IPSEC Working Group
INTERNET-DRAFT
draft-nsa-isakmp-00.txt,.ps

Douglas Maughan, Barbara Patrick, Mark Schertler
National Security Agency
March 20, 1995

Internet Security Association and Key Management Protocol (ISAKMP)

Abstract

This memo describes a combination of security concepts and protocols for establishing Security Associations (SA) and cryptographic keys in an Internet environment. A Security Association Protocol which negotiates, establishes, modifies and deletes Security Associations and their attributes is required for an evolving Internet, where there will be numerous security mechanisms and several options for each security mechanism. The key management protocol must be robust in order to handle public key generation for the Internet community at large and private key requirements for those enclaves with that requirement.

The Internet Security Association and Key Management Protocol (ISAKMP) defines the procedures for authenticating a communicating peer, creation and management of Security Associations, key generation techniques, and an anti-clogging mechanism, all of which are necessary to establish and maintain secure communications (via IPSP or any other security protocol) in an Internet environment.

Status of this memo

1 Introduction

This document describes the use of a combination of security concepts and protocols to provide the desired security for government, commercial, and private users of the Internet. We identify the necessary components to provide security for communications in the Internet, provide detailed descriptions of each of these components along with the algorithms and mechanisms to realize the expected security associated with these components, and outline possible scenarios that include these components.

1.1 Authentication

A very important step in establishing secure communications is authentication of the entity at the other end of the communication. There are many authentication mechanisms for this purpose. An example of weak authentication is the use of passwords. Digital signatures such as the Digital Signature Standard (DSS) and Rivest-Shamir-Adleman (RSA) signature are public key based strong authentication mechanisms that require a trusted third party to sign and properly distribute certificates. Kerberos is an example of an authentication system that relies on a trusted third party during the authentication. ISAKMP must allow a party initiating communications to indicate which authentication mechanism it is using and support the message exchanges required by that mechanism.

Certificates bind a specific identity (host, network, user, application) to public keys, privileges, clearances, compartments and other security-related information. Certificates are an essential part of strong authentication mechanisms. There must be an infrastructure available to verify, manage and distribute certificates. There is currently work going on in industry to develop X.500 Directory Services which would provide X.509 certificates to users. There has also been work done in this area by the NIST Public Key Infrastructure Working Group. Alternatively, if no infrastructure exists, the PGP Web of Trust could be used to provide user authentication in a community of users who know and trust each other.

ISAKMP does not specify a specific certificate authority or type (i.e. X.509 certificates), but it must allow the identification of different certificate authorities and types and facilitate the exchange of the chosen certificate type. This protocol will support the use of a variety of digital signatures to provide the strong authentication function. The DSS and RSA are examples of digital signatures which will provide that strong authentication. There are many others, as well. Details of DSS, RSA, and other signature algorithms may be found in [Schn94].

1.2 Security Associations and Management

A Security Association (SA) is a relationship between two or more entities. The relationship describes how the entities will utilize security services to communicate securely. This relationship is represented by a set of information that can be considered a contract between the entities. The information must be agreed upon and shared between all the entities. Sometimes the information alone is referred to as an SA, but this is just a physical instantiation of the existing relationship. The existence of the relationship, represented by the information, is what allows the entities to communicate securely. All entities must adhere to the SA for secure communications to be possible. The Security Association Identifier (SAID) is a pointer or identifier an entity uses to name the SA.

The types of information needed to represent an SA include, but are not limited to, authentication mechanisms, cryptographic algorithms, algorithm mode, key length, Initialization Vector (IV), integrity mechanisms, hash algorithms, etc. . ISAKMP allows communicating entities to negotiate the information needed to create an SA. The protocol includes the ability to establish, modify and delete an SA and negotiate the SA attributes.

NOTE: See Appendix B for initial list of SA attributes.

1.3 Public Key Cryptography

In an Internet environment with large numbers of users, there are many ways those users can interconnect. There are also many key management techniques and algorithms available to the users of the network. All users will not choose the same combination of capabilities. Therefore, users need a way to determine the capabilities of the entities with which they want to communicate. ISAKMP is intended to provide that service.

Because of the large number of different ways Internet users can connect, the use of public key cryptography is the most flexible and efficient way for users to obtain the keys they need.

There are two methods for using public key cryptography to place keys. In the first method, user A generates a random key. The random key is then encrypted, using a public key algorithm (e.g. RSA), with user B's public key. The encrypted random key is then sent to user B. In the second method, users A and B use a public key algorithm (e.g. Diffie-Hellman) to exchange public information. They then each use the other's public information along with their own secret keys to compute the same value which they use as the session key or the key encryption key for encrypting the session key.

If public key cryptography is used in this way for exchanging or agreeing upon a new key each time they communicate, then both back traffic protection and perfect forward secrecy will be provided. Each key is independent and the compromise of one key will not automatically compromise any other keys. The second method described above is preferred as the key used for encrypting messages is based on information held by both A and B.

1.4 Back Traffic Protection / Perfect Forward Secrecy

The concept of back traffic protection is concerned with the cryptographic protection of previous traffic, even when cryptographic keys used to encrypt future traffic are compromised. The use of public key cryptography for the establishment of cryptographic keys provides back traffic protection. The difficulty of numerical factoring of large numbers has proven that cryptographic keys can protect information for a considerable length of time. However, this is based on the assumption that keys used for protection of communications are destroyed after use and not kept for any reason. This concept of back traffic protection is provided by the independent generation of each key such that subsequent keys are not dependent on any previous key.

The concept of perfect forward secrecy is aimed at guaranteeing future communications are cryptographically protected, even in the event of compromise of current cryptographic keys. This concept of perfect forward secrecy is provided by the independent generation of each key such that subsequent keys are not dependent on any previous key.

1.5 Anti-Clogging

Of the numerous security services available, protection against denial of service always seems to be one of the most difficult to address. Phil Karn in his Internet-Draft [Karn94] has introduced a mechanism to provide a measure of denial of service protection through the use of a "cookie" exchange. This anti-clogging token exchange (ACTE) is aimed at protecting the computing resources from attack without spending excessive CPU resources to determine its authenticity. As described in [Karn94], an exchange prior to CPU-intensive public key operations can thwart some denial of service attempts (e.g. simple flooding with bogus IP source addresses). As noted by Karn, absolute protection against denial of service is impossible, but this anti-clogging token exchange provides a technique for making it easier to handle.

1.6 Multicast Communications

While it is envisioned that future communications will increasingly be of a multicast nature, this document is addressing security and key management for unicast communications. It is expected that multicast communications will require the same security services as unicast communications and may introduce the need for additional security services. Upon agreement and implementation of security mechanisms for the unicast Internet community, we fully intend to examine any additional security requirements for multicast communications. For an introduction to the issues related to multicast security consult the Internet Drafts, [Spar94a] and [Spar94b], describing Sparta's research in this area.

2 Description of the Protocol

The Internet Security Association and Key Management Protocol (ISAKMP) is not a single protocol, but a combination of cryptographic techniques, mechanisms, and protocols. This coalition is designed to provide maximum flexibility, while providing the requisite security. The ISAKMP consists of four components that can be applied in various combinations to provide security services according to a stated requirement. The four components of the ISAKMP are listed below in no particular order:

- A strong authentication exchange.
- An anti-clogging token exchange (ACTE) similar to the work presented in [Karn94].
- A Security Association Management Protocol (SAMP) based on the IEEE Key Management Protocol (KMP).
- A public key exchange to establish a session key for communications using conventional encryption.

The ordering of these ISAKMP components to create a specific protocol exchange is called a scenario. Scenarios depicting possible component orderings are given in section 8 with a high-level state diagram given in appendix A.

2.1 General Packet Format

ISAKMP has a specific header format, described below, followed by the component dependent payload formats.

SAID (32)
Scenario (8)
Payload Type (8)
Payload

ISAKMP Packet Format

1. SAID - The SAID field indicates a predefined SAID that is used to determine the key generation and/or the authentication method in those scenarios that require this information prior to SA negotiation. For scenarios that start with an SA negotiation this field is set to a default of TBD.
2. Scenerio - The scenario indicator is encoded in this field.
   (a) Authenticated Clear Setup: 01h.
   (b) Authenticated Protected Setup: 02h.
   (c) Negotiated Clear Setup: 03h.
   (d) Anonymous Setup: 04h.
3. Payload Type - The component of the ISAKMP exchange is encoded in this field.
   (a) Authentication: 01h.
   (b) SAMP: 02h
   (c) Key Exchange: 03h.
   (d) Anti-clogging token: 04h.
4. Payload - The format of the payload as defined in the following sections.

The following sections describe the details of each component of the ISAKMP.

3 Strong Authentication Details

This section specifies the encoding of the payload of the ISAKMP PDU for the authentication component.

Authentication Authority (16)
Authentication Type (8)
Length (8)
Authentication Data

Authentication Payload Format

1. Authentication Authority - This field encodes the party that generated the certificates used for authentication.

   PGP certificates based on the "web of trust" have a standard encoding. Other third party certificate authorities must be assigned an identifier by the Internet Assigned Numbers Authority (IANA).

   PGP certificate - 0001h

   Examples of third party certificate authorities that would have to register for an identifier are RSA Commercial Certificate Authority (https://www_csc.rsa.com/netsite), the Stable Large E-mail Database (SLED) (http://www.four11.com) and the U.S. Postal Service.

2. Authentication Type - This field indicates the authentication payload format.

   RSA certificates

   PGP certificates

   DSS certificates

   X.509 certificates

   Other certificate formats

3. Length - Length of the Authentication Data field in bytes.

4. Authentication Data - This field contains the actual authentication data. The certificates as indicated by the Authentication type field. Examples are RSA, PGP, and DSS certificates.

4 Anti-Clogging Token Exchange Details

This section will detail the Anti-Clogging Token Exchange (ACTE) outlined in Phil Karn's Internet Draft [Karn94].

5 Security Association Management Protocol

This section describes the SAMP component of ISAKMP and payload format for each SA service. The SAMP component provides the ability to establish (including negotiation), modify, and delete Security Associations and their attributes. The SAMP services are listed below:

1. SA Negotiation.
2. SA Commit.
3. SA Modify.
4. SA Delete.

The SAMP payload format is the same for all the SAMP services, but depending on the type of SA service provided all the fields may not be present. Those fields not present for a service are defined in sections 5.1 through 5.4.

SA Indicator (4)
SA Flags (4)
SAID Initiator (32)
SAID Responder (32)
SA Attributes

SAMP Payload Format

1. SA Indicator - This field encodes the SA service carried in the SAMP payload.
   (a) SA Negotiation: 1h.
   (b) SA Commit: 2h.
   (c) SA Modify: 3h.
   (d) SA Delete: 4h.
2. SA Flags - This field encodes flags specific to an SA service.
3. SAID Initiator - The SAID used by the initiating entity to uniquely identify an SA within its local system.
4. SAID Responder - The SAID used by the responding entity to uniquely identify an SA within its local system.
5. SA Attributes - A list of SA attributes. SA Attribute specifications are discussed in section 5.1.1.

5.1 SA Negotiation

The SA negotiation service allows the initiating entity to present a series of SA attributes, that it wishes to use for secure communications, to a responding entity. Each attribute may contain a number of options presented in priority order. The responding entity must choose the highest priority option it can support for each attribute. The responding entity sends back the SA attribute list with the option it has chosen for each attribute.

Example:

(Letters are SA Attributes and Numbers are options for the attributes.)

Initiating entity sends (A:1,2,3; B:1,2; C:2,3,4) to responding entity.

Responding entity sends back (A:3; B:1; C:3) to initiating entity.

Another option for exchanging SA Attributes for negotiation is to present the sets of attributes with a single option that would represent the SA. The sets are presented in priority order. The responding entity would choose the highest priority set of which it can support the option for each attribute. This method of presenting SA Attributes for negotiation has the possibility of having a number of sets presented that differ only slightly. The advantage to this presentation method is that SA Attribute combinations that are illegal cannot be choosen by the responding entity.

Example:

(Letters are SA Attributes and Numbers are the option for the attribute.)

Initiating entity sends (A2,B2,C4; A3,B1,C3; A1,B2,C2) to responding entity.

Responding entity sends back (A3,B1,C3) to initiating entity.

If the responding entity cannot support any option for a particular attribute, it can respond with a NULL and an SA will not be established. The responding entity may also provide a single alternative for any of the SA attributes for which it can't support the offered options. The initiating entity may choose to accept the alternative offered using the SA Commit service, see section 5.2. If the initiating party does not accept the alternative offered by the responding entity an SA is not established.

The SA Negotiation must use all fields in the SAMP payload.

The SAID Initiator field is set by the initiating entity on the first negotiation exchange. The responding entity cannot change this field.

The SAID Responder field is set by the responding entity on the second negotiation exchange. The initiating entity cannot change this field.

The SA Flags field can be set by the entity that initiates the negotiation to indicate that the SA commit service will be performed following negotiation. If the initiating entity sets the flag the responding entity may not reset it. If the initiating entity does not set the flag the responding entity may set it, thereby forcing the initiating entity to issue an SA Commit. If neither entity sets the flag then the SA Commit service shall not be invoked.

5.1.1 SA Attributes

SA attributes are listed in Appendix B. This is an initial set of attributes to begin discussion. The final set of SA attributes should be defined in a separate RFC so additions or modifications to the attributes do not require a modification to the protocol RFC and visa versa. An SA object and SA attributes should become part of the MIB, see [RFC-1213], on a special branch called the SMIB. This puts ISAKMP in line with the SNMP concept of separate RFCs for the protocol and the information managed. SA attributes will be defined using the syntax in [RFC-1155] and [RFC-1212].

5.2 SA Commit

The SA Commit service allows an initiating entity to send a commitment to the responding entity after negotiations are complete. This informs the responding party that the initiating party agrees to the SA and secure communications can begin. This service is optional when the responding entity selected the options offered by the initiating party. It is required when the responding party request this service when it presents alternatives to the SA attributes offered. Section 5.1 contains the details on how to set the SA Flags field within an SA Negotiation payload in order to request this service.

The SA Flags field is set to a NOT USED default of TBD.

The SAID Initiator field and SAID Responder field must contain the respective SAIDs set during the SA Negotiation Service.

The SA Commit service will not have the SA Attributes field in the SAMP payload.

5.3 SA Modify

The SA Modify service provides the ability to update an existing SA.

The SA Modify must use all fields in the SAMP payload.

The SAID Initiator field and SAID Responder field must contain the respective SAIDs that are being modified.

The SA Attributes field contains only those attributes being updated.

5.4 SA Delete

The SA Delete service informs the remote entity that the SA within the local system of the initiating entity is being deleted.

The SAID Initiator field and SAID Responder field must contain the SAIDs that correspond to the SA being deleted.

The SA Delete service will not have the SA Attributes field in the SAMP payload.

6 Key Exchange Details

A variety of key exchanges can be supported by this protocol. Some examples of key exchanges which may be used in this protocol are Diffie-Hellman, the enhanced Diffie-Hellman key exchange described in X9.42 [ANSI94], the key exchange on the FORTEZZA card, and the RSA-based key exchange used by PGP. This protocol will also support the government key escrow requirement, but does not mandate its use.

The encoding of the ISAKMP payload for the key exchange is dependent on the specific key exchange and therefore is not specified in this Internet draft. When a key exchange is registered in the SMIB, see 5.1.1, its field specifications must also be registered so the appropriate parsers can be constructed to process the exchange.

7 Concatenation Details

The Authentication and Key Exchange payloads can be concatenated together in the scenarios described in sections 8.2.2, 8.2.3, and 8.2.4. This requires an additional payload type (SAUTH-KE) in the ISAKMP header. This is provided to limit the number of exchanges and speed-up SA establishment. Additional concatenations of components will be considered as long as they do not compromise security or negate the benefit of a set-up scenario.

8 Protocol Scenarios

8.1 Introduction

Sections 3 through 6 provide the details of the four components of ISAKMP. This section describes a number of scenarios or exchanges that can be built from these four components. The scenarios highlight the positive aspect of having 4 protocol components whose order of exchange is flexible. Each scenario provides a different type of protection and security attribute negotiation and each is suitable for a different situation. The benefits and constraints to each scenario are also presented. There are three categories of exchanges: SA Establishment or Setup, SA Modification and SA Deletion. The SA Establishment category presents six basic protocol component exchanges orderings that are possible using the four components. The SA Modification and SA Deletion categories each have one scenario. None of these scenarios is mandated. All of these scenarios have positive aspects that make them preferable under circumstances that benefit from their strengths.

8.2 SA Establishment Scenarios

8.2.1 Authenticated Clear Setup

This scenario provides the capability to do a strong authentication exchange prior to establishing any additional security relevant information. Once the authentication is complete, the SAMP is initiated to establish the security association attributes and key exchange parameters for the ensuing communications. The Key Exchange is then performed using these negotiated parameters (e.g. Algorithm ID, algorithm mode, key length, etc.).

The benefit of this scenario is that it allows the negotiation of the security association and key exchange parameters in a strongly authenticated mode.

The constraint to this scenario is that the authentication mechanism must be known prior to communications and the security attribute establishment is not encrypted. Once the is SA established it can be used to renegotiate security attributes which are encrypted.

To Be Done: SAuth -> SAMP -> KE State Table

8.2.2 Authenticated Protected Setup

This scenario provides the capability to do a strong authentication exchange and key exchange prior to establishing any additional security relevant information. Upon successful completion of both the authentication and key exchange, the security attributes are negotiated using the SAMP.

The benefit of this scenario is that the negotiation of the security association attributes is both strongly authenticated and cryptographically protected.

The constraint to this scenario is that the authentication mechanism and key exchange parameters must be known prior to communications.

To Be Done: SAuth -> KE -> SAMP State Table

8.2.3 Negotiated Clear Setup

This scenario allows the negotiation of security relevant information, including the authentication mechanism and key exchange technique, using the SAMP. After the negotiation of the security attributes, the strong authentication and key exchange are performed. Once the strong authentication and key exchange are complete the SA Commit service of the SAMP component may be re-issued to provide cryptographically protected confirmation of the SA attributes.

This scenario allows all attributes of the future communication to be negotiated. The benefit of this scenario is that two entities who both are capable of providing security, but have never "met" before, can establish an SA. When connecting to a remote entity with which there is no pre-placed SA, to allow authentication or key exchange, this scenario provides the ability to attempt to establish an SA in-band, rather than having to use an out-of-band method such as the telephone, fax, or mail.

The constraint to this scenario is that the SAMP negotiation is completely unprotected and the actual identity of the communicating peer is not known until the authentication is performed.

To Be Done: SAMP -> SAuth -> KE State Table

8.2.4 Negotiated Clear Setup with Anti-Clogging

This scenario adds the anti-clogging technique to the beginning of the scenario presented in section 8.2.3.

The added benefit of this scenario is the protection against denial of service, provided by the anti-clogging technique, prior to the negotiation phase.

The additional constraint to this scenario is the added exchange to perform the anti-clogging technique. Additionally, this anti-clogging technique provides only limited denial of service protection.

To Be Done: ACTE -> SAMP -> SAuth -> KE State Table

8.2.5 Anonymous Setup

This scenario performs the key exchange technique followed by the strong authentication mechanism prior to any SAMP negotiation. This scenario provides anonymity for the initiating entity and full cryptographic protection of all exchanges as well as strongly authenticated communications.

The benefit of this scenario is the ability to communicate with a person with whom you've pre-established encryption and authentication attributes without disclosing your or their identity to passive attackers on the network. Also, the security attribute negotiation is cryptographically protected.

The constraint to this scenario is that the authentication mechanism and key exchange parameters must be known prior to communications.

To Be Done: KE -> SAuth -> SAMP State Table

8.2.6 Anonymous Setup with Anti-Clogging

This scenario adds the anti-clogging technique to the beginning of the scenario presented in section 8.2.5.

The added benefit of this scenario is the protection against denial of service, provided by the anti-clogging technique, prior to the key exchange technique.

The additional constraint to this scenario is the added exchange to perform the anti-clogging technique. Additionally, this anti-clogging technique provides only limited denial of service protection.

To Be Done: ACTE -> KE -> SAuth -> SAMP State Table

8.3 SA Modification Scenario

The procedure for modifying an SA using the ISAKMP components will be described in this section.

8.4 SA Deletion Scenario

The procedure for deleting an SA using the ISAKMP components will be described in this section.

9 Conclusions

A Protocol State Diagram

B Security Association Attributes

This appendix is an exact copy of an e-mail message [Kent94] to the IPSEC mail list from Steve Kent and is reproduced here to start a discussion on SA attributes. Inclusion in this appendix is not a wholesale endorsement.

```
The following is a set of possible parameters for each security
association (SA), e.g., candidate MIB data items where each SA
has its own MIB entry.  They may be negotiated or pre-determined,
but all are important for each SA in the most general case.

SAID.INBOUND
SAID.OUTBOUND
     These variables contain the SAIDs for this SA and thus are
used to select this MIB entry for IPSP protected traffic.

ENCAPSULATION
     This Boolean indicates whether IPSP is used to encapsulate
packets on this SA.  If TRUE, then encapsulation is employed,
otherwise packets are not encapsulated.

INBOUND-CRITERIA
     This data structure specifies the checking applied to
inbound packet received on this SA before releasing the packets
to the client.  Each data item is either null, or is a value
against which the inbound packet is checked.  If encapsulation is
employed, the checks are applied to the encapsulated header(s),
otherwise the checks are applied to the outer header(s). {needs
work, e.g., to express star names}
     IP-DESTINATION-ADDRESS
     IP-SOURCE-ADDRESS
     NEXT-PROTOCOL
     IP-SECURITY-LABEL
     TRANSPORT-DESTINATION-PORT
     TRANSPORT-SOURCE-PORT


PEER-ADDRESS
     This variable specifies the IP address of the IPSPE that
defines the other end of the SA.  For a multicast SA, this is an
IP multicast address.

ENCRYPTION
```

This data structure provides the parameters required for provision of a connectionless confidentiality service, if employed for this SA.

ENABLED

This is a Boolean and is TRUE if encryption is to be employed for this SA.

ALGORTIHM

This is a structured, IANA-registered algorithm ID that also specifies the mode of use, e.g., DES-CBC or DES-EDE2-CBC, or DES-CFB-8.

KEY.INBOUND
KEY.OUTBOUD

If the same key is used for encryption in both directions, then the values are the same for both of these variables, otherwise different values must be present. (For some algorithms and modes, this value may be an interchange key, and the key to be used to decrypt an individual packet may encrypted by this key and carried in the packet in the manner of an IV.) The size of the key is determined by ENCRYPTION.ALGORITHM and if a multi-key algorithm mode is employed, e.g., DES-EDE, all of the keys for a given direction are stored in the appropriate variable, and the algorithm mode specifies how each key is extracted from the variable.

IV.INBOUND
IV.OUTBOUND

If the algorithm and mode require use of an IV, and if the IV (or a portion thereof) is constant for the SA, then these per-SA values are stored here. The size of these variable and the portion used in forming an IV is determined by ENCRYPTION.ALGORITHM.

INTEGRITY

This data structure contains all of the parameters required for provision of a connectionless integrity service, if employed for this SA.

ENABLED

This is a Boolean that is TRUE if an integrity transformation is applied for this SA.

PLAINTEXT

This value is TRUE if the integrity transformation is applied to plaintext, and is FALSE if it is applied to ciphertext; FALSE may be selected only if ENCRYPTION.ENABLED = TRUE

DIRECTION.ENABLED

This is a Boolean that is TRUE if a direction flag is used in packets.

DIRECTION.VALUE

This is a Boolean that specifies the one-bit direction flag value inserted into outbound packets. The value is meaningful only if INTEGRITY.DIRECTION.ENABLED is TRUE.

ALGORITHM

This is a structured, IANA-registered algorithm ID that specifies the integrity algorithm employed and the mode of use, e.g., MD5, MD5-keyed, SHA, DES-MAC, etc.

KEY.OUTBOUND

KEY.INBOUND

These items have meaning only if a keyed integrity function is employed, as defined in INTEGRITY.ALGORITHM. If ENCRYPTION.ENABLED = FALSE, then a keyed integrity function must be employed, if INTEGRITY.ENABLED = TRUE. If the same key is used in both directions, then these variables have the same value, otherwise different values must be present. The size of key is a function of the algorithm selected in INTEGRITY.ALGORIHTM and if a multi-key algorithm mode is employed, all of the keys for a given direction are stored in the appropriate variable, and the algorithm mode specifies how each key is extracted from the variable.

COMPRESSION

This data structure provides parameters required for provision of compression (and corresponding decompression) of plaintext data.

ENABLED

This Boolean is TRUE if compression is to be applied to traffic on this SA.

ALGORITHM

This is a structured, IANA-registered algorithm ID that specifies the compression algorithm employed.

REPLAY

This data structure provides the parameters required for provision of replay countermeasures, if employed for this SA.

ENABLED

This Boolean is TRUE if a sequence number is included in each outbound packet and is checked on each inbound packet for this SA.

SIZE

This item defines the size of sequence number, expressed in 16-bit words. {we could bound this at 64 bits}

NUMBER.OUTBOUND

This is the value of the sequence number to be assigned to next outbound packet. The size of this variable is determined by REPLAY.SIZE.

NUMBER.INBOUND

This is the sequence number expected in next inbound packet; if an integrity checked packet arrives with a sequence number greater than or equal to this value, the variable is updated to be one greater than that received value, and REPLAY.WINDOW is updated to reflect the received packet sequence number.

WINDOW.SIZE

This is the size of the window within which late arriving packets will be checked for duplication. {we could fix this value, or allow it to be only one of a few small values, e.g., 32/64/128, rather than allowing arbitrary window sizes}

WINDOW

This is logically an array of packet sequence numbers for packets received within the current window. Its dimension is REPLAY.WINDOW.SIZE. It may be implemented as a bit map, rather than an array of sequence numbers, using OR operations and end-off shifts to efficiently manage the list. It is checked only when a packet arrives with a sequence number less than REPLAY.NUMBER.INBOUND and greater than REPLAY.NUMBER.INBOUND - REPLAY.WINDOW.SIZE + 1 (since all sequence numbers less than this are, by definition, too old). It is updated every time a sequence numbered packet is received

FRAGMENTATION

This data structure provides the parameters required for provision of denial of service protection with regard to attacks exploiting the fragmentation features of IP.

INBOUND

If FALSE, no fragments are accepted on inbound traffic, otherwise fragments are acceptable for inbound traffic.

OUTBOUND

If FALSE, no fragments may be transmitted by the IPSPE (unless encapsulated), otherwise fragments may be transmitted.

KEY-MANAGEMENT

This data structure provides the key management parameters associated with this SA.

NEGOTIATED

If TRUE, then IKMP is used to negotiate the key management technique for this SA. If FALSE, a pre-defined key management technique is used.

TECHNIQUE

This variable specifies the key management technique used for this SA. The identifier used here is structured, e.g., to specify symmetric vs. asymmetric techniques, and is IANA registered.

PARAMETERS

Technique-specific parameters ???

REKEY

This data structure indicates what criteria are used to determine when a new key must be established for this traffic flow. When a new key is established, the traffic will be migrated to a new SA, identified by the NEXT.SA pointer below.

GRACE

This is the time value (in seconds) that incoming traffic is permitted to be processed on the old SA after a new SA has been established, in response to a rekey.

NEXT-SA

This is a pointer to the data structure for the new SA to which traffic will be routed when the rekey is complete.

TIME-BASED

This data structure contains variables for time based rekey management.

ENABLE

This Boolean is TRUE if time-based rekey management is employed.

TRIGGER

This is a local time value that specifies when a new key and SAID should be created, after which the traffic for this SA will be transferred to the new SAI. This variable should be zero unless this IPSPE initiated the SA and unless REKEY.TIME-BASED.ENABLE is TRUE. {is this an acceptable simplification, or should either end be able to initiate a rekey?}

TRAFFIC-BASED

This structure contains parameters for traffic-based rekey control.

ENABLE

```
                    This Boolean is TRUE if traffic-based rekey
management is employed.

               PACKET-COUNT.INBOUND
               PACKET-COUNT.OUTBOUND
                    These are counters for traffic that will be
used to trigger a rekey.  They are maintained only if
REKEY.TRAFFIC-BASED.ENABLE is TRUE.

               TRIGGER.INBOUND
               TRIGGER.OUTBOUND
                    These are values that are compared against
the packet counters to trigger a rekey. They are non-zero only if
REKEY.TRAFFIC-BASED.ENABLE is TRUE.
```

Security Considerations

This Internet Draft presents a Security Association and Key Management Protocol. Cryptographic mechanisms require keys to provide data confidentiality and authentication services on a network. Secure communications may utilize a variety of security services and mechanisms. The ability to choose from a variety of security services and mechanisms and to exchange attributes required by the mechanisms is important to security in the complex structure of the Internet. Therefore it is essential to secure communications to provide a Security Association Protocol capability on the Internet.

Acknowledgements

Thanks to Carl Muckenhirn of SPARTA, Inc. for his assistance with LaTeX.

References

[ANSI94] ANSI, *X9.42: Public Key Cryptography Using Irreversible Algorithms for the Financial Services Industry – Management of Symmetric Keys Using Diffie-Hellman*, Draft, September, 1994.

[Karn94] Phil Karn, *The Photuris Key Management Protocol*, Internet-Draft, December, 1994.

[Kent94] Steve Kent, *IPSEC SMIB*, e-mail to ipsec@ans.net, August 10, 1994.

[RFC-1155] Rose M. and K. McCloghrie, *Structure and Identification of Management Information for TCP/IP-based Internets*, RFC-1155, May, 1990.

[RFC-1212] McCloghrie K. and M. Rose, *Concise MIB Definitions*, RFC-1212-, March 26, 1991.

[RFC-1213] McCloghrie K. and M. Rose, *Management Information Base for Network Management of TCP/IP-based Internets: MIB-II*, RFC-1213, March 26, 1991.

[Schn94] Bruce Schneier, *Applied Cryptography - Protocols, Algorithms, and Source Code in C*, John Wiley & Sons, Inc., 1994.

[Spar94a] Harney H., C. Muckenhirn, and T. Rivers, *Group Key Management (GKMP) Architecture*, SPARTA, Inc., Internet-Draft, September, 1994.

[Spar94b] Harney H., C. Muckenhirn, and T. Rivers, *Group Key Management (GKMP) Specification*, SPARTA, Inc., Internet-Draft, September, 1994.

Addresses of Authors

The three authors are with:

National Security Agency
ATTN: R2
9800 Savage Road
Ft. Meade, MD. 20755-6000

Douglas Maughan
Phone: 301-688-0847
E-mail: `wdmaugh@tycho.ncsc.mil`

Barbara Patrick
Phone: 301-688-0298
E-mail: `bspatri@orion.ncsc.mil`

Mark Schertler
Phone: 301-688-0849
E-mail: `mjs@tycho.ncsc.mil`