Internet Draft
draft-ietf-ipsp-spp-01.txt
Expires July, 2002

L.A. Sanchez, Megisto
M.N. Condell, BBN

January 29, 2002

Security Policy Protocol

Status of this Memo

This document is an Internet-Draft and is in full conformance with 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 six 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 "work in progress."

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

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

Abstract

This document describes a protocol for discovering, accessing and processing security policy information of hosts, subnets or networks of a security domain. The Security Policy Protocol defines how the policy information is exchanged, processed, and protected by clients and servers. The protocol is extensible and flexible. It allows the exchange of complex policy objects between clients and servers.

Table of Contents

1. Introduction

The IPsec protocols [Kent98] provide a mechanism for securing communications at the IP layer and IKE [Harkins98] can be used to provide keys for IPsec. Currently practice with these protocols maintains an assumption that communicating hosts have some a-priori knowledge of which communications with particular newtwork entities must be secured. While this assumption is valid in some environments (e.g. some VPN environments), it does not support more general IPsec senarios in a scalable manner.

In order to allow IPsec to scale in general cases, it is necessary to be able to identify which entities involved in a communication will require IPsec to protect the communication and what their policies are regarding it.

The Security Policy Protocol (SPP) defines how the policy information is exchanged, processed, and protected by clients and servers. The

protocol also defines what policy information is exchanged and the format used to encode the information. The protocol specifies six different message types used to exchange policy information. An SPP

message contains a message header section followed by zero or more SPP payloads, depending on the message type.

SPP is part of the Security Policy System architecture [SPS]. This document uses terms and references functionality described in [SPS].

The remainder of this section defines terms and concepts that will be used throughout this document. Section 2 provides and overview of the protocol. The remainder of the document describes the encoding of the protocol and how SPP messages are processed.

1.1 Definitions

The following terms are used throughout this document, in addition to the terms defined in [SPS] and defined for general policy terminology [RGSC00].

Authoritative

> Host A is authoritative over host B if host A has the right to represent policy for host B. Host A may assert its relationship to host B using policy server records (section 5.4), but MUST be able to cryptographically prove the assertion.

Transitively Authoritative

> A host is transitively authoritative over another host, A, if it is either authoritative over host A or authoritative over a host, B, which is trasitively athoritative over host A. For example, if host X is authoritative for host Y and host Y is authoritative for host Z, then host X is transitively authoritative for host Z.

Chain of Trust

> A chain of trust is a set of cryptographically-proven authoritative assertions that prove that a policy server is transitively authoritative over the source or destination of a communication. The chain of trust is used to prove that a policy server has a right to be involved in an SPP exchange. See section 10 for more about the chain of trust.

Keywords "MUST", "MUST NOT", "REQUIRED", "SHOULD", "SHOULD NOT" and "MAY" that appear in this document are to be interpreted as described in [Bra97].

1.2 Policies

Defining and storing policies are beyond the scope if this document. However, this section describes SPP's policy requirements and a brief high-level look at its representation.

Policy Representation

SPP provides both the comsec record (section 5.2) and the Security Association (SA rec) record (section 5.3) to describe policies. The comsec record defines the selectors that describe a communication along with a permit or deny action. The SA rec defines the actions, specifically the IPsec and IKE security associations, necessary for the communication to proceed. A policy transferred by SPP, therefore, MUST consist of one comsec record to describe the selectors of the communication and zero or more SA recs which describe the security associations that are required to complete the communication.

Decorrelation

Policies exchanged using SPP MUST be decorrelated as described in Appendix C. **Two policies are decorrelated if there exists at least** one selector in both policies for which their values do not intersect. Decorrelation is necessary to permit policy servers to properly cache policies.

2. Overview

This section provides an overview of the SPP operation. A more detailed and complex example of SPP operation is available in appendix B. **This overview assumes the policy servers have been loaded with** policies for their security domains and the policy has been appropriately decorrelated.

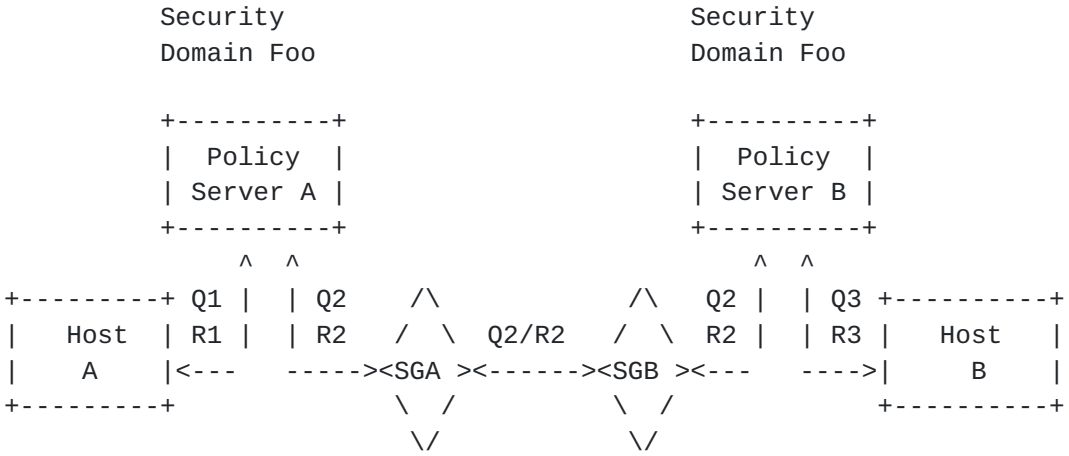


Figure 1: Overview of SPP operation

Host A, wanting to communicate with Host B, invokes its policy client. Host A's client sends a Query (Q1) to its configured local policy server, Policy Server A. Policy Server A looks in its cache for a policy record that matches the query. If it doesn't find one, it sends a Query (Q2) containing the same policy request information to Host B. Q2 is sent to Host B since Policy Server A may not know about the

existence of SGB or Policy Server B. This message includes a signature that validates the authenticity and integrity of the query's content.

(Q2) is intercepted by SGB. SGB forwards the message (Q2) to Policy Server B. Policy server B verifies that it can accept queries from Policy Server A and validates the signature in Q2. It searches its database for the appropriate policy information after verifying that it is authoritative over Policy Client B.

Policy Server B merges its local policy with the policy information in (Q2) and it sends a Reply (R2) to Policy Server A. The reply includes the original query information and all policy information needed to allow Policy Client A to establish a secure communication with Host **B. Policy Server B also attaches additional information to the reply** asserting its authority over Host B.

When Policy Server A receives the reply (R2) from Policy Server A, it validates the signature in R2 and cryptographically verifies that Policy Server B is authoritative over Host B. It then merges is local policy with the policy information in (R2) and sends a Reply (R1) to Host A. Policy Server A caches the merged policy to use when answering future queries. Host A may then use this information to establish necessary security associations with Host B.

If, however, Policy Server B is not authoritative over Host B, it would query Host B for its policy with respect to this particular communication. Policy Server B would generate a third query (Q3). Host B would respond with its policy in (R3). Policy Server B merges its policy for this communication and the policy in (R3) before replying to Policy Server A. Policy Server A processes the reply as it did above.

SPP accommodates topology changes, hence policy changes, rather easily without the scalability constraints imposed by static reconfiguration of each client. The protocol is extensible and flexible It allows the exchange of complex policy objects between clients and servers.

3. SPP Message

The SPP header is present in every message. It contains fields identifying the message, the type of message, the status of the message, the number of queries and/or record payloads, and the host requesting policy information. The header also includes a timestamp field that provides anti-replay protection. Following the header there might be zero or more SPP payloads. Currently, there are three payload types defined in SPP: Query, Record, and Signature payloads. See section 3.2 for encoding details.

SPP has six distinct message types. Query messages contain a specific request for policy information. Reply messages include policy records that answer specific policy queries. Policy messages include policy

information and are utilized for up/downloading security policies to and from a policy server. Policy Acknowledgment messages are utilized to acknowledge corresponding Policy messages but do not themselves contain policy information. Transfer messages, which include policy

information, are utilized by policy servers to exchange bulk policy information between servers. Finally, policy servers use keep alive messages to inform security gateways and/or other monitoring devices of the status of the server.

SPP messages MUST be authenticated either using IPsec [Kent98] or another security mechanism. SPP provides a basic security mechanism that can be used to provide authentication and integrity to its messages when other security mechanisms are not in use. The SPP authentication is especially useful when traversing heterogenous domains and the identity of the policy server authoritative for the destination is unknown. These services are provided using digital signatures.

SPP caries signatures in the signature payload. The signature is calculated over the entire SPP message. When this service is used, the entity (host, policy server, or security gateway) verifying the signature must have access to the public key that corresponds to the private key used to sign the SPP message.

Certificate fetching is out of the scope of SPP. However, SPP provides a simple certificate fetching mechanism for entities that elect to use it as an alternative to other mechanisms. SPP suports several Public Key certificates formats.

SPP is modular and extensible (see section 10 for IANA considerations). New policy queries and records can be defined and incorporated easily. This document defines a minimum set of queries and policy records required in a policy-based security management system.

3.1 SPP Message Format

An SPP message follows the format depicted in figure 2. It is comprised of a header and zero or more SPP payloads. This section defines the encoding for the SPP header. Sections 3.2 and 3.3 cover the encoding for the SPP payload and message types, respectively.

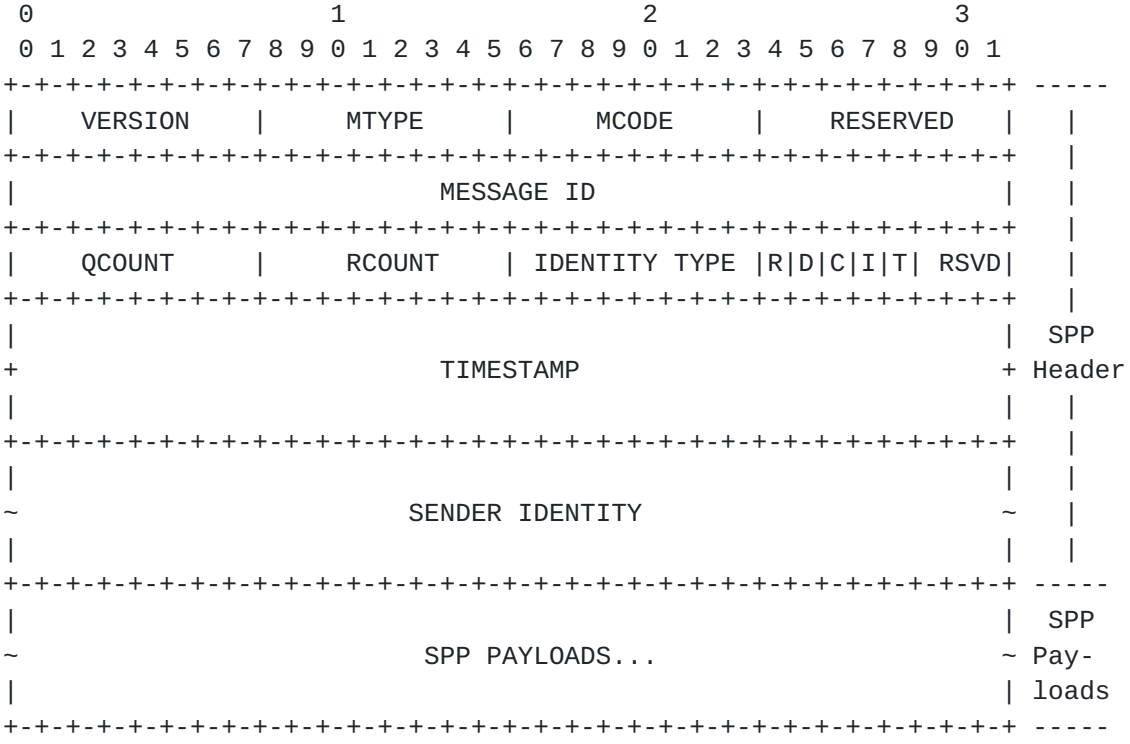


Figure 2: Format of an SPP Message

The SPP header includes the following fields:

VERSION

A 1-octect field containing the version of the Security Policy Protocol. This document describes version 1 of the protocol.

MTYPE

A 1-octet field indicating the SPP message type. The currently defined values are:

Message Type	Value
Value Not Assigned	0
SPP-QUERY	1
SPP-REPLY	2
SPP-POL	3
SPP-POL_ACK	4
SPP-XFR	5
SPP-KEEP_ALIVE	6

values 7-250 are reserved to IANA. Values 251-255 are for private use among mutually consenting parties.

MCODE

A 1-octet field providing information about this message. All MTYPEs share a common MCODE space, although each message type may not use all the defined message codes. See section 3.3 for the codes applicable to each message type.

Action Type	Code Field
Value Not Assigned	0
message accepted	1
denied, administratively prohibited	2
denied, timestamp failed	3
denied, failed signature	4
denied, insufficient resources	5
denied, malformed message	6
denied, unspecified	7
partially available	8
unavailable	9
communication prohibited	10
partially available, server unreachable	11

values 12-250 are reserved to IANA. Values 251-255 are for private use among mutually consenting parties.

RESERVED

A one octet field reserved for future use. Set value to all zeros (0).

MESSAGE ID

A 4 octet field used to match messages and their responses (e.g. queries to replies and policy to policy acknowledgement messages). This value starts at "zero" and MUST be incremented by (1) with every new message.

QCOUNT

A 1 octet field indicating the number of Query payloads included in the message.

RCOUNT

A 1 octet field indicating the number of Record payloads included in the message.

IDENTITY TYPE

This 1 octet field indicates the type of indentity found in the Sender Identity field. Valid values are:

Identity Type	Value
Value Not Assigned	0
IPV4_ADDR	1
IPV6_ADDR	2
Host DNS Name	3

values 4-250 are reserved to IANA. Values 251-255 are for private use among mutually consenting parties.

R

Raw policy flag. When this flag is set, policy servers MUST NOT resolve the policies that they return.

D

Domain flag. Only resolve the policies as far as the last policy server that is transitively authoritative over the host requesting the policy resolution.

C

Dont cache flag. Don't cache the policies generated by the query.

I

Ignore cache flag. Ignore any cached policies when processing the query.

T

No chain-of-trust. A client indicates to its server that it does not need chain-of-trust information. Policy Servers MUST NOT set this flag. Only Policy Clients have the option to set it.

RSVD

A 4 bit field reserved for future use.
Set value to all zeros (0).

TIMESTAMP

This 8-octet field contains a timestamp used to provide limited protection against replay attacks. The timestamp is formatted as specified by the Network Time Protocol [RFC1305].

SENDER IDENTITY

A variable length field containing the identity of the sender (host, security gateway, or policy server) of the SPP message. The IDENTITY_TYPE field indicates the format of the content in this field:

Identity Type	Sender Identity
IPV4_ADDR	An IPv4 Address
IPV6_ADDR	An IPv6 Address
Host DNS Name	A DNS name encoded as described in [rfc1035]

This field does not allow IP address ranges or wildcards. If this field is not aligned at the 4 octet boundary, the field MUST be padded on the right with (00)hex to align on the next 32-bit boundary.

3.2 SPP Payloads

3.2.1 Query Payload

The Query payload contains fields to express a particular request for policy information. Hosts, security gateways, or policy servers can generate and transmit Query payloads in SPP messages to policy servers. Figure 3 shows the format of the Query payload.

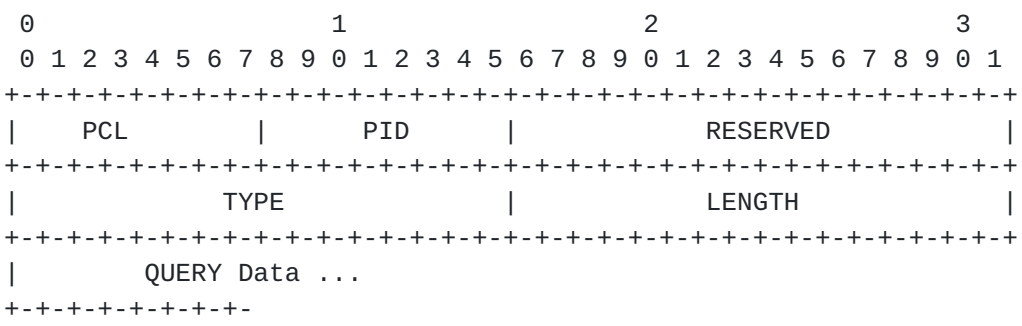


Figure 3: Format of Query Payload

The Query Payload fields are defined as follows:

PCL

A 1 octet field indicating the payload class. Query payloads MUST contain (1) in the PCL field.

PID

A 1 octet field containing the ID number that identifies a particular Query payload within an SPP message. Since one SPP message can contain multiple Query payloads, each one MUST be uniquely identified. This number MUST be unique among the Query payloads within an SPP message.

RESERVED

A 2 octet field reserved for future use. Set value to all zeros (0).

TYPE

A 2 octet field that specifies the type of query contained in the QUERY Data fields. The currently defined queries are:

Query Payload Type	Value
Value Not Assigned	0
Security Gateway Query	1
Communication Security Query	2
Certificate Query	3

values 4-65000 are reserved to IANA. Values 65001-65535 are for private use among mutually consenting parties.

LENGTH

A 2 octet field indicating the length in octets of the query data field.

QUERY Data

A variable length field containing a single policy query. See section 7 for encoding format.

3.2.2 Record Payload

The Record payload contains fields that assert policy information. Hosts, security gateways, or policy servers can generate and transmit Record payloads in SPP messages. Figure 4 shows the format of the Record payload.

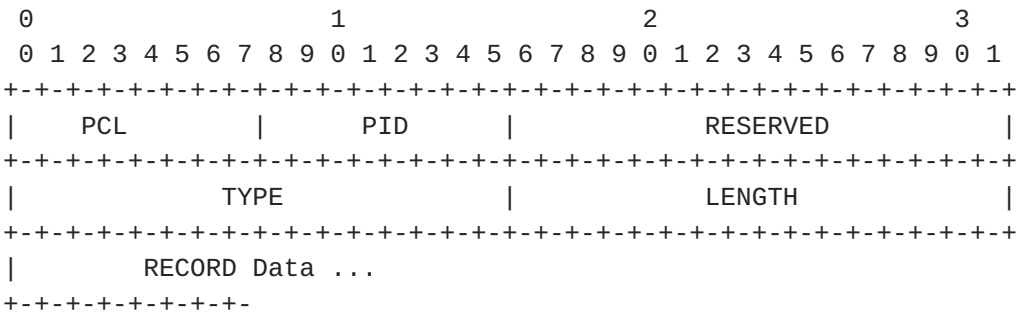


Figure 4: Format of Record Payload

The Record Payload fields are defined as follows:

PCL

A 1 octet field indicating the payload class. Record payloads MUST contain (2) in the PCL field.

PID

This field is used to match queries to Record payloads. If the record is a reply to a query, then the value for this field MUST match the correspondent Query payload PID. If it is not a reply to a query, the value SHOULD be set to zero.

RESERVED

A 2 octet field reserved for future use. Set value to all zeros (0).

TYPE

A 2 octet field that specifies the type of Record. The currently defined records are:

Record Type	Value
Value Not Assigned	0
Security Gateway Record	1
Communication Security Record	2
Security Association Record	3
Certificate Record	4
Policy Server Record	5
Transfer Record	6

values 7-65000 are reserved to IANA. Values 65001-65535 are for private use among mutually consenting parties.

LENGTH

A 2 octet field indicating the length in octets of the RECORD data field.

RECORD Data

A variable length field containing a single policy record. See section 8 for encoding format.

3.2.3 Signature Payload

The Signature Payload contains data generated by the digital signature function (selected by the originator), over the entire SPP message, except for part of the Signature payload. This payload is used to verify the integrity of the data in the SPP message. Figure 5 shows the format of the Signature payload.

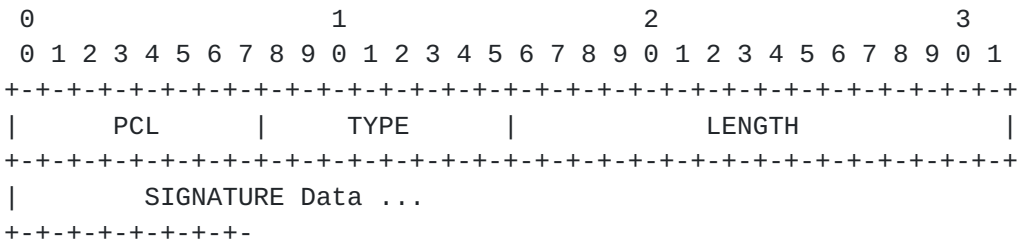


Figure 5: Signature Payload Format

The Signature payload fields are defined as follows:

PCL

A 1 octet field indicating the payload class. Signature payloads MUST contain (3) in the PCL field.

TYPE

A 1 octet field that specifies the signature algorithm employed. The currently defined signature types are:

Algorithm Type	Value
Value Not Assigned	0
RSA	1
DSA	2

values 3-250 are reserved to IANA. Values 251-255 are for private use among mutually consenting parties.

LENGTH

A 2 octet field indicating the length in octets of the SIGNATURE Data field.

SIGNATURE Data

A variable length field that contains the results from applying the digital signature function to the entire SPP message (including the PCL, TYPE, and LENGTH fields of the Signature payload), except for the Signature Data field of the Signature payload.

3.3 SPP Messages

3.3.1 Query Message

An SPP-QUERY message is comprised of an SPP header, one or more Query payloads, zero or more Record payloads, and a Signature payload, if one is required. Query messages MUST always contain a Query payload. Record payloads may optionally be included to pass policy information along with the query. If the Signature payload is employed it MUST be the last payload in the message. The Query message MTYPE value is (1). The MCODE field must be set to zero (0).

3.3.2 Reply Message

An SPP-REPLY message is comprised of an SPP header, one or more Query payloads, zero or more Record payloads which answer the corresponding Query payload, and a Signature payload, if one is required. Reply messages MUST contain a Query payload. Reply messages MUST include a Record payload unless the reply contains an MCODE of values 2-8. If the Signature payload is employed it MUST be the last payload in the message. The MTYPE value for a Reply message is (2). The following MCODE values may be used for Reply messages:

Action Type	Code Field
Value Not Assigned	0
message accepted	1
denied, administratively prohibited	2
denied, timestamp failed	3
denied, failed signature	4
denied, insufficient resources	5
denied, malformed message	6
denied, unspecified	7
partially available	8
unavailable	9
communication prohibited	10
partially available, server unreachable	11

3.3.3 Policy Message

An SPP-POL message is comprised of an SPP header, one or more Record payloads, and a Signature payload, if one is required. Policy messages MUST NOT include Query payloads. If the Signature payload is employed it MUST be the last payload in the message. The MTYPE value for a Policy message is (3). The MCODE field must be set to zero (0).

3.3.4 Policy Acknowledgement Message

An SPP-POL_ACK message is comprised of an SPP header and a Signature payload, if one is required. These messages MUST NOT contain Query or Record payloads. The status of the associated Policy message is expressed within the MCODE field. If the Signature payload is employed it MUST be the only payload in the message. The MTYPE value for a Policy Acknowledgement message is (4). The following MCODE values may be used for Policy Acknowledgement messages:

Action Type	Code Field
Value Not Assigned	0
message accepted	1
denied, administratively prohibited	2
denied, timestamp failed	3
denied, failed signature	4
denied, insufficient resources	5
denied, malformed message	6
denied, unspecified	7

3.3.5 Transfer Message

An SPP-XFR message is comprised of an SPP header, one or more Record payloads, and a Signature payload, if one is required. Transfer

messages MUST NOT include Query payloads. If the Signature payload is employed it MUST be the last payload in the message. The MTYPE value for a Transfer message is (5). The MCODE field must be set to zero (0).

3.3.6 KeepAlive Message

An SPP-KEEP_ALIVE message is comprised of an SPP header and a Signature payload, if one is required. These messages MUST NOT contain Query or Record payloads. If the Signature payload is employed it MUST be the only payload in the message. The MTYPE value for a KeepAlive message is (6). The MCODE field must be set to zero (0).

4. Policy Queries

4.1 Security Gateway Query

This basic query provides a dynamic mechanism to determine which relevant security gateways, both primary and backup, are in the path to a particular destination address. Since the answer to a request for information could depend on the identity of the requestor, the host address of the source of the intended communicaton is included in the query. Figure 6 shows the format of the Security Gateway Query.

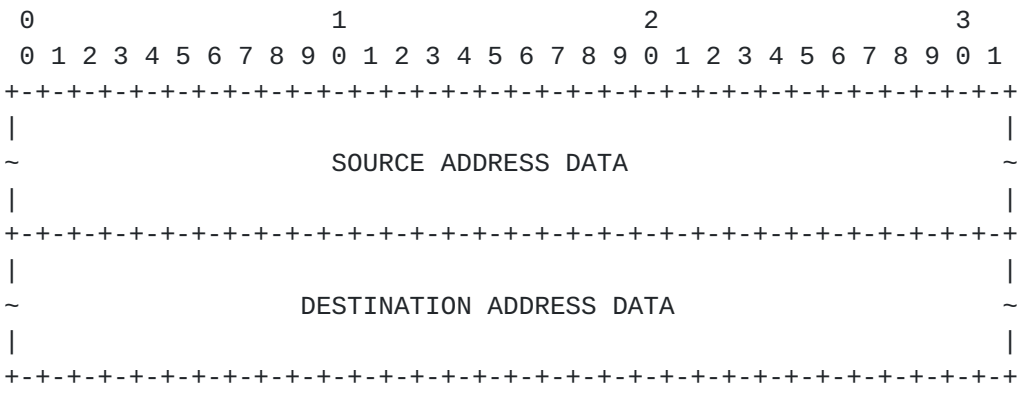


Figure 6: Security Gateway Query Format

The Security Gateway Query fields are defined as follows:

SOURCE ADDRESS DATA

This variable length field contains a single IP address (unicast) either in IPv4 or IPv6 format. The encoding format is specified in section 7. The acceptable DATA_TYPE values are 3 and 9.

DESTINATION ADDRESS DATA

This variable length field contains a single IP address (unicast) either in IPv4 or IPv6 format. The encoding format is specified in section 7. The acceptable DATA_TYPE values are 6 and 12.

4.2 COMSEC Query

The Communication Security Query (or COMSEC query) provides a dynamic mechanism for a host or security gateway to inquire if a communication having a particular set of characteristics is allowed. The communication is described in terms of source and destination addresses, protocols, source port, destination port, and other parameters as defined in section 7. These parameters are known as selectors in the IPsec context and are primarily the contents of the IP, TCP, and UDP headers. Figure 7 shows the format of the COMSEC Query.

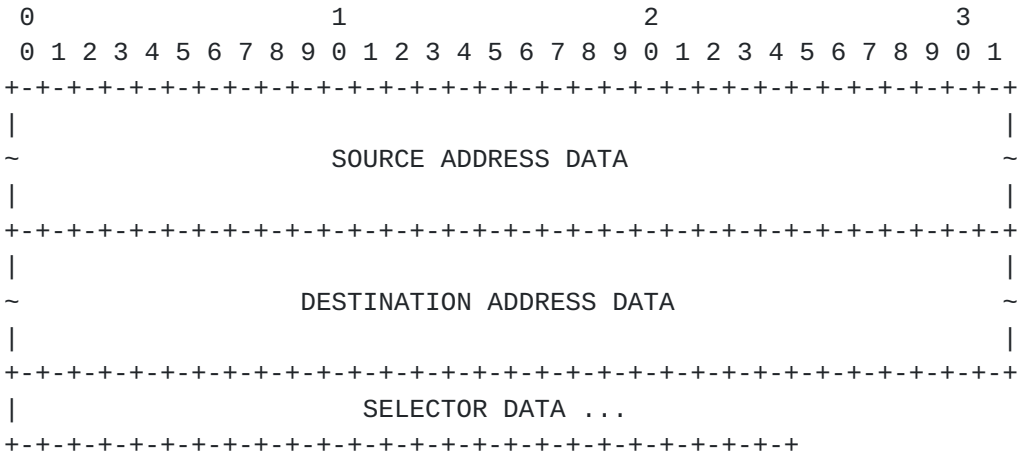


Figure 7: COMSEC Query Format

The COMSEC Query fields are defined as follows:

SOURCE ADDRESS DATA

This variable length field contains a single IP address (unicast) either in IPv4 or IPv6 format. The encoding format is specified in section 7. The acceptable DATA_TYPE values are 3 and 9.

DESTINATION ADDRESS DATA

This variable length field contains a single IP address (unicast) either in IPv4 or IPv6 format. The encoding format is specified in section 7. The acceptable DATA_TYPE values are 6 and 12.

SELECTOR DATA

This includes one or more fields following the encoding format specified in section 7. The acceptable DATA_TYPE values are 15-29, inclusive.

4.3 CERT Query

Mechanisms to dispatch and fetch public-key certificates are not part of SPP. However, in the absence of external request/dispatch mechanisms, SPP provides for a certificate request query that allows a host, security gateway, or server to solicit a certificate. Figure 8 shows the format of the CERT Query.

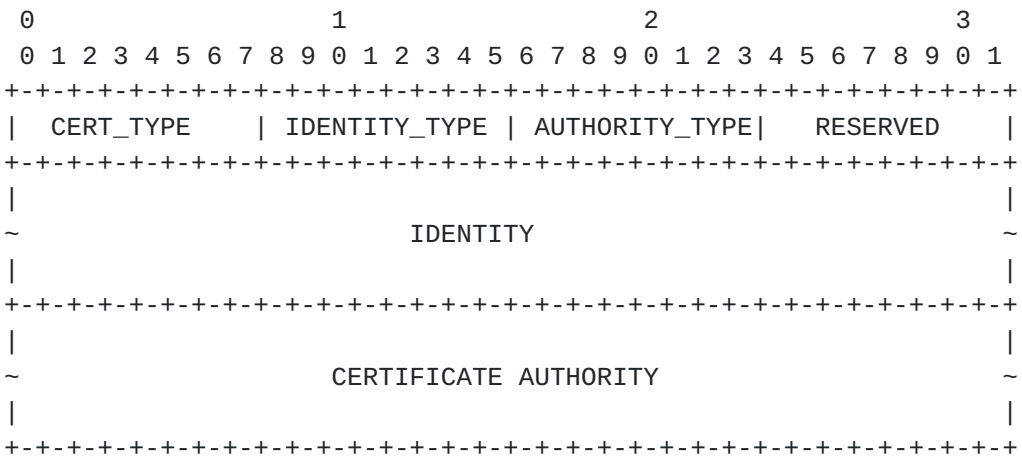


Figure 8: Certificate Query Format

The Certificate query fields are defined as follows:

CERT_TYPE

A 1 octet field that contains an encoding of the type of certificate requested. Acceptable values are listed below:

Certificate Type	Value
Value Not Assigned	0
PKCS #7 wrapped X.509 certificate	1
PGP Certificate	2
DNS Signed Key	3
X.509 Certificate - Signature	4
X.509 Certificate - Key Exchange	5
Kerberos Tokens	6
SPKI Certificate	7

values 8-250 are reserved to IANA. Values 251-255 are for private use among mutually consenting parties.

IDENTITY_TYPE

This 1 octet field indicates the type of indentity found in the Identity field. Valid values are listed below:

Value	Identity Type
0	Value Not Assigned
1	IPV4_ADDR
2	IPV6_ADDR
3	DNS Name
4	X.500 Distinguished Name

values 5-250 are reserved to IANA. Values 251-255 are for private use among mutually consenting parties.

AUTHORITY_TYPE

This 1 octet field indicates the type of authority found in the Certificate Authority field. Valid values are the same as IDENTITY_TYPE.

IDENTITY

This variable length field contains the identity of the principal by which the certificate should be located. The value MUST be of the type stated in IDENTITY_TYPE.

CERTIFICATE AUTHORITY

A variable length field containing an encoding of an acceptable certificate authority for the type of certificate requested. The value MUST be of the type stated in AUTHORITY_TYPE.

5. Policy Records

5.1 Security Gateway Record

This record contains information that indicates the IP addresses of the interfaces for the the primary and secondary security gateways protecting a host or group of hosts. The record contains the primary and secondary gateways at one point in the communication path between the source and destination addresses listed in the Security Gateway query. If the IP datagram must traverse multiple gateways, a Security Gateway Record must be included for each gateway. The list of secondary security gateways is optional. Figure 9 shows the format of the Security Gateway Record.

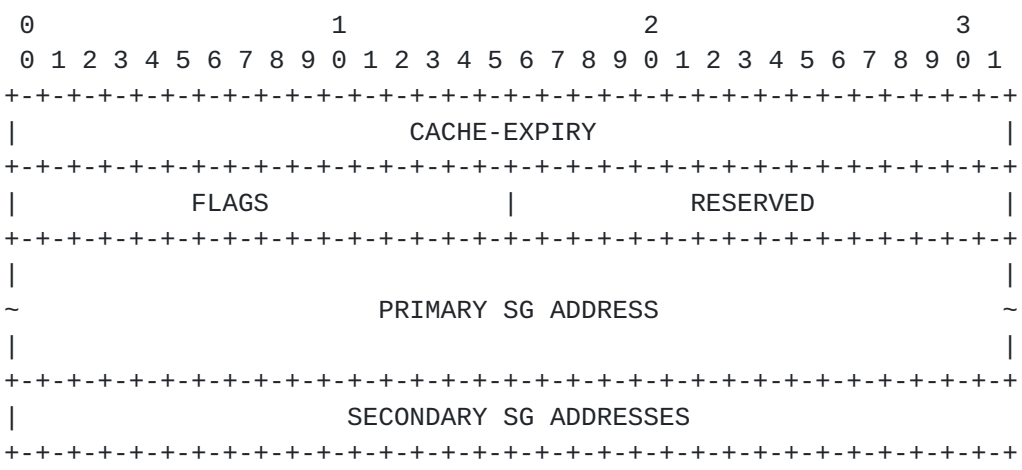


Figure 9: Security Gateway Record Format

The Security Gateway Record fields are defined as follows:

CACHE-EXPIRY

A 4 octet field indicating the maximum amount of time, in seconds, this policy record MAY be cached.

FLAGS

A 2 octet field indicating different options to aid in interpreting the security gateway data. If not in use, set value to all zeros (00)hex. Currently, no flag values are defined so this field MUST be set to (00)hex.

RESERVED

A 2 octet field reserved for future use. Set value to all zeros (0).

PRIMARY SG ADDRESS

A variable length field containing the IP address of the primary security gateway for protecting a particular host. This variable length field contains a single unicast IP address. The encoding format is specified in section 7. The acceptable DATA_TYPE values are 1 and 2.

SECONDARY SG ADDRESSES

This variable length field contains the IP addresses of one or more secondary security gateways protecting a particular host. This field may contain a list of single unicast IP addresses. The encoding format is specified in section 7. The acceptable DATA_TYPE values are 1 and 2.

5.2 COMSEC Record

The COMSEC record indicates if a communication having a particular set of characteristics is allowed or not. The communication is described in terms of source and destination addresses, protocols, source ports, destination ports, and other attributes defined in section 7. Figure 10 shows the format of the COMSEC Record.

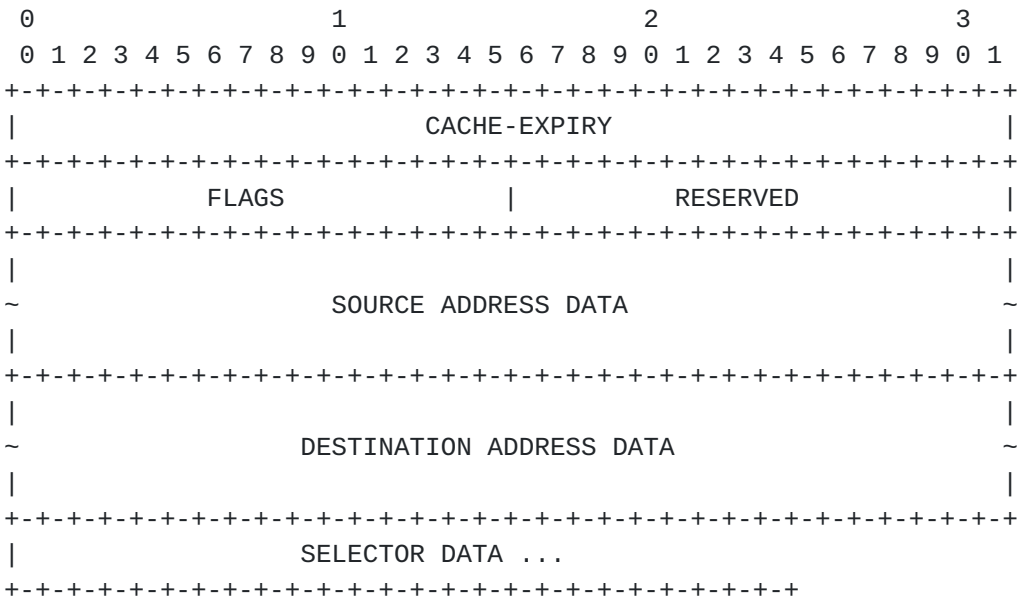


Figure 10: COMSEC Record Format

The COMSEC Record fields are defined as follows:

CACHE-EXPIRY

A 4 octet field indicating the maximum amount of time, in seconds, this policy record MAY be cached.

FLAGS

A 2 octet field indicating different options to aid in interpreting the selector data. If not in use, set value to all zeros (0). Currently, no flag values are defined so this field MUST be set to zero (0).

RESERVED

A 2 octet field reserved for future use. Set value to all zeros (0).

SOURCE ADDRESS DATA

This variable length field contains a single IP address (unicast, anycast, broadcast (IPv4 only), or multicast group), range of addresses (low and high values, inclusive), address + mask, or a wildcard address. The encoding format is specified in section 7. The acceptable DATA_TYPE values are 3-5 and 9-11, inclusive.

DESTINATION ADDRESS DATA

This variable length field contains a single IP address (unicast, anycast, broadcast (IPv4 only), or multicast group), range of addresses (low and high values, inclusive), address + mask, or a wildcard address. The encoding format is specified in section 7. The acceptable DATA_TYPE values are 6-8 and 12-14, inclusive.

SELECTOR DATA

This includes one or more fields following the encoding format specified in section 7. The acceptable DATA_TYPE values are 15-29, inclusive.

5.3 Security Association Record

Security Association Records contain selectors and security association attributes (appliers) that characterize a particular Security Association between the source and destination addresses listed in the record. This record contains data types as defined in the section 7. Figure 11 shows the format of the SA Record.

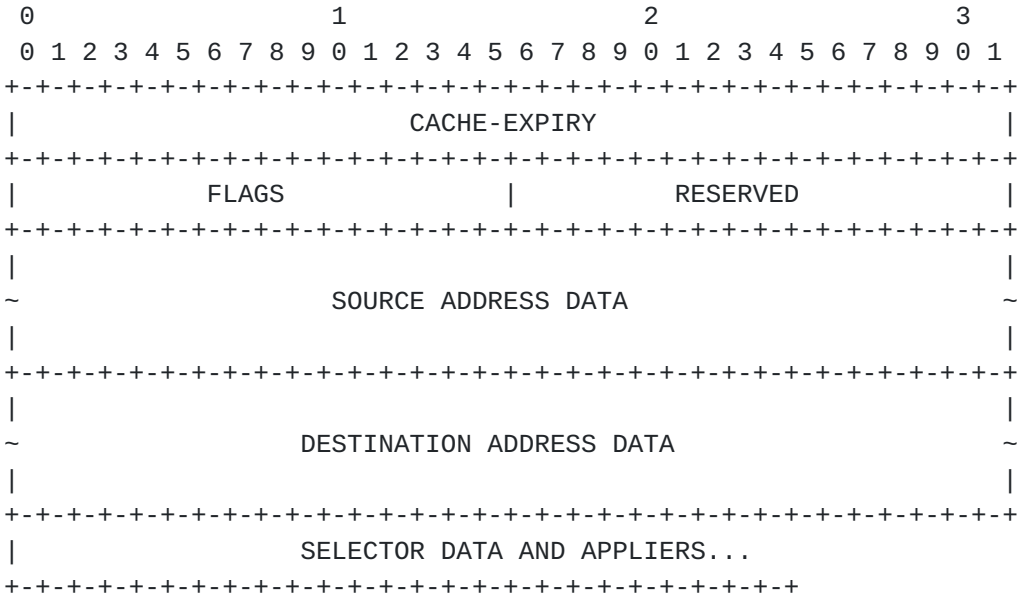


Figure 11: SA Record Format

The SA record fields are defined as follows:

CACHE-EXPIRY

A 4 octet field indicating the maximum amount of time, in seconds, this policy record MAY be cached.

FLAGS

A 2 octet field indicating different options to aid in interpreting the selector data. If not in use, set value to all zeros (0). Currently, no flag values are defined so this field MUST be set to zero(0).

RESERVED

A 2 octet field reserved for future use. Set value to all zeros (0).

SOURCE ADDRESS DATA

This variable length field contains a single IP address (unicast, anycast, broadcast (IPv4 only), or multicast group), range of addresses (low and high values, inclusive), address + mask, or a wildcard address. The encoding format is specified in section 7. The acceptable DATA_TYPE values are 3-5 and 9-11, inclusive.

DESTINATION ADDRESS DATA

This variable length field contains a single IP address (unicast, anycast, broadcast (IPv4 only), or multicast group), range of addresses (low and high values, inclusive), address + mask, or a wildcard address. The encoding format is specified in section 7. The acceptable DATA_TYPE values are 6-8 and 12-14, inclusive.

SELECTOR DATA AND APPLIERS

This includes one or more fields following the encoding format specified in section 7. The acceptable DATA_TYPE values are 15-29 and 50-51, inclusive.

5.4 Policy Server Record

The Policy Server record indicates the host, security gateway, or policy server for which a particular policy server is authoritative. It represents an assertion, typically made by a policy server, with repect to a member of a security domain that the server represents. The record includes the Identity of the policy server and the identity of a node (host, security gateway, another server, etc.). Figure 12 shows the format of the Policy Server Record.

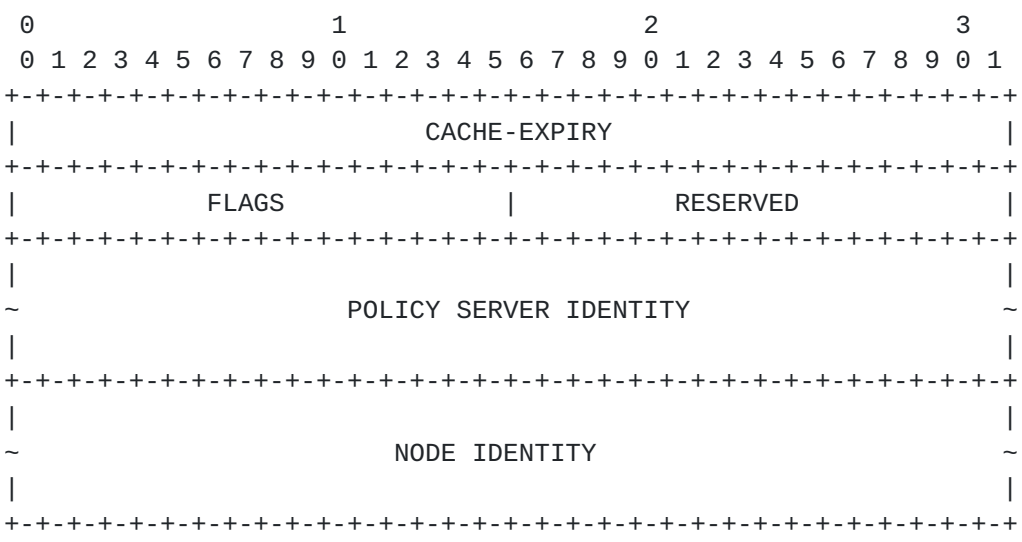


Figure 12: Policy Server record format

The Policy Server Record fields are defined as follows:

CACHE-EXPIRY

A 4 octet field indicating the maximum amount of time, in seconds, this policy record MAY be cached.

FLAGS

A 2 octet field indicating different options to aid in interpreting the server and node data. If not in use, set value to all zeros (0). Currently, no flag values are defined so this field MUST be set to zero (0).

RESERVED

A 2 octet field reserved for future use. Set value to all zeros (0).

POLICY SERVER IDENTITY

This variable length field contains the identity of the policy server. It may contain an IP address (unicast) either in IPv4 or IPv6 format. The encoding format is specified in section 7. The acceptable DATA_TYPE values are 1 and 2.

NODE IDENTITY

This variable length field contains the identity of a node for which the policy server is authoritative. It may contain an IP address (unicast) either in IPv4 or IPv6 format. The

encoding format is specified in section 7. The acceptable DATA_TYPE values are 1 and 2.

5.5 CERT Record

The CERT record contains one public key certificate. This record is provided in SPP as an alternate mechanism for certificate dispatching. Figure 13 shows the format of the CERT Record.

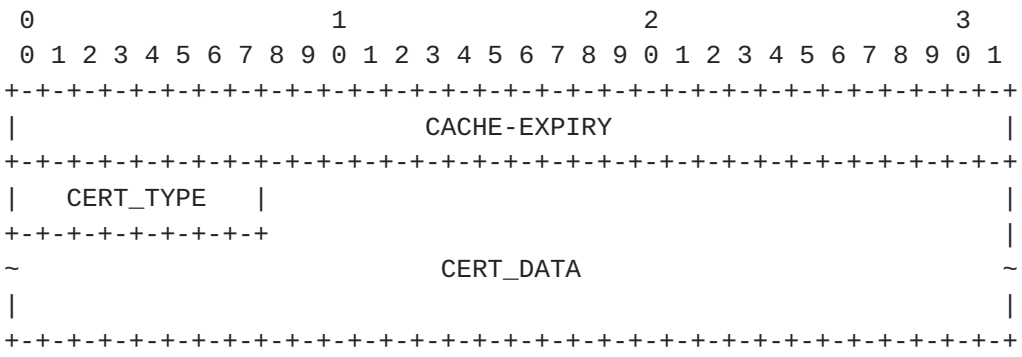


Figure 13: Certificate Record Format

CACHE-EXPIRY

A 4 octet field indicating the maximum amount of time, in seconds, this policy record MAY be cached.

CERT_TYPE

This 1 octet field indicates the type of certificate or certificate-related information contained in the Certificate Data field. The values for this field are described in Section 4.3.

CERT_DATA

This variable length field contains the actual encoding of certificate data. The type of certificate is indicated by the Certificate Type field.

6. Transfer Records

This record contains the text of the master file that is used to configure the primary policy server.

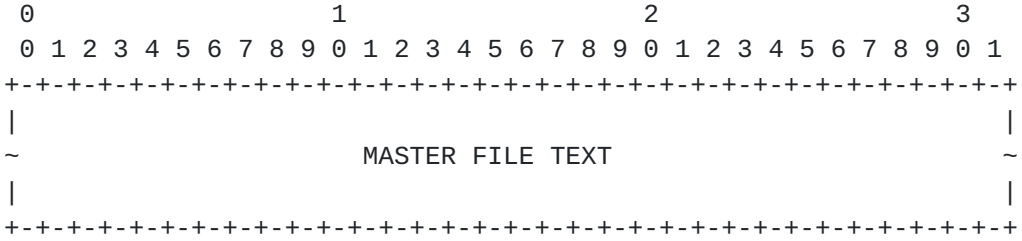


Figure 14: Security Gateway Record Format

The Transfer Record field is defined as follows:

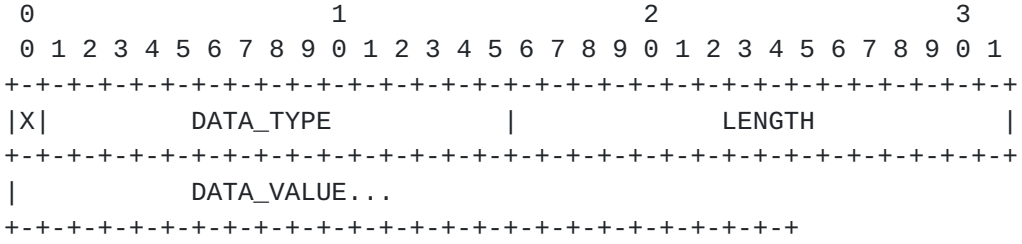
MASTER FILE TEXT

This variable length field contains the text of the master file that is used to configure the policy server.

7. Policy Attribute Encoding

Query and Record payloads include several different selector types and SA attributes with their associated values. These data are encoded following a Type/Length/Value (TLV) format to provide flexibility for representing different kinds of data within a payload. Certain Data_Types with values of length equal to 2 octets follow the Type/Value (T/V) format. The first bit of the DATA_TYPE field is used to distinguished between the two formats. A value of (0) indicates a TLV format while a value of (1) indicates TV format. This generic encoding format is depicted in figure 15.

X = 0:



X = 1:

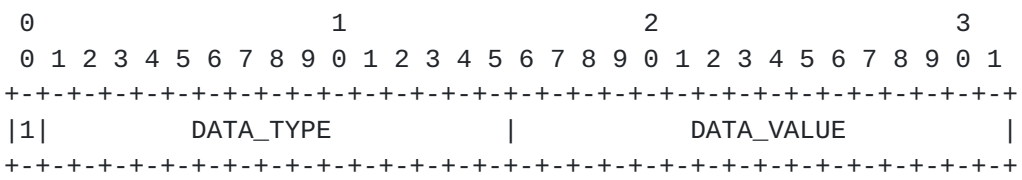


Figure 15: Generic Data Attribute Formats

The generic data attribute fields are defined as follows:

X

This bit indicates if the DATA_TYPE follows the TLV(0) or the TV(1) format.

DATA_TYPE

A 2 octet field indicating the selector type. The currently defined values are:

DATA	DATA_TYPE	X
IPV4_ADDR	1	0
IPV6_ADDR	2	0
SRC_IPV4_ADDR	3	0
SRC_IPV4_ADDR_SUBNET	4	0
SRC_IPV4_ADDR_RANGE	5	0
DST_IPV4_ADDR	6	0
DST_IPV4_ADDR_SUBNET	7	0
DST_IPV4_ADDR_RANGE	8	0
SRC_IPV6_ADDR	9	0
SRC_IPV6_ADDR_SUBNET	10	0
SRC_IPV6_ADDR_RANGE	11	0
DST_IPV6_ADDR	12	0
DST_IPV6_ADDR_SUBNET	13	0
DST_IPV6_ADDR_RANGE	14	0
DIRECTION	15	1
USER_NAME	16	0
SYSTEM_NAME	17	0
XPORT_PROTOCOL	18	0
SRC_PORT	19	0
SRC_PORT_DYNAMIC	20	0
DST_PORT	21	0
DST_PORT_DYNAMIC	22	0
SEC_LABELS	23	0
V6CLASS	24	1
V6FLOW	25	0
V4TOS	26	1
ACTION	27	1
SRC_PORT_RANGE	28	0
DST_PORT_RANGE	29	0
IPSEC_ACTION	50	0
ISAKMP_ACTION	51	0

values 30-49 and 52-3200 are reserved to IANA. Values 3200-32767 are for private use among mutually consenting parties.

LENGTH

A 2 octet field indicating the length of the selector value in octets, not including any trailing padding added to the DATA_VALUE field. The padding length is implicit.

DATA_VALUE

A variable length field containing the value of the selector specified by DATA_TYPE. If the Selector value is not aligned at

the 4 octet boundary the field MUST be padded on the right with (00)hex to align on the next 32-bit boundary.

8. SPP Message Processing

SPP messages use UDP or TCP as their transport protocol. Messages carried by UDP are restricted to 512 bytes (not counting the IP or UDP headers). Fragmentation is allowed for messages containing more than 512 **bytes. The SPP-XFR message SHOULD use TCP to transfer the contents** of the SPS Database from a primary to secondary policy server. A port number has not yet been assigned for use with SPP. For now SPP uses private UDP and TCP ports 55555.

8.1 General Message processing

For an SPP-Query or SPP-Pol message, the transmitting entity MUST do the following:

1. Set a timer and initialize a retry counter.

2. If an SPP-Reply or SPP-Pol_Ack message corresponding to the appropriate SPP-Query or SPP-Pol message is received within the time interval, or before the retry counter reaches 0, the transmitting entity continues normal operation.

3. If an SPP-Reply or SPP-Pol_Ack message corresponding to the appropriate SPP-Query or SPP-Pol message is not received within the time interval and a secondary policy server, which has not been attempted on this value of the retry counter, is available, the message is sent to the secondary server. If all the secondary servers fail to respond within the time interval, decrement the retry counter and resend the message to the primary server.

4. If the retry counter reaches zero (0) the event MAY be logged in the appropriate system audit file.

5. Following step 4, processing is more specific:

   - For hosts and security gateways:

     o the transmitting entity will clear state for this peer and revert to using conventional security mechanisms.

   - For policy servers:

     o For SPP-Pol messages, clear state for this peer.

     o For SPP-Query messages, clear state for this peer, lookup policy in the server's SPS database and send an SPP-Reply message per section 8.3 with the policy and MCODE 11.

8.2 Query Message (SPP-Query) Processing

When creating a SPP-Query message, the entity (host, security gateway, or policy server) MUST do the following:

1. Generate the Message ID value. This value starts at zero (0) and MUST be incremented by (1) with every new message.

2. Set the value of the MTYPE field to 1

3. Set the MCODE value to zero (0).

4. Set the QCOUNT and RCOUNT fields. All fields MUST be set to zero (0) when their corresponding payloads do no exist.

5. Set the flag bits accordingly and set the RESERVED field to zero (0).

6. Set the IDENTITY_TYPE and IDENTITY of the Sender of the SPP message.

7. Construct the SPP data payloads. A Query payload MUST be present in this message.

8. Local policy information related to the query MAY be included as Record payloads following the Query payloads.

9. A Policy Server record and a Cert Record SHOULD also be included in the message. A Cert record MUST be included if the query is a Cert Query to avoid a possible certificate query loop.

10. Calculate and place the timestamp value used for anti-replay attack protection.

11. If the Signature payload is required for message integrity and authentication, calculate a signature over the SPP header, SPP payloads, the PCL, TYPE, and LENGTH fields of the Signature payload. If required, the Signature payload MUST be the last payload in the message.

When a policy server receives an SPP-Query message it MUST do the following:

1. Check for SPP access control. If enabled, read the IP address in the Sender's field of the SPP header.

   - Verify whether or not the message is allowed. If the message is not allowed then:

     o send an SPP-Reply message with the MCODE 2 or 7

o discard message and the event MAY be logged.

- If the message is allowed, continue.

2. Check timestamp field for anti-replay protection. If a replayed message is detected:

   - send an SPP-Reply message with the MCODE 3 or 7
   - discard message and the event MAY be logged.

3. If the message requires signature validation.

   - If a certificate record is present, the server MUST process it, however, if the server already has a valid key for the host or server identified in the certificate, the certificate MAY be ignored.

   - Fetch certificate or key corresponding to the IP address found in the sender field of the SPP header.

   - If a certificate or key is not available the entity MAY, depending on configuration:

     o choose to abort validation process, send SPP-Reply message with MCODE 5 or 7, discard the message, and MAY log the event.

     o send an SPP-Query message to the source of the IP address found in the sender field of the SPP header with a CERT Query payload. Keep the SPP-Query message until the SPP-Reply returns. Extract certificate or key, validate it and proceed.

   - Once a validated certificate or key is available then validate signature.

     o If validation fails, send SPP-Reply message with MCODE 5 or 7, discard the message, and the event MAY be logged.

4. Parse the Query records.

   - If the SPP-Query message only contains cert queries:

     o If the Identity field identifies the server or a member of the server's security domain, send an SPP-Reply message per section 8.3 with one or more cert records with the certificates in the certification chain between the requested Identity and Certificate_Authority and MCODE 1.

     o Otherwise, send an SPP-Reply message per section 8.3 with with MCODE 9 or 7.

- If the SPP-Query message does not only contain cert queries:

  o Check the Destination Address Data field to determine if the message received was intended for a node that is a member of the server's security domain.

  o Continue processing.

5. If the message is for a node that is a member of the server's security domain or the D bit in the SPP header is set and the server is the outermost server that is authoritative over the client or server that sent the message, then:

   - Using src, dst, and any other applicable fields found in the Query Payload, search the SPS database for an appropriate policy.

     o If a policy is found then construct an SPP-Reply message per section 8.3 with appropriate payloads and MCODE 1.

     o If a policy is not found then construct an SPP-Reply message per section 8.3 with appropriate payloads and MCODE 9 or 7.

6. If the message is for a node that is not part of the server's security domain then:

   - If the I and R bits are not set in the SPP header, check the Cache database for any relevant policies that apply to this communication.

     o If a policy is found then construct a SPP-Reply message per section 8.3 with appropriate payloads and MCODE 1.

     o If a policy is not found then continue.

   - Check the Local database for any relevant policies that apply to this communication.

   - If the server is authoritative for the source address of the query or a matching policy is found (A matching policy is defined as a policy that either produces a comsec record with an action attribute with the value "deny", or a policy that would produce an SA record with one or more IPsec action and IKE action attributes. A policy that only produces a comsec record with an action attribute with the value "permit" is not considered matching for this step.):

     o Generate a new SPP-Query message. The new message MUST use the same query payload as the old message. If the new query will include policy from the server, then the policy

used SHOULD be the server's local policy merged with any policies received with the query message. The Sender Address will be the address of the server. The destination for this message MUST be the one in the original SPP-Query received.

o Keep state. Upon reception of the corresponding SPP-Reply the policy server MUST send an SPP-Reply addressed to sender of the original SPP-Query.

- If the server is not authoritative for the source address of the query and a matching policy is not found then:

o The policy server MUST send the SPP-Query message unchanged. The SPP-Query message MUST use the same source port that was used to send it to the server so the next server that processes the query will return it to the correct port. This SPP-Query message MUST NOT be added to the retry queue (section 8.1).

8.3 Reply Message (SPP-Reply) Processing

When creating a SPP-Reply message, the policy server MUST do the following:

1. Copy the Message ID value from the corresponding SPP-Query message into the Message ID field.

2. Set the value of the MTYPE field to 2

3. Set the MCODE value.

4. Set the QCOUNT and RCOUNT fields. All fields MUST be set to zero (0) when their corresponding payloads do no exist.

5. Set the flag bits accordingly and set the RESERVED field to (0).

6. Set the IDENTITY_TYPE and IDENTITY of the Sender of the SPP message.

7. Copy the Query payloads from the SPP-Query message to the SPP-Reply message.

8. Construct the SPP data payload. Unless there is an error, at least one record corresponding to each Query payload MUST be present.

9. A policy server record and a CERT record MUST be added to the SPP-Reply message if the the query to which this is a reply did NOT have the T bit set. If the T bit is set, the records SHOULD NOT be added.

10. Calculate and place the timestamp value used for anti-replay attack protection.

11. If the Signature payload is required for message integrity and

authentication, calculate a signature over the SPP header, SPP payloads, the PCL, TYPE, and LENGTH fields of the Signature payload. If present, the Signature payload MUST be the last payload in the message.

12. The SPP-Reply message is sent to the host that is listed in the Sender ID field of the SPP-Query to which this is a reply.

When a host or security gateway receives an SPP-Reply message it MUST do the following:

1. Read the Message ID field. If the value does not match the value of an outstanding SPP-Query message from a policy server then:
   - silently discard the message and the event MAY be logged.

2. If Message ID matches, Check the timestamp field for anti-replay protection. If a replayed message is detected the message is silently discarded and the event MAY be logged.

3. Establish if the message requires signature validation by searching for a Signature payload:
   - if signature validation is required proceed with step 4.
   - if signature validation is not required proceed with step 6.

4. Validate the signature on the message.

   - If a certificate record is present, the server MUST process it, however, if the server already has a valid key for the host or server identified in the certificate, the certificate MAY be ignored.

   - Fetch certificate or key corresponding to the IP address found in the sender field of the SPP header.

   - If a certificate or key is not available the entity MAY:

     o choose, depending on configuration, to abort validation process, discard the message and MAY log the event.

     o send an SPP-Query message to the source of the IP address found in the sender field of the SPP header with a CERT query payload. Keep SPP-Reply message until the corresponding SPP-Reply returns. Extract certificate or key, validate it and proceed.

5. Once a validated certificate or key is available then validate signature.

   If validation fails:
   - the message is silently discarded and the event MAY be logged

   If validation succeeds, continue processing.

6. For Policy Servers, validate the chain of trust.

A valid chain proves that policy has only been applied by servers authorized to control policy over either the source or destination host of the requested policy. The "chain" represents the hierarchy of policy servers authoritative over

the source of the communication and the heirarchy over the destination. The chain may have a single "break" between the two policy servers that represent the top of the two heirarchies. It is formed by the information in the Policy Server records and must be cryptographically proven that the relationships described in those records are true.

- For each Policy Server record, verify that the Policy Server is authoritative over the Node. This MUST be verified cryptographically which MAY be accomplished using X.509 certificates [PKIXP1]. See section 11 for more details.

- Use the Policy Server records to Create a chain of hosts from the destination host to this policy server where two records are linked if the Node in one is the Policy Server in another.

- If the chain has no breaks, then this policy server MUST be authoritative over the sender of the reply, otherwise drop the message and stop processing it.

- If the chain has one break, then the last policy server on the chain MUST be the sender of the reply, otherwise drop the message and stop processing it.

- If the chain has two or more breaks, then there is an error in the chain so drop the message and stop processing it.

- Verify that the Policy Server that is authoritative over the destination host is authoritative over ALL destination hosts in any comsec records.

7. If MCODE value is 2-7, 9 or 10:

   For hosts or security gateways:
   - clear all the state and stop processing
   For policy servers:
   - create an SPP-Reply message using the same MCODE value.

8. If the reply received correponds to a Cert query and the MCODE is either (1) or (8) (accept or partially unavailable), process message that was held up waiting for the cert.

9. For Policy Servers:

   - Search the Local Policy Database for a policy entry that matches the policy expressed in Query payload.

   - If the R bit is not set, merge the local and non-local policy information using the policy resolution proccess outlined in section 9.

- If the R bit is set, include both the policies found in the Local Policy Database and the policies in the reply to send in the new reply.

- If the merge fails send an SPP-Reply message with MCODE 10 or 7 and cache the failure.

- If the merge succeeds or the R bit is set:

  o If the R and C bits are not set, cache policy information. This includes all Record payloads.
  o send an SPP-Reply message with the resulting policy records and MCODE 1.
  o If the R and D bits are not set and the merge indicated that policies should be sent to one or more security gateways, construct a signal for each gateway by creating an SPP-Pol message with the appropriate policy from the merge.

10. For hosts or security gateways:

- verify that the information in the Record payload corresponds to the information in the Query payload.
- extract the records and create a policy entry in the SPD according to local format. The policy is entered in the SPD ONLY if local configuration allows.

8.4 Policy Message (SPP-Pol) Processing

When creating a SPP-Pol message, the entity (host, security gateway, or policy server) MUST do the following:

1. Generate the Message ID value. This value starts at zero (0) and MUST be incremented by (1) with every new message.

2. Set the value of the MTYPE field to 3.

3. Set the MCODE value to zero (0).

4. Set the RCOUNT field. The QCOUNT field MUST be set to zero (0) since no query payloads exist.

5. Set the flag bits accordingly and set the RESERVED field to (0).

6. Set the IDENTITY_TYPE and IDENTITY of the Sender of the SPP message.

7. Construct the SPP data payloads. A Record payload MUST be present in this message. Query payloads MUST NOT be present.

8. Calculate and place the timestamp value used for anti-replay attack protection.

9. If the Signature payload is required for message integrity and

authentication, calculate a signature over the SPP header, SPP payloads, the PCL, TYPE, and LENGTH fields of the Signature payload. If required, the Signature payload MUST be the last payload in the message.

When a policy server receives an SPP-Pol message it MUST do the following:

1. Check for SPP access control. If enabled, read the IP address in the Sender's field of the SPP header.

   - Verify whether or not the message is allowed. If the message is not allowed then:

     o send an SPP-Pol_Ack message with the MCODE 2 or 7
     o discard message and the event MAY be logged.

   - If the message is allowed, continue.

2. Check timestamp field for anti-replay protection. If a replayed message is detected:

   - send an SPP-Pol_Ack message with the MCODE 3 or 7
   - discard message and the event MAY be logged.

3. If the message requires signature validation:

   - If a certificate record is present, the server MUST process it, however, if the server already has a valid key for the host or server identified in the certificate, the certificate MAY be ignored.

   - Fetch certificate or key corresponding to the IP address found in the sender field of the SPP header.

   - If a certificate or key is not available the entity MAY, depending on configuration:

     o choose to abort validation process, send SPP-Pol_Ack message with MCODE 5 or 7, discard the message and MAY log the event.

     o send an SPP-Query message to the source of the IP address found in the sender field of the SPP header with a CERT Query payload. Keep SPP-Pol message until the SPP-Reply returns. Extract certificate or key, validate it and proceed.

   - Once a validated certificate or key is available then validate signature.

     o If validation fails the message is silently discarded and the event MAY be logged

4. If signature was not required or upon successful validation of a

signature parse the payloads.

5. For hosts and security gateways:

   - extract the records and create a policy entry in the cache database. The policy MAY also be entered in the SPD, ONLY if configuration allows.

6. For Policy Servers:

   - extract the records, find corresponding policies in the server's SPS database, merge the two sets of policies, and create a policy entry in the cache database.

7. Send an SPP-Pol_Ack message with MCODE 1.

8.5 Policy Acknowledgement Message (SPP-Pol_Ack) Processing

When creating a SPP-Pol_Ack message, the policy server MUST do the following:

1. Copy the Message ID value from the corresponding SPP-Pol message into the Message ID field.

2. Set the value of the MTYPE field to 4

3. Set the MCODE value.

4. Set the QCOUNT and RCOUNT fields. All fields MUST be set to zero (0).

5. Set the flag bits accordingly and set the RESERVED field to (0).

6. Set the IDENTITY_TYPE and IDENTITY of the Sender of the SPP message.

7. Query or Record payloads MUST NOT be present.

8. Calculate and place the timestamp value used for anti-replay attack protection.

9. If the Signature payload is required for message integrity and authentication, calculate a signature over the SPP header, the PCL, TYPE, and LENGTH fields of the Signature payload.

When a host, security gateway, or policy server receives an SPP-Pol_Ack message the entity MUST do the following:

1. Read the Message ID field. If the value does not match the value of an outstanding SPP-Pol message from a policy server then:
   - silently discard the message and the event MAY be logged.

2. If Message ID matches then check the timestamp field for anti-replay protection. If a replayed message is detected the message is silently discarded and the event MAY be logged.

3. If the message is original (not replayed) and the message requires signature validation then:

   - If a certificate record is present, the server MUST process it, however, if the server already has a valid key for the host or server identified in the certificate, the certificate MAY be ignored.

   - Fetch certificate or key corresponding to the IP address found in the sender field of the SPP header.

   - If a certificate or key is not available the entity MAY:

     o choose, depending on configuration, to abort validation process, discard the message and MAY log the event.

     o send an SPP-Query message to the source of the IP address found in the sender field of the SPP header with a CERT Query payload. Keep SPP-Pol_ack message until the SPP-Reply returns. Extract certificate or key, validate it and proceed.

4. Once a validated certificate or key is available then validate the signature.
   If validation fails:
   - the message is silently discarded and the event MAY be logged
   If validation succeeds:
   - read the MCODE field and proceed accordingly. If no errors, clear all the state for this message and proceed.

8.6 Transfer Message (SPP-XFER) Processing

When creating an SPP-Xfer message, the policy server MUST do the following:

1. Generate the Message ID value. This value starts at zero (0) and MUST be incremented by (1) with every new message.

2. Set the value of the MTYPE field to 5.

3. Set the MCODE value to (0).

4. Set the RCOUNT field. The QCOUNT field MUST be set to zero (0) since no query payloads exist.

5. Set the flag bits accordingly and set the RESERVED field to zero (0).

6. Set the IDENTITY_TYPE and IDENTITY of the Sender of the SPP message.

7. Construct the SPP data payloads. A single Transfer Record MUST be present in this payload and MUST contain the master file used to configure this policy server.

8. Calculate and place the timestamp value used for anti-replay attack protection.

9. If the Signature payload is required for message integrity and authentication, calculate a signature over the SPP header, SPP payloads, the PCL, TYPE, and LENGTH fields of the Signature payload. If required, the Signature payload MUST be the last payload in the message.

When a security gateway receives an SPP-Xfer message it MUST do the following:

1. Check for SPP access control. If enabled, read the IP address in the Sender's field of the SPP header.

   - Verify whether or not the message is allowed. If the message is not allowed then:

     o discard message and the event MAY be logged.

   - If the message is allowed, continue.

2. Check timestamp field for anti-replay protection. If a replayed message is detected:

   - discard message and the event MAY be logged.

3. If the message requires signature validation:

   - If a certificate record is present, the server MUST process it, however, if the server already has a valid key for the host or server identified in the certificate, the certificate MAY be ignored.

   - Fetch certificate or key corresponding to the IP address found in the sender field of the SPP header.

   - If a certificate or key is not available the entity MAY, depending on configuration:

     o choose to discard the message, and MAY log the event.

     o send an SPP-Query message to the source of the IP address found in the sender field of the SPP header with a CERT Query payload. Keep the SPP-Query message until the SPP-Reply returns. Extract certificate or key, validate it and proceed.

   - Once a validated certificate or key is available then

validate signature.

o discard the message, and the event MAY be logged.

4. If signature was not required or upon successful validation of a signature parse the payload.

   - extract the Transfer Record and save the master file that it contains.

   - Flush the contents of the SPS database, domain database, and cache.

   - Load the new information from the transferred master file into the databases.

8.7 KeepAlive Message (SPP-KEEP_ALIVE) Processing

When creating an SPP-Keep_Alive message, the policy server MUST do the following:

1. Generate the Message ID value. This value starts at zero (0) and MUST be incremented by (1) with every new message.

2. Set the value of the MTYPE field to 6.

3. Set the MCODE value to zero (0).

4. Set the QCOUNT and RCOUNT fields. All fields MUST be set to zero (0).

5. Set the flag bits accordingly and set the RESERVED field to (0).

6. Set the IDENTITY_TYPE and IDENTITY of the Sender of the SPP message.

7. Query or Record payloads MUST NOT be present.

8. Calculate and place the timestamp value used for anti-replay attack protection.

9. If the Signature payload is required for message integrity and authentication, calculate a signature over the SPP header, the PCL, TYPE, and LENGTH fields of the Signature payload.

When a host or security gateway receives an SPP-Keep_Alive message it MUST do the following:

1. Check for SPP access control. If enabled, read the IP address in the Sender's field of the SPP header.

   - Verify whether or not the message is allowed. If the message is not allowed then discard message and the event MAY be logged.

2. Check timestamp field for anti-replay protection. If a replayed message is detected discard message and the event MAY be logged.

3. If the message requires signature validation then:

   - If a certificate record is present, the server MUST process it, however, if the server already has a valid key for the host or server identified in the certificate, the certificate MAY be ignored.

   - Fetch certificate or key corresponding to the IP address found in the sender field of the SPP header.

   - If a certificate or key is not available the entity MAY depending on configuration:

     o choose to abort validation process, discard the message and the event MAY be logged.

     o send an SPP-Query message to the source of the IP address found in the sender field of the SPP header with a CERT Query payload. Keep SPP-Keep_Alive message until the SPP-Reply returns. Extract certificate or key, validate it and proceed.

   - Once a validated certificate or key is available then validate signature.

     o If validation fails the message is silently discarded and the event MAY be logged

4. If signature validation was not required or upon successful validation of a signature, the event MAY be logged.

9. Policy Resolution

When a policy server receives a reply, it must merge its local policy for the communication with any non-local policies contained in the reply. The merging process creates a new policy that is the intersection of the local and remote policies. It then uses the merged policy as its reply to the query and caches it. The policy resolution process is as follows.

A message (set of policies) consists of one comsec record and zero or more SA recs that apply to the communication in the comsec record.

1. **Get local and remote policies for the requested communication.**

2. **Verify that the remote policy answer the query.** This may be accomplished by intersecting the query with the comsec record int the answer and verify that they have a non-nil intersection.

3. **Merge the local and remote comsec records.** If they don't merge return error. If they do put merged comsec record in answer.

4. Merge the two sets of policies (SA recs). The merge must:

- Find the intersection of the policies between matching endpoints.

  o If the intersection is nil, then the policies do not permit the communication and an error should be returned. It is not necessary to continue processing other endpoints.

  o If the intersection is not nil, then the intersection should be added to the reply policy.

- Take into account ipsec action locations when determining the endpoints to intersect.

- preserve the ordering of the policies so that the SA recs are generated in the correct order.

5. The policy created by the intersections is the policy that should be cached and used as a reply to the query.

Step 4 requires that the policy server must be able to determine all the sets of endpoints described by the policy. The endpoint information comes from two places: the source and destination addresses in the query (which is possibly more specific than those fields in the policies) and the location information in the ipsec_action attribute. Section 9.1 describes a method of processing this step in more detail.

The location information may offer the policy server some flexibility in how it interprets endpoints for the communication. For example, if the policy indicates a tunnel must be established with any host or gateway in the source or destination host's domain, the policy server can choose the endpoint within the bounds of the policy. This choice can be made randomly, using a set policy (e.g., always choose the outermost gateway permitted by the policy), or using additional information the server may maintain for this purpose. For example, the server may keep track of previous policy decisions it made and use those as hints to which security associations may already exist. It can then try to make decisions that will allow these security associations to be reused.

9.1 Expansion of step 4

This section describes a method of merging two sets of policies. It describes which policies should be merged together and how to maintain the appropriate order of the policies. It does not describe the merging of individual policies which involves taking the intersecting the selectors and appliers. Other methods to implement the merge may be used.

Start with two sets of ordered policies. One set is the remote set of policies that has been received through an SPP exchange. The other is a local set of policies that was found in a local policy database.

1) Attempt to merge any records that may be merged and interleave messages to preserve ordering:

for each SA rec in remote message:

Check remote src, dst, location src, and location dst endpoints against SA recs in local message, in order. Check against the following rules which are grouped into three priorities. The high priority is for those policies that match and should be merged. The low priority is for those policies that don't merge, but care must be taken to insure that they are ordered correctly. The final group is policies that do not match in any way.

Attempt to match each of the unprocessed local SA recs to the remote SA rec. Find the SA rec (or SA recs, there might be more than one applicable match) to find the highest priority match.

If the highest priority match is a low priority match, compare it to the remaining remote SA recs. If there is a remote SA rec to which it has a high priority match, take the local SA rec and goto step 3.

Otherwise, process the romote SA rec against each of the highest matching local SA recs as specified by their priority.

High Priority Matches:

o if both the remote src and dst match the local src and dst and either

the remote location src and dst and the local location src and dst match

or

the local or remote location src and dst are set to zero and the other location src and dst match the src and dst

- take each unprocessed SA rec in the local message before this matching one and goto 3.

- merge the two SA recs and goto 3.

o if (remote src and dst match the local location src and dst and the remote location src and dst either match the remote src and dst or are zero) or (local src and dst match the remote location src and dst and the local location src and dst either match the local src and dst or are zero), then:

- take each unprocessed SA rec in the local message
  before this one and goto 3.

- Take the SA rec with the location endpoints that matched the other src and dst and send it to step 3, but return the results here for further processing.

- merge the other SA rec with the SA rec resulting from 3 that has the same src and dst. Take this merged SA rec and goto 3. If the sa rec that was sent to step 3 above was the local sa rec, send the remaining SA recs that resulted from 3 to step 3, otherwise send them to step 2.

Low Priority Matches:

o If both the remote src and dst match the local src and dst but the remote location src and dst and the local location src and dst do not match, then they do not merge.

- take each unprocessed SA rec in the local message before this one and goto 3.

- On a query, order the local SA rec before the remote SA rec. On a reply, order the remote SA rec before the local SA rec. Maintaining query/reply order, take the local SA rec to step 3 and the remote SA rec to step 2.

o if the remote src and local src match, but the dsts do not, or the remote dst and local dst match and the srcs do not, then:

- take each unprocessed SA rec in the local message before this one and goto 3.

- On a query, order the local SA rec before the remote SA rec. On a reply, order the remote SA rec before the local SA rec. Maintaining query/reply order, take the local SA rec to step 3 and the remote SA rec to step 2.

No Matches:

o if no endpoints match, then goto 2.

If there are any SA recs remaining in the local message, take each of them to step 3.

2) Verify local policies for non end-to-end SA recs. This involves finding policies that are being merged which involve intermediate enforcement points and check the local policy for the intermediate points.

The SA rec is directly from the remote message so the communication must be verified:

A) Check the src and dst and the location src and dst.
If a pair is the same as the communication endpoints,
zero, or is ambiguous, do not continue processing it.
Continue processing any that do not fit these conditions.
If neither pair continues processing, goto 3.

- o look in the local database for a policy that matches the communication described by these endpoints.
- o check the comsec record to verify that it is permitted. (If not deny entire communication).
- o For each SA rec from the local policy, except a matching SA rec, goto 3.
- o If there is a matching SA rec, merge it with the remote SA rec and goto 3. Else take the remote SA rec and goto 3.

3) Process location fields to resolve any ambiguities that they may describe and define any new SAs that the location fields may specify.

If the loc fields are empty, then goto 4.

- If the location fields and local decision making over any ambiguities indicate that a host or GW controlled by this PS should be a LOC DST, then replace the LOC DST with the gateway's ipaddress or DNS name.
- If the location fields and local decision making over any ambiguities indicate that a host or GW controlled by this PS should be a LOC SRC, then replace the LOC SRC with the gateway's ipaddress or DNS name.
- If both the location src and location dst fields are specific hosts or gateways (i.e. not ambiguous) and not the same as the src and dst fields, create a new SA rec to reflect the policy between the location src and dst.

  o create the new SA rec.

- Take the original SA rec and any that have been added by this process and goto 4.

4) Create a reply message and signals to enforcement points as needed.

If this is a query:

Add the SA rec to the answer.

If this is a reply:

If SA rec dst is for GW controlled by this PS:
- If no signal message exists for this GW, yet, create one with a comsec record formed from the information in the previous SA rec (in the order that has been established).
- Add the SA rec to the signal for this GW.
- Add SA rec to answer.

If SA rec src is for GW controlled by this PS:
- If no signal message exists for this GW, yet, create one with a comsec record formed from the information in the previous SA rec (in the order that has been established).
- Add the SA rec to the signal for this GW.

If neither endpoint is for GW controlled by this PS:
- Add SA rec to answer.

10. IANA Considerations

This document contains many "magic numbers" to be maintained by the IANA. This section explains the criteria to be used by the IANA to assign numbers beyond the ones defined in this document.

10.1 Message Type

The MTYPE field of the SPP Header (section 3.1) defines message exchange types for SPP. Requests for assignment of new message type values 7-250 must be accompanied by a reference to a standards-track or Informational RFC which describes the new message type and how it should be processed. Values 251-255 are for private use.

10.2 Message Code

The MCODE field of the SPP Header (section 3.1) defines the acceptable return codes for an SPP message. Requests for assignment of new message code values 12-250 must be accompanied by a description of the conditions under which the code is returned. Values 251-255 are for private use.

10.3 Identity Type

The Identity Type field of the SPP Header (section 3.1) defines the acceptable formats for identifying the sender of an SPP message. Requests for assignment of new identity types 4-250 must be accompanied by a description of the format of the corresponding SENDER IDENTITY field in the header. Values 251-255 are for private use.

10.4 Payload Class

The first octect of each payload header (section 3.2) defines the type of payload that follows it. Requests for assignment of new message type values 4-250 must be accompanied by a reference to a standards-track or Informational RFC which describes the format of the payload's header and data. Values 251-255 are for private use.

10.5 Query Type

The query type (section 3.2.1) defines how the payload data will be interpreted and answered. Requests for assignment of new query type values 4-65000 must be accompanied by a reference to a standards-track or Informational RFC which describes the format of the data and how it should be used. Values 65001-65535 are for private use.

10.6 Record Type

The record type (section 3.2.2) defines how the payload data will be interpreted. Requests for assignment of new record type values 4-65000 must be accompanied by a reference to a standards-track or Informational RFC which describes the format of the data and how it should be used. Values 65001-65535 are for private use.

10.7 Signature Type

The signature type (section 3.2.3) defines the signature algorithm used to sign the SPP message. Requests for assignment of new signature type values 3-250 must be accompanied by a reference to a standards-track or Informational RFC or a reference to published cryptographic literature which describes this algorithm. Values 251-255 are for private use.

10.8 Certificate Type

The Cert Type field of the Certificate query and record (section 3.1) defines the type of certificate requested or included in the payload. Requests for assignment of new certificate types 8-250 must be accompanied by a description of certificate and its encoding. Values 251-255 are for private use.

10.9 Certificate Identity Type

The Identity Type and Authority Type fields of the certificate query (section 4.3) define the acceptable formats for identifying the host and its certificate authority for which a certificate is requested. Requests for assignment of new certificate identity types 5-250 must be accompanied by a description of the format of the corresponding IDENTITY and CERTIFICATE AUTHORITY fields in the payload. Values 251-255 are for private use.

10.10 Attribute Data Type

The Data_Type field of the attribute encoding (section 7) defines the type of attribute included in the data_value field. Requests for assignment of new attribute data types 30-49 and 52-3200 must be accompanied by a description of the X bit indicating if it is in TLV or TV format, a detailed description of the Data_Value field corresponding to the attribute type, and in which record and query data fields the type may be used. Values 3200-32767 are for private use.

10.11 User Name Type

The Name_Type field of the user name attribute (section A.16) defines the data in the Name field of the attribute. Requests for assignment of new user name types 2-250 must be accompanied by a description of the corresponding Name field. Values 251-255 are for private use.

10.12 System Name Type

The Name_Type field of the system name attribute (section A.17) defines the data in the Name field of the attribute. Requests for assignment of new system name types 9-249 must be accompanied by a description of the corresponding Name field. Values 251-255 are for private use.

10.13 IPsec Action Attribute

The assigned values of Lifetime_Type, Cipher_Alg, Int_Alg_Esp, Int_Alg_Ah, and Ipcomp_Alg use the values of their associated fields in [Piper98] and are updated when the IANA updates their values in [Piper98].

The Loc_Type field of the IPsec action attribute (section A.30) defines the type of location address in the Loc_Src and Loc_Dst fields. Requests for assignment of new location types 5-250 must be accompanied by a description of the corresponding Loc_Src and Loc_Dst field. Values 251-255 are for private use.

The Loc_Src and Loc_Dst fields of the IPsec action attribute (section A.30) may define a general location type. Requests for assignment of new general location values 5-250 must be accompanied by a description of the general location type. Values 251-255 are for private use.

10.14 IKE Action Attribute

The assigned values of Group Description, Group_Type, Auth_Method, PRF, Lifetime_Type, Cipher_Alg, and Hash_Alg use the values of their associated fields in [Harkins98] and are updated when the IANA updates their values in [Harkins98].

The Mode field of the IKE action attribute (section A.31) defines the IKE Mode. Requests for assignment of new Modes 3-250 must only be done when new modes are added to the IKE protocol. Values 251-255 are for private use.

11. Security Considerations

All SPP messages MUST be authenticated to prove which policy server sent the message and that it hasn't been modified en-route. The authentication MAY be provided using the signature payload provided by SPP or some other mechanism such as IPsec.

However, since the policy data may change during SPP exchanges, the messages cannot maintain a signature from every policy server that is involved in the policy exchange. SPP depends on a chain-of-trust for end-to-end authentication. Messages between policy servers are authenticated and contain policy server records, which claim authorization over a node, and certificates, which include signed proof that the server is authoritative the node. The receiving server can use this information to create a chain of servers involved in the policy exchange from itself to the server authoritative over the destination of the query. This chain of authorized servers is used to prove that only servers that have authorization to be involved in the communication were involved. Section 8.3 for details on how the chain is created and verified.

Policy information may be considered sensitive, since examining policies may expose expoitable weaknesses in the policies. The distribution of policies may be limited to reduce this risk. Policy distribution MAY be limited to those nodes that need to know the information. Limiting distribution any further negates the purpose of the protocol so is not allowed for proper use of SPP.

Additional protections, such as privacy protection, may be desired by some domains. This can be achieved by encrypting SPP data. Encrypting SPP messages is out of scope of this document and may be discussed elsewhere.

SPP uses timestamps to protect against replay attacks. This requires that nodes have adequately synchronized time-of-day clocks. It is necessary to choose an appropriately sized window of time in which timestamps may be accepted. If the window is too small, valid messages may be discarded. On the other hand, if the window is too large it may leave the server open to replay attacks.

Acknowledgments

The authors thank Charles Lynn, Steve Kent and John Zao for their participation in requirements discussions for the Security Policy System. Our gratitude to Charlie Lynn, Matt Fredette, Alden Jackson, Dave Mankins, Marla Shepard and Pam Helinek for the contributions to this document. We thank Joel Levin and Mary Hendrix (INS Corp.) for reviewing this document. We thank Isidro Castineyra for his contributions to the early parts of this work.

References

[Bra97] Bradner, S., "Key Words for use in RFCs to indicate Requirement Levels", RFC2119, March 1997.

[Kent98] S. Kent, R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

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

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

[RFC1035] Mockapetris, P., "Domain Names - Implementation and Specification", RFC 1035, November 1987.

[RFC1305] Mills, D., "Network Time Protocol (Version 3): Specification, Implementation and Analysis", RFC 1305, March 1992.

[RGSC00] F. Reichmeyer , D. Grossman, J. Strassner, M. Condell, "A Common Terminology for Policy Management" Internet Draft draft-reichmeyer-polterm-terminology-00.txt, March 2000.

[PKIXP1] R. Housley, W. Ford, W. Polk, D. Solo, "Internet Public Key Infrastructure: X.509 Certificate and CRL Profile". RFC 2459, January 1999.

[Harkins98] D. Harkins, D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, November 1998.

[Piper98] D. Piper, "The Internet IP Security Domain of Interpretation for ISAKMP", RFC 2407, November 1998.

[SPS] M. Richardson, A. Keromytis, L. Sanchez, "IPsec Policy Discovery Protocol Requirements", Internet Draft, draft-richardson-ipsp-requirements-00.txt, October 1999.

[SPSL] M. Condell, C. Lynn, J. Zao "Security Policy Specification Language", Internet Draft draft-ietf-ipsp-spsl-00.txt, March 2000

APPENDIX A

DATA_TYPE Definitions

The encoding of each selector and SA attribute is decribed here.
Each attribute is described using the following set of data:

X	The value of the X bit in the attribute encoding.
DATA_TYPE	The value of the DATA_TYPE field in the attribute encoding.
LENGTH	The length of the data to use if X = 0.
list	'yes' indicates the attribute may be used as a list as described below.
DATA_VALUE	Encoding of the DATA_VALUE field in the attribute encoding.

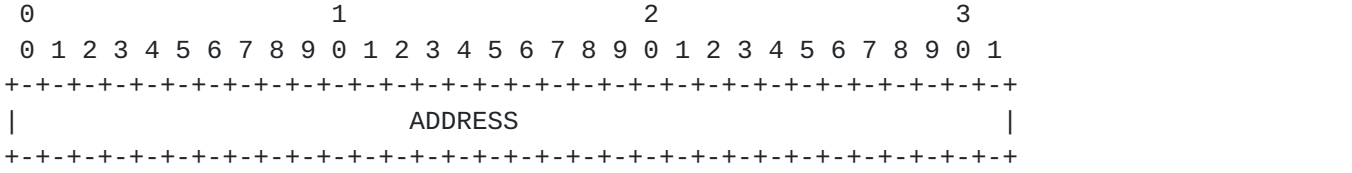
Attributes generally encode "any" in one of two ways. If using the TLV format (X = 0) then the length is set to 0 to indicate any. If the TV format (X = 1) is used, then the value is set to 0;

Attributes that may be expressed as lists provide the DATA_VALUE encoding for one element of the list. Multiple list elements may be expressed by concatenating multiple list elements. The LENGTH of attribute is the number of elements present times the length of one list element. Therefore, when reading an attribute that can be expressed as a list, the number of list elements may be determined by dividing the length by the size of a single list element.

The remainder of this appendix describes the values and encoding for each selector and SA attribute specified in section 7.

A.1 IPV4_ADDR

X	0
DATA_TYPE	1
LENGTH	4 if an IP address is present, 0 if no IP address is present.
list	No
DATA_VALUE	

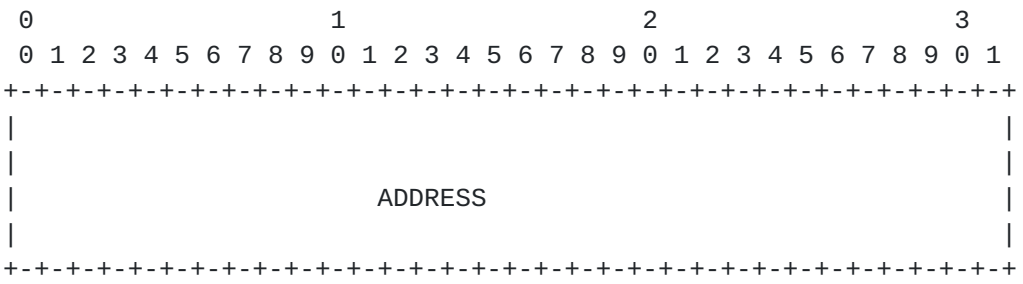


ADDRESS

An IPV4 address

A.2 IPV6_ADDR

X 0
DATA_TYPE 2
LENGTH 16 if an IP address is present,
0 if no IP address is present.
list No
DATA_VALUE

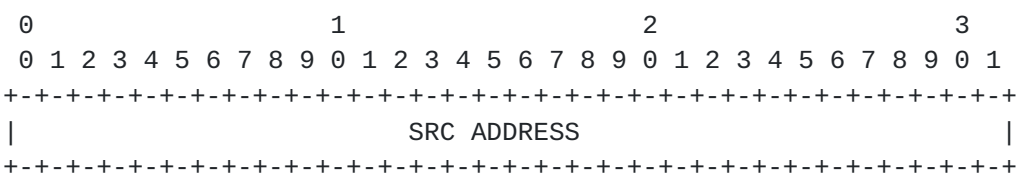


ADDRESS

An IPV6 address

A.3 SRC_IPV4_ADDR

X 0
DATA_TYPE 3
LENGTH 4 times the number of addresses in the list.
A length of 0 indicates any address.
list Yes
DATA_VALUE

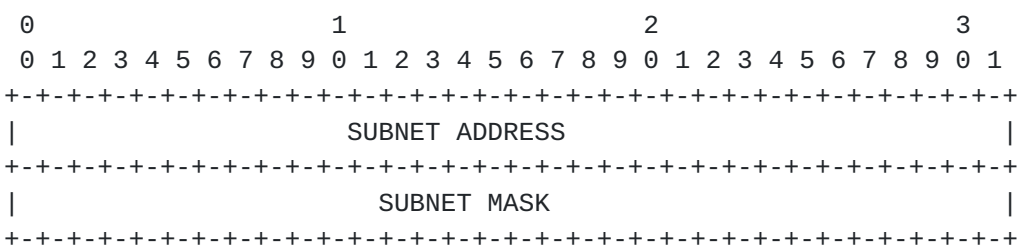


SRC ADDRESS

An IPV4 address representing the source host of a communication

A.4 SRC_IPV4_ADDR_SUBNET

X 0
DATA_TYPE 4
LENGTH 8 times the number of subnets in the list.
A length of 0 indicates any subnet.
list Yes
DATA_VALUE



SUBNET ADDRESS

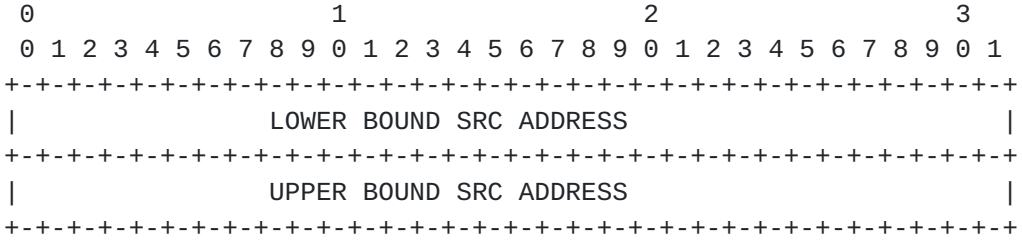
An IPV4 address representing the source subnet of a communication

SUBNET MASK

An IPV4 address representing the source subnet mask of a communication

A.5 SRC_IPV4_ADDR_RANGE

X	0
DATA_TYPE	5
LENGTH	8 times the number of address ranges in the list. A length of 0 indicates any address.
list	Yes
DATA_VALUE	



LOWER BOUND SRC ADDRESS

An IPV4 address representing the includsive lower-bound of a range of source addresses of a communication.

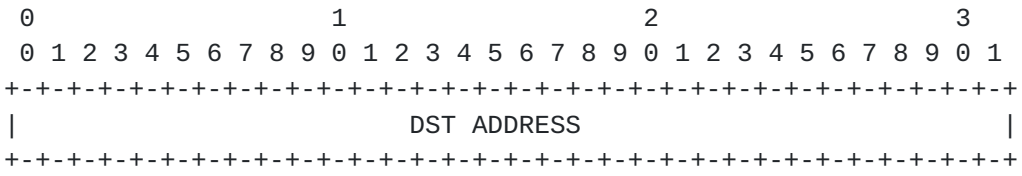
UPPER BOUND SRC ADDRESS

An IPV4 address representing the includsive upper-bound of a range of source addresses of a communication.

A.6 DST_IPV4_ADDR

X	0
DATA_TYPE	6

LENGTH	4 times the number of addresses in the list. A length of 0 indicates any address.	
list DATA_VALUE	Yes	

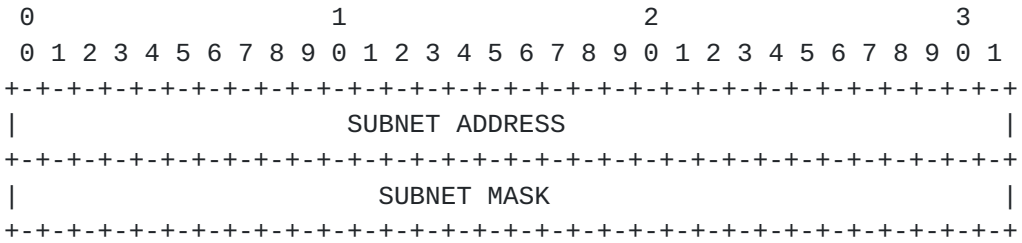


DST ADDRESS

An IPV4 address representing the destination host of a communication

A.7 DST_IPV4_ADDR_SUBNET

X	0
DATA_TYPE	7
LENGTH	8 times the number of subnets in the list. A length of 0 indicates any subnet.
list	Yes
DATA_VALUE	



SUBNET ADDRESS

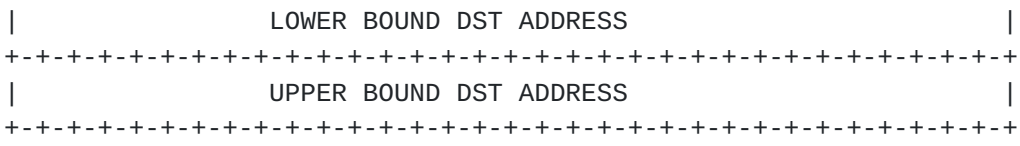
An IPV4 address representing the destination subnet of a communication

SUBNET MASK

An IPV4 address representing the destination subnet mask of a communication

A.8 DST_IPV4_ADDR_RANGE

X	0
DATA_TYPE	8
LENGTH	8 times the number of address ranges in the list. A length of 0 indicates any address.
list	Yes
DATA_VALUE	


LOWER BOUND DST ADDRESS
UPPER BOUND DST ADDRESS

LOWER BOUND DST ADDRESS

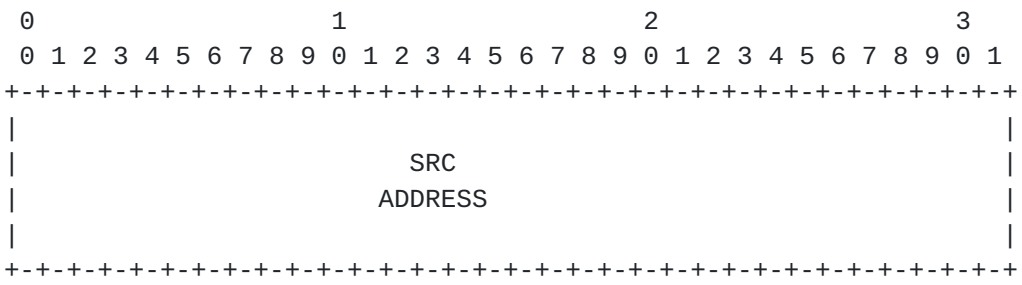
An IPV4 address representing the includsive lower-bound of a range of destination addresses of a communication.

UPPER BOUND DST ADDRESS

An IPV4 address representing the includsive upper-bound of a range of destination addresses of a communication.

A.9 SRC_IPV6_ADDR

X	0
DATA_TYPE	9
LENGTH	16 times the number of addresses in the list. A length of 0 indicates any address.
list	Yes
DATA_VALUE	

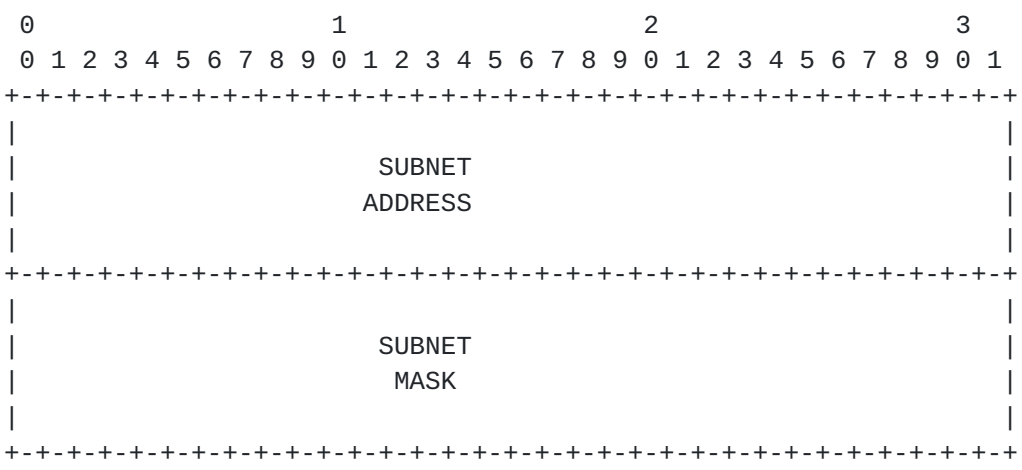


SRC ADDRESS

An IPV6 address representing the source host of a communication

A.10 SRC_IPV6_ADDR_SUBNET

X	0
DATA_TYPE	10
LENGTH	32 times the number of subnets in the list. A length of 0 indicates any subnet.
list	Yes
DATA_VALUE	



SUBNET ADDRESS

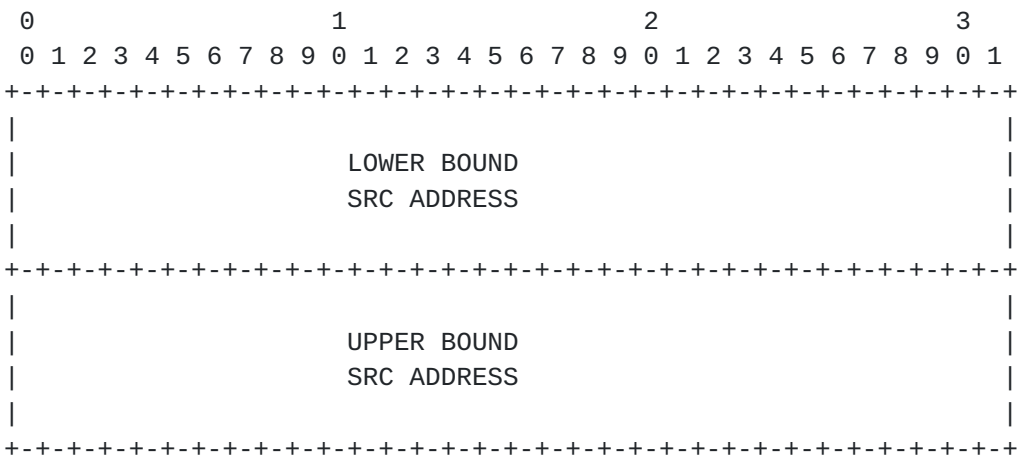
An IPV6 address representing the source subnet of a communication

SUBNET MASK

An IPV6 address representing the source subnet mask of a communication

A.11 SRC_IPV6_ADDR_RANGE

X	0
DATA_TYPE	11
LENGTH	32 times the number of address ranges in the list. A length of 0 indicates any address.
list	Yes
DATA_VALUE	



LOWER BOUND SRC ADDRESS

An IPV6 address representing the includsive lower-bound

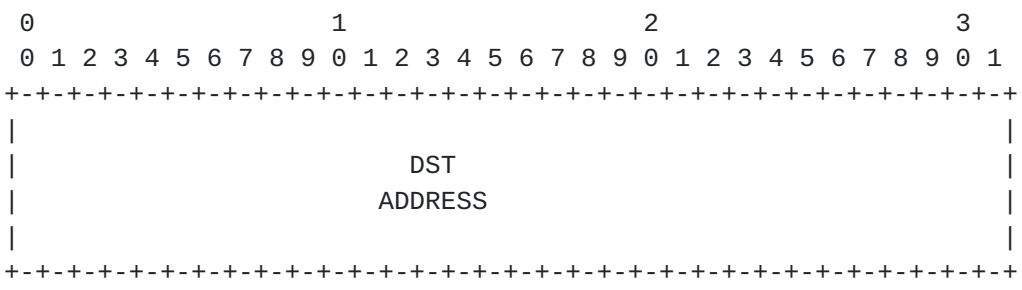
of a range of source addresses of a communication.

UPPER BOUND SRC ADDRESS

An IPV6 address representing the includsive upper-bound of a range of source addresses of a communication.

A.12 DST_IPV6_ADDR

X	0
DATA_TYPE	12
LENGTH	16 times the number of addresses in the list. A length of 0 indicates any address.
list	Yes
DATA_VALUE	

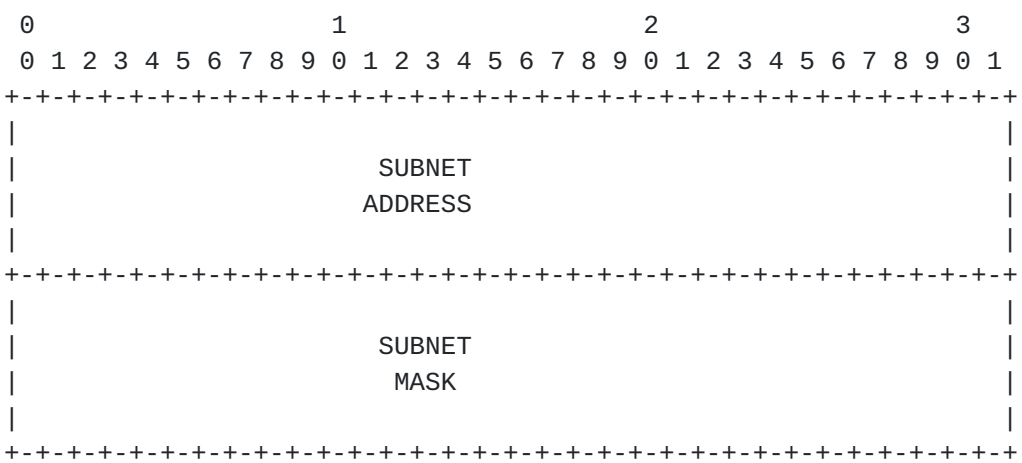


DST ADDRESS

An IPV6 address representing the destination host of a communication

A.13 DST_IPV6_ADDR_SUBNET

X	0
DATA_TYPE	13
LENGTH	32 times the number of subnets in the list. A length of 0 indicates any subnet.
list	Yes
DATA_VALUE	



SUBNET ADDRESS

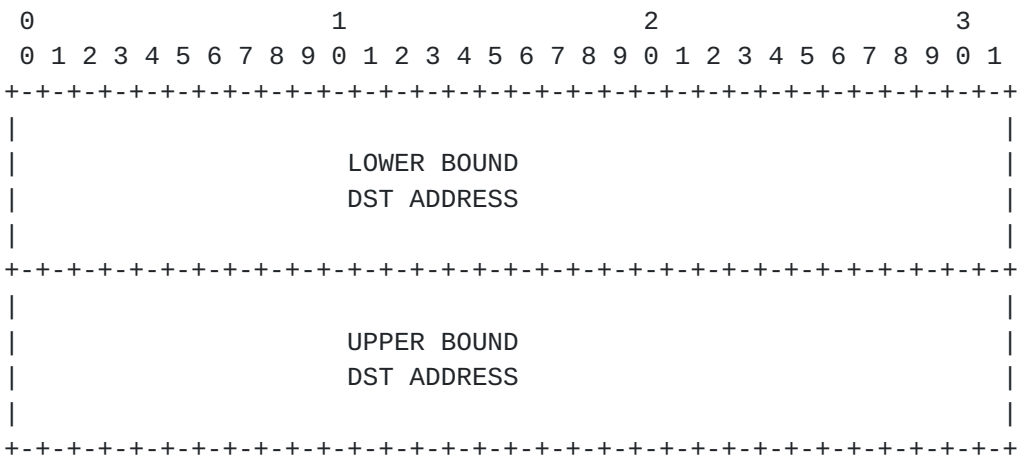
An IPV6 address representing the destination subnet of a communication

SUBNET MASK

An IPV6 address representing the destination subnet mask of a communication

A.14 DST_IPV6_ADDR_RANGE

X 0
DATA_TYPE 14
LENGTH 32 times the number of address ranges in the list. A length of 0 indicates any address.
list Yes
DATA_VALUE



LOWER BOUND DST ADDRESS

An IPV6 address representing the includsive lower-bound of a range of destination addresses of a communication.

UPPER BOUND DST ADDRESS

An IPV6 address representing the includsive upper-bound of a range of destination addresses of a communication.

A.15 DIRECTION

X 1
DATA_TYPE 15
LENGTH TV attribute, no length
list No
DATA_VALUE

```
 1       2                   3
 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          DIRECTION            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

DIRECTION

In/Outbound	0
Inbound	1
Outbound	2

Direction is with respect to the senders interface.

A.16 USER_NAME

X	0
DATA_TYPE	16
LENGTH	1 plus the length of NAME A length of 0 indicates any name.
list	No

DATA_VALUE

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   NAME_TYPE   |               NAME                            ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

NAME_TYPE

822_EMAIL	0
DIST_NAME	1

values 2-250 are reserved to IANA. Values 251-255 are for private use among mutually consenting parties.

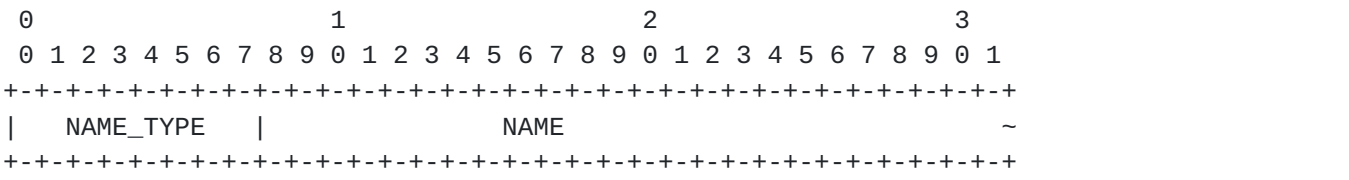
NAME

Name of type NAME_TYPE:

NAME_TYPE	Description of NAME
822_EMAIL	A fully-qualified user name string (e.g. "jdoe@example.com") as defined in RFC 822. The string must not contain any terminators
DIST_NAME	A fully-qualified distinguished name string (e.g. "CN=John Doe, O=Example, Inc, C=US ") as defined in RFC 1779. The string must not contain any terminators

A.17 SYSTEM_NAME

X 0
DATA_TYPE 17
LENGTH 1 plus the length of NAME
A length of 0 indicates any name.
list No
DATA_VALUE



NAME_TYPE

DNS_NAME	0
DIST_NAME	1
822_NAME	2
X400_ADDR	3
DIR_NAME	4
EDI_PARTY_NAME	5
URI	6
IPADDR	7
REGID	8
OTHER	250

values 9-249 are reserved to IANA. Values 251-255 are for private use among mutually consenting parties.

NAME

Name of type NAME_TYPE. Strings must not contain any terminators.

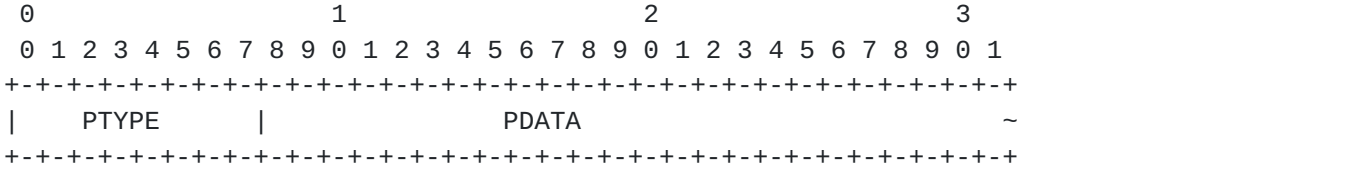
NAME_TYPE	Description of NAME
DNS_NAME	A DNS name string (e.g. "host.example.com") as defined in RFC 1034.
DIST_NAME	A fully-qualified distinguished name string (e.g. "CN=John Doe, O=Example Inc, C=US ") as defined in RFC 1779.
822_EMAIL	A fully-qualified user name string (e.g. "jdoe@example.com") as defined in RFC 822.
X400_ADDR	A textual representation of an X.400 OR address string (e.g. "/CN=John Doe/O=Example Inc/C=US/")

as defined in RFC 2156.

DIR_NAME	A relative distinguished name string (e.g. "OU=Engineering + CN=John Doe, O=Example Inc, C=US ") as defined in RFC 1779.
EDI_PARTY_NAME	An electronic data interchange name string.
URI	A uniform resource identifier string (e.g. "ftp://ftp.example.com/pub/doc.html") as defined in RFC 2396.
IPADDR	A 32-bit or 128-bit IP address. Note that this is NOT the string representation of the IP address.
REGID	A registered ID is represented by a string representation of the dotted integer representation of an object ID (OID) (e.g. "4.5.8.2.1").
OTHER	This is an object identifier followed by object specific information. The OID is represented as above, however, its end is indicated by a colon ":" which is followed by the object specified information. (e.g. "4.5.8.2.1:" 98 "jdoe")

A.18 XPORT_PROTOCOL

X	0
DATA_TYPE	18
LENGTH	1 plus length of pdata A length of 0 indicates any address.
list	No (see below)
DATA_VALUE	



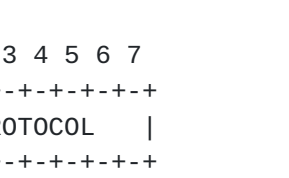
PTYPE Describes the rest of the data:

ANY	0
OPAQUE	1
LIST	2
RANGE	3

PDATA

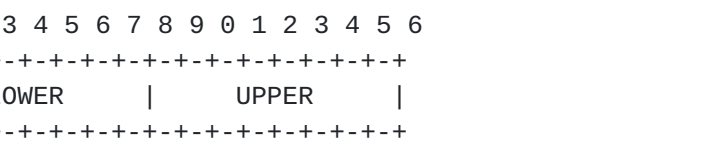
Not used if PTYPE is ANY or OPAQUE.

LIST
    indicates a list whose elements look like the following:



The length of pdata to be used as part of the LENGTH field is 1 times the number of elements in the list.

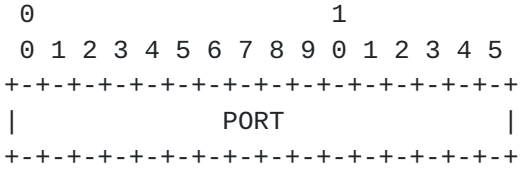
RANGE
indicates a range of protocol values whose inclusive lower-bound is LOWER, and inclusive upper-bound is UPPER.



The length of pdata to be used as part of the LENGTH field is 2.

A.19 SRC_PORT

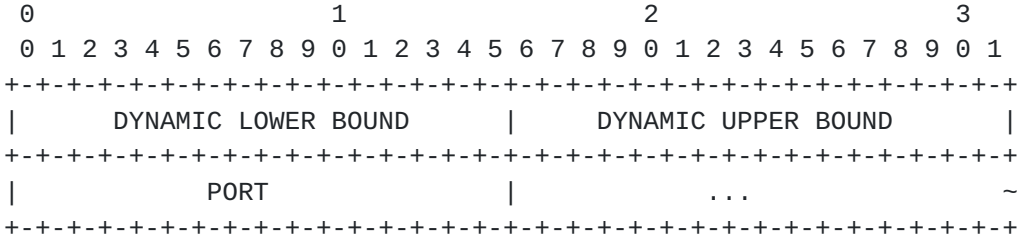
X	0
DATA_TYPE	19
LENGTH	2 times the number of ports in the list. A length of 0 indicates any port.
list	Yes
DATA_VALUE	



PORT
Port that the communication must be initiated with. This may be a list of ports.

A.20 SRC_PORT_DYNAMIC

X	0
DATA_TYPE	20
LENGTH	4 plus 2 times the number of ports in the list. A length of 4 indicates any port.
list	See Below
DATA_VALUE	



The use of this attribute indicates that dynamic port allocation is permitted. Communications that are intitiated with any of the ports indicated, may then dynamically allocate any of the ports listed within the LOWER and UPPER BOUNDS, inclusive.

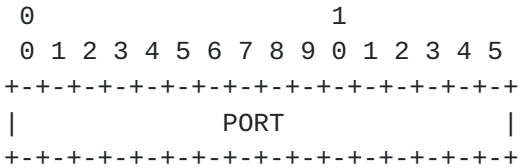
DYNAMIC LOWER BOUND
Lower bound of the range of ports that may be dynamically allocated. If this and DYNAMIC UPPER BOUND are both 0, then any port may be dynamically allocated.

DYNAMIC UPPER BOUND
Upper bound of the range of ports that may be dynamically allocated. If this and DYNAMIC LOWER BOUND are both 0, then any port may be dynamically allocated.

PORT
Port that the communication must be initiated with. This may be a list of ports.

A.21 DST_PORT

X	0
DATA_TYPE	21
LENGTH	2 times the number of ports in the list. A length of 0 indicates any port.
list	Yes
DATA_VALUE	

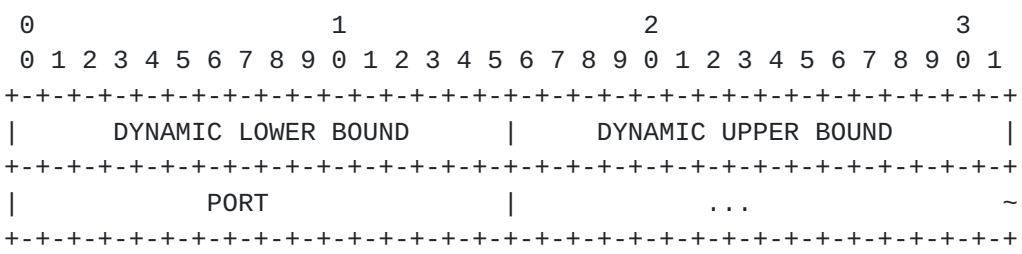


PORT
Port that the communication must be initiated with. This may be a list of ports.

A.22 DST_PORT_DYNAMIC

X	0
DATA_TYPE	22

LENGTH	4 plus 2 times the number of ports in the list. A length of 4 indicates any port.
list	See Below
DATA_VALUE	



The use of this attribute indicates that dynamic port allocation is permitted. Communications that are intitiated with any of the ports indicated, may then dynamically allocate any of the ports listed within the LOWER and UPPER BOUNDS, inclusive.

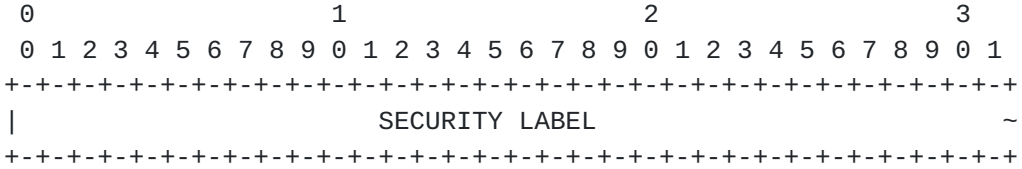
DYNAMIC LOWER BOUND
Lower bound of the range of ports that may be dynamically allocated. If this and DYNAMIC UPPER BOUND are both 0, then any port may be dynamically allocated.

DYNAMIC UPPER BOUND
Upper bound of the range of ports that may be dynamically allocated. If this and DYNAMIC LOWER BOUND are both 0, then any port may be dynamically allocated.

PORT
Port that the communication must be initiated with. This may be a list of ports.

A.23 SEC_LABELS

X	0
DATA_TYPE	23
LENGTH	Variable.
list	No
DATA_VALUE	



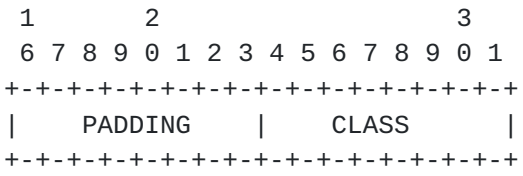
SECURITY LABEL

Bit representation of the security label for the IP security option field.

A.24 V6CLASS

X	1
DATA_TYPE	24

LENGTH	TV attribute, no length
list	No
DATA_VALUE	



PADDING

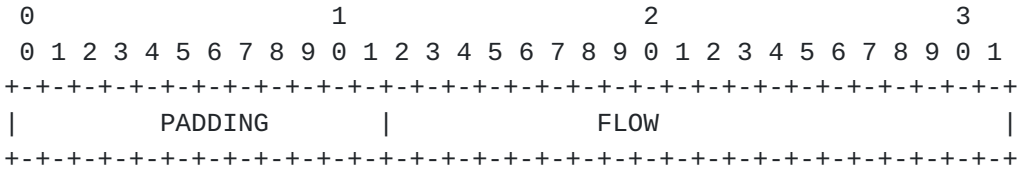
set to 0

CLASS

class value

A.25 V6FLOW

X 0
DATA_TYPE 25
LENGTH 4
list No
DATA_VALUE



PADDING

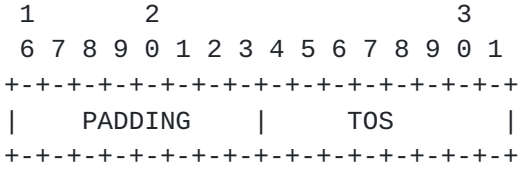
set to 0

FLOW

set to flow value

A.26 V4TOS

X 1
DATA_TYPE 26
LENGTH TV attribute, no length
list No
DATA_VALUE



PADDING

set to 0

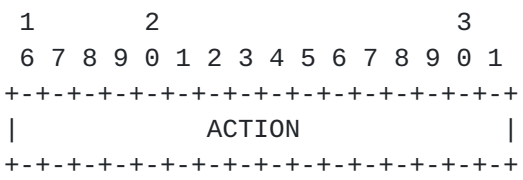
TOS

type of service value

A.27 ACTION

X 1
DATA_TYPE 27
LENGTH TV attribute, no length
list No

DATA_VALUE

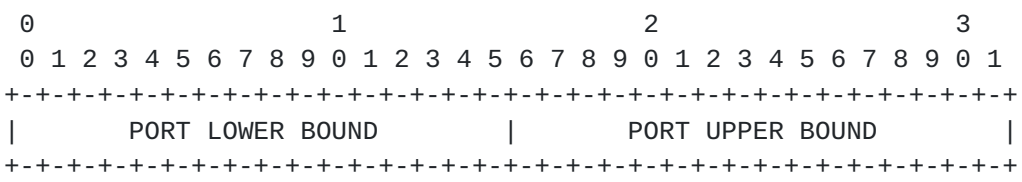


ACTION

Deny 0
Permit 1

A.28 SRC_PORT_RANGE

X 0
DATA_TYPE 28
LENGTH 4 times the number of port ranges in the list.
A length of 0 indicates any port.
list Yes
DATA_VALUE



PORT LOWER BOUND

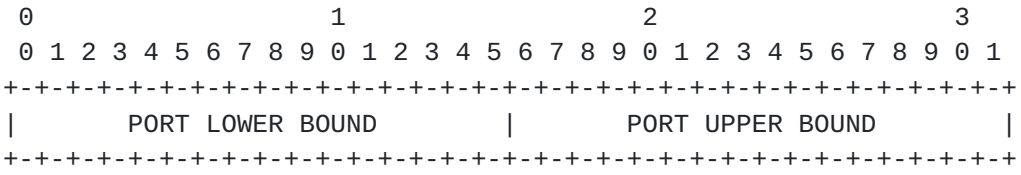
Inclusive lower-bound of a range of port numbers that the communication must be initiated with.

PORT UPPER BOUND

Inclusive upper-bound of a range of port numbers that the communication must be initiated with.

A.29 DST_PORT_RANGE

X	0
DATA_TYPE	29
LENGTH	4 times the number of port ranges in the list. A length of 0 indicates any port.
list	Yes
DATA_VALUE	



PORT LOWER BOUND

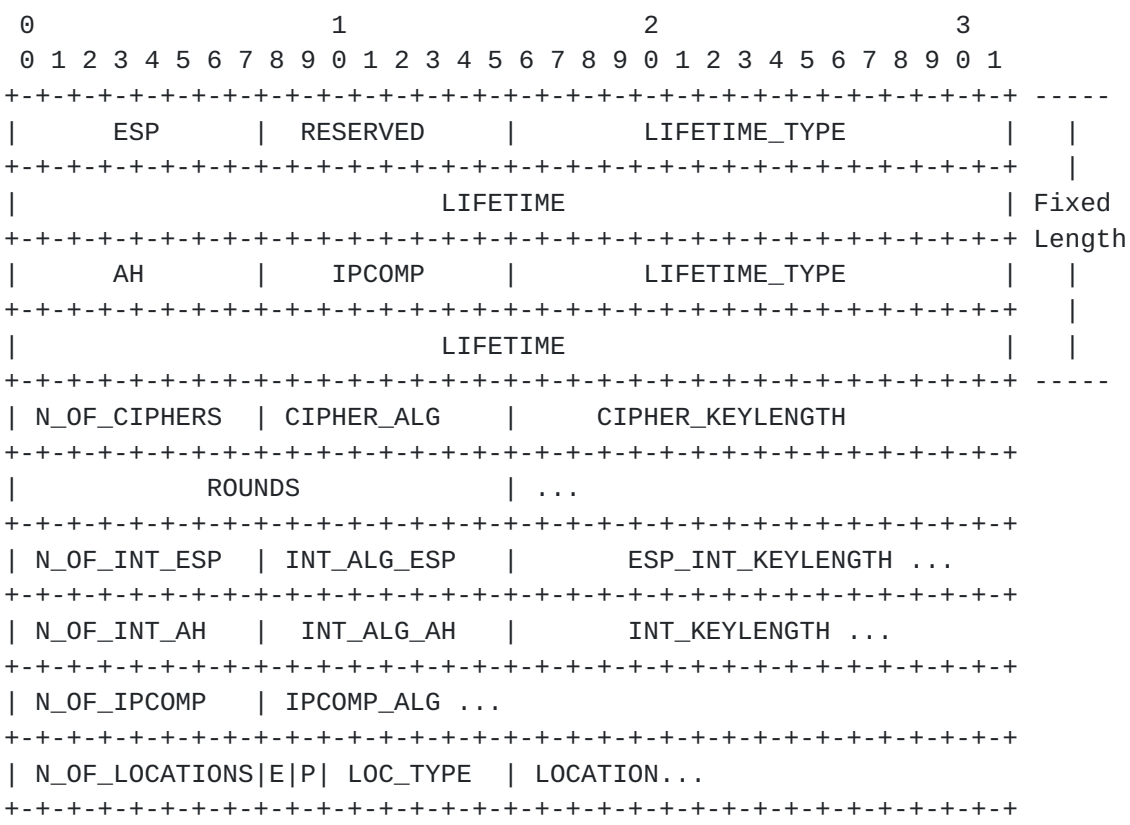
Inclusive lower-bound of a range of port numbers that the communication must be initiated with.

PORT UPPER BOUND

Inclusive upper-bound of a range of port numbers that the communication must be initiated with.

A.30 IPSEC_ACTION

X	0
DATA_TYPE	50
LENGTH	Variable
list	Yes
DATA_VALUE	



ESP

This octet indicates if ESP is to be used and in what mode. NOT REQUIRED means that ESP is not necessary but if used it MUST be negotiated based on the parameters defined below. TUNNEL_MODE or TRANSP_MODE means that ESP MUST be negotiatiated in this mode. ANY_MODE means that ESP MUST be negotited and that any mode (Tunnel or transport) will suffice. NOT ALLOWED means that ESP SHOULD NOT be negotiated and it MUST NOT be part of this SA.

NOT_REQUIRED	0
TUNNEL_MODE	1
TRANSP_MODE	2
TUNNEL_MODE_OPT	3
TRANSP_MODE_OPT	4
ANY_MODE	5
NOT ALLOWED	6

LIFETIME_TYPE

This 2 octet field indicates type of lifetime.

RESERVED	0
SECONDS	1

These values are assigned in section 4.5 of [Piper98] and are updated when those assigned values change.

RESERVED

This 1 octet field primarily used for alignment purposes. Its value is always 0.

LIFETIME

This 4 octet field indicates the SA lifetime. For a given "Lifetime_Type" the value of the "Lifetime" attribute defines the actual length of the SA life--either a number of seconds, or a number of kilobytes protected. 0 is not used.

AH

This octet indicates if AH is to be used and in what mode. NOT REQUIRED means that AH is not necessary but if used it MUST be negotiated based on the parameters defined below. TUNNEL_MODE or TRANSP_MODE means that AH MUST be negotiatiated in this mode. ANY_MODE means that AH MUST be negotited and that any mode (Tunnel or transport) will suffice. NOT ALLOWED means that AH SHOULD NOT be negotiated and it MUST not be part of this SA.

NOT_REQUIRED	0
TUNNEL_MODE	1
TRANSP_MODE	2
TUNNEL_MODE_OPT	3
TRANSP_MODE_OPT	4
ANY_MODE	5
NOT ALLOWED	6

IPCOMP

This field indicates if IP Compression is to be used. NOT REQUIRED means that IPCOMP is not necessary but if used it MUST be negotiated based on the parameters defined below. REQUIRED means that IPCOMP MUST be negotiated as part of this SA. NOT ALLOWED means that IPCOMP MUST NOT be part of this SA.

NOT_REQUIRED	0
REQUIRED	1
NOT ALLOWED	2

N_OF_CIPHERS

This octet indicates the number of CIPHER_ALG fields in octets that will follow this field and that could be used during an IKE phase 2 negotiation. If the value of the ESP field is (04)hex this field MUST be set to 0.

CIPHER_ALG

This octet indicates which ciphers should be used for the IKE phase 2 negotiation. If the ANY identifier is used, it MUST be the only identifier in the list, and its meaning does not include the NULL cipher. If the value of the N_OF_CIPHERS field is 0 the CIPHER_ALG, the CIPHER_KEYLENTH and the ROUNDS fields are ignored.

ANY	0
NULL	1
RFC1829_IV64	2
DES	3
DES3	4
RC5	5
IDEA	6
CAST	7
BLOWFISH	8
3IDEA	9
RFC1829_IV32	10
RC4	11

These values are assigned in section 4.4.4 of [Piper98], with the exception of 0 being defined as ANY, and are updated when those assigned values change.

CIPHER_KEYLENGTH

The first octet corresponds to the minimum value and the second octet corresponds to the maximum value. If no range exist the first octet indicates the keylength. The second octet contains a value of (00)hex.

ROUNDS

The first octet corresponds to the minimum value and the second octet corresponds to the maximum value. If no range exist the first octet indicates the rounds. The second octet contains a value of (00)hex.

N_OF_INT_ESP

This octet indicates the number of INTEGRITY_ALG fields in octets that will follow this field and that could be used during an IKE phase 1 negotiation. If this field is 0 no authentication/integrity is used with ESP.

INT_ALG_ESP

This octet indicates which algorithm should be used for the

IKE phase 2 negotiation. If the ANY identifier is used, it MUST be the only identifier in the list. If the value of the N_OF_INT_ESP field is 0 this INT_ALG_ESP and ESP_INT_KEYLENGTH are ignored.

ANY	0
HMAC_MD5	1
HMAC_SHA1	2
DES_MAC	3
KPDK	4

These values are assigned in section 4.5 of [Piper98], with the exception of 0 being defined as ANY, and are updated when those assigned values change.

ESP_INT_KEYLENGTH

The first octet corresponds to the minimum value and the second octet corresponds to the maximum value. If no range exist the first octet indicates the keylength. The second octet contains a value of (00)hex.

N_OF_INT_AH

This octet indicates the number of INTEGRITY_ALG fields in octets that will follow this field and that could be used during an IKE phase 1 negotiation. If the value of the AH field is (04)hex this field MUST be set to 0.

INT_ALG_AH

This octet indicates which algorithm should be used for the IKE phase 2 negotiation. If the value of the N_OF_INT_AH field is 0 the INT_ALG_AH and the INT_KEYLENGTH fields are ignored.

ANY	0
HMAC_MD5	1
HMAC_SHA1	2
DES_MAC	3
KPDK	4

These values are assigned in section 4.5 of [Piper98], with the exception of 0 being defined as ANY, and are updated when those assigned values change.

INT_ KEYLENGTH

The first octet corresponds to the minimum value and the second octet corresponds to the maximum value. If no range exist the first octet indicates the keylength. The second octet contains a value of (00)hex.

N_OF_IPCOMP

This octet indicates the number of IPCOMP_ALG fields in octets that will follow this field and that could be used during an IKE phase 2 negotiation. If the value of the IPCOMP field is (04)hex this field MUST be set to 0.

IPCOMP_ALG

This octet indicates which algorithm should be used for the IKE phase 2 negotiation. If the ANY identifier is used, it MUST be the only identifier in the list. If the value of the N_OF_IPCOMP field is 0 this field is ignored.

ANY	0
OUI	1
DEFLATE	2
LZS	3

These values are assigned in section 4.4.5 of [Piper98], with the exception of 0 being defined as ANY, and are updated when those assigned values change.

N_OF_LOCATIONS

This octet indicates the number of E/P/LOC_TYPE fields that will follow this field.

E

This bit indicates whether this location represents a source or destination location. E = 0 indicates a source location and E = 1 indications a destination location.

P

This bit indicates whether the location is for the AH or ESP SA. P = 0 indicates the location is for the AH SA while P = 1 indicates the location is for the ESP SA.

LOC_TYPE

This 6-bit field indicates the contents of the LOCATION field. If this field is 0 then the LOCATION will be omitted.

NONE	0
IPv4 address	1
IPv6 address	2
DNS Name	3
General	4

values 5-250 are reserved to IANA. Values 251-255 are for private use among mutually consenting parties.

LOCATION

Variable length field depending on LOC_TYPE.

IF LOC_TYPE is (04) then this field is 1 octet in length an it may only take the following values:

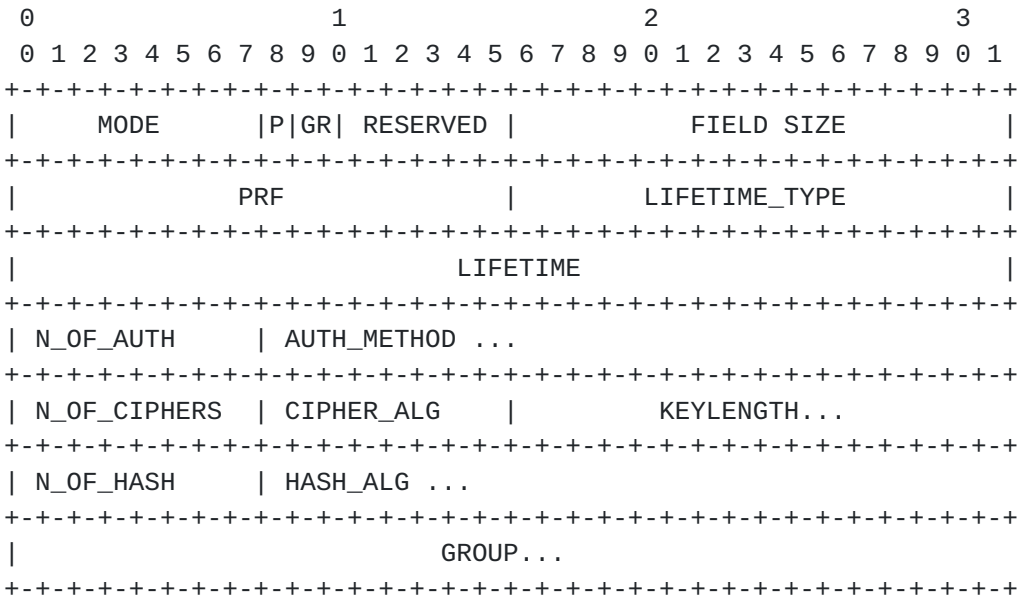
ANY	0
DEST	1
HOST	2
LOCAL-SG	3
REMOTE-SG	4

values 5-250 are reserved to IANA. Values 251-255 are for private use among mutually consenting parties.

If multiple locations are indicated that specify the same end-point for the same SA (i.e. E and P bits are the same), it indicates that they are possible alternatives for the end-point.

A.31 IKE_ACTION

X	0
DATA_TYPE	51
LENGTH	Variable
list	No
DATA_VALUE	



MODE

This octet indicates the IKE mode of operation.

MAIN	0
AGRESSIVE	1
QUICK	2

values 3-250 are reserved to IANA. Values 251-255 are for private use among mutually consenting parties.

P

Indicates if PFS is to be used for the SA negotiation.

FALSE	0
TRUE	1

GR

Indicates if a group description or group type fields are included in this IKE action.

NO GROUP	0
GROUP_DESCRIPTION	1
GROUP_TYPE	2

See the GROUP field below for more information.

RESERVED

Reserved for future use. Set to zero.

FIELD SIZE

The field size, in bits, of a Diffie-Hellman Group

PRF

There are currently no pseudo-random functions defined.

These values are assigned in Appendix A of [Harkins98] and are updated when those assigned values change.

LIFETIME_TYPE

This 2 octet field indicates type of lifetime.

seconds	1
kilobytes	2

These values are assigned in Appendix A of [Harkins98]

and are updated when those assigned values change.

LIFETIME

This 4 octet field indicates the SA lifetime. For a given "Lifetime_Type" the value of the "Lifetime" attribute defines the actual length of the SA life-- either a number of seconds, or a number of kilobytes protected.

N_OF_AUTH

This octet indicates the number of AUTH_METHOD fields in octets that will follow this field and that could be used during an IKE phase 1 negotiation.

AUTH_METHOD

This octet indicates which authentication methods should be used. The number of auth_methods that could be used is N_OF_AUTH

pre-shared key	1
DSS signatures	2
RSA signatures	3
Encryption with RSA	4
Revised encryption with RSA	5

These values are assigned in Appendix A of [Harkins98] and are updated when those assigned values change.

N_OF_CIPHERS

This octet indicates the number of CIPHER_ALG fields in octets that will follow this field and that could be used during an IKE phase 1 negotiation.

KEYLENGTH

The first octet corresponds to the minimum value and the second octet corresponds to the maximum value. If no range exist the first octet indicates the keylength. The second octet contains a value of (00)hex.

CIPHER_ALG

This octet indicates which ciphers should be used for the IKE phase 1 negotiation. For IKE phase 2 negotiations this field is ignored. The number of ciphers that could be used is N_OF_CIPHERS

ANY	0
DES	1
IDEA	2

```
        BLOWFISH 3
        RC5      4
        DES3     5
        CAST     6
```

These values are assigned in Appendix A of [Harkins98], with the exception of 0 being defined as ANY, and are updated when those assigned values change.

N_OF_HASH

This octet indicates the number of HASH_ALG fields in octets that will follow this field and that could be used during an IKE phase 1 negotiation.

HASH_ALG

This octet indicates which algorithm should be used for the IKE phase 1 negotiation. For IKE phase 2 negotiations this field is ignored.

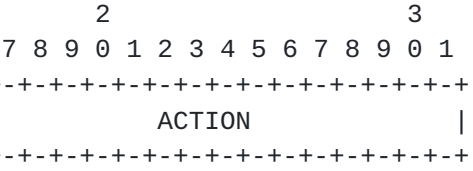
ANY	0
MD5	1
SHA1	2
TIGER	3

These values are assigned in Appendix A of [Harkins98], with the exception of 0 being defined as ANY, and are updated when those assigned values change.

GROUP

This field describes the group to be used during ISAKMP negotiation. It is only present if GR is 1 or 2. If GR is 1 it takes the form of the GROUP_DESCRIPTION field below. If it is 2, it takes the form of the GROUP_DEFINITION field below.

GROUP DESCRIPTION

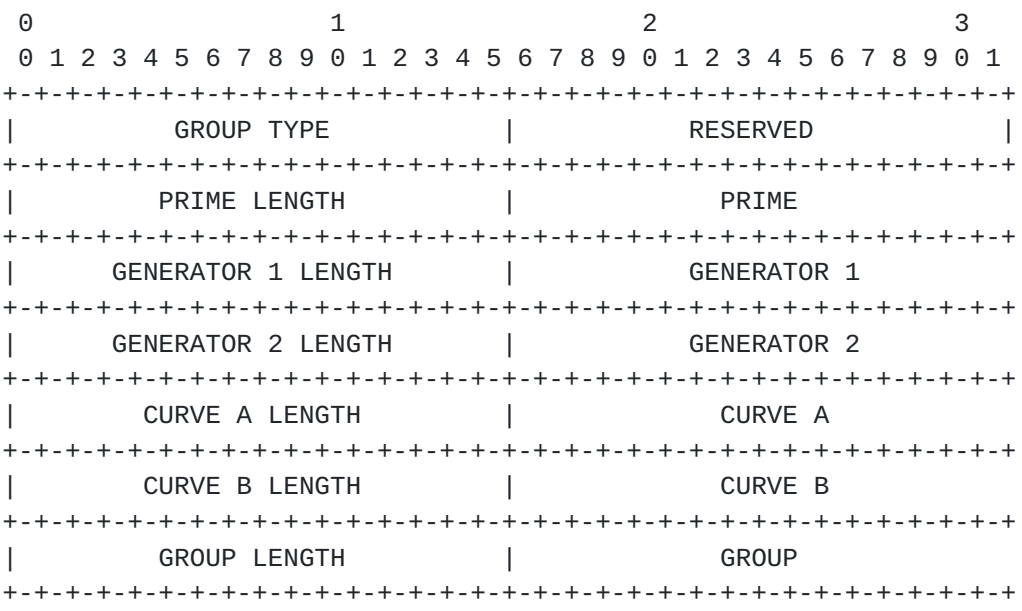


This 2 octet field indicates which group should be used during the ISAKMP phase 1 or phase 2 negotiation.

Not Used	0
default 768-bit MODP group	1
alternate 1024-bit MODP group	2
EC2N group on GP[2^155]	3
EC2N group on GP[2^185]	4

These values are assigned in Appendix A of [Harkins98] and are updated when those assigned values change.

GROUP DEFINITION



GROUP TYPE

This 2 octet field indicates which group type should be used during the ISAKMP phase 1 or phase 2 negotiation.

Not Used	0
MODP	1
ECP	2
EC2N	3

These values are assigned in Appendix A of [Harkins98] and are updated when those assigned values change.

RESERVED

Reserved for future use. Set to zero.

PRIME LENGTH
GENERATOR 1 LENGTH
GENERATOR 2 LENGTH
CURVE A LENGTH
CURVE B LENGTH
GROUP LENGTH

Length of their respective fields in bytes. If their respective field does not exist, the length is set to zero.

PRIME
GENERATOR 1
GENERATOR 2

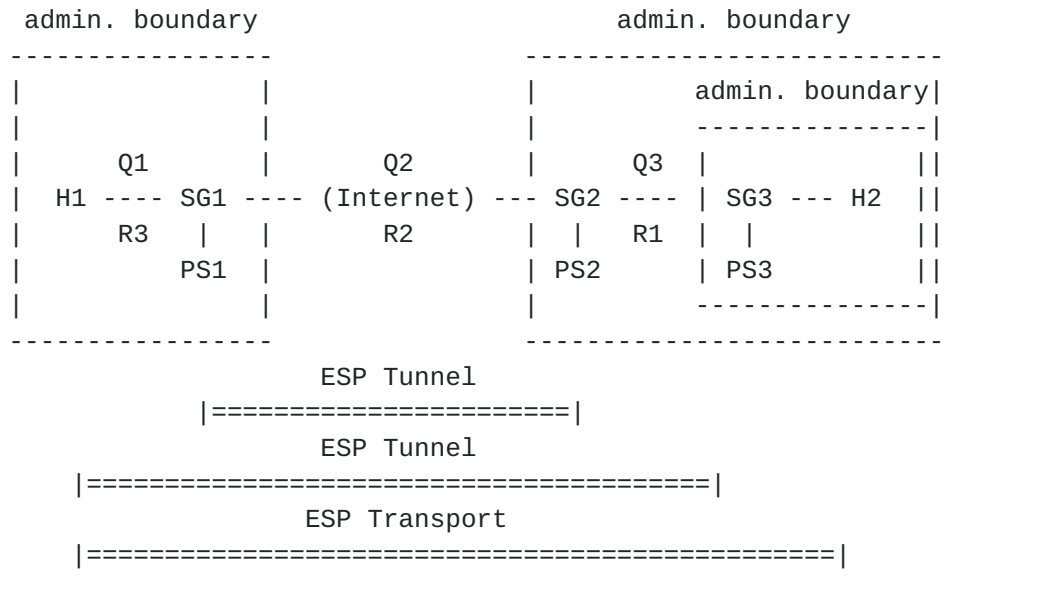
CURVE A
CURVE B
GROUP

The group prime/irreducible polynomial, group generator one, group generator two, group curve A, group curve B, and group order, respectively. These are defined in [Harkins98].

APPENDIX B

An SPP Example

This appendix provides a detailed example of SPP in use.



|==| = security association required by policy
---- = connectivity (or if so labeled, administrative boundary)
Hx = host x
SGx = security gateway x
PSx = policy server x
Qx = query x
Rx = reply x

The following entities have these policies for a communication between H1 and H2 for UDP port 79:

H1: requires an ESP Transport SA with H2
PS1: requires an ESP Tunnel SA between SG1 and SG2
PS2: requires an ESP Tunnel SA between SG1 and SG2
PS3: requires an ESP Tunnel SA between H1 and SG3
H2: requires an ESP Transport SA with H1

PS1, PS2, PS3 also have policies allowing ESP to pass through their respective Security Gateways.

1. The policy client at H1 is asked for a policy for a communication: H1 to H2 using UDP port 79.

2. **H1's policy client does not have an answer so it creates an SPP query, Q1:**

```
          SPP Header [Query, Sender H1, qcount 1, rcount 2]
          Query Payload [comsec]:
            src H1, dst H2, UDP, 79
          Record Payload [comsec]:
            src H1, dst H2, UDP, 79, permit
          Record Payload [SA rec]:
            src H1, dst H2, UDP, 79, permit, ESP transport H1->H2
          Signature Payload
```

H1 sends Q1 to PS1, its configured policy server.

3. **PS1 receives the query and verifies the signature. Its domain database indicates that it is not authoritative over H2 so it** checks its cache to see if it has a cached answer. For this example, it does not, so it creates a new SPP query, Q2, with the query and records formed by merging the local policy with the policy from Q1:

```
          SPP Header [Query, Sender PS1, qcount 1, rcount 3]
          Query Payload [comsec]:
            src H1, dst H2, UDP, 79
          Record Payload [comsec]:
            src H1, dst H2, UDP, 79, permit
          Record Payload [SA rec]:
            src SG1, dst SG2, UDP, 79, permit, ESP tunnel SG1->SG2
          Record Payload [SA rec]:
            src H1, dst H2, UDP, 79, permit, ESP transport H1->H2
          Signature Payload
```

PS1 sends Q2 to H2.

4. **SG2 intercepts Q2 and passes it to PS2.**

5. **PS2 receives the query and verifies the signature. Its domain database indicates that it is not authoritative over H2 so it** checks its cache to see if it has a cached answer. For this example, it does not, so it creates a new SPP query, Q3, with the query and records formed by merging the local policy with the policy from Q2:

```
          SPP Header [Query, Sender PS1, qcount 1, rcount 3]
          Query Payload [comsec]:
            src H1, dst H2, UDP, 79
          Record Payload [comsec]:
            src H1, dst H2, UDP, 79, permit
          Record Payload [SA rec]:
            src SG1, dst SG2, UDP, 79, permit, ESP tunnel SG1->SG2
          Record Payload [SA rec]:
            src H1, dst H2, UDP, 79, permit, ESP transport H1->H2
          Signature Payload
```

PS2 sends Q3 to H2.

6. SG3 intercepts Q3 and passes it to PS3.

7. **PS3 receives the query and verifies the signature. Its domain database indicates that it is authoritative over H2 so it** will send a reply. It checks its cache to see if it has a cached answer. For this example, it does have one cached from previous information sent to it by H2. PS3 merges the cached policy with the policy it received from Q3. The merge indicates that a signal and a reply will be needed. PS3 caches the merged policy.

   PS3 creates a reply with the query payload from Q3, the merged policy and policy server and cert records:

```
         SPP Header [Reply, Sender PS3, qcount 1, rcount 6]
         Query Payload [comsec]:
           src H1, dst H2, UDP, 79
         Record Payload [comsec]:
           src H1, dst H2, UDP, 79, permit
         Record Payload [SA rec]:
           src SG1, dst SG2, UDP, 79, permit, ESP tunnel SG1->SG2
         Record Payload [SA rec]:
           src H1, dst SG3, UDP, 79, permit, ESP tunnel H1->SG3
         Record Payload [SA rec]:
           src H1, dst H2, UDP, 79, permit, ESP transport H1->H2
         Record Payload [policy server]:
           policy server PS3, node H2
         Record Payload [cert]:
           cert for PS3
         Signature Payload
```

   PS3 sends R1 to PS2.

   PS3 creates a signal with a comsec record derived from knowing the traffic that will pass through SG3 and, the part of the merged policy that terminates at SG3:

```
         SPP Header [Pol, Sender PS3, qcount 0, rcount 2]
         Record Payload [comsec]:
           src H1, dst H2, ESP, OPAQUE, permit
         Record Payload [SA rec]:
           src H1, dst SG3, UDP, 79, permit, ESP tunnel H1->SG3
         Signature Payload
```

   PS3 sends the signal to SG3.

8. **SG3 receives the signal and verifies the signature.** SG3 creates an Ack message to indicate that it has received the policy message:

```
         SPP Header [Pol-Ack, Sender SG3, qcount 0, rcount 0]
         Signature Payload
```

   SG3 sends the signal to PS3.

9. **PS3 receives the Pol-Ack and verifies the signature.** PS3 removes the corresponding policy message from its retry queue.

10. **Meanwhile, PS2 receives the reply R1 and verifies the signature and the chain-of-trust to verify the policy came from a server** authoritative for H2. It matches an outstanding query message, so it will send a reply. PS2 merges the policy received in R1 with its local policy and the policy information it received from Q2. The merge indicates that a signal and a reply will be needed. PS2 caches the merged policy.

PS2 creates a reply with the query payload from R1, the merged policy and policy server and cert records:

```
          SPP Header [Reply, Sender PS2, qcount 1, rcount 8]
          Query Payload [comsec]:
            src H1, dst H2, UDP, 79
          Record Payload [comsec]:
            src H1, dst H2, UDP, 79, permit
          Record Payload [SA rec]:
            src SG1, dst SG2, UDP, 79, permit, ESP tunnel SG1->SG2
          Record Payload [SA rec]:
            src H1, dst SG3, UDP, 79, permit, ESP tunnel H1->SG3
          Record Payload [SA rec]:
            src H1, dst H2, UDP, 79, permit, ESP transport H1->H2
          Record Payload [policy server]:
            policy server PS3, node H2
          Record Payload [cert]:
            cert for PS3
          Record Payload [policy server]:
            policy server PS2, node PS3
          Record Payload [cert]:
            cert for PS2
          Signature Payload
```

PS2 sends R2 to PS1.

PS2 creates a signal with a comsec record derived from knowing the traffic that will pass through SG2 and, the part of the merged policy that terminates at SG2:

```
          SPP Header [Pol, Sender PS2, qcount 0, rcount 2]
          Record Payload [comsec]:
            src H1, dst SG3, ESP, OPAQUE, permit
          Record Payload [SA rec]:
            src SG1, dst SG2, UDP, 79, permit, ESP tunnel SG1->SG2
          Signature Payload
```

PS2 sends the signal to SG2.

11. **SG2 receives the signal and verifies the signature.** SG2 creates an Ack message to indicate that it has received the policy message:

```
          SPP Header [Pol-Ack, Sender SG2, qcount 0, rcount 0]
          Signature Payload
```

SG2 sends the signal to PS2.

12. **PS2 receives the Pol-Ack and verifies the signature.** PS2 removes the corresponding policy message from its retry queue.

11. **Meanwhile, PS1 receives the reply R2 and verifies the signature and the chain-of-trust to verify the policy came from a server** authoritative for H2. R2 matches an outstanding query message, so it will send a reply. PS1 merges the policy received in R2 with its local policy and the policy information it received from Q1. The merge indicates that a signal and a reply will be needed. PS1 caches the merged policy.

    PS1 creates a reply with the query payload from R2 and the merged policy. Policy server and cert records are not necessary since PS1 is authoritative for H1:

```
        SPP Header [Reply, Sender PS1, qcount 1, rcount 3]
        Query Payload [comsec]:
          src H1, dst H2, UDP, 79
        Record Payload [comsec]:
          src H1, dst H2, UDP, 79, permit
        Record Payload [SA rec]:
          src H1, dst SG3, UDP, 79, permit, ESP tunnel H1->SG3
        Record Payload [SA rec]:
          src H1, dst H2, UDP, 79, permit, ESP transport H1->H2
        Signature Payload
```

    PS1 sends R3 to H1.

    PS1 creates a signal with a comsec record derived from knowing the traffic that will pass through SG1 and, the part of the merged policy that terminates at SG1:

```
        SPP Header [Pol, Sender PS1, qcount 0, rcount 2]
        Record Payload [comsec]:
          src H1, dst SG3, ESP, OPAQUE, permit
        Record Payload [SA rec]:
          src SG1, dst SG2, UDP, 79, permit, ESP tunnel SG1->SG2
        Signature Payload
```

    PS1 sends the signal to SG1.

12. **SG1 receives the signal and verifies the signature.** SG1 creates an Ack message to indicate that it has received the policy message:

```
        SPP Header [Pol-Ack, Sender SG1, qcount 0, rcount 0]
        Signature Payload
```

    SG1 sends the signal to PS1.

13. **PS1 receives the Pol-Ack and verifies the signature.** PS1 removes the corresponding policy message from its retry queue.

14. **Meanwhile, H1 receives the reply R3 and verifies the signature. The client can now use the policy as it is needed.**

Appendix C

Decorrelation

It is not possible for a Policy Server to use policies as they are written in the SPS master file, since those policies are likely to be correlated. Two policies are correlated if there is a non-nil intersection between the values of each of their selectors. Caching correlated policies can lead to incorrect policy implementation based on those cached policies, as the following example shows.

PS2 contains the following policies in its master file:

	src	dst	proto	direction	action
1)	*	H2	*	inbound	permit
2)	*	*	*	inbound	deny

These two policies are correlated since all the selectors (src, dst, proto, and direction) overlap. The following SPP exchanges occur:

1) PS1 requests policy for H3.
2) PS2 returns policy #2 to PS1 which then caches policy #2.
3) PS1 now looks up the policy for H2 and discovers that it already has a matching policy (policy #2) and uses that.

This is clearly wrong, since policy #2 indicates that the communication to H2 should be denied, though PS2's policy actually indicates that it should be allowed.

The solution is to remove the ambiguities that may exist in the master file. The policies of the master file MUST be decorrelated before they are entered into the Local Policy Database. That is, the policies must be rewritten so that for every pair of policies there exists a selector for which there is a nil intersection between the values in both of the policies.

The policies in the above example could be decorrelated as follows:

	src	dst	proto	direction	action
1')	*	H2	*	inbound	permit
2')	*	not H2	*	inbound	deny

Now the exchange is a bit different:

1) PS1 requests policy for H3.
2) PS2 returns policy #2' to PS1 which then caches policy #2'.
3) PS1 now looks up the policy for H2, doesn't have a matching

policy, so it requests a policy for H2.

4) PS2 returns policy #1' to PS1 which then caches policy #1', which is the correct policy for H2.

All SPAs and SPRs, therefore, MUST decorrelate their master files before using those policies for SPP. Once the policies are decorrelated, there is no longer any ordering requirement on the policies, since only one policy will match any requested communication. The next section describes decorrelation in more detail and presents an algorithm that may be used to implement decorrelation.

C.1 Decorrelation Algorithm

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

The basic algorithm does not guarantee an optimal set of decorrelated policies. That is, the policies 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 policies
Ci The ith policy in C.
U  The set of decorrelated policies being built from C
Ui The ith policy in U.
```

A policy P may be expressed as a mapping of selector values to actions:

```
Pi = Si1 x Si2 x ... x Sik -> Ai
```

1) Put C1 in set U as U1

For each policy Cj (j > 1) in C

2) If Cj is decorrelated with every policy in U, then add it to U.

3) If Cj is correlated with one or more policies in U, create a tree rooted at the policy Cj that partitions Cj into a set of decorrelated policies. The algorithm starts with a root node where no selectors have yet been chosen.

A) Choose a selector in Cj, Scjn, 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 policies in U that are correlated with the policy to this node.

The policy at this node is the policy 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 Cj, since the values in Cj represent the maximum value for each selector.

B) Add a branch to the tree for each value of the selector Scjn that appears in any of the policies in T. (If the value is a superset of the value of Scjn in Cj, then use the value in Cj, since that value represents the universal set.) Also add a branch for the compliment of the union of all the values of the selector Scjn in T. When taking the compliment, remember that the universal set is the value of Scjn in Cj. A branch need not be created for the nil set.

C) Repeat A and B until the tree is completed.

D) The policy to each leaf now represents a policy that is a subset of Cj. The policies at the leaves completely partition Cj in such a way that each policy is either completely overridden by a policy in U, or is decorrelated with the policies in U.

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

5) Get next Cj and goto 2.

6) When all policies 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 Scjn can be chosen so that all the values for that selector in T are equal to or a superset of the value of Scjn in Cj, then only a single branch need to be created (since the compliment will be nil).

Branches of the tree do not have to proceed with the entire decorrelation algorithm. For example, if a node represents a policy that is decorrelated with all the policies in U, then there is no reason to continue decorrelating that branch. Also, if a branch is completely overridden by a policy 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 policies in set C that has already been decorrelated. That is, if the branch is part of decorrelating Cj, then check to see if it was overridden by a policy Cm, m < j.

This is a valid check, since all the policies Cm are already expressed in U.

Along with checking if a policy is already decorrelated in step 2, check if Cj is overridden by any policy in U. If it is, skip it since it is not relevant. A policy x is overridden by another policy y if every selector in x is equal to or a subset of the corresponding selector in policy y. Appendix A shows an example of applying the algorithm to a set of correlated policies.

C.2 Decorrelation Example

This appendix section demonstrates the decorrelation algorithm and the optimizations presented on a sample set of policies. We start with the following set of correlated policies, set C:

	src	dst	prot	sport	dport	user	sec level
C1	199.93/16	199.100.2/24	TCP	*	22	lsanchez	sec
C2	199.93/16	199.100.2/24	TCP	*	*	lsanchez	conf
C3	199.93/16	199.100.2/24	UDP	*	*	lsanchez	*
C4	199.93/16	199.100.2/24	UDP	*	52	*	*
C5	199.93/16	199.100.2/24	*	*	*	*	*
C6	*	*	*	*	*	*	*

C.2.1 policy C1

We start with policy C1:

	src	dst	prot	sport	dport	user	sec level
C1:	199.93/16	199.100.2/24	TCP	*	22	lsanchez	sec

By step 1 of the algorithm, C1 is put directly into set U as policy U1.

The current decorrelated policy set U is:

U1	199.93/16	199.100.2/24	TCP	*	22	lsanchez	sec

C.2.2 policy C2

Next, we look at policy C2:

	src	dst	prot	sport	dport	user	sec level
C2:	199.93/16	199.100.2/24	TCP	*	*	lsanchez	conf

C2 is decorrelated with the policies already in U (U1) because the security levels do not overlap. By step 2, C2 is added to U as U2.

The current decorrelated policy set U is:

U1	199.93/16	199.100.2/24	TCP	*	22	lsanchez	sec
U2	199.93/16	199.100.2/24	TCP	*	*	lsanchez	conf

C.2.3 policy C3

Next, we look at policy C3:

	src	dst	prot	sport	dport	user	sec level
C3:	199.93/16	199.100.2/24	UDP	*	*	lsanchez	*

C3 is decorrelated with the policies already in U (U1 and U2) because it uses UDP while both policies in U use TCP. By step 2, C3 is added to U as U3.

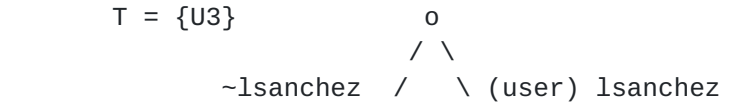
The current decorrelated policy set U is:

	src	dst	prot	sport	dport	user	sec level
U1	199.93/16	199.100.2/24	TCP	*	22	lsanchez	sec
U2	199.93/16	199.100.2/24	TCP	*	*	lsanchez	conf
U3	199.93/16	199.100.2/24	UDP	*	*	lsanchez	*

C.2.4 policy C4

Next, we look at policy C4:

	src	dst	prot	sport	dport	user	sec level
C4:	199.93/16	199.100.2/24	UDP	*	52	*	*



Policy C4 is correlated with policy U3 in U, so T = {U3}. First we choose to decorrelate the user selector. The policy in T as the value "lsanchez" for this selector, so we create a branch for "lsanchez" and its compliment.

The lsanchez branch:

199.93/16	199.100.2/24	UDP	*	52	lsanchez	*

is overriden by the policy U3.

The compliment branch:

199.93/16	199.100.2/24	UDP	*	52	~lsanchez	*

is decorrelated with T since ~lsanchez does not overlap any policies in T. Since this branch is decorrelated, it is added to set U.

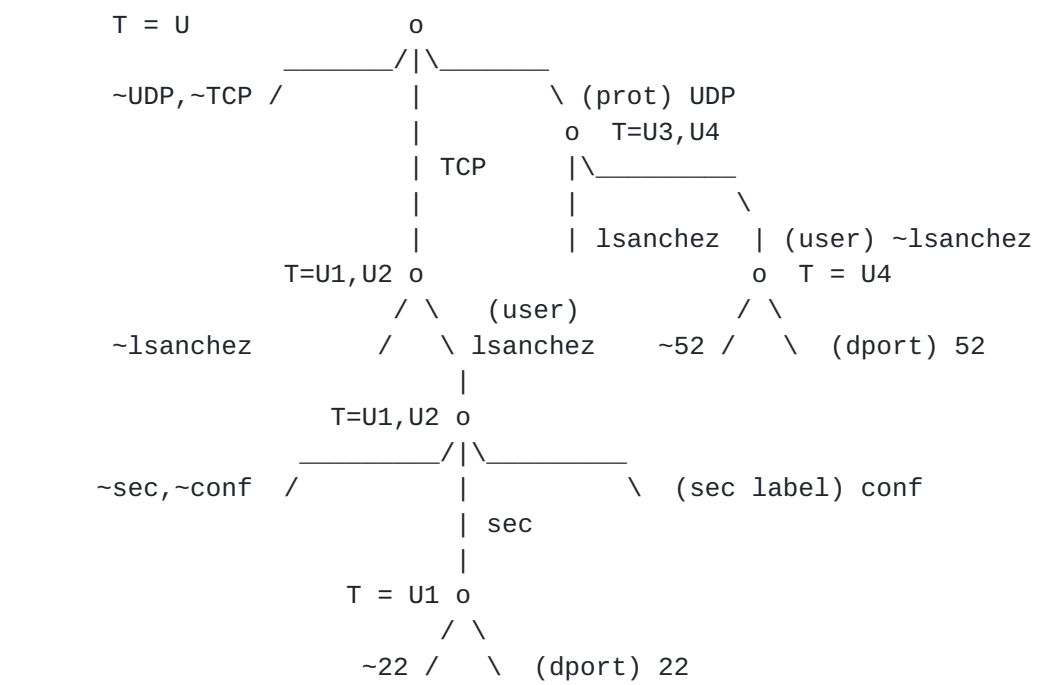
The current decorrelated policy set U is:

U1	199.93/16	199.100.2/24	TCP	*	22	lsanchez	sec
U2	199.93/16	199.100.2/24	TCP	*	*	lsanchez	conf
U3	199.93/16	199.100.2/24	UDP	*	*	lsanchez	*
U4	199.93/16	199.100.2/24	UDP	*	52	~lsanchez	*

C.2.5 policy C5

Next, we look at policy C5:

	src	dst	prot	sport	dport	user	sec level
C5:	199.93/16	199.100.2/24	*	*	*	*	*



Policy C5 is correlated with all the policies currently in U, so T = U. First we choose to decorrelate the protocol selector. The policies in T have the values ``UDP'' and ``TCP'' for this selector, so we create a branch for each of them and a branch for the complement of their union.

We can stop processing the complement branch:
199.93/16 199.100.2/24 ~UDP,~TCP * * * *
since it is decorrelated with T. This policy will be added to the decorrelated set.

The "UDP" and "TCP" branches still require more processing since they are both still correlated with policies in U. We will start by processing the "UDP" branch. The policy through this branch is correlated with policies U3 and U4, so T = {U3, U4}. We choose to decorrelate on the user selector. The policies in T have "lsanchez" and "~lsanchez" as their values for this selector so we create branches for "lsanchez" and "~lsanchez." The compliment branch is redundant to these branches.

We can stop processing the "lsanchez" branch:
199.93/16 199.100.2/24 UDP * * lsanchez *
since it is overridden by policy U3.

The "~lsanchez" branch, however, requires more processing since it is correlated with policy U4 (T = {U4}). We choose to decorrelate on the dport selector. The policy in T has "52" as its value for this selector so we create a "52" branch and a branch for its compliment

"~52".

We can stop processing the complement branch:
199.93/16 199.100.2/24 UDP * ~52 ~lsanchez *
since it is decorrelated with T. This policy will be added to the decorrelated set.

We can also stop processing the "52" branch:
199.93/16 199.100.2/24 UDP * 52 ~lsanchez *
since it is overridden by U4.

Now we need to go back and process the "TCP" branch. The policy through this branch is correlated with policies U1 and U2, so T = {U1, U2}. We choose to decorrelate on the user selector. The policies in T have "lsanchez" as their values for this selector so we create branches for "lsanchez" and its compliment, "~lsanchez."

We can stop processing the complement branch:
199.93/16 199.100.2/24 TCP * * ~lsanchez *
since it is decorrelated with T. This policy will be added to the decorrelated set.

The "~lsanchez" branch, however, requires more processing since it is correlated with both policies in T. We choose to decorrelate on the sec label selector. The policies in T have "sec" and "conf" as their values for this selector so we create branches "sec", "conf", and the complement of their union, "~sec,~conf"

We can stop processing the complement branch:
199.93/16 199.100.2/24 TCP * * lsanchez ~sec,~conf
since it is decorrelated with T. This policy will be added to the decorrelated set.

We can stop processing the "conf" branch:
199.93/16 199.100.2/24 TCP * * lsanchez conf
since it is overridden by policy U2.

The "sec" branch, however, requires more processing since it is correlated with policy U1 (T = U1). We choose to decorrelate on the dport selector. The policy in T has "22" as its value for this selector so we create a "22" branch and a "~22" branch for its compliment.

We can stop processing the complement branch:
199.93/16 199.100.2/24 TCP * ~22 lsanchez sec
since it is decorrelated with T. This policy will be added to the decorrelated set.

We can stop processing the "22" branch:
199.93/16 199.100.2/24 TCP * 22 lsanchez sec
since it is overridden by policy U1.

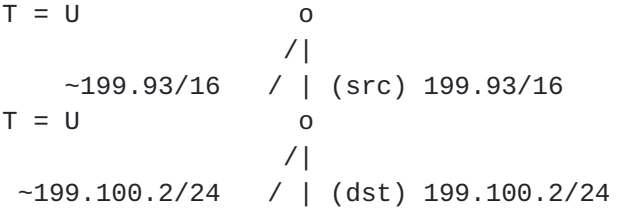
The decorrelated policy set after decorrelating C5 is:

U1	199.93/16	199.100.2/24	TCP	*	22	lsanchez	sec
U2	199.93/16	199.100.2/24	TCP	*	*	lsanchez	conf
U3	199.93/16	199.100.2/24	UDP	*	*	lsanchez	*
U4	199.93/16	199.100.2/24	UDP	*	52	~lsanchez	*
U5	199.93/16	199.100.2/24	~UDP,~TCP	*	*	*	*
U6	199.93/16	199.100.2/24	UDP	*	~52	~lsanchez	*
U7	199.93/16	199.100.2/24	TCP	*	*	~lsanchez	*
U8	199.93/16	199.100.2/24	TCP	*	*	lsanchez	~sec,~conf
U9	199.93/16	199.100.2/24	TCP	*	~22	lsanchez	sec

C.2.6 policy C6

Finally, we look at policy C6:

	src	dst	prot	sport	dport	user	sec level
C6:	*	*	*	*	*	*	*



Policy C6 is correlated with all the policies currently in U, so T = U. First we choose to decorrelate the src selector. The policies in T have the value "199.93/16" for this selector, so we create a branch for "199.93/16" and one for its compliment, "~199.93/16".

We can stop processing the complement branch:

~199.93/16	*	*	*	*	*	*

since it is decorrelated with all the policies in T. This policy will be added to the decorrelated set.

The "199.93/16" branch, however, requires more processing since it is correlated with all the policies in T. We choose to decorrelate on the dst selector. The policies in T have "199.100.2/24" as their value for this selector so we create a "199.100.2/24" branch and a "~199.100.2/24" branch for its compliment.

We can stop processing the complement branch:

199.93/16	~199.100.2/24	*	*	*	*	*

since it is decorrelated with all the policies in T. This policy will be added to the decorrelated set.

We can stop processing the "199.100.2/24" branch:

199.93/16 199.100.2/24   *   *   *   *   *
since it is overridden by policy C5.

The full decorrelated version of C is:

U1	199.93/16	199.100.2/24	TCP	*	22	lsanchez	sec
U2	199.93/16	199.100.2/24	TCP	*	*	lsanchez	conf
U3	199.93/16	199.100.2/24	UDP	*	*	lsanchez	*
U4	199.93/16	199.100.2/24	UDP	*	52	~lsanchez	*
U5	199.93/16	199.100.2/24	~UDP,~TCP	*	*	*	*
U6	199.93/16	199.100.2/24	UDP	*	~52	~lsanchez	*
U7	199.93/16	199.100.2/24	TCP	*	*	~lsanchez	*
U8	199.93/16	199.100.2/24	TCP	*	*	lsanchez	~sec,~conf
U9	199.93/16	199.100.2/24	TCP	*	~22	lsanchez	sec
U10	199.93/16	~199.100.2/24	*	*	*	*	*
U11	~199.93/16	*	*	*	*	*	*

Disclaimer

The views and specification here are those of the authors and are not necessarily those of their employers. The authors and their employers specifically disclaim responsibility for any problems arising from correct or incorrect implementation or use of this specification.

Author Information

Luis A. Sanchez
Megisto Systems

USA
Email: lsanchez@megisto.com
Telephone:

Matthew N. Condell
BBN Technologies
10 Moulton Street
Cambridge, MA 02138
USA
Email: mcondell@bbn.com
Telephone: +1 (617) 873-6203