                   The Internet Key Exchange (IKE)

Status of this Memo

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

Table Of Contents

1. Abstract

   ISAKMP ([MSST98]) provides a framework for authentication and key
   exchange but does not define them. ISAKMP is designed to be key
   exchange independant; that is, it is designed to support many
   different key exchanges.

   Oakley ([Orm96]) describes a series of key exchanges-- called
   "modes"-- and details the services provided by each (e.g. perfect
   forward secrecy for keys, identity protection, and authentication).

   SKEME ([SKEME]) describes a versatile key exchange technique which
   provides anonymity, repudiability, and quick key refreshment.

   This document describes a protocol using part of Oakley and part of
   SKEME in conjunction with ISAKMP to obtain authenticated keying
   material for use with ISAKMP, and for other security associations
   such as AH and ESP for the IETF IPsec DOI.

2. Discussion

   This memo describes a hybrid protocol. The purpose is to negotiate,
   and provide authenticated keying material for, security associations
   in a protected manner.

   Processes which implement this memo can be used for negotiating
   virtual private networks (VPNs) and also for providing a remote user
   from a remote site (whose IP address need not be known beforehand)
   access to a secure host or network.

   Client negotiation is supported. Client mode is where the
   negotiating parties are not the endpoints for which security
   association negotiation is taking place. When used in client mode,
   the identities of the end parties remain hidden.

This does not implement the entire Oakley protocol, but only a subset necessary to satisfy its goals. It does not claim conformance or compliance with the entire Oakley protocol nor is it dependant in any way on the Oakley protocol.

Likewise, this does not implement the entire SKEME protocol, but only the method of public key encryption for authentication and its concept of fast re-keying using an exchange of nonces. This protocol is not dependant in any way on the SKEME protocol.

3. Terms and Definitions

3.1 Requirements Terminology

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

3.2 Notation

The following notation is used throughout this memo.

HDR is an ISAKMP header whose exchange type is the mode. When writen as HDR* it indicates payload encryption.

SA is an SA negotiation payload with one or more proposals. An initiator MAY provide multiple proposals for negotiation; a responder MUST reply with only one.

<P>_b indicates the body of payload <P>-- the ISAKMP generic vpayload is not included.

SAi_b is the entire body of the SA payload (minus the ISAKMP generic header)-- i.e. the DOI, situation, all proposals and all transforms offered by the Initiator.

CKY-I and CKY-R are the Initiator's cookie and the Responder's cookie, respectively, from the ISAKMP header.

g^xi and g^xr are the Diffie-Hellman ([DH]) public values of the initiator and responder respectively.

g^xy is the Diffie-Hellman shared secret.

KE is the key exchange payload which contains the public information exchanged in a Diffie-Hellman exchange. There is no particular encoding (e.g. a TLV) used for the data of a KE payload.

Nx is the nonce payload; x can be: i or r for the ISAKMP initiator
and responder respectively.

IDx is the identification payload for "x". x can be: "ii" or "ir"
for the ISAKMP initiator and responder respectively during phase
one negotiation; or "ui" or "ur" for the user initiator and
responder respectively during phase two. The ID payload format for
the Internet DOI is defined in [Pip97].

SIG is the signature payload. The data to sign is exchange-
specific.

CERT is the certificate payload.

HASH (and any derivitive such as HASH(2) or HASH_I) is the hash
payload. The contents of the hash are specific to the
authentication method.

prf(key, msg) is the keyed pseudo-random function-- often a keyed
hash function-- used to generate a deterministic output that
appears pseudo-random. prf's are used both for key derivations and
for authentication (i.e. as a keyed MAC). (See [KBC96]).

SKEYID is a string derived from secret material known only to the
active players in the exchange.

SKEYID_e is the keying material used by the ISAKMP SA to protect
the confidentiality of its messages.

SKEYID_a is the keying material used by the ISAKMP SA to
authenticate its messages.

SKEYID_d is the keying material used to derive keys for non-ISAKMP
security associations.

<x>y indicates that "x" is encrypted with the key "y".

--> signifies "initiator to responder" communication (requests).

<-- signifies "responder to initiator" communication (replies).

 | signifies concatenation of information-- e.g. X | Y is the
concatentation of X with Y.

[x] indicates that x is optional.

Message encryption (when noted by a '*' after the ISAKMP header) MUST begin immediately after the ISAKMP header. When communication is protected, all payloads following the ISAKMP header MUST be encrypted. Encryption keys are generated from SKEYID_e in a manner that is defined for each algorithm.

3.3 Perfect Forward Secrecy

When used in the memo Perfect Forward Secrecy (PFS) refers to the notion that compromise of a single key will permit access to only data protected by a single key. For PFS to exist the key used to protect transmission of data MUST NOT be used to derive any additional keys, and if the key used to protect transmission of data was derived from some other keying material, that material MUST NOT be used to derive any more keys.

Perfect Forward Secrecy for both keys and identities is provided in this protocol. (Sections 5.5 and 8).

3.4 Security Association

A security association (SA) is a set of policy and key(s) used to protect information. The ISAKMP SA is the shared policy and key(s) used by the negotiating peers in this protocol to protect their communication.

4. Introduction

Oakley and SKEME each define a method to establish an authenticated key exchange. This includes payloads construction, the information payloads carry, the order in which they are processed and how they are used.

While Oakley defines "modes", ISAKMP defines "phases". The relationship between the two is very straightforward and IKE presents different exchanges as modes which operate in one of two phases.

Phase 1 is where the two ISAKMP peers establish a secure, authenticated channel with which to communicate. This is called the ISAKMP Security Association (SA). "Main Mode" and "Aggressive Mode" each accomplish a phase 1 exchange. "Main Mode" and "Aggressive Mode" MUST ONLY be used in phase 1.

Phase 2 is where Security Associations are negotiated on behalf of services such as IPsec or any other service which needs key material and/or parameter negotiation. "Quick Mode" accomplishes a phase 2 exchange. "Quick Mode" MUST ONLY be used in phase 2.

"New Group Mode" is not really a phase 1 or phase 2. It follows
phase 1, but serves to establish a new group which can be used in
future negotiations. "New Group Mode" MUST ONLY be used after phase
1.

The ISAKMP SA is bi-directional. That is, once established, either
party may initiate Quick Mode, Informational, and New Group Mode
Exchanges. Per the base ISAKMP document, the ISAKMP SA is identified
by the Initiator's cookie followed by the Responder's cookie-- the
role of each party in the phase 1 exchange dictates which cookie is
the Initiator's. The cookie order established by the phase 1 exchange
continues to identify the ISAKMP SA regardless of the direction the
Quick Mode, Informational, or New Group exchange. In other words, the
cookies MUST NOT swap places when the direction of the ISAKMP SA
changes.

With the use of ISAKMP phases, an implementation can accomplish very
fast keying when necessary. A single phase 1 negotiation may be used
for more than one phase 2 negotiation. Additionally a single phase 2
negotiation can request multiple Security Associations. With these
optimizations, an implementation can see less than one round trip per
SA as well as less than one DH exponentiation per SA. "Main Mode"
for phase 1 provides identity protection. When identity protection
is not needed, "Aggressive Mode" can be used to reduce round trips
even further. Developer hints for doing these optimizations are
included below. It should also be noted that using public key
encryption to authenticate an Aggressive Mode exchange will still
provide identity protection.

This protocol does not define its own DOI per se. The ISAKMP SA,
established in phase 1, MAY use the DOI and situation from a non-
ISAKMP service (such as the IETF IPSec DOI [Pip97]). In this case an
implementation MAY choose to restrict use of the ISAKMP SA for
establishment of SAs for services of the same DOI. Alternately, an
ISAKMP SA MAY be established with the value zero in both the DOI and
situation (see [MSST98] for a description of these fields) and in
this case implementations will be free to establish security services
for any defined DOI using this ISAKMP SA. If a DOI of zero is used
for establishment of a phase 1 SA, the syntax of the identity
payloads used in phase 1 is that defined in [MSST98] and not from any
DOI-- e.g. [Pip97]-- which may further expand the syntax and
semantics of identities.

The following attributes are used by IKE and are negotiated as part
of the ISAKMP Security Association. (These attributes pertain only
to the ISAKMP Security Association and not to any Security
Associations that ISAKMP may be negotiating on behalf of other
services.)

     - encryption algorithm

     - hash algorithm

     - authentication method

     - information about a group over which to do Diffie-Hellman.

   All of these attributes are mandatory and MUST be negotiated. In
   addition, it is possible to optionally negotiate a psuedo-random
   function ("prf"). (There are currently no negotiable pseudo-random
   functions defined in this document. Private use attribute values can
   be used for prf negotiation between consenting parties). If a "prf"
   is not negotiation, the HMAC (see [KBC96]) version of the negotiated
   hash algorithm is used as a pseudo-random function. Other non-
   mandatory attributes are described in Appendix A. The selected hash
   algorithm MUST support both native and HMAC modes.

   The Diffie-Hellman group MUST be either specified using a defined
   group description (section 6) or by defining all attributes of a
   group (section 5.6). Group attributes (such as group type or prime--
   see Appendix A) MUST NOT be offered in conjunction with a previously
   defined group (either a reserved group description or a private use
   description that is established after conclusion of a New Group Mode
   exchange).

   IKE implementations MUST support the following attribute values:

     - DES [DES] in CBC mode with a weak, and semi-weak, key check
     (weak and semi-weak keys are referenced in [Sch96] and listed in
     Appendix A). The key is derived according to Appendix B.

     - MD5 [MD5] and SHA [SHA}.

     - Authentication via pre-shared keys.

     - MODP over default group number one (see below).

   In addition, IKE implementations SHOULD support: 3DES for encryption;
   Tiger ([TIGER]) for hash; the Digital Signature Standard, RSA [RSA]
   signatures and authentication with RSA public key encryption; and
   MODP group number 2. IKE implementations MAY support any additional
   encryption algorithms defined in Appendix A and MAY support ECP and
   EC2N groups.

   The IKE modes described here MUST be implemented whenever the IETF
   IPsec DOI [Pip97] is implemented. Other DOIs MAY use the modes
   described here.

5. Exchanges

   There are two basic methods used to establish an authenticated key
   exchange: Main Mode and Aggressive Mode. Each generates authenticated
   keying material from an ephemeral Diffie-Hellman exchange. Main Mode
   MUST be implemented; Aggressive Mode SHOULD be implemented. In
   addition, Quick Mode MUST be implemented as a mechanism to generate
   fresh keying material and negotiate non-ISAKMP security services. In
   addition, New Group Mode SHOULD be implemented as a mechanism to
   define private groups for Diffie-Hellman exchanges. Implementations
   MUST NOT switch exchange types in the middle of an exchange.

   Exchanges conform to standard ISAKMP payload syntax, attribute
   encoding, timeouts and retransmits of messages, and informational
   messages-- e.g a notify response is sent when, for example, a
   proposal is unacceptable, or a signature verification or decryption
   was unsuccessful, etc.

   The SA payload MUST precede all other payloads in a phase 1 exchange.
   Except where otherwise noted, there are no requirements for ISAKMP
   payloads in any message to be in any particular order.

   The Diffie-Hellman public value passed in a KE payload, in either a
   phase 1 or phase 2 exchange, MUST be the length of the negotiated
   Diffie-Hellman group enforced, if necessary, by pre-pending the value
   with zeros.

   The length of nonce payload MUST be between 8 and 256 bytes
   inclusive.

   Main Mode is an instantiation of the ISAKMP Identity Protect
   Exchange: The first two messages negotiate policy; the next two
   exchange Diffie-Hellman public values and ancillary data (e.g.
   nonces) necessary for the exchange; and the last two messages
   authenticate the Diffie-Hellman Exchange. The authentication method
   negotiated as part of the initial ISAKMP exchange influences the
   composition of the payloads but not their purpose. The XCHG for Main
   Mode is ISAKMP Identity Protect.

   Similarly, Aggressive Mode is an instantiation of the ISAKMP
   Aggressive Exchange. The first two messages negotiate policy,
   exchange Diffie-Hellman public values and ancillary data necessary
   for the exchange, and identities. In addition the second message
   authenticates the responder. The third message authenticates the
   initiator and provides a proof of participation in the exchange. The
   XCHG for Aggressive Mode is ISAKMP Aggressive. The final message MAY
   NOT be sent under protection of the ISAKMP SA allowing each party to

postpone exponentiation, if desired, until negotiation of this
exchange is complete. The graphic depictions of Aggressive Mode show
the final payload in the clear; it need not be.

Exchanges in IKE are not open ended and have a fixed number of
messages. Receipt of a Certificate Request payload MUST NOT extend
the number of messages transmitted or expected.

Security Association negotiation is limited with Aggressive Mode. Due
to message construction requirements the group in which the Diffie-
Hellman exchange is performed cannot be negotiated. In addition,
different authentication methods may further constrain attribute
negotiation. For example, authentication with public key encryption
cannot be negotiated and when using the revised method of public key
encryption for authentication the cipher and hash cannot be
negotiated. For situations where the rich attribute negotiation
capabilities of IKE are required Main Mode may be required.

Quick Mode and New Group Mode have no analog in ISAKMP. The XCHG
values for Quick Mode and New Group Mode are defined in Appendix A.

Main Mode, Aggressive Mode, and Quick Mode do security association
negotiation. Security Association offers take the form of Tranform
Payload(s) encapsulated in Proposal Payload(s) encapsulated in
Security Association (SA) payload(s). If multiple offers are being
made for phase 1 exchanges (Main Mode and Aggressive Mode) they MUST
take the form of multiple Transform Payloads for a single Proposal
Payload in a single SA payload. To put it another way, for phase 1
exchanges there MUST NOT be multiple Proposal Payloads for a single
SA payload and there MUST NOT be multiple SA payloads. This document
does not proscribe such behavior on offers in phase 2 exchanges.

There is no limit on the number of offers the initiator may send to
the responder but conformant implementations MAY choose to limit the
number of offers it will inspect for performance reasons.

During security association negotiation, initiators present offers
for potential security associations to responders. Responders MUST
NOT modify attributes of any offer, attribute encoding excepted (see
Appendix A). If the initiator of an exchange notices that attribute
values have changed or attributes have been added or deleted from an
offer made, that response MUST be rejected.

Four different authentication methods are allowed with either Main
Mode or Aggressive Mode-- digital signature, two forms of
authentication with public key encryption, or pre-shared key. The
value SKEYID is computed seperately for each authentication method.

```
   For signatures:            SKEYID = prf(Ni_b | Nr_b, g^xy)
   For public key encryption: SKEYID = prf(hash(Ni_b | Nr_b), CKY-I |
CKY-R)
   For pre-shared keys:       SKEYID = prf(pre-shared-key, Ni_b |
Nr_b)
```

The result of either Main Mode or Aggressive Mode is three groups of
authenticated keying material:

```
   SKEYID_d = prf(SKEYID, g^xy | CKY-I | CKY-R | 0)
   SKEYID_a = prf(SKEYID, SKEYID_d | g^xy | CKY-I | CKY-R | 1)
   SKEYID_e = prf(SKEYID, SKEYID_a | g^xy | CKY-I | CKY-R | 2)
```

and agreed upon policy to protect further communications. The values
of 0, 1, and 2 above are represented by a single octet. The key used
for encryption is derived from SKEYID_e in an algorithm-specific
manner (see appendix B).

To authenticate either exchange the initiator of the protocol
generates HASH_I and the responder generates HASH_R where:

```
 HASH_I = prf(SKEYID, g^xi | g^xr | CKY-I | CKY-R | SAi_b | IDii_b )
 HASH_R = prf(SKEYID, g^xr | g^xi | CKY-R | CKY-I | SAi_b | IDir_b )
```

For authentication with digital signatures, HASH_I and HASH_R are
signed and verified; for authentication with either public key
encryption or pre-shared keys, HASH_I and HASH_R directly
authenticate the exchange. The entire ID payload (including ID type,
port, and protocol but excluding the generic header) is hashed into
both HASH_I and HASH_R.

As mentioned above, the negotiated authentication method influences
the content and use of messages for Phase 1 Modes, but not their
intent. When using public keys for authentication, the Phase 1
exchange can be accomplished either by using signatures or by using
public key encryption (if the algorithm supports it). Following are
Phase 1 exchanges with different authentication options.

5.1 IKE Phase 1 Authenticated With Signatures

   Using signatures, the ancillary information exchanged during the
   second roundtrip are nonces; the exchange is authenticated by signing
   a mutually obtainable hash. Main Mode with signature authentication
   is described as follows:

```
     Initiator                         Responder
     -----------                       -----------
     HDR, SA                     -->
                                 <--     HDR, SA
     HDR, KE, Ni                 -->
                                 <--     HDR, KE, Nr
     HDR*, IDii, [ CERT, ] SIG_I -->
                                 <--     HDR*, IDir, [ CERT, ] SIG_R
```

Aggressive mode with signatures in conjunction with ISAKMP is
described as follows:

```
     Initiator                         Responder
     -----------                       -----------
     HDR, SA, KE, Ni, IDii       -->
                                 <--     HDR, SA, KE, Nr, IDir,
                                           [ CERT, ] SIG_R
     HDR, [ CERT, ] SIG_I        -->
```

In both modes, the signed data, SIG_I or SIG_R, is the result of the
negotiated digital signature algorithm applied to HASH_I or HASH_R
respectively.

In general the signature will be over HASH_I and HASH_R as above
using the negotiated prf, or the HMAC version of the negotiated hash
function (if no prf is negotiated). However, this can be overridden
for construction of the signature if the signature algorithm is tied
to a particular hash algorithm (e.g. DSS is only defined with SHA's
160 bit output). In this case, the signature will be over HASH_I and
HASH_R as above, except using the HMAC version of the hash algorithm
associated with the signature method. The negotiated prf and hash
function would continue to be used for all other prescribed pseudo-
random functions.

Since the hash algorithm used is already known there is no need to
encode its OID into the signature. In addition, there is no binding
between the OIDs used for RSA signatures in PKCS #1 and those used in
this document. Therefore, RSA signatures MUST be encoded as a private
key encryption in PKCS #1 format and not as a signature in PKCS #1
format (which includes the OID of the hash algorithm). DSS signatures
MUST be encoded as r followed by s.

One or more certificate payloads MAY be optionally passed.

5.2 Phase 1 Authenticated With Public Key Encryption

   Using public key encryption to authenticate the exchange, the
   ancillary information exchanged is encrypted nonces. Each party's
   ability to reconstruct a hash (proving that the other party decrypted
   the nonce) authenticates the exchange.

   In order to perform the public key encryption, the initiator must
   already have the responder's public key. In the case where the
   responder has multiple public keys, a hash of the certificate the
   initiator is using to encrypt the ancillary information is passed as
   part of the third message. In this way the responder can determine
   which corresponding private key to use to decrypt the encrypted
   payloads and identity protection is retained.

   In addition to the nonce, the identities of the parties (IDii and
   IDir) are also encrypted with the other party's public key. If the
   authentication method is public key encryption, the nonce and
   identity payloads MUST be encrypted with the public key of the other
   party. Only the body of the payloads are encrypted, the payload
   headers are left in the clear.

   When using encryption for authentication, Main Mode is defined as
   follows.

         Initiator   Responder
        -----------   -----------
        HDR, SA   -->
                                  <--   HDR, SA
        HDR, KE, [HASH(1),]
          <IDii_b>PubKey_r,
            <Ni_b>PubKey_r   -->
                                          HDR, KE, <IDir_b>PubKey_i,
                                  <--   <Nr_b>PubKey_i
        HDR*, HASH_I   -->
                                  <--   HDR*, HASH_R

   Aggressive Mode authenticated with encryption is described as
   follows:

         Initiator   Responder
        -----------   -----------
        HDR, SA, [HASH(1),] KE,
          <IDii_b>Pubkey_r,
            <Ni_b>Pubkey_r   -->
                                          HDR, SA, KE, <IDir_b>PubKey_i,
                                  <--   <Nr_b>PubKey_i, HASH_R
        HDR, HASH_I   -->

Where HASH(1) is a hash (using the negotiated hash function) of the
certificate which the initiator is using to encrypt the nonce and
identity.

RSA encryption MUST be encoded in PKCS #1 format. While only the body
of the ID and nonce payloads is encrypted, the encrypted data must be
preceded by a valid ISAKMP generic header. The payload length is the
length of the entire encrypted payload plus header. The PKCS #1
encoding allows for determination of the actual length of the
cleartext payload upon decryption.

Using encryption for authentication provides for a plausably deniable
exchange. There is no proof (as with a digital signature) that the
conversation ever took place since each party can completely
reconstruct both sides of the exchange. In addition, security is
added to secret generation since an attacker would have to
successfully break not only the Diffie-Hellman exchange but also both
RSA encryptions. This exchange was motivated by [SKEME].

Note that, unlike other authentication methods, authentication with
public key encryption allows for identity protection with Aggressive
Mode.

5.3 Phase 1 Authenticated With a Revised Mode of Public Key Encryption

Authentication with Public Key Encryption has significant advantages
over authentication with signatures (see section 5.2 above).
Unfortunately, this is at the cost of 4 public key operations-- two
public key encryptions and two private key decryptions. This
authentication mode retains the advantages of authentication using
public key encryption but does so with half the public key
operations.

In this mode, the nonce is still encrypted using the public key of
the peer, however the peer's identity (and the certificate if it is
sent) is encrypted using the negotiated symmetric encryption
algorithm (from the SA payload) with a key derived from the nonce.
This solution adds minimal complexity and state yet saves two costly
public key operations on each side. In addition, the Key Exchange
payload is also encrypted using the same derived key. This provides
additional protection against cryptanalysis of the Diffie-Hellman
exchange.

As with the public key encryption method of authentication (section
5.2), a HASH payload may be sent to identify a certificate if the
responder has multiple certificates which contain useable public keys
(e.g. if the certificate is not for signatures only, either due to
certificate restrictions or algorithmic restrictions). If the HASH

payload is sent it MUST be the first payload of the second message
exchange and MUST be followed by the encrypted nonce. If the HASH
payload is not sent, the first payload of the second message exchange
MUST be the encrypted nonce. In addition, the initiator my optionally
send a certificate payload to provide the responder with a public key
with which to respond.

When using the revised encryption mode for authentication, Main Mode
is defined as follows.

```
    Initiator                        Responder
    -----------                      -----------
    HDR, SA                  -->
                             <--     HDR, SA
    HDR, [ HASH(1), ]
      <Ni_b>Pubkey_r,
      <KE_b>Ke_i,
      <IDii_b>Ke_i,
      [<<Cert-I_b>Ke_i]      -->
                                     HDR, <Nr_b>PubKey_i,
                                          <KE_b>Ke_r,
                             <--          <IDir_b>Ke_r,
    HDR*, HASH_I             -->
                             <--     HDR*, HASH_R
```

Aggressive Mode authenticated with the revised encryption method is
described as follows:

```
    Initiator                        Responder
    -----------                      -----------
    HDR, SA, [ HASH(1),]
      <Ni_b>Pubkey_r,
      <KE_b>Ke_i, <IDii_b>Ke_i
      [, <Cert-I_b>Ke_i ]    -->
                                     HDR, SA, <Nr_b>PubKey_i,
                                          <KE_b>Ke_r, <IDir_b>Ke_r,
                             <--          HASH_R
    HDR, HASH_I             -->
```

where HASH(1) is identical to section 5.2. Ke_i and Ke_r are keys to
the symmetric encryption algorithm negotiated in the SA payload
exchange. Only the body of the payloads are encrypted (in both public
key and symmetric operations), the generic payload headers are left
in the clear. The payload length includes that added to perform
encryption.

The symmetric cipher keys are derived from the decrypted nonces as
follows. First the values Ne_i and Ne_r are computed:

```
   Ne_i = prf(Ni_b, CKY-I)
   Ne_r = prf(Nr_b, CKY-R)
```

The keys Ke_i and Ke_r are then taken from Ne_i and Ne_r respectively
in the manner described in Appendix B used to derive symmetric keys
for use with the negotiated encryption algorithm. If the length of
the output of the negotiated prf is greater than or equal to the key
length requirements of the cipher, Ke_i and Ke_r are derived from the
most significant bits of Ne_i and Ne_r respectively. If the desired
length of Ke_i and Ke_r exceed the length of the output of the prf
the necessary number of bits is obtained by repeatedly feeding the
results of the prf back into itself and concatenating the result
until the necessary number has been achieved. For example, if the
negotiated encryption algorithm requires 320 bits of key and the
output of the prf is only 128 bits, Ke_i is the most significant 320
bits of K, where

```
   K = K1 | K2 | K3 and
   K1 = prf(Ne_i, 0)
   K2 = prf(Ne_i, K1)
   K3 = prf(Ne_i, K2)
```

For brevity, only derivation of Ke_i is shown; Ke_r is identical. The
length of the value 0 in the computation of K1 is a single octet.
Note that Ne_i, Ne_r, Ke_i, and Ke_r are all ephemeral and MUST be
discarded after use.

Save the requirements on the location of the optional HASH payload
and the mandatory nonce payload there are no further payload
requirements. All payloads-- in whatever order-- following the
encrypted nonce MUST be encrypted with Ke_i or Ke_r depending on the
direction.

If CBC mode is used for the symmetric encryption then the
initialization vectors (IVs) are set as follows. The IV for
encrypting the first payload following the nonce is set to 0 (zero).
The IV for subsequent payloads encrypted with the ephemeral symmetric
cipher key, Ke_i, is the last ciphertext block of the previous
payload. Encrypted payloads are padded up to the nearest block size.
All padding bytes, except for the last one, contain 0x00. The last
byte of the padding contains the number of the padding bytes used,
excluding the last one. Note that this means there will always be
padding.

5.4 Phase 1 Authenticated With a Pre-Shared Key

   A key derived by some out-of-band mechanism may also be used to
   authenticate the exchange. The actual establishment of this key is
   out of the scope of this document.

   When doing a pre-shared key authentication, Main Mode is defined as
   follows:

```
            Initiator                       Responder
           ----------                      -----------
           HDR, SA             -->
                               <--     HDR, SA
           HDR, KE, Ni         -->
                               <--     HDR, KE, Nr
           HDR*, IDii, HASH_I  -->
                               <--     HDR*, IDir, HASH_R
```

   Aggressive mode with a pre-shared key is described as follows:

```
            Initiator                       Responder
           -----------                     -----------
           HDR, SA, KE, Ni, IDii -->
                               <--     HDR, SA, KE, Nr, IDir, HASH_R
           HDR, HASH_I         -->
```

   When using pre-shared key authentication with Main Mode the key can
   only be identified by the IP address of the peers since HASH_I must
   be computed before the initiator has processed IDir. Aggressive Mode
   allows for a wider range of identifiers of the pre-shared secret to
   be used. In addition, Aggressive Mode allows two parties to maintain
   multiple, different pre-shared keys and identify the correct one for
   a particular exchange.

5.5 Phase 2 - Quick Mode

   Quick Mode is not a complete exchange itself (in that it is bound to
   a phase 1 exchange), but is used as part of the SA negotiation
   process (phase 2) to derive keying material and negotiate shared
   policy for non-ISAKMP SAs. The information exchanged along with Quick
   Mode MUST be protected by the ISAKMP SA-- i.e. all payloads except
   the ISAKMP header are encrypted. In Quick Mode, a HASH payload MUST
   immediately follow the ISAKMP header and a SA payload MUST
   immediately follow the HASH. This HASH authenticates the message and
   also provides liveliness proofs.

The message ID in the ISAKMP header identifies a Quick Mode in
progress for a particular ISAKMP SA which itself is identified by the
cookies in the ISAKMP header. Since each instance of a Quick Mode
uses a unique initialization vector (see Appendix B) it is possible
to have multiple simultaneous Quick Modes, based off a single ISAKMP
SA, in progress at any one time.

Quick Mode is essentially a SA negotiation and an exchange of nonces
that provides replay protection. The nonces are used to generate
fresh key material and prevent replay attacks from generating bogus
security associations. An optional Key Exchange payload can be
exchanged to allow for an additional Diffie-Hellman exchange and
exponentiation per Quick Mode. While use of the key exchange payload
with Quick Mode is optional it MUST be supported.

Base Quick Mode (without the KE payload) refreshes the keying
material derived from the exponentiation in phase 1. This does not
provide PFS. Using the optional KE payload, an additional
exponentiation is performed and PFS is provided for the keying
material.

The identities of the SAs negotiated in Quick Mode are implicitly
assumed to be the IP addresses of the ISAKMP peers, without any
implied constraints on the protocol or port numbers allowed, unless
client identifiers are specified in Quick Mode. If ISAKMP is acting
as a client negotiator on behalf of another party, the identities of
the parties MUST be passed as IDci and then IDcr. Local policy will
dictate whether the proposals are acceptable for the identities
specified. If the client identities are not acceptable to the Quick
Mode responder (due to policy or other reasons), a Notify payload
with Notify Message Type INVALID-ID-INFORMATION (18) SHOULD be sent.

The client identities are used to identify and direct traffic to the
appropriate tunnel in cases where multiple tunnels exist between two
peers and also to allow for unique and shared SAs with different
granularities.

All offers made during a Quick Mode are logically related and must be
consistant. For example, if a KE payload is sent, the attribute
describing the Diffie-Hellman group (see section 6.1 and [Pip97])
MUST be included in every transform of every proposal of every SA
being negotiated. Similarly, if client identities are used, they MUST
apply to every SA in the negotiation.

Quick Mode is defined as follows:

```
     Initiator                         Responder
     -----------                       -----------
     HDR*, HASH(1), SA, Ni
       [, KE ] [, IDci, IDcr ] -->
                                  <--  HDR*, HASH(2), SA, Nr
                                         [, KE ] [, IDci, IDcr ]
     HDR*, HASH(3)             -->
```

Where:
HASH(1) is the prf over the message id (M-ID) from the ISAKMP header
concatenated with the entire message that follows the hash including
all payload headers, but excluding any padding added for encryption.
HASH(2) is identical to HASH(1) except the initiator's nonce-- Ni,
minus the payload header-- is added after M-ID but before the
complete message. The addition of the nonce to HASH(2) is for a
liveliness proof. HASH(3)-- for liveliness-- is the prf over the
value zero represented as a single octet, followed by a concatenation
of the message id and the two nonces-- the initiator's followed by
the responder's-- minus the payload header. In other words, the
hashes for the above exchange are:

HASH(1) = prf(SKEYID_a, M-ID | SA | Ni [| KE] [| IDci | IDcr)
HASH(2) = prf(SKEYID_a, M-ID | Ni_b | SA | Nr [| KE] [| IDci |
IDcr)
HASH(3) = prf(SKEYID_a, 0 | M-ID | Ni_b | Nr_b)

With the exception of the HASH, SA, and the optional ID payloads,
there are no payload ordering restrictions on Quick Mode. HASH(1) and
HASH(2) may differ from the illustration above if the order of
payloads in the message differs from the illustrative example or if
any optional payloads, for example a notify payload, have been
chained to the message.

If PFS is not needed, and KE payloads are not exchanged, the new
keying material is defined as

    KEYMAT = prf(SKEYID_d, protocol | SPI | Ni_b | Nr_b).

If PFS is desired and KE payloads were exchanged, the new keying
material is defined as

    KEYMAT = prf(SKEYID_d, g(qm)^xy | protocol | SPI | Ni_b | Nr_b)

where g(qm)^xy is the shared secret from the ephemeral Diffie-Hellman
exchange of this Quick Mode.

In either case, "protocol" and "SPI" are from the ISAKMP Proposal
Payload that contained the negotiated Transform.

A single SA negotiation results in two security assocations-- one
inbound and one outbound. Different SPIs for each SA (one chosen by
the initiator, the other by the responder) guarantee a different key
for each direction. The SPI chosen by the destination of the SA is
used to derive KEYMAT for that SA.

For situations where the amount of keying material desired is greater
than that supplied by the prf, KEYMAT is expanded by feeding the
results of the prf back into itself and concatenating results until
the required keying material has been reached. In other words,

```
    KEYMAT = K1 | K2 | K3 | ...
    where
      K1 = prf(SKEYID_d, [ g(qm)^xy | ] protocol | SPI | Ni_b | Nr_b)
      K2 = prf(SKEYID_d, K1 | [ g(qm)^xy | ] protocol | SPI | Ni_b |
      Nr_b)
      K3 = prf(SKEYID_d, K2 | [ g(qm)^xy | ] protocol | SPI | Ni_b |
      Nr_b)
      etc.
```

This keying material (whether with PFS or without, and whether
derived directly or through concatenation) MUST be used with the
negotiated SA. It is up to the service to define how keys are derived
from the keying material.

In the case of an ephemeral Diffie-Hellman exchange in Quick Mode,
the exponential (g(qm)^xy) is irretreivably removed from the current
state and SKEYID_e and SKEYID_a (derived from phase 1 negotiation)
continue to protect and authenticate the ISAKMP SA and SKEYID_d
continues to be used to derive keys.

Using Quick Mode, multiple SA's and keys can be negotiated with one
exchange as follows:

```
     Initiator                        Responder
     -----------                      -----------
     HDR*, HASH(1), SA0, SA1, Ni,
       [, KE ] [, IDci, IDcr ] -->
                                 <--   HDR*, HASH(2), SA0, SA1, Nr,
                                         [, KE ] [, IDci, IDcr ]
     HDR*, HASH(3)              -->
```

The keying material is derived identically as in the case of a single
SA. In this case (negotiation of two SA payloads) the result would be
four security associations-- two each way for both SAs.

5.6 New Group Mode

   New Group Mode MUST NOT be used prior to establishment of an ISAKMP
   SA. The description of a new group MUST only follow phase 1
   negotiation. (It is not a phase 2 exchange, though).

         Initiator   Responder
         -----------   -----------
         HDR*, HASH(1), SA   -->
                                 <--   HDR*, HASH(2), SA

   where HASH(1) is the prf output, using SKEYID_a as the key, and the
   message-ID from the ISAKMP header concatenated with the entire SA
   proposal, body and header, as the data; HASH(2) is the prf output,
   using SKEYID_a as the key, and the message-ID from the ISAKMP header
   concatenated with the reply as the data. In other words the hashes
   for the above exchange are:

      HASH(1) = prf(SKEYID_a, M-ID | SA)
      HASH(2) = prf(SKEYID_a, M-ID | SA)

   The proposal will specify the characteristics of the group (see
   appendix A, "Attribute Assigned Numbers"). Group descriptions for
   private Groups MUST be greater than or equal to 2^{15}. If the group
   is not acceptable, the responder MUST reply with a Notify payload
   with the message type set to ATTRIBUTES-NOT-SUPPORTED (13).

   ISAKMP implementations MAY require private groups to expire with the
   SA under which they were established.

   Groups may be directly negotiated in the SA proposal with Main Mode.
   To do this the component parts-- for a MODP group, the type, prime
   and generator; for a EC2N group the type, the Irreducible Polynomial,
   Group Generator One, Group Generator Two, Group Curve A, Group Curve
   B and Group Order-- are passed as SA attributes (see Appendix A).
   Alternately, the nature of the group can be hidden using New Group
   Mode and only the group identifier is passed in the clear during
   phase 1 negotiation.

5.7 ISAKMP Informational Exchanges

   This protocol protects ISAKMP Informational Exchanges when possible.
   Once the ISAKMP security association has been established (and
   SKEYID_e and SKEYID_a have been generated) ISAKMP Information
   Exchanges, when used with this protocol, are as follows:

```
        Initiator                        Responder
        -----------                      -----------
        HDR*, HASH(1), N/D        -->
```

   where N/D is either an ISAKMP Notify Payload or an ISAKMP Delete
   Payload and HASH(1) is the prf output, using SKEYID_a as the key, and
   a M-ID unique to this exchange concatenated with the entire
   informational payload (either a Notify or Delete) as the data. In
   other words, the hash for the above exchange is:

       HASH(1) = prf(SKEYID_a, M-ID | N/D)

   As noted the message ID in the ISAKMP header-- and used in the prf
   computation-- is unique to this exchange and MUST NOT be the same as
   the message ID of another phase 2 exchange which generated this
   informational exchange. The derivation of the initialization vector,
   used with SKEYID_e to encrypt this message, is described in Appendix
   B.

   If the ISAKMP security association has not yet been established at
   the time of the Informational Exchange, the exchange is done in the
   clear without an accompanying HASH payload.

6 Oakley Groups

   With IKE, the group in which to do the Diffie-Hellman exchange is
   negotiated. Four groups-- values 1 through 4-- are defined below.
   These groups originated with the Oakley protocol and are therefore
   called "Oakley Groups". The attribute class for "Group" is defined in
   Appendix A. All values 2^{15} and higher are used for private group
   identifiers. For a discussion on the strength of the default Oakley
   groups please see the Security Considerations section below.

   These groups were all generated by Richard Schroeppel at the
   University of Arizona. Properties of these groups are described in
   [Orm96].

6.1 First Oakley Default Group

   Oakley implementations MUST support a MODP group with the following
   prime and generator. This group is assigned id 1 (one).

       The prime is: $2^{768} - 2^{704} - 1 + 2^{64} * \{ [2^{638} pi] + 149686 \}$
       Its hexadecimal value is

```
        FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1
        29024E08 8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD
        EF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245
        E485B576 625E7EC6 F44C42E9 A63A3620 FFFFFFFF FFFFFFFF
```

    The generator is: 2.

6.2 Second Oakley Group

   IKE implementations SHOULD support a MODP group with the following
   prime and generator. This group is assigned id 2 (two).

   The prime is 2^1024 - 2^960 - 1 + 2^64 * { [2^894 pi] + 129093 }.
   Its hexadecimal value is

```
        FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1
        29024E08 8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD
        EF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245
        E485B576 625E7EC6 F44C42E9 A637ED6B 0BFF5CB6 F406B7ED
        EE386BFB 5A899FA5 AE9F2411 7C4B1FE6 49286651 ECE65381
        FFFFFFFF FFFFFFFF
```

    The generator is 2 (decimal)

6.3 Third Oakley Group

   IKE implementations SHOULD support a EC2N group with the following
   characteristics. This group is assigned id 3 (three). The curve is
   based on the Galois Field GF[2^155]. The field size is 155. The
   irreducible polynomial for the field is:
        u^155 + u^62 + 1.
   The equation for the elliptic curve is:
        y^2 + xy = x^3 + ax^2 + b.

   Field Size: 155
   Group Prime/Irreducible Polynomial:
                    0x0800000000000000000000004000000000000001
   Group Generator One: 0x7b
   Group Curve A: 0x0
   Group Curve B: 0x07338f

   Group Order: 0X0800000000000000000057db5698537193aef944

   The data in the KE payload when using this group is the value x from
   the solution (x,y), the point on the curve chosen by taking the
   randomly chosen secret Ka and computing Ka*P, where * is the
   repetition of the group addition and double operations, P is the
   curve point with x coordinate equal to generator 1 and the y

coordinate determined from the defining equation. The equation of
curve is implicitly known by the Group Type and the A and B
coefficients. There are two possible values for the y coordinate;
either one can be used successfully (the two parties need not agree
on the selection).

6.4 Fourth Oakley Group

   IKE implementations SHOULD support a EC2N group with the following
   characteristics. This group is assigned id 4 (four). The curve is
   based on the Galois Field GF[2^185]. The field size is 185. The
   irreducible polynomial for the field is:
           u^185 + u^69 + 1. The
   equation for the elliptic curve is:
           y^2 + xy = x^3 + ax^2 + b.

   Field Size: 185
   Group Prime/Irreducible Polynomial:
                      0x020000000000000000000000000000000200000000000000001
   Group Generator One: 0x18
   Group Curve A: 0x0
   Group Curve B: 0x1ee9

   Group Order: 0X01ffffffffffffffffffffffffdbf2f889b73e484175f94ebc

   The data in the KE payload when using this group will be identical to
   that as when using Oakley Group 3 (three).

   Other groups can be defined using New Group Mode. These default
   groups were generated by Richard Schroeppel at the University of
   Arizona. Properties of these primes are described in [Orm96].

7. Payload Explosion for a Complete IKE Exchange

   This section illustrates how the IKE protocol is used to:

      - establish a secure and authenticated channel between ISAKMP
      processes (phase 1); and

      - generate key material for, and negotiate, an IPsec SA (phase 2).

7.1 Phase 1 using Main Mode

   The following diagram illustrates the payloads exchanged between the
   two parties in the first round trip exchange. The initiator MAY
   propose several proposals; the responder MUST reply with one.

```
         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 with XCHG of Main Mode,              ~
        ~                and Next Payload of ISA_SA                    ~
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        !      0       !   RESERVED   !        Payload Length          !
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        !                   Domain of Interpretation                  !
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        !                         Situation                           !
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        !      0       !   RESERVED   !        Payload Length          !
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        ! Proposal #1  ! PROTO_ISAKMP ! SPI size = 0  | # Transforms   !
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        !   ISA_TRANS  !   RESERVED   !        Payload Length          !
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        ! Transform #1 ! KEY_OAKLEY   |          RESERVED2             !
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        ~                     prefered SA attributes                  ~
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        !      0       !   RESERVED   !        Payload Length          !
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        ! Transform #2 ! KEY_OAKLEY   |          RESERVED2             !
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        ~                     alternate SA attributes                 ~
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The responder replies in kind but selects, and returns, one transform
proposal (the ISAKMP SA attributes).

The second exchange consists of the following payloads:

```
         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 with XCHG of Main Mode,              ~
        ~                and Next Payload of ISA_KE                    ~
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        !   ISA_NONCE  !   RESERVED   !        Payload Length          !
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        ~   D-H Public Value  (g^xi from initiator g^xr from responder) ~
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        !      0       !   RESERVED   !        Payload Length          !
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        ~          Ni (from initiator) or  Nr (from responder)         ~
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The shared keys, SKEYID_e and SKEYID_a, are now used to protect and
authenticate all further communication. Note that both SKEYID_e and
SKEYID_a are unauthenticated.

```
      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 with XCHG of Main Mode,            ~
     ~     and Next Payload of ISA_ID and the encryption bit set    ~
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     !    ISA_SIG    !    RESERVED    !       Payload Length         !
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     ~         Identification Data of the ISAKMP negotiator         ~
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     !      0       !    RESERVED    !       Payload Length         !
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     ~        signature verified by the public key of the ID above  ~
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The key exchange is authenticated over a signed hash as described in
section 5.1. Once the signature has been verified using the
authentication algorithm negotiated as part of the ISAKMP SA, the
shared keys, SKEYID_e and SKEYID_a can be marked as authenticated.
(For brevity, certificate payloads were not exchanged).

7.2 Phase 2 using Quick Mode

The following payloads are exchanged in the first round of Quick Mode
with ISAKMP SA negotiation. In this hypothetical exchange, the ISAKMP
negotiators are proxies for other parties which have requested
authentication.

```
      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 with XCHG of Quick Mode,           ~
     ~   Next Payload of ISA_HASH and the encryption bit set        ~
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     !    ISA_SA     !    RESERVED    !       Payload Length         !
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     ~                   keyed hash of message                      ~
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     !   ISA_NONCE   !    RESERVED    !       Payload Length         !
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     !                   Domain Of Interpretation                   !
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     !                         Situation                           !
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     !      0       !    RESERVED    !       Payload Length         !
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
! Proposal #1  ! PROTO_IPSEC_AH! SPI size = 4  | # Transforms  !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                       SPI (4 octets)                          ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!   ISA_TRANS  !   RESERVED   !         Payload Length          !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Transform #1 !     AH_SHA   |            RESERVED2            !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                     other SA attributes                       !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!      0       !   RESERVED   !         Payload Length          !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Transform #2 !     AH_MD5   |            RESERVED2            !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                     other SA attributes                       !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!    ISA_ID    !   RESERVED   !         Payload Length          !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                           nonce                               ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!    ISA_ID    !   RESERVED   !         Payload Length          !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~          ID of source for which ISAKMP is a client            ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!      0       !   RESERVED   !         Payload Length          !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~        ID of destination for which ISAKMP is a client         ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where the contents of the hash are described in 5.5 above. The
responder replies with a similar message which only contains one
transform-- the selected AH transform. Upon receipt, the initiator
can provide the key engine with the negotiated security association
and the keying material. As a check against replay attacks, the
responder waits until receipt of the next message.

```
 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 with XCHG of Quick Mode,              ~
~    Next Payload of ISA_HASH and the encryption bit set        ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!      0       !   RESERVED   !         Payload Length          !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                         hash data                             ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where the contents of the hash are described in 5.5 above.

8. Perfect Forward Secrecy Example

   This protocol can provide PFS of both keys and identities. The
   identies of both the ISAKMP negotiating peer and, if applicable, the
   identities for whom the peers are negotiating can be protected with
   PFS.

   To provide Perfect Forward Secrecy of both keys and all identities,
   two parties would perform the following:

      o A Main Mode Exchange to protect the identities of the ISAKMP
        peers.
        This establishes an ISAKMP SA.
      o A Quick Mode Exchange to negotiate other security protocol
        protection.
        This establishes a SA on each end for this protocol.
      o Delete the ISAKMP SA and its associated state.

   Since the key for use in the non-ISAKMP SA was derived from the
   single ephemeral Diffie-Hellman exchange PFS is preserved.

   To provide Perfect Forward Secrecy of merely the keys of a non-ISAKMP
   security association, it in not necessary to do a phase 1 exchange if
   an ISAKMP SA exists between the two peers. A single Quick Mode in
   which the optional KE payload is passed, and an additional Diffie-
   Hellman exchange is performed, is all that is required. At this point
   the state derived from this Quick Mode must be deleted from the
   ISAKMP SA as described in section 5.5.

9. Implementation Hints

   Using a single ISAKMP Phase 1 negotiation makes subsequent Phase 2
   negotiations extremely quick. As long as the Phase 1 state remains
   cached, and PFS is not needed, Phase 2 can proceed without any
   exponentiation. How many Phase 2 negotiations can be performed for a
   single Phase 1 is a local policy issue. The decision will depend on
   the strength of the algorithms being used and level of trust in the
   peer system.

   An implementation may wish to negotiate a range of SAs when
   performing Quick Mode. By doing this they can speed up the "re-
   keying". Quick Mode defines how KEYMAT is defined for a range of SAs.
   When one peer feels it is time to change SAs they simply use the next
   one within the stated range. A range of SAs can be established by
   negotiating multiple SAs (identical attributes, different SPIs) with
   one Quick Mode.

An optimization that is often useful is to establish Security
Associations with peers before they are needed so that when they
become needed they are already in place. This ensures there would be
no delays due to key management before initial data transmission.
This optimization is easily implemented by setting up more than one
Security Association with a peer for each requested Security
Association and caching those not immediately used.

Also, if an ISAKMP implementation is alerted that a SA will soon be
needed (e.g. to replace an existing SA that will expire in the near
future), then it can establish the new SA before that new SA is
needed.

The base ISAKMP specification describes conditions in which one party
of the protocol may inform the other party of some activity-- either
deletion of a security association or in response to some error in
the protocol such as a signature verification failed or a payload
failed to decrypt. It is strongly suggested that these Informational
exchanges not be responded to under any circumstances. Such a
condition may result in a "notify war" in which failure to understand
a message results in a notify to the peer who cannot understand it
and sends his own notify back which is also not understood.

10. Security Considerations

This entire memo discusses a hybrid protocol, combining parts of
Oakley and parts of SKEME with ISAKMP, to negotiate, and derive
keying material for, security associations in a secure and
authenticated manner.

Confidentiality is assured by the use of a negotiated encryption
algorithm. Authentication is assured by the use of a negotiated
method: a digital signature algorithm; a public key algorithm which
supports encryption; or, a pre-shared key. The confidentiality and
authentication of this exchange is only as good as the attributes
negotiated as part of the ISAKMP security association.

Repeated re-keying using Quick Mode can consume the entropy of the
Diffie-Hellman shared secret. Implementors should take note of this
fact and set a limit on Quick Mode Exchanges between exponentiations.
This memo does not prescribe such a limit.

Perfect Forward Secrecy (PFS) of both keying material and identities
is possible with this protocol. By specifying a Diffie-Hellman group,
and passing public values in KE payloads, ISAKMP peers can establish
PFS of keys-- the identities would be protected by SKEYID_e from the
ISAKMP SA and would therefore not be protected by PFS. If PFS of both
keying material and identities is desired, an ISAKMP peer MUST

establish only one non-ISAKMP security association (e.g. IPsec
Security Association) per ISAKMP SA. PFS for keys and identities is
accomplished by deleting the ISAKMP SA (and optionally issuing a
DELETE message) upon establishment of the single non-ISAKMP SA. In
this way a phase one negotiation is uniquely tied to a single phase
two negotiation, and the ISAKMP SA established during phase one
negotiation is never used again.

The strength of a key derived from a Diffie-Hellman exchange using
any of the groups defined here depends on the inherent strength of
the group, the size of the exponent used, and the entropy provided by
the random number generator used. Due to these inputs it is difficult
to determine the strength of a key for any of the defined groups. The
default Diffie-Hellman group (number one) when used with a strong
random number generator and an exponent no less than 160 bits is
sufficient to use for DES. Groups two through four provide greater
security. Implementations should make note of these conservative
estimates when establishing policy and negotiating security
parameters.

Note that these limitations are on the Diffie-Hellman groups
themselves. There is nothing in IKE which prohibits using stronger
groups nor is there anything which will dilute the strength obtained
from stronger groups. In fact, the extensible framework of IKE
encourages the definition of more groups; use of elliptical curve
groups will greatly increase strength using much smaller numbers.

For situations where defined groups provide insufficient strength New
Group Mode can be used to exchange a Diffie-Hellman group which
provides the necessary strength. In is incumbent upon implementations
to check the primality in groups being offered and independently
arrive at strength estimates.

It is assumed that the Diffie-Hellman exponents in this exchange are
erased from memory after use. In particular, these exponents must not
be derived from long-lived secrets like the seed to a pseudo-random
generator.

IKE exchanges maintain running initialization vectors (IV) where the
last ciphertext block of the last message is the IV for the next
message. To prevent retransmissions (or forged messages with valid
cookies) from causing exchanges to get out of sync IKE
implementations SHOULD NOT update their running IV until the
decrypted message has passed a basic sanity check and has been
determined to actually advance the IKE state machine-- i.e. it is not
a retransmission.

While the last roundtrip of Main Mode (and optionally the last
message of Aggressive Mode) is encrypted it is not, strictly
speaking, authenticated. An active substitution attack on the
ciphertext could result in payload corruption. If such an attack
corrupts mandatory payloads it would be detected by an authentication
failure, but if it corrupts any optional payloads (e.g. notify
payloads chained onto the last message of a Main Mode exchange) it
might not be detectable.

11. IANA Considerations

This document contains many "magic numbers" to be maintained by the
IANA. This section explains the criteria to be used by the IANA to
assign additional numbers in each of these lists.

11.1 Attribute Classes

Attributes negotiated in this protocol are identified by their class.
Requests for assignment of new classes must be accompanied by a
standards-track RFC which describes the use of this attribute.

11.2 Encryption Algorithm Class

Values of the Encryption Algorithm Class define an encryption
algorithm to use when called for in this document. Requests for
assignment of new encryption algorithm values must be accompanied by
a reference to a standards-track or Informational RFC or a reference
to published cryptographic literature which describes this algorithm.

11.3 Hash Algorithm

Values of the Hash Algorithm Class define a hash algorithm to use
when called for in this document. Requests for assignment of new hash
algorithm values must be accompanied by a reference to a standards-
track or Informational RFC or a reference to published cryptographic
literature which describes this algorithm. Due to the key derivation
and key expansion uses of HMAC forms of hash algorithms in IKE,
requests for assignment of new hash algorithm values must take into
account the cryptographic properties-- e.g it's resistance to
collision-- of the hash algorithm itself.

11.4 Group Description and Group Type

Values of the Group Description Class identify a group to use in a
Diffie-Hellman exchange. Values of the Group Type Class define the
type of group. Requests for assignment of new groups must be
accompanied by a reference to a standards-track or Informational RFC
which describes this group. Requests for assignment of new group

types must be accompanied by a reference to a standards-track or
Informational RFC or by a reference to published cryptographic or
mathmatical literature which describes the new type.

11.5 Life Type

Values of the Life Type Class define a type of lifetime to which the
ISAKMP Security Association applies. Requests for assignment of new
life types must be accompanied by a detailed description of the units
of this type and its expiry.

12. Acknowledgements

This document is the result of close consultation with Hugo Krawczyk,
Douglas Maughan, Hilarie Orman, Mark Schertler, Mark Schneider, and
Jeff Turner. It relies on protocols which were written by them.
Without their interest and dedication, this would not have been
written.

Special thanks Rob Adams, Cheryl Madson, Derrell Piper, Harry Varnis,
and Elfed Weaver for technical input, encouragement, and various
sanity checks along the way.

We would also like to thank the many members of the IPSec working
group that contributed to the development of this protocol over the
past year.

13. References

[CAST]   Adams, C., "The CAST-128 Encryption Algorithm", RFC 2144,
          May 1997.

[BLOW]   Schneier, B., "The Blowfish Encryption Algorithm", Dr.
          Dobb's Journal, v. 19, n. 4, April 1994.

[Bra97]   Bradner, S., "Key Words for use in RFCs to indicate
          Requirement Levels", BCP 14, RFC 2119, March 1997.

[DES]   ANSI X3.106, "American National Standard for Information
          Systems-Data Link Encryption", American National Standards
          Institute, 1983.

[DH]   Diffie, W., and Hellman M., "New Directions in
          Cryptography", IEEE Transactions on Information Theory, V.
          IT-22, n. 6, June 1977.

    [DSS]   NIST, "Digital Signature Standard", FIPS 186, National
                Institute of Standards and Technology, U.S. Department of
                Commerce, May, 1994.

    [IDEA]   Lai, X., "On the Design and Security of Block Ciphers," ETH
                Series in Information Processing, v. 1, Konstanz: Hartung-
                Gorre Verlag, 1992

    [KBC96]   Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-
                Hashing for Message Authentication", RFC 2104, February
                1997.

    [SKEME]   Krawczyk, H., "SKEME: A Versatile Secure Key Exchange
                Mechanism for Internet", from IEEE Proceedings of the 1996
                Symposium on Network and Distributed Systems Security.

    [MD5]   Rivest, R., "The MD5 Message Digest Algorithm", RFC 1321,
                April 1992.

    [MSST98]   Maughhan, D., Schertler, M., Schneider, M., and J. Turner,
                "Internet Security Association and Key Management Protocol
                (ISAKMP)", RFC 2408, November 1998.

    [Orm96]   Orman, H., "The Oakley Key Determination Protocol", RFC
                2412, November 1998.

    [PKCS1]   RSA Laboratories, "PKCS #1: RSA Encryption Standard",
                November 1993.

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

    [RC5]   Rivest, R., "The RC5 Encryption Algorithm", Dr. Dobb's
                Journal, v. 20, n. 1, January 1995.

    [RSA]   Rivest, R., Shamir, A., and Adleman, L., "A Method for
                Obtaining Digital Signatures and Public-Key Cryptosystems",
                Communications of the ACM, v. 21, n. 2, February 1978.

    [Sch96]   Schneier, B., "Applied Cryptography, Protocols, Algorithms,
                and Source Code in C", 2nd edition.

    [SHA]   NIST, "Secure Hash Standard", FIPS 180-1, National Institue
                of Standards and Technology, U.S. Department of Commerce,
                May 1994.

    [TIGER]   Anderson, R., and Biham, E., "Fast Software Encryption",
                Springer LNCS v. 1039, 1996.

Appendix A

   This is a list of DES Weak and Semi-Weak keys. The keys come from
   [Sch96]. All keys are listed in hexidecimal.

      DES Weak Keys
      0101 0101 0101 0101
      1F1F 1F1F E0E0 E0E0
      E0E0 E0E0 1F1F 1F1F
      FEFE FEFE FEFE FEFE

      DES Semi-Weak Keys
      01FE 01FE 01FE 01FE
      1FE0 1FE0 0EF1 0EF1
      01E0 01E0 01F1 01F1
      1FFE 1FFE 0EFE 0EFE
      011F 011F 010E 010E
      E0FE E0FE F1FE F1FE

      FE01 FE01 FE01 FE01
      E01F E01F F10E F10E
      E001 E001 F101 F101
      FE1F FE1F FE0E FE0E
      1F01 1F01 0E01 0E01
      FEE0 FEE0 FEF1 FEF1

   Attribute Assigned Numbers

   Attributes negotiated during phase one use the following definitions.
   Phase two attributes are defined in the applicable DOI specification
   (for example, IPsec attributes are defined in the IPsec DOI), with
   the exception of a group description when Quick Mode includes an
   ephemeral Diffie-Hellman exchange. Attribute types can be either
   Basic (B) or Variable-length (V). Encoding of these attributes is
   defined in the base ISAKMP specification as Type/Value (Basic) and
   Type/Length/Value (Variable).

   Attributes described as basic MUST NOT be encoded as variable.
   Variable length attributes MAY be encoded as basic attributes if
   their value can fit into two octets. If this is the case, an
   attribute offered as variable (or basic) by the initiator of this
   protocol MAY be returned to the initiator as a basic (or variable).

Attribute Classes

```
        class                            value              type
   ------------------------------------------------------------------
    Encryption Algorithm              1                  B
    Hash Algorithm                    2                  B
    Authentication Method             3                  B
    Group Description                 4                  B
    Group Type                        5                  B
    Group Prime/Irreducible Polynomial 6                 V
    Group Generator One               7                  V
    Group Generator Two               8                  V
    Group Curve A                     9                  V
    Group Curve B                     10                 V
    Life Type                         11                 B
    Life Duration                     12                 V
    PRF                               13                 B
    Key Length                        14                 B
    Field Size                        15                 B
    Group Order                       16                 V
```

values 17-16383 are reserved to IANA. Values 16384-32767 are for
private use among mutually consenting parties.

Class Values

```
- Encryption Algorithm                      Defined In
   DES-CBC                       1          RFC 2405
   IDEA-CBC                      2
   Blowfish-CBC                  3
   RC5-R16-B64-CBC               4
   3DES-CBC                      5
   CAST-CBC                      6
```

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

```
- Hash Algorithm                            Defined In
   MD5                           1          RFC 1321
   SHA                           2          FIPS 180-1
   Tiger                         3          See Reference [TIGER]
```

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

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

   values 6-65000 are reserved to IANA. Values 65001-65535 are for
   private use among mutually consenting parties.

```
   - Group Description
      default 768-bit MODP group (section 6.1)        1

      alternate 1024-bit MODP group (section 6.2)     2

      EC2N group on GP[2^155] (section 6.3)           3

      EC2N group on GP[2^185] (section 6.4)           4
```

   values 5-32767 are reserved to IANA. Values 32768-65535 are for
   private use among mutually consenting parties.

```
   - Group Type
      MODP (modular exponentiation group)          1
      ECP  (elliptic curve group over GF[P])       2
      EC2N (elliptic curve group over GF[2^N])     3
```

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

```
   - Life Type
      seconds                           1
      kilobytes                         2
```

   values 3-65000 are reserved to IANA. Values 65001-65535 are for
   private use among mutually consenting parties. For a given "Life
   Type" the value of the "Life Duration" attribute defines the actual
   length of the SA life-- either a number of seconds, or a number of
   kbytes protected.

```
   - PRF
     There are currently no pseudo-random functions defined.
```

   values 1-65000 are reserved to IANA. Values 65001-65535 are for
   private use among mutually consenting parties.

- Key Length

  When using an Encryption Algorithm that has a variable length key,
  this attribute specifies the key length in bits. (MUST use network
  byte order). This attribute MUST NOT be used when the specified
  Encryption Algorithm uses a fixed length key.

- Field Size

  The field size, in bits, of a Diffie-Hellman group.

- Group Order

  The group order of an elliptical curve group. Note the length of
  this attribute depends on the field size.

Additional Exchanges Defined-- XCHG values
  Quick Mode   32
  New Group Mode   33

Appendix B

   This appendix describes encryption details to be used ONLY when
   encrypting ISAKMP messages. When a service (such as an IPSEC
   transform) utilizes ISAKMP to generate keying material, all
   encryption algorithm specific details (such as key and IV generation,
   padding, etc...) MUST be defined by that service. ISAKMP does not
   purport to ever produce keys that are suitable for any encryption
   algorithm. ISAKMP produces the requested amount of keying material
   from which the service MUST generate a suitable key. Details, such
   as weak key checks, are the responsibility of the service.

   Use of negotiated PRFs may require the PRF output to be expanded due
   to the PRF feedback mechanism employed by this document. For example,
   if the (ficticious) DOORAK-MAC requires 24 bytes of key but produces
   only 8 bytes of output, the output must be expanded three times
   before being used as the key for another instance of itself. The
   output of a PRF is expanded by feeding back the results of the PRF
   into itself to generate successive blocks. These blocks are
   concatenated until the requisite number of bytes has been acheived.
   For example, for pre-shared key authentication with DOORAK-MAC as the
   negotiated PRF:

     BLOCK1-8 = prf(pre-shared-key, Ni_b | Nr_b)
     BLOCK9-16 = prf(pre-shared-key, BLOCK1-8 | Ni_b | Nr_b)
     BLOCK17-24 = prf(pre-shared-key, BLOCK9-16 | Ni_b | Nr_b)
   and
     SKEYID = BLOCK1-8 | BLOCK9-16 | BLOCK17-24

   so therefore to derive SKEYID_d:

     BLOCK1-8 = prf(SKEYID, g^xy | CKY-I | CKY-R | 0)
     BLOCK9-16 = prf(SKEYID, BLOCK1-8 | g^xy | CKY-I | CKY-R | 0)
     BLOCK17-24 = prf(SKEYID, BLOCK9-16 | g^xy | CKY-I | CKY-R | 0)
   and
     SKEYID_d = BLOCK1-8 | BLOCK9-16 | BLOCK17-24

   Subsequent PRF derivations are done similarly.

   Encryption keys used to protect the ISAKMP SA are derived from
   SKEYID_e in an algorithm-specific manner. When SKEYID_e is not long
   enough to supply all the necessary keying material an algorithm
   requires, the key is derived from feeding the results of a pseudo-
   random function into itself, concatenating the results, and taking
   the highest necessary bits.

For example, if (ficticious) algorithm AKULA requires 320-bits of key
(and has no weak key check) and the prf used to generate SKEYID_e
only generates 120 bits of material, the key for AKULA, would be the
first 320-bits of Ka, where:

     Ka = K1 | K2 | K3
and
     K1 = prf(SKEYID_e, 0)
     K2 = prf(SKEYID_e, K1)
     K3 = prf(SKEYID_e, K2)

where prf is the negotiated prf or the HMAC version of the negotiated
hash function (if no prf was negotiated) and 0 is represented by a
single octet. Each result of the prf provides 120 bits of material
for a total of 360 bits. AKULA would use the first 320 bits of that
360 bit string.

In phase 1, material for the initialization vector (IV material) for
CBC mode encryption algorithms is derived from a hash of a
concatenation of the initiator's public Diffie-Hellman value and the
responder's public Diffie-Hellman value using the negotiated hash
algorithm. This is used for the first message only. Each message
should be padded up to the nearest block size using bytes containing
0x00. The message length in the header MUST include the length of the
pad since this reflects the size of the ciphertext. Subsequent
messages MUST use the last CBC encryption block from the previous
message as their initialization vector.

In phase 2, material for the initialization vector for CBC mode
encryption of the first message of a Quick Mode exchange is derived
from a hash of a concatenation of the last phase 1 CBC output block
and the phase 2 message id using the negotiated hash algorithm. The
IV for subsequent messages within a Quick Mode exchange is the CBC
output block from the previous message. Padding and IVs for
subsequent messages are done as in phase 1.

After the ISAKMP SA has been authenticated all Informational
Exchanges are encrypted using SKEYID_e. The initiaization vector for
these exchanges is derived in exactly the same fashion as that for a
Quick Mode-- i.e. it is derived from a hash of a concatenation of the
last phase 1 CBC output block and the message id from the ISAKMP
header of the Informational Exchange (not the message id from the
message that may have prompted the Informational Exchange).

Note that the final phase 1 CBC output block, the result of
encryption/decryption of the last phase 1 message, must be retained
in the ISAKMP SA state to allow for generation of unique IVs for each
Quick Mode. Each post- phase 1 exchange (Quick Modes and

Informational Exchanges) generates IVs independantly to prevent IVs
from getting out of sync when two different exchanges are started
simultaneously.

In all cases, there is a single bidirectional cipher/IV context.
Having each Quick Mode and Informational Exchange maintain a unique
context prevents IVs from getting out of sync.

The key for DES-CBC is derived from the first eight (8) non-weak and
non-semi-weak (see Appendix A) bytes of SKEYID_e. The IV is the first
8 bytes of the IV material derived above.

The key for IDEA-CBC is derived from the first sixteen (16) bytes of
SKEYID_e. The IV is the first eight (8) bytes of the IV material
derived above.

The key for Blowfish-CBC is either the negotiated key size, or the
first fifty-six (56) bytes of a key (if no key size is negotiated)
derived in the aforementioned pseudo-random function feedback method.
The IV is the first eight (8) bytes of the IV material derived above.

The key for RC5-R16-B64-CBC is the negotiated key size, or the first
sixteen (16) bytes of a key (if no key size is negotiated) derived
from the aforementioned pseudo-random function feedback method if
necessary. The IV is the first eight (8) bytes of the IV material
derived above. The number of rounds MUST be 16 and the block size
MUST be 64.

The key for 3DES-CBC is the first twenty-four (24) bytes of a key
derived in the aforementioned pseudo-random function feedback method.
3DES-CBC is an encrypt-decrypt-encrypt operation using the first,
middle, and last eight (8) bytes of the entire 3DES-CBC key. The IV
is the first eight (8) bytes of the IV material derived above.

The key for CAST-CBC is either the negotiated key size, or the first
sixteen (16) bytes of a key derived in the aforementioned pseudo-
random function feedback method. The IV is the first eight (8) bytes
of the IV material derived above.

Support for algorithms other than DES-CBC is purely optional. Some
optional algorithms may be subject to intellectual property claims.

Authors' Addresses

    Dan Harkins
    cisco Systems
    170 W. Tasman Dr.
    San Jose, California, 95134-1706
    United States of America

    Phone: +1 408 526 4000
    EMail: dharkins@cisco.com

    Dave Carrel
    76 Lippard Ave.
    San Francisco, CA 94131-2947
    United States of America

    Phone: +1 415 337 8469
    EMail: carrel@ipsec.org

Authors' Note

    The authors encourage independent implementation, and
    interoperability testing, of this hybrid protocol.

Full Copyright Statement