IPSEC Working Group
INTERNET-DRAFT
draft-ietf-ipsec-isakmp-01.txt, .ps

Douglas Maughan, Barbara Patrick, Mark Schertler
National Security Agency
October 31, 1995

Internet Security Association and Key Management Protocol (ISAKMP)

Abstract

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

The Internet Security Association and Key Management Protocol (ISAKMP) defines the procedures for authenticating a communicating peer, creation and management of Security Associations, key generation techniques, and threat mitigation (e.g. denial of service and replay attacks). All of these are necessary to establish and maintain secure communications (via IP Security Service or any other security protocol) in an Internet environment.

Status of this memo

Contents

1 Introduction **3**

1.1 Authentication . . . 3

1.2 Security Associations and Management . . . 3

1.3 Public Key Cryptography . . . 4

1.4 Back Traffic Protection / Perfect Forward Secrecy . . . 4

1.5 Anti-Clogging . . . 5

1.5.1 Anti-Clogging Token Creation . . . 5

1.6 Multicast Communications . . . 5

2 Description of the Protocol **6**

2.1 ISAKMP Header Format . . . 6

2.1.1 General Message Processing . . . 8

2.2 ISAKMP Details . . . 9

2.2.1 Security Association Attributes . . . 9

2.2.2 Transport Protocol . . . 10

2.2.3 RESERVED Fields . . . 11

2.3 Security Association Establishment . . . 11

2.3.1 Security Association Initialization . . . 11

2.3.2 Key and Authentication Phase . . . 13

2.3.3 Security Association Negotiation Phase . . . 16

2.3.4 Packet Exchanges . . . 19

2.4 Security Association Modification . . . 20

2.5 Security Association Deletion . . . 21

2.6 Notification Message . . . 22

3 Conclusions **23**

A Key Exchange Examples **24**

A.1 Photuris KE 24

A.2 FORTEZZA Key Exchange Algorithm (KEA) 24

B Security Association Attributes **26**

1 Introduction

This document describes an Internet Security Association and Key Management Protocol (ISAKMP). ISAKMP combines the security concepts of authentication, key management, and security associations to establish the desired security for government, commercial, and private communications on the Internet. ISAKMP does not bind itself to any specific cryptographic algorithm, key generation technique, or security mechanism. This flexibility is beneficial because new attacks are constantly being developed that make today's security certainties obsolete. ISAKMP will guard against denial of service, replay, and connection hijacking attacks. This is important because these are the types of attacks that are targeted against protocols. Independence from specific security mechanisms that will eventually be replaced by better ones and protection from protocol threats are the strengths of ISAKMP.

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 allows 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. Currently, Domain Name System (DNS) Security Extensions [EK94] are being developed which will provide signed host keys in DNS. There is also work going on in industry to develop X.500 Directory Services which would provide X.509 certificates to users. The NIST Public Key Infrastructure Working Group has also been doing work in this area. 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 (e.g. 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 supports the use of a variety of digital signatures to provide the strong authentication function. The DSS and RSA are examples of digital signatures which provide 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 Parameter Index (SPI) 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. It includes the ability to establish, modify and delete an SA and negotiate the SA attributes.

NOTE: See Appendix B for example lists 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. Then, they 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 [Karn95] has introduced a mechanism to provide a measure of denial of service protection through the use of a "cookie" exchange. This anti-clogging token (ACT) is aimed at protecting the computing resources from attack without spending excessive CPU resources to determine its authenticity. As described in [Karn95], 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 provides a technique for making it easier to handle.

1.5.1 Anti-Clogging Token Creation

Phil Karn's Internet Draft [Karn95] states that cookie generation is implementation dependent, but must satisfy some basic requirements:

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

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

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

Karn's suggested method for creating the cookie is to perform a fast hash (e.g. MD5) over the IP Source and Destination Address, the UDP Source and Destination Ports and a locally generated secret random value. ISAKMP requires that the cookie be unique for each SA establishment, SA modify and SA delete to help prevent replay attacks, therefore we suggest adding the date and time to the information hashed.

1.6 Multicast Communications

While future communications will increasingly be of a multicast nature, this document is presenting a security association and key management protocol from the unicast point of view. It is expected that multicast communications will require the same security services as unicast communications and may introduce the need for additional security services. The issues of distributing SPIs for multicast traffic are presented in [Atki95]. Upon agreement and implementation of a security association protocol for the Internet unicast environment, 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) defines procedures and packet formats to establish (including negotiate), modify and delete Security Associations (SA). SAs contain all the information required for execution of IP security services, such as the IP Authentication Header (AH), the IP Encapsulating Security Payload (ESP), and routing protocol authentication mechanisms. ISAKMP includes packet formats for exchanging key generation and authentication data. These formats provide a consistent method of transferring key and authentication data that is independent of the key generation technique, encryption algorithm or authentication mechanism. These generic packets provide flexibility by allowing the protocol to be independent of key generation techniques and authentication mechanisms used to establish SAs.

The following sections contain the details of ISAKMP. Sections 2.1 through 2.2 cover details that are pertinent to the entire protocol. Sections 2.3 through 2.6 define the establishment, modification, and deletion services, defined as exchanges, offered by the protocol. The appendices provide examples of SAs and key exchanges.

2.1 ISAKMP Header Format

ISAKMP has a fixed header format. A fixed header simplifies parsing, providing the benefit of less complex and easier to implement parsing software. The fixed header contains the information required by the protocol to maintain state, process payloads and prevent attacks (e.g. denial of service and replay). Based on the message type each header is followed by a payload specific to the message type. The payload for each message is define in sections 2.3 through 2.6.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Message Type  |    Exchange   |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              SPI                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Initiator-Cookie                       |
~                                                               ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Responder-Cookie                       |
~                                                               ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Payload                            |
~                                                               ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

ISAKMP Header Format

- Message Type (1 octet) - Indicates the type of message. A suffix of REQ denotes a message of type request and RESP suffix denotes a message of type response. The format and processing for each message is defined in sections 2.3 through 2.6.

ISAKMP Message	Message Type
ISA_INIT_REQ	1
ISA_INIT_RESP	2
ISA_KE_REQ	3
ISA_KE_RESP	4
ISA_AUTH_REQ	5
ISA_AUTH_RESP	6
ISA_AUTH&KE_REQ	7
ISA_AUTH&KE_RESP	8
ISA_NEG_REQ	9
ISA_NEG_RESP	10
ISA_MODIFY_REQ	11
ISA_MODIFY_RESP	12
ISA_NOTIFY	13
ISA_DELETE	14
ISA_COMMIT	15

- Exchange (1 octet) - indicates the type of exchange, see Sections 2.3.4 and 2.3.4

ISAKMP Exchange	Exchange Type
Base	1
Identity Protection	2

- Length (2 octets) - Length of total message (header + payload) in octets.
- SPI (4 octets) - Security Parameter Index. The receiving entity's SPI is always in this field, except for the ISA_INIT packets. The ISA_INIT packets contain the SPI the issuer expects to receive in all subsequest packets.
- Initiator Cookie (16 octets) - Cookie of entity that initiated SA establishment, SA modify or SA delete.
- Responder Cookie (16 octets) - Cookie of entity that is responding to SA establishment, SA modify or SA delete request
- Payload (variable) - The format of the payload is based on the message type and defined in sections 2.3 through 2.6.

2.1.1 General Message Processing

Every ISAKMP message has basic processing applied to insure protocol reliability and minimize threats, such as denial of service and replay attacks.

When issuing an ISAKMP packet:

1. Sets a timer and initializes a retry counter
2. If timer expires the message is resent and the retry counter decremented.
3. If the retry counter reaches zero (0), the event is logged in the appropriate system file.
4. Clears all state and return to IDLE.

When an ISAKMP packet is received:

1. Verify the appropriate "cookie".
2. Check exchange type and message fields to confirm they are valid types.
3. Check SPI to ensure it is valid for the exchange being preformed.
4. If any of these fields fails its check, the message is discarded.

   Log Event in the appropriate system file.

   No response is sent to the message orginator.
5. If all fields pass the checks, the message payload is processed.

   Individual message processing (described in sections 2.3 through 2.6) may result in the message being invalidated, in which case:

   Log Event in the appropriate system file.

   No response is sent to the message orginator.

   A valid message results in a response being sent to the message orginator.

2.2 ISAKMP Details

2.2.1 Security Association Attributes

A Security Association (SA) is a relationship between two enities that describes how they will utilize security services. This relationship is represented by a collection of security related information. The SA Attributes are the individual elements that comprise all security relevant information necessary to form the SA.

The following syntax encodes the security attributes to be exchanged by ISAKMP. This syntax is used in the ISA_INIT_REQ, ISA_INIT_RESP, ISA_NEG_REQ, ISA_NEG_RESP, ISA_MOD_REQ, and ISA_MOD_RESP messages. The syntax groups security attributes needed to perform a security function into an SA set or SA list format. The set format must be supported by ISAKMP implementations. The list format is an optional format.

The set format groups all necessary attributes together. Each set has a unique identifier (Set Number), supported security service (Supports), such as IP AH, IP ESP, OSPF authentication, and a list of Attribute Classes and Attribute Types. The Attribute Class is the broad category of Attribute Type, such as encryption algorithms. Attribute Type is a specific attribute identifier. DES is an example of an attribute type for the encryption algorithm attribute class. A set has only one instance of an Attribute Class and one type for that class. This syntax maintains flexibility by allowing many different (and some still undefined) types of SA attributes to be exchanged.

The figure below depicts the syntax for exchanging security attributes using the set format. It shows a single set from which multiple sets would be grouped for a specific message type.

```
                     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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Set Number   |   Supports    |         Number of Attr        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Attribute Class          |         Attribute Type        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                                                               ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Attribute Class          |         Attribute Type        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Generic Set Exchange Format

- Set number (1 octet) - Unique identifier for each proposed set
- Supports (1 octet) - Security service proposed set supports. Examples are IP AH, IP ESP, and OSPF authentication
- Number of Attributes (2 octets) - Number of attributes contained in the proposed set
- Attribute Class (2 octets) - examples are Encryption Algorithms, Key Exchange Algorithms
- Attribute Type (2 octets) - examples of attribute type for the encryption algorithms attribute class are DES, Triple DES, and IDEA.

The size of a set formatted exchange is 4 octets + (Number of Attrs * 4 octets). Computing the size of a particular set allows determining the beginning of the next set without completely parsing the current set, should it not be an acceptable SA set.

The SA list format presents several attribute types for an attribute class. Each type within the class is then ordered to indicate its precedence. Specifically, the first attribute type is the highest priority type, followed by other choices. Each subsequent choice are listed in descending priority order. An attribute type must be chosen from each attribute class to establish a complete SA.

The figure below shows the sytax for the optional list exchange format. Note that multiple attribute types appear within an attribute class. The number of types is determined from the Count field.

```
                     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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Attribute Class              |   Count                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Attribute  Type              |   Attribute Type              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                                                               ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Attribute  Type              |   Attribute Type              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Generic List Exchange Format

- Attribute Class (2 octets) - Examples are Encryption Algorithms, Key Exchange Algorithms
- Count - Number of proposed types for a class
- Attribute Type (2 octets) - Presented in priority order

Appendix B presents an outline containing a more comprehensive set of SA attributes. These sets of attributes are initial definitions and are presented to stimulate thought and discussion on SAs. 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 Internet Key Management Protocol (IKMP) RFC and vice versa. An SA container object and SA attribute definitions should become part of the Management Information Base (MIB), see [RFC-1213], in a separate protected section called the Security MIB. IKMP should emulate the SNMP concept of separate RFCs for the protocol and the information managed. SA attribute identifiers MAY be defined using the syntax in [RFC-1155] and [RFC-1212].

2.2.2 Transport Protocol

The User Datagram Protocol (UDP) is the transport protocol for ISAKMP. UDP checksumming discards UDP packets with an incorrect or zero (0) checksum. ISAKMP is unaware of the discard, but will resend the packet when its resend timer expires.

2.2.3 RESERVED Fields

All RESERVED fields in the ISAKMP protocol MUST be set to zero (0) when a packet is issued. The receiver SHOULD check the RESERVED fields for zero (0) and discard the packet if other values are found.

2.3 Security Association Establishment

SA Establishment is the process of agreeing upon and exchanging all the security information that is required in an SA. The following sections, 2.3.1 to 2.3.3, describe the three basic phases, – SA Initialization, key and Authentication information exchange, and SA Negotiation –, that comprise SA Establishment.

2.3.1 Security Association Initialization

The initialization exchange of SA establishment is composed of the ISA_INIT_REQ and ISA_INIT_RESP packets. The ISA_INIT packets exchange "cookies", and options for a key generation technique, an encryption algorithm and an authentication mechanism. The "cookies" are used to prevent replay and denial of service attacks. Authentication and encryption for the ISAKMP exchanges is provided by the authentication mechanism and encryption algorithm selected. The key generation technique selected creates keys for use by the authentication mechanism and encryption algorithm. The keys may also be used either as the session keys, to create the session keys, or protect the exchange of the actual session keys for the SA. If the key, encryption algorithm, and authentication mechanism are only used to protect ISAKMP exchanges then new options can be picked during the negotiation phase (described in Section 2.3.3) for use in protecting the actual data communications. If encryption is not required for the SA the encryption algorithm options need not be exchanged.

```
                     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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                         ISAKMP Header                         ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| SA Syntax Type|    SA Flags   |  Num of Sets  |    RESERVED   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                       SA Attribute Set #1                     ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                       SA Attribute Set #2                     ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                              ...                              ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                       SA Attribute Set #N                     ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

ISA_INIT_REQ and ISA_INIT_RESP Packet Format

- SAKMP Header - Described in Section 2.1

- SA Syntax Type (1 octet) - Presentation format of SAs

SA Syntax	SA Syntax Type
Set	1
List	2

- SA Flags (1 octet) - Flags specific to an SA service. The SA Flags field is zero (0) for the ISA_INIT messages.
- Number of Sets (1 octet) - Number of SA Attribute Sets being proposed
- SA Attributes (variable) - A list of SA Attributes. SA Attribute specifications are discussed in Section 2.2.1.

Initialization Procedures

When issuing an ISA_INIT_REQ message:

1. Create initiator cookie. See Section 1.5.1 for details.
2. Generate a unique pseudo-random SPI for future communications with the initiating host.
3. Construct an ISA_INIT_REQ packet.
4. Send the packet to the destination host as described in Section 2.1.1.

When an ISA_INIT_REQ message is received:

1. Check the ISAKMP header as described in Section 2.1.1.
2. Unpack ISA_INIT_REQ payload and determine the highest priority attribute set supported.
3. Create responder cookie.
4. Create a unique pseudo-random SPI for future communications with the responding host.
5. Construct an ISA_INIT_RESP packet.
6. Send the packet to the initiating host as described in Section 2.1.1.

When an ISA_INIT_RESP message is received:

1. Check the ISAKMP header as described in Section 2.1.1.
2. Unpack ISA_INIT_RESP payload.
3. Determine if attribute set selected is valid. If attribute set is invalid or responder rejected all proposed attribute sets:

   Log Event in the appropriate system file.

   Clear all state and return to IDLE.
4. Configure protocol machine based on attribute set selected.
5. Transition to Key and Authentication Phase.

2.3.2 Key and Authentication Phase

The authentication and key exchange phase exchange information required to confirm the identities of the parties wishing to establish the SA and establish a session key for use during the SA establishment. Based on user preferences the key may be used during data communications or a new one may be created/exchanged during the negotiation phase, described in section 2.3.3, for use in protecting the actual data communications.

The authentication and key payloads are general formats which support many types of key exchange and authentication. The detailed specification of these fields are specified in companion RFCs. These companion RFCs will define the standard authentication and key exchange mechanisms that need to be implemented and assure compliance with this specification.

The packets that carry the authentication and key exchange payloads have the format shown below. When the ISA_AUTH&KE_REQ and ISA_AUTH&KE_RESP packets are used the authentication payload SHOULD be processed first to strongly authenticate the packet issuer, before the key generation processing is executed. In the ISA_AUTH_REQ and ISA_AUTH_RESP packets the key exchange payload is not present. In the ISA_KE_REQ and ISA_KE_RESP packets the authentication payload is not present.

```
                     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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                        ISAKMP Header                          ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                                                               ~
|                   Authentication Payload                      |
~                                                               ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                                                               ~
|                    Key Exchange Payload                       |
~                                                               ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

ISA_AUTH&KE_REQ and ISA_AUTH&KE_RESP Packet Format

Strong Authentication Details

This section specifies the encoding of the authentication payload for the ISA_AUTH_REQ, ISA_AUTH_RESP, ISA_AUTH&KE_REQ, and ISA_AUTH&KE_RESP messages.

```
                     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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Authentication Authority     |           Reserved            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Authentication Type          |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                                                               ~
|                     Authentication Data                       |
~                                                               ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Authentication Payload Format

- Authentication Authority (2 octets) - Indentifies the party that generated the certificates used for authentication. Authorities must be assigned an identifier by the Internet Assigned Numbers Authority (IANA). Before being assigned an identifier, an authority must publish an RFC defining the authority's domain.

  If PGP certificates, based on the "web of trust", are carried in the authentication payload the Authentication Authority value is one (1).

  Examples certificate authorities that would have to register for an identifier are:

  - RSA Commercial Certificate Authority (https://www_csc.rsa.com/netsite)
  - Stable Large E-mail Database (SLED) (http://www.four11.com)
  - U.S. Postal Service.

- Authentication Type (2 octets) - Indicates the authentication payload format. This field is used by authentication authorities that support more than one certificate type. The authentication types supported by an authentication authority must be defined in the RFC required for authentication authority registration. Examples are:

  - RSA certificates
  - PGP certificates
  - DSS certificates
  - DNS Signed Keys
  - Kerberos Tokens
  - X.509 certificates

- Length (2 octets) - Length of the Authentication Data field in octets.

- Authentication Data (variable) - Actual authentication data. Type of certificate is indicated by the Authentication type field.

Key Exchange Details

A variety of key exchanges can be supported in the key exchange phase. 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 key exchange payload is dependent on the specific key exchange and therefore is not specified in this Internet draft. There can be both public and private key generation techniques. Both types must register with IANA to obtain a Key Exchange Identifier (KEI). Before public key exchanges can obtain a KEI, an RFC defining the key exchange payload format and key generation procedures MUST be submitted. Private key exchanges are not REQUIRED to provide an RFC when registering for a KEI.

Example key exchange payload encodings are shown in Appendix A.

```
                     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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 KEI           |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                                                               ~
|                     Key Exchange Payload                      |
~                                                               ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Key Exchange Payload Format

KEI (2 octets) - Key Exchange Identifier

Length (2 octets) - Length of payload in octets

Key Exchange Payload (variable) - Data (i.e. public values) required to create session key.

ISA_AUTH&KE Procedures

When issuing an ISA_AUTH&KE_REQ packet:

1. Generate an authentication signature using the authentication attributes and options selected in initialization phase.
2. Create key exchange payload based on KEI.
3. Construct an ISA_AUTH&KE_REQ packet.
4. Send the packet to the responding host as described in Section 2.1.1.

When an ISA_AUTH&KE_REQ packet is received:

1. Check the ISAKMP header as described in Section 2.1.1.
2. Unpack ISA_AUTH&KE_REQ packet.
3. Verify Initiator's signature.

   If verification fails

    Log Event in the appropriate system file.

    Terminate with error.

   ELSE

    Discard packet.

    Log Event in the appropriate system file.

    RETURN to WAIT for ISA_AUTH&KE_REQ state.
4. Generate an authentication signature, to authenticate responder to initiator, using the authentication attributes and options selected.
5. Create key exchange payload for initiator based on KEI.
6. Construct an ISA_AUTH&KE_RESP packet.
7. Send the packet to the initiating host as described in Section 2.1.1.
8. Begin key calculation in the background.

When an ISA_AUTH&KE_RESP message is received:

1. Check the ISAKMP header as described in Section 2.1.1.
2. Verify Responder's signature.

   If verification fails, either:

    Log Event in the appropriate system file.

    Terminate with error.

   ELSE

    Discard packet.

    Log Event in the appropriate system file.

    RETURN to WAIT for ISA_AUTH&KE_RESP state.
3. Calculate key.
4. Transition to Security Association Negotiation Phase.

2.3.3 Security Association Negotiation Phase

The SA Negotiation phase allows the initiating entity to present SA attributes, that it wishes to use for secure communications, to a responding entity. These SA attributes may included additional options for the attributes agreed upon during the initialization phase, as well as selection of the additional attributes required for an SA. The REQUIRED and RECOMMENDED SA parameters for the IP AH and IP ESP security mechanisms are cited in the Security Architecture for the Internet Protocol [Atki95].

```
                     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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                          ISAKMP Header                        ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| SA Syntax Type|  Num of Sets  |    SA Flags   |    RESERVED   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                        SA Attribute Set #1                    ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                        SA Attribute Set #2                    ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                               ...                             ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                        SA Attribute Set #N                    ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

ISA_NEG_REQ and ISA_NEG_RESP Packet Format

- SA Msg Type (1 octet) - Defined in Section 2.3.1.
- Num of Sets (1 octet) - Number of Attribute Sets being proposed
- SA Flags (1 octet) - Flags specific to an SA service. See Section 2.3.3 for flag settings in the ISA_NEG messages.
- SA Attributes (variable) - A list of SA attributes. SA Attribute specifications are discussed in section 2.2.1.

SA Negotiation Procedures

When issuing an ISA_NEG_REQ packet:

1. Determine SA attributes to be negotiated. This may include changing or confirming the attributes from the SA initialization phase.
2. Encrypt and/or sign ISA_NEG_REQ payload only (not header).
3. Construct an ISA_NEG_REQ packet.
4. Send the packet to the responding host as described in Section 2.1.1.

When an ISA_NEG_REQ packet is received:

1. Check the ISAKMP header as described in Section 2.1.1.
2. Decrypt ISA_NEG_REQ payload and verify signature.

3. Unpack ISA_NEG_REQ packet payload and determine the highest priority SA attributes supported.

   If none of the SA attribute options are supported, the ISA_NEG_RESP will have the value zero (0) in the Number of Sets field and an SA will not be established.
4. If the SA negotiation is requesting a key change or new authentication mechanism, then, generate appropriate information and include it as an attribute/option in the ISA_NEG_RESP payload.
5. Encrypt and/or sign ISA_NEG_RESP payload only (not header).
6. Construct an ISA_NEG_RESP packet.
7. Send the packet to the initiating host as described in Section 2.1.1.
8. If required, begin calculation of the new session key in the background.
9. Transition to SA Negotation Conclusion.

When an ISA_NEG_RESP message is received:

1. Check the ISAKMP header as described in Section 2.1.1.
2. Decrypt ISA_NEG_RESP payload and verify signature.
3. Unpack ISA_NEG_RESP payload and verify the SA attributes selected by responder are valid.

   If response is invalid or responder rejected all proposed SA Attributes:

   Log Event in the appropriate system file.

   Clear all state and return to an IDLE.
4. If required, begin calculation of the new session key in the background.
5. Transition to SA Negotiation Conclusion.

SA Negotiation Conclusion

SA Commit Message The SA Commit message allows the initiating entity to inform the responding party that it has completed the processing required to set-up the SA and therefore, secure communications may begin.

The Least Significant Bit (LSB) in the SA Flags field is set to one (1) in the ISA_NEG packet if an ISA_COMMIT packet is issued and zero (0) if the ISA_COMMIT packet is not issued. All other bits in the SA Flags field are zero (0).

The SA Flags field may be set by the entity that initiated the negotiation to indicate that the ISA_COMMIT packet will follow the negotiation exchange. If the initiating entity sets the flag the responding entity can 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 ISA_COMMIT packet. If neither entity sets the flag then the ISA_COMMIT packet will not be issued.

The ISA_COMMIT packet is the ISAKMP header with no payload.

The SPI is set to the Responder SPI.

Transmiting ISA_COMMIT packet is optional and determined by the policy of the parties establishing the SA. All implementations MUST be able to generate, transmit, and receive this message.

SA_Negotiation Conclusion Procedures

1. Both initiator and responder place SA in appropriate database for the security service it supports.
2. Based on the SA Flags field, the initiator constructs an ISA_COMMIT packet.
3. Initiator sends the packet to the responding host as described in Section 2.1.1.
4. When responder received ISA_COMMIT packet it checks the ISAKMP header as described in Section 2.1.1.
5. Clear all state and return to IDLE.

2.3.4 Packet Exchanges

The "Exchange" field in the ISAKMP header currently has two values defined, the base exchange (BASE) and the anonomous exchange (ANON). These exchanges define the flow of the ISAKMP packets during SA establishment. The diagrams in 2.3.4 and 2.3.4 shows the packet exchange ordering for each exchange type and provides basic notes describing what has happened after each packet exchange.

Base Exchange

The previous sections, 2.3.1 to 2.3.3, described the three basic phases, – SA Initialization, key and authentication information exchange, and SA negotiation –, that comprise the BASE exchange. The base exchange contains the minimum number of packet exchanges in order to reduce latency associated with SA establishment.

Base Exchange

	Initiator	—	Responder	Note
	ISA_INIT_REQ	=>		
		<=	ISA_INIT_RESP	
				Basic SA selected
	ISA_AUTH&KE_REQ	=>		
		<=	ISA_AUTH&KE_RESP	
				Identity Verified Key Generated Encryption Begun
	ISA_NEG_REQ	=>		
		<=	ISA_NEG_RESP	SA Completed
(optional)	ISA_COMMIT	=>		

Identity Protection SA Establishment Variation

The identity protect exchange starts and ends the same as the base exchange, but separates the key exchange payload and the authentication payloads into separate packets. In this exchange the key exchange is transmited first followed by the strong authentication exchange. The benefit of this exchange is the ability to communicate with a person without disclosing either party's identity to passive attackers on the network.

The ISA_KE_REQ and ISA_KE_RESP packets used for the key exchange in this variation contain an ISAKMP header followed by the key exchange payload. The ISA_AUTH_REQ and ISA_AUTH_RESP packet used for the authentication exchange in this variation contain an ISAKMP header followed by the authentication payload.

Identity Protection Exchange

	Initiator		Responder	Note
	ISA_INIT_REQ	=>		
		<=	ISA_INIT_RESP	
				Basic SA selected
	ISA_KE_REQ	=>		
		<=	ISA_KE_RESP	
				Key Generated Encryption Begun
	ISA_AUTH_REQ	=>		
		<=	ISA_AUTH_RESP	
				Identity Verified
	ISA_NEG_REQ	=>		
		<=	ISA_NEG_RESP	SA Completed
(optional)	ISA_COMMIT	=>		

2.4 Security Association Modification

SA modification provides the ability to update attributes and parameters within an existing SA. The most common use of this exchange will be to re-key an SA.

```
                     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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                         ISAKMP Header                         ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| SA Syntax Type|  Num of Sets  |    SA Flags   |    RESERVED   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                       SA Attribute Set #1                     ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                       SA Attribute Set #2                     ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                              ...                              ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                       SA Attribute Set #N                     ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

ISA_MODIFY_REQ and ISA_MODIFY_RESP Packet Format

- SA Syntax Type (1 octet) - Defined in Section 2.3.1.
- Num of Sets (1 octet) - Number of Attribute Sets being modified.

- SA Flags (1 octet) - Flags specific to an SA service. Currently the SA Flags field is set to zero (0) for the ISA_MODIFY packets.
- SA Attributes (variable) - A list of SA attributes. SA Attributes field contains only those attributes being updated. SA Attribute specifications are discussed in section 2.2.1.

The procedure for exchanging information to modify an SA are similiar to the SA negotiation exchange. The details of SA modification will be described in this section as they are solidified during prototype development.

2.5 Security Association Deletion

The ISA_DELETE packet provide a controlled method of informing a peer entity that the initiating entity has deleted an SA(s). The ISA_DELETE packet provides for the deletion of any number of SAs. The receiving entity SHOULD clean up its SA database. The receiving entity may be using the SA for secure communications with more than one party and would not want to actually delete the SA from it's database, however, upon receipt of an ISA_DELETE packet the SAs in the packet can not be used with the initiating entity. The SA Establishment must be repeated to resume secure communications.

```
                     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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                          ISAKMP Header                        ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           SPI Count           |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                                                               ~
|                              SPIs                             |
~                                                               ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

SA Delete Payload Format

- SPI Count - Number of security associations to be deleted
- Length - length of payload in octets
- SPIs - Initiator's Security Parameter Index(s) to be deleted

Deletion Procedures

When issuing an ISA_DELETE message:

1. Create initiator cookie. See Section 1.5.1 for details.

2. Determine SPI of receiving entity.
3. Construct ISA_DELETE packet.
4. Send the packet to the destination host as described in Section 2.1.1.

Upon receipt of an ISA_DELETE message:

1. Check the ISAKMP header as described in Section 2.1.1.
2. Unpack ISA_DELETE payload.

2.6 Notification Message

ISAKMP ISA_NOTIFY packet contains information one party wants to send to another. Notification information can be error messages specifying why a SA could not be established. It can also be status data that a process managing an SA database, such would be required on a security gateway, wishes to communicate with a peer process. The ISA_NOTIFY packet is unidirectional.

```
                     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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                         ISAKMP Header                         ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Notify Message Type      |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                                                               ~
|                         Notify Payload                        |
~                                                               ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

SA NOTIFY Payload Format

- Notify Message Type (2 octets)

Notification	Notify Message Type
Error	1
Status	2

- Length (2 octets) - length of payload in octets
- Notify Payload (variable) - Value dependent on the Notify Message Type

3 Conclusions

The ISAKMP provides the flexibility needed to support multiple key exchange techniques, encryption algorithms, authentication mechanisms, security services, and security attributes. These item may be publicly or privately defined. The added benefit of supporting multiple techniques is that as new techniques are developed they can easily be added to the protocol. This provided a path for the growth of Internet security services.

A Key Exchange Examples

Two key exchanges examples are presented to help illustrate the ISAKMP's ability to support multiple key exchanges.

A.1 Photuris KE

Based on [Karn95] an example of how the Photuris Key Exchange could be accomplished in ISAKMP is presented.

1. Photuris "groups", K-Transform, and S-Transform would be exchanged in the ISA_INIT packets.
2. The following Photuris fields would be in the ISA_KE packets.

ISAKMP Packet	Value
ISA_KE_REQ	Initiator-Public-Value
ISA_KE_RESP	Responder-Public-Value

3. The following Photuris fields would be in the ISA_AUTH packets.

ISAKMP Packet	Value
ISA_AUTH_REQ	Signature [Initiator]
ISA_AUTH_REQ	Certificate [Initiator]
ISA_AUTH_RESP	Signature [Responder]
ISA_AUTH_RESP	Certificate [Resonder]

4. The session key would be created as described in [Karn95] after each key exchange payload is received.
5. Finally the Transforms, I-Transform and Parameters, R-Transform and Parameters, and Lifetime would be exchanged in the ISA_NEG packets.

A.2 FORTEZZA Key Exchange Algorithm (KEA)

One of the benefits of ISAKMP is that it is not limited to one key exchange. An example of how the FORTEZZA KEA is accomplished in ISAKMP is now presented.

1. Options for Encryption algorithm, Authentication Authority and Key Exchange Algorithm would be exchanged in the ISA_INIT packets.
2. The following FORTEZZA fields would be in the ISA_AUTH&KE packets.

Packet Payload	FORTEZZA Value
Authentication Payload	Signed Information [Initiator]
Authentication Payload	FORTEZZA Certificate [Initiator]
Authentication Payload	Signed Information [Responder]
Authentication Payload	FORTEZZA Certificate [Resonder]
Key Exchange Payload	Message Encryption Key encrypted in Token Encryption Key
Key Exchange Payload	Initiator-Random-Value
Key Exchange Payload	Responder-Random-Value

3. The Token Encryption Key is generated.
4. Message Encryption Key is decrypted.
5. Additional Fortezza attributes would be exchanged in the ISA_NEG packets.

Another benefit of ISAKMP is that classified key exchanges, such as the FORTEZZA KEA, can be performed using a public KMP without revealing the algorithm. This is an important Department of Defense requirement.

B Security Association Attributes

This appendix is based upon an e-mail message [Kent94] to the IPSEC mail list from Steve Kent and is reproduced here to start a discussion on SA attributes. The authors welcome input on what are meaningful security attributes for an SA.

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.

1. SAID.INBOUND
2. SAID.OUTBOUND
3. ENCAPSULATION
4. INBOUND-CRITERIA
   (a) IP-DESTINATION-ADDRESS
   (b) IP-SOURCE-ADDRESS
   (c) NEXT-PROTOCOL
   (d) IP-SECURITY-LABEL
   (e) TRANSPORT-DESTINATION-PORT
   (f) TRANSPORT-SOURCE-PORT
5. PEER-ADDRESS
6. ENCRYPTION
   (a) ENABLED
   (b) ALGORTIHM
   (c) KEY.INBOUND
   (d) KEY.OUTBOUND
   (e) IV.INBOUND
   (f) IV.OUTBOUND
7. INTEGRITY
   (a) ENABLED
   (b) PLAINTEXT
   (c) DIRECTION.ENABLED
   (d) DIRECTION.VALUE
   (e) ALGORITHM
   (f) KEY.OUTBOUND
   (g) KEY.INBOUND
8. COMPRESSION
   (a) ENABLED

   (b) ALGORITHM

9. REPLAY
   (a) ENABLED
   (b) SIZE
   (c) NUMBER.OUTBOUND
   (d) NUMBER.INBOUND
   (e) WINDOW.SIZE
   (f) WINDOW

10. FRAGMENTATION
   (a) INBOUND
   (b) OUTBOUND

11. KEY-MANAGEMENT
   (a) NEGOTIATED
   (b) TECHNIQUE
   (c) PARAMETERS
   (d) REKEY
      i. GRACE
      ii. NEXT-SA
      iii. TIME-BASED
         A. ENABLE
         B. TRIGGER
      iv. TRAFFIC-BASED
         A. ENABLE
         B. PACKET-COUNT.INBOUND
         C. PACKET-COUNT.OUTBOUND
         D. TRIGGER.INBOUND
         E. TRIGGER.OUTBOUND

Security Considerations

Cryptographic analysis techniques are improving at a steady pace. The continuing improvement in processing power makes once computational prohibitive cryptographic attacks more realistic. New cryptographic algorithms and public key generation techniques are also being developed at a steady pace. New security services and mechanisms are being developed at an accelerated pace. A consistent method of choosing from a variety of security services and mechanisms and to exchange attributes required by the mechanisms is important to security in the complex structure of the Internet. However a system that locks itself into a single cryptographic algorithm, key exchange technique, or security mechanism will become increasingly vulnerable as time passes.

UDP is an unreliable datagram protocol and therefore its use in ISAKMP introduces a number of security considerations. Since UDP is unreliable, but a key management protocol must be reliable, the reliability is built into ISAKMP. While ISAKMP utilizes UDP as its transport mechanism, it doesn't soley rely on any UDP information (e.g. checksum, length) for its processing.

Another issue that must be considered in the development of IKMP is the effect of firewalls on the protocol. Many firewalls filter out all UDP packets, making reliance on UDP questionable in certian environments.

A number of very important security considerations are presented in [Atki95]. One bares repeating. Once a private session key is created it must be safely stored. Failure to properly protect the private key from access both internal and external to the system completely nullifies any protect provided by the IP Security services.

Acknowledgements

Marsha Gross, Mike Oehler, Mark Schneider, and Pete Sell provided significant input and review to this document.

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.

[Atki95] Randell Atkinson, *Security Architecture for the Internet Protocol*, Internet-Draft, working in progress, 8 May, 1995.

[EK94] Eastlake III, D. and c. Kaufman, *Domain Name System Protocol Security Extensions*, Internet-Draft, working in progress, March, 1994.

[Karn95] Karn P. and B. Simpson, *The Photuris Key Management Protocol*, Internet-Draft, working in progress, March, 1995.

[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`