back the packet. The packet is compared against SPD-I (see Figure 2), matches SPD rule 1, and so it is BYPASSED. The packet is treated as an outbound packet and compared against the SPD for a third time. This time it matches SPD rule 2, so ESP is applied in tunnel mode. This time the forwarding table doesn't loop back the packet, because the outer destination address is B, so the packet goes out onto the wire.

An inbound TCP packet from C to A, is wrapped in two ESP headers; the outer header (ESP in tunnel mode) shows B as the source whereas the inner header (ESP transport mode) shows C as the source. Upon arrival at A, the packet would be mapped to an SA based on the SPI, have the outer header removed, and be decrypted and integrity-checked. Then it would be matched against the SAD selectors for this SA, which would specify C as the source and A as the destination, derived from SPD rule 2. The protected-side forwarding function would then send it back to the unprotected side based on the addresses and the next layer protocol (ESP), indicative of nesting. It is compared against SPD-O (see figure 3) and found to match SPD rule 1, so it is BYPASSED. The packet is mapped to an SA based on the SPI, integrity-checked, and compared against the SAD selectors derived from SPD rule 3. The forwarding function then passes it up to the next layer, because it isn't an ESP packet.

References

Normative

- [BBCDWW98] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "An Architecture for Differentiated Service", [RFC 2475](#), December 1998.
- [Bra97] Bradner, S., "Key words for use in RFCs to Indicate Requirement Level", [BCP 14](#), [RFC 2119](#), March 1997.
- [CD98] Conta, A. and S. Deering, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", [RFC 2463](#), December 1998.
- [DH98] Deering, S., and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", [RFC 2460](#), December 1998.
- [Eas03] Eastlake, D., "Cryptographic Algorithm Implementation Requirements For ESP And AH", ???, ??? 2004.
- [Kau04] Kaufman, C., "The Internet Key Exchange (IKEv2) Protocol", RFC ???, ??? 2004.
- [Ken04a] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC ???, ??? 2004.
- [Ken04b] Kent, S., "IP Authentication Header", RFC ???, ??? 2004.
- [MD90] Mogul, J. and S. Deering, "Path MTU discovery", [RFC 1191](#), November 1990.
- [Pos81a] Postel, J., "Internet Protocol", STD 5, [RFC 791](#), September 1981
- [Pos81b] Postel, J., "Internet Control Message Protocol", [RFC 792](#), September 1981
- [Sch03] Schiller, J., "Cryptographic Algorithms for use in the Internet Key Exchange Version 2", RFC ???, ??? 2003

Informative

- [FaLiHaMeTr00] Farinacci, D., Li, T., Hanks, S., Meyer, D., Traina, P., "Generic Routing Encapsulation (GRE)", [RFC 2784](#), March 2000.

- [Gro02] Grossman, D., "New Terminology and Clarifications for Diffserv", [RFC 3260](#), April 2002.
- [HC03] Holbrook, H., and Cain, B., "Source Specific Multicast for IP", WWork in Progress, November 3, 2002.
- [HA94] Haller, N., and Atkinson, R., "On Internet Authentication", [RFC 1704](#), October 1994
- [Mobip] Johnson, D., Perkins, C., Arkko, J., "Mobility Support in IPv6", Work in Progress, June 2003
- [NiBlBaBL98]Nichols, K., Blake, S., Baker, F., Black, D., "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", [RFC2474](#), December 1998.
- [Per96] Perkins, C., "IP Encapsulation within IP", [RFC 2003](#), October 1996.
- [RaFlBl01]Ramakrishnan, K., Floyd, S., Black, D., "The Addition of Explicit Congestion Notification (ECN) to IP", [RFC 3168](#), September 2001.
- [RFC3547] Baugher, M., Weis, B., Hardjono, T., Harney, H., "The Group Domain of Interpretation", [RFC 3547](#), July 2003.
- [RFC3740] Hardjono, T., Weis, B., "The Multicast Group Security Architecture", [RFC 3740](#), March 2004.
- [Sch94] Schneier, B., Applied Cryptography, [Section 8.6](#), John Wiley & Sons, New York, NY, 1994.
- [Shi00] Shirey, R., "Internet Security Glossary", [RFC 2828](#), May 2000.
- [SMPT01] Shacham, A., Monsour, B., Pereira, R., and M. Thomas, "IP Payload Compression Protocol (IPComp)", [RFC 3173](#), September 2001.
- [VK83] V.L. Voydock & S.T. Kent, "Security Mechanisms in High-level Networks", ACM Computing Surveys, Vol. 15, No. 2, June 1983.

Author Information

Stephen Kent
BBN Technologies
10 Moulton Street
Cambridge, MA 02138
USA

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

Karen Seo
BBN Technologies
10 Moulton Street
Cambridge, MA 02138
USA

Phone: +1 (617) 873-3152
EMail: kseo@bbn.com

Notices

"The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in [BCP 78](#) and [BCP 79](#).

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org."

Copyright (C) The Internet Society (2004).

This document is subject to the rights, licenses and restrictions contained in [BCP 78](#), and except as set forth therein, the authors retain all their rights.

"This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE."

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other

Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

Expires March 2005