                    The Group Domain of Interpretation

Status of this Memo

Copyright Notice

Abstract

   This document presents an ISAMKP Domain of Interpretation (DOI) for
   group key management to support secure group communications. The
   GDOI manages group security associations, which are used by IPSEC and
   potentially other data security protocols running at the IP or
   application layers. These security associations protect one or more
   key-encrypting keys, traffic-encrypting keys, or data shared by group
   members.

Table of Contents

1. Introduction

   This document presents an ISAMKP Domain of Interpretation (DOI) for
   group key management called the "Group Domain of Interpretation"
   (GDOI). In this group key management model, the GDOI protocol is run
   between a group member and a "group controller/key server" (GCKS),
   which establishes security associations [Section 4.6.2 RFC2401] among
   authorized group members. ISAKMP defines two "phases" of negotiation
   [p.16 RFC2408]. The GDOI MUST be protected by a Phase 1 security
   association. This document incorporates the Phase 1 security
   association (SA) definition from the Internet DOI [RFC2407, RFC2409].
   Other possible Phase 1 security association types are noted in
   Appendix A. The Phase 2 exchange is defined in this document, and
   proposes new payloads and exchanges according to the ISAKMP standard
   [p. 14 RFC2408].

   There are six new payloads:

      1) GDOI SA
      2) SA KEK (SAK) which follows the SA payload
      3) SA TEK (SAT) which follows the SA payload

          4) Key Download Array (KD)
          5) Sequence number (SEQ)
          6) Proof of Possession (POP)

     There are two new exchanges.

     1) A Phase 2 exchange creates Re-key and Data-Security Protocol SAs.

     The new Phase 2 exchange, called "GROUPKEY-PULL," downloads keys for
     a group's "Re-key" SA and/or "Data-security" SA. The Re-key SA
     includes a key encrypting key, or KEK, common to the group; a
     Data-security SA includes a data encryption key, or TEK, used by a
     data-security protocol to encrypt or decrypt data traffic [Section
     2.1 RFC2407]. The SA for the KEK or TEK includes authentication
     keys, encryption keys, cryptographic policy, and attributes. The
     GROUPKEY-PULL exchange uses "pull" behavior since the member
     initiates the retrieval of these SAs from a GCKS.

     2) A datagram subsequently establishes additional Rekey and/or
     Data-Security Protocol SAs.

     The GROUPKEY-PUSH datagram is "pushed" from the GCKS to the members
     to create or update a Re-key or Data-security SA. A Re-key SA
     protects GROUPKEY-PUSH messages. Thus, a GROUPKEY-PULL is necessary
     to establish at least one Re-key SA in order to protect subsequent
     GROUPKEY-PUSH messages. The GCKS encrypts the GROUPKEY-PUSH message
     using the KEK Re-key SA. GDOI accommodates the use of arrays of KEKs
     for group key management algorithms using the Logical Key Hierarchy
     (LKH) algorithm to efficiently add and remove group members
     [RFC2627]. Implementation of the LKH algorithm is OPTIONAL.

     Although the GROUPKEY-PUSH specified by this document can be used to
     refresh a Re-key SA, the most common use of GROUPKEY-PUSH is to
     establish a Data-security SA for a data security protocol. GDOI can
     accommodate future extensions to support a variety of data security
     protocols. This document only specifies data-security SAs for one
     security protocol, IPsec ESP. A separate RFC will specify support
     for other data security protocols such as a future secure Real-time
     Transport Protocol. A security protocol uses the TEK and "owns" the
     data-security SA in the same way that IPsec ESP uses the IKE Phase 2
     keys and owns the Phase 2 SA; for GDOI, IPsec ESP uses the TEK.

     Thus, GDOI is a group security association management protocol: All
     GDOI messages are used to create, maintain, or delete security
     associations for a group. As described above, these security
     associations protect one or more key-encrypting keys,
     traffic-encrypting keys, or data shared by group members for
     multicast and groups security applications.

   The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD,
   SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this
   document, are to be interpreted as described in BCP 14, RFC 2119
   [RFC2119].

1.1. GDOI Applications

   Secure multicast applications include video broadcast and multicast
   file transfer. In a business environment, many of these applications
   require network security and may use IPsec ESP to secure their data
   traffic. Section 5.4.1 specifies how GDOI carries the needed SA
   parameters for ESP. In this way, GDOI supports multicast ESP with
   group authentication of ESP packets using the shared, group key
   (authentication of unique sources of ESP packets is not possible).

   GDOI can also secure group applications that do not use multicast
   transport such as video-on-demand. For example, the GROUPKEY-PUSH
   message may establish a pair-wise IPsec ESP SA for a member of a
   subscription group without the need for key management exchanges and
   costly asymmetric cryptography.

1.2. Extending GDOI

   Not all secure multicast or multimedia applications can use IPsec
   ESP. Many Real Time Transport Protocol applications, for example,
   require security above the IP layer to preserve RTP header
   compression efficiencies and transport-independence [RFC3550]. A
   future RTP security protocol may benefit from using GDOI to establish
   group SAs.

   In order to add a new data security protocol, a new RFC MUST specify
   the data-security SA parameters conveyed by GDOI for that security
   protocol; these parameters are listed in section 5.4.2 of this
   document.

   Data security protocol SAs MUST protect group traffic. GDOI provides
   no restriction on whether that group traffic is transmitted as
   unicast or multicast packets. However, GDOI MUST NOT be used as a
   key management mechanism by a data security protocol when the packets
   protected by the data-security SA are intended to be private and
   never become part of group communications.

2. GDOI Phase 1 protocol

   GDOI is a "phase 2" protocol which MUST be protected by a "phase 1"
   protocol. The "phase 1" protocol can be any protocol which provides
   for the following protections:

   o Peer Authentication
   o Confidentiality
   o Message Integrity

   The following sections describe one such "phase 1" protocol. Other
   protocols which may be potential "phase 1" protocols are described in
   Appendix A. However, the use of the protocols listed there are not
   considered part of this document.

2.1. ISAKMP Phase 1 protocol

   This document defines how the ISAKMP phase 1 exchanges as defined in
   [RFC2409] can be used a "phase 1" protocol for GDOI. The following
   sections define characteristics of the ISAKMP phase 1 protocols that
   are unique for these exchanges when used for GDOI.

   Section 6.1 describes how the ISAKMP Phase 1 protocols meet the
   requirements of a GDOI "phase 1" protocol.

2.1.1. DOI value

   The Phase 1 SA payload has a DOI value. That value MUST be the GDOI
   DOI value as defined later in this document.

2.1.2. UDP port

   GDOI MUST NOT run on port 500 (the port commonly used for IKE). IANA
   has assigned port 848 for the use of GDOI.

3. GROUPKEY-PULL Exchange

   The goal of the GROUPKEY-PULL exchange is to establish a Re-key
   and/or Data-security SAs at the member for a particular group. A
   Phase 1 SA protects the GROUPKEY-PULL; there MAY be multiple
   GROUPKEY-PULL exchanges for a given Phase 1 SA. The GROUPKEY-PULL
   exchange downloads the data security keys (TEKs) and/or group key
   encrypting key (KEK) or KEK array under the protection of the Phase 1
   SA.

3.1. Authorization

   There are two alternative means for authorizing the GROUPKEY-PULL
   message. First, the Phase 1 identity can be used to authorize the
   Phase 2 (GROUPKEY-PULL) request for a group key. Second, a new
   identity can be passed in the GROUPKEY-PULL request. The new
   identity could be specific to the group and use a certificate that is
   signed by the group owner to identify the holder as an authorized
   group member. The Proof-of-Possession payload validates that the
   holder possesses the secret key associated with the Phase 2 identity.

3.2. Messages

   The GROUPKEY-PULL is a Phase 2 exchange. Phase 1 computes SKEYID_a
   which is the "key" in the keyed hash used in the GROUPKEY-PULL HASH
   payloads. When using the Phase 1 defined in this document, SKEYID_a
   is derived according to [RFC2409]. As with the IKE HASH payload
   generation [RFC 2409 section 5.5], each GROUPKEY-PULL message hashes
   a uniquely defined set of values. Nonces permute the HASH and
   provide some protection against replay attacks. Replay protection is
   important to protect the GCKS from attacks that a key management
   server will attract.

   The GROUPKEY-PULL uses nonces to guarantee "liveliness", or against
   replay of a recent GROUPKEY-PULL message. The replay attack is only
   useful in the context of the current Phase 1. If a GROUPKEY-PULL
   message is replayed based on a previous Phase 1, the HASH calculation
   will fail due to a wrong SKEYID_a. The message will fail processing
   before the nonce is ever evaluated. In order for either peer to get
   the benefit of the replay protection, it must postpone as much
   processing as possible until it receives the message in the protocol
   that proves the peer is live. For example, the Responder MUST NOT
   compute the shared Diffie-Hellman number (if KE payloads were
   included) or install the new SAs until it receives a message with Nr
   included properly in the HASH payload.

   Nonces require an additional message in the protocol exchange to
   ensure that the GCKS does not add a group member until it proves
   liveliness. The GROUPKEY-PULL member-initiator expects to find its
   nonce, Ni, in the HASH of a returned message. And the GROUPKEY-PULL
   GKCS responder expects to see its nonce, Nr, in the HASH of a
   returned message before providing group-keying material as in the
   following exchange.

```
          Initiator (Member)                  Responder (GCKS)
          ------------------                  ----------------
          HDR*, HASH(1), Ni, ID      -->
                                     <--      HDR*, HASH(2), Nr, SA
          HDR*, HASH(3) [,KE_I]      -->
             [,CERT] [,POP_I]
                                     <--      HDR*, HASH(4),[KE_R,][SEQ,]
                                                KD [,CERT] [,POP_R]
```

Hashes are computed as follows:

```
  HASH(1) = prf(SKEYID_a, M-ID | Ni | ID)
  HASH(2) = prf(SKEYID_a, M-ID | Ni_b | Nr | SA)
  HASH(3) = prf(SKEYID_a, M-ID | Ni_b | Nr_b [ | KE_I ] [ | CERT ]
            [ | POP_I ])
  HASH(4) = prf(SKEYID_a, M-ID | Ni_b | Nr_b [ | KE_R ] [ | SEQ | ]
            KD [ | CERT ] [ | POP_R])
```

   POP payload is constructed as described in Section 5.7.
   * Protected by the Phase 1 SA, encryption occurs after HDR

HDR is an ISAKMP header payload that uses the Phase 1 cookies and a
message identifier (M-ID) as in IKE [RFC2409]. Note that nonces are
included in the first two exchanges, with the GCKS returning only the
SA policy payload before liveliness is proven. The HASH payloads
[RFC2409] prove that the peer has the Phase 1 secret (SKEYID_a) and
the nonce for the exchange identified by message id, M-ID. Once
liveliness is established, the last message completes the real
processing of downloading the KD payload.

In addition to the Nonce and HASH payloads, the member-initiator
identifies the group it wishes to join through the ISAKMP ID payload.
The GCKS responder informs the member of the current value of the
sequence number in the SEQ payload; the sequence number orders the
GROUPKEY-PUSH datagrams (section 4); the member MUST check to see
that the sequence number is greater than in the previous SEQ payload
the member holds for the group (if it holds any) before installing
any new SAs. The SEQ payload MUST be present if the SA payload
contains an SA KEK attribute. The GCKS responder informs the member
of the cryptographic policies of the group in the SA payload, which
describes the DOI, KEK and/or TEK keying material, and authentication
transforms. The SPIs are also determined by the GCKS and downloaded
in the SA payload chain (see section 5.2). The SA KEK attribute
contains the ISAKMP cookie pair for the Re-key SA, which is not
negotiated but downloaded. The SA TEK attribute contains an SPI as
defined in section 5.4 of this document. The second message
downloads this SA payload. If a Re-key SA is defined in the SA
payload, then KD will contain the KEK; if one or more Data-security

SAs are defined in the SA payload, KD will contain the TEKs. This is
useful if there is an initial set of TEKs for the particular group
and can obviate the need for future TEK GROUPKEY-PUSH messages
(described in section 4).

As described above, the member may establish an identity in the
GROUPKEY-PULL exchange in an optional CERT payload that is separate
from the Phase 1 identity. When the member passes a new CERT, a
proof of possession (POP) payload accompanies it. The POP payload
demonstrates that the member or GCKS has used the very secret that
authenticates it. POP_I is an ISAKMP SIG payload containing a hash
including the nonces Ni and Nr signed by the member, when the member
passes a CERT, signed by the Group Owner to prove its authorization.
POP_R contains the hash including the concatenated nonces Ni and Nr
signed by the GCKS, when the GCKS passes a CERT, signed by the group
owner, to prove its authority to provide keys for a particular group.
The use of the nonce pair for the POP payload, transformed through a
pseudo-random function (prf) and encrypted, is designed to withstand
compromise of the Phase 1 key. Implementation of the CERT and POP
payloads is OPTIONAL.

3.2.1. Perfect Forward Secrecy

If PFS is desired and the optional KE payload is used in the
exchange, then both sides compute a DH secret and use it to protect
the new keying material contained in KD. The GCKS responder will xor
the DH secret with the KD payload and send it to the member
Initiator, which recovers the KD by repeating this operation as in
the Oakley IEXTKEY procedure [RFC2412]. Implementation of the KE
payload is OPTIONAL.

3.2.2. ISAKMP Header Initialization

Cookies are used in the ISAKMP header as a weak form of denial of
service protection. The GDOI GROUPKEY-PULL exchange uses cookies
according to ISAKMP [RFC2408].

Next Payload identifies an ISAKMP or GDOI payload (see Section 5.0).

Major Version is 1 and Minor Version is 0 according to ISAKMP
[RFC2408, Section 3.1].

The Exchange Type has value 32 for the GDOI GROUPKEY-PULL exchange.

Flags, Message ID, and Length are according to ISAKMP [RFC2408,
Section 3.1]

3.3. Initiator Operations

   Before a group member (GDOI initiator) contacts the GCKS, it must
   determine the group identifier and acceptable Phase 1 policy via an
   out-of-band method such as SDP. Phase 1 is initiated using the GDOI
   DOI in the SA payload. Once Phase 1 is complete, the initiator state
   machine moves to the GDOI protocol.

   To construct the first GDOI message the initiator chooses Ni and
   creates a nonce payload, builds an identity payload including the
   group identifier, and generates HASH(1).

   Upon receipt of the second GDOI message, the initiator validates
   HASH(2), extracts the nonce Nr, and interprets the SA payload. If
   the policy in the SA payload is acceptable (e.g., the security
   protocol and cryptographic protocols can be supported by the
   initiator), the initiator continues the protocol.

   If the group policy uses certificates for authorization, the
   initiator generates a hash including Ni and Nr and signs it. This
   becomes the contents of the POP payload. If necessary, a CERT
   payload is constructed which holds the public key corresponding to
   the private key used to sign the POP payload.

   The initiator constructs the third GDOI message by including the CERT
   and POP payloads (if needed) and creating HASH(3).

   Upon receipt of the fourth GDOI message, the initiator validates
   HASH(4). If the responder sent CERT and POP_R payloads, the POP
   signature is validated.

   If SEQ payload is present, the sequence number in the SEQ payload
   must be checked against any previously received sequence number for
   this group. If it is less than the previously received number, it
   should be considered stale and ignored. This could happen if two
   GROUPKEY-PULL messages happened in parallel, and the sequence number
   changed between the times the results of two GROUPKEY-PULL messages
   were returned from the GCKS.

   The initiator interprets the KD key packets, matching the SPIs in the
   key packets to SPIs previously sent in the SA payloads identifying
   particular policy. For TEKs, once the keys and policy are matched,
   the initiator is ready to send or receive packets matching the TEK
   policy. (If policy and keys had been previously received for this
   TEK policy, the initiator may decide instead to ignore this TEK
   policy in case it is stale.) If this group has a KEK, the KEK policy
   and keys are marked as ready for use.

3.4. Receiver Operations

   The GCKS (responder) passively listens for incoming requests from
   group members. The Phase 1 authenticates the group member and sets
   up the secure session with them.

   Upon receipt of the first GDOI message the GCKS validates HASH(1),
   extracts the Ni and group identifier in the ID payload. It verifies
   that its database contains the group information for the group
   identifier.

   The GCKS constructs the second GDOI message, including a nonce Nr,
   and the policy for the group in an SA payload, followed by SA TEK
   payloads for traffic SAs, and SA KEK policy (if the group controller
   will be sending Re-key messages to the group).

   Upon receipt of the third GDOI message the GCKS validates HASH(3).
   If the initiator sent CERT and POP_I payloads, the POP signature is
   validated.

   The GCKS constructs the fourth GDOI message, including the SEQ
   payload (if the GCKS sends rekey messages), the KD payload containing
   keys corresponding to policy previously sent in the SA TEK and SA KEK
   payloads, and the CERT and POP payloads (if needed).

4. GROUPKEY-PUSH Message

   GDOI sends control information securely using group communications.
   Typically this will be using IP multicast distribution of a
   GROUPKEY-PUSH message but it can also be "pushed" using unicast
   delivery if IP multicast is not possible. The GROUPKEY-PUSH message
   replaces a Re-key SA KEK or KEK array, and/or creates a new
   Data-security SA.

              Member   GCKS or Delegate
              ------   ----------------

                        <---- HDR*, SEQ, SA, KD, [CERT,] SIG

   * Protected by the Re-key SA KEK; encryption occurs after HDR

   HDR is defined below. The SEQ payload is defined in the Payloads
   section. The SA defines the policy (e.g., protection suite) and
   attributes (e.g., SPI) for a Re-key and/or Data-security SAs. The
   GCKS or delegate optionally provides a CERT payload for verification
   of the SIG. KD is the key download payload as described in the
   Payloads section.

The SIG payload is a signature of a hash of the entire message before
encryption (including the header and excluding the SIG payload
itself), prefixed with the string "rekey". The prefixed string
ensures that the signature of the Rekey datagram cannot be used for
any other purpose in the GDOI protocol.

If the SA defines an LKH KEK array or single KEK, KD contains a KEK
or KEK array for a new Re-key SA, which has a new cookie pair. When
the KD payload carries a new SA KEK attribute (section 5.3), a Re-key
SA is replaced with a new SA having the same group identifier (ID
specified in message 1 of section 3.2) and incrementing the same
sequence counter, which is initialized in message 4 of section 3.2.
If the SA defines an SA TEK payload, this informs the member that a
new Data-security SA has been created, with keying material carried
in KD (Section 5.5).

If the SA defines a large LKH KEK array (e.g., during group
initialization and batched rekeying), parts of the array MAY be sent
in different unique GROUPKEY-PUSH datagrams. However, each of the
GROUPKEY-PUSH datagrams MUST be a fully formed GROUPKEY-PUSH
datagram. This results in each datagram containing a sequence number
and the policy in the SA payload, which corresponds to the KEK array
portion sent in the KD payload.

4.1. Perfect Forward Secrecy (PFS)

The GROUPKEY-PUSH message is protected by the group KEK though in all
cases, the GROUPKEY-PUSH message carries new key downloads, among
other information. A freshly generated secret must protect the key
download for the GROUPKEY-PUSH message to have PFS. This issue is
for further study.

4.2. Forward and Backward Access Control

Through GROUPKEY-PUSH, the GDOI supports algorithms such as LKH that
have the property of denying access to a new group key by a member
removed from the group (forward access control) and to an old group
key by a member added to the group (backward access control). An
unrelated notion to PFS, "forward access control" and "backward
access control" have been called "perfect forward security" and
"perfect backward security" in the literature [RFC2627].

Group management algorithms providing forward and backward access
control other than LKH have been proposed in the literature,
including OFT [OFT] and Subset Difference [NNL]. These algorithms
could be used with GDOI, but are not specified as a part of this
document.

Support for group management algorithms is supported via the
KEY_MANAGEMENT_ALGORITHM attribute which is sent in the SA_KEK
payload. GDOI specifies one method by which LKH can be used for
forward and backward access control. Other methods of using LKH, as
well as other group management algorithms such as OFT or Subset
Difference may be added to GDOI as part of a later document. Any
such addition MUST be due to a Standards Action as defined in
[RFC2434].

4.2.1. Forward Access Control Requirements

When group membership is altered using a group management algorithm
new SA_TEKs (and their associated keys) are usually also needed. New
SAs and keys ensure that members who were denied access can no longer
participate in the group.

If forward access control is a desired property of the group, new
SA_TEKs and the associated key packets in the KD payload MUST NOT be
included in a GROUPKEY-PUSH message which changes group membership.
This is required because the SA_TEK policy and the associated key
packets in the KD payload are not protected with the new KEK. A
second GROUPKEY-PUSH message can deliver the new SA_TEKS and their
associated keys because it will be protected with the new KEK, and
thus will not be visible to the members who were denied access.

If forward access control policy for the group includes keeping group
policy changes from members that are denied access to the group, then
two sequential GROUPKEY-PUSH messages changing the group KEK MUST be
sent by the GCKS. The first GROUPKEY-PUSH message creates a new KEK
for the group. Group members, which are denied access, will not be
able to access the new KEK, but will see the group policy since the
GROUPKEY-PUSH message is protected under the current KEK. A
subsequent GROUPKEY-PUSH message containing the changed group policy
and again changing the KEK allows complete forward access control. A
GROUPKEY-PUSH message MUST NOT change the policy without creating a
new KEK.

If other methods of using LKH or other group management algorithms
are added to GDOI, those methods MAY remove the above restrictions
requiring multiple GROUPKEY-PUSH messages, providing those methods
specify how forward access control policy is maintained within a
single GROUPKEY-PUSH message.

4.3. Delegation of Key Management

   GDOI supports delegation of GROUPKEY-PUSH datagrams through the
   delegation capabilities of the PKI. However, GDOI does not
   explicitly specify how the GCKS identifies delegates, but leaves this
   to the PKI that is used by a particular GDOI implementation.

4.4. Use of signature keys

   The GCKS SHOULD NOT use the same key to sign the SIG payload in the
   GROUPKEY-PUSH message as was used for authorization in the
   GROUPKEY-PULL POP payload. If the same key must be used, a different
   hash function SHOULD be used as a base for the POP payload than is
   used as a base for the SIG payload.

4.5. ISAKMP Header Initialization

   Unlike ISAKMP or IKE, the cookie pair is completely determined by the
   GCKS. The cookie pair in the GDOI ISAKMP header identifies the Re-
   key SA to differentiate the secure groups managed by a GCKS. Thus,
   GDOI uses the cookie fields as an SPI.

   Next Payload identifies an ISAKMP or GDOI payload (see Section 5.0).

   Major Version is 1 and Minor Version is 0 according to ISAKMP
   [RFC2408, Section 3.1].

   The Exchange Type has value 33 for the GDOI GROUPKEY-PUSH message.

   Flags MUST have the Encryption bit set according to [RFC2008, Section
   3.1]. All other bits MUST be set to zero.

   Message ID MUST be set to zero.

   Length is according to ISAKMP [RFC2408, Section 3.1]

4.6. Deletion of SAs

   There are times the GCKS may want to signal to receivers to delete
   SAs, for example at the end of a broadcast. Deletion of keys may be
   accomplished by sending an ISAKMP Delete payload [RFC2408, Section
   3.15] as part of a GDOI GROUPKEY-PUSH message.

   One or more Delete payloads MAY be placed following the SEQ payload
   in a GROUPKEY-PUSH message. If a GCKS has no further SAs to send to
   group members, the SA and KD payloads MUST be omitted from the
   message.

The following fields of the Delete Payload are further defined as follows:

o   The Domain of Interpretation field contains the GDOI DOI.

o   The Protocol-Id field contains TEK protocol id values defined
    in Section 5.4 of this document. To delete a KEK SA, the value
    of zero MUST be used as the protocol id. Note that only one
    protocol id value can be defined in a Delete payload. If a TEK
    SA and a KEK SA must be deleted, they must be sent in different
    Delete payloads.

4.7. GCKS Operations

GCKS or its delegate may initiate a Rekey message for one of several
reasons, e.g., the group membership has changed or keys are due to
expire.

To begin the rekey datagram the GCKS builds an ISAKMP HDR with the
correct cookie pair, and a SEQ payload that includes a sequence
number which is one greater than the previous rekey datagram.

An SA payload is then added. This is identical in structure and
meaning to a SA payload sent in a GROUPKEY-PULL exchange. If there
are changes to the KEK (in the case of a static KEK) or in group
membership (in the case of LKH) an SA_KEK attribute is added to the
SA. If there are one or more new TEKs then SA_TEK attributes are
added to describe that policy.

A KD payload is then added. This is identical in structure and
meaning to a KD payload sent in a GROUPKEY-PULL exchange. If an
SA_KEK attribute was included in the SA payload then corresponding
KEK keys (or a KEK array) is included. TEK keys are sent for each
SA_TEK attribute included in the SA payload.

A CERT payload is added if the initiator needs to provide its
certificate.

In the penultimate step, the initiator hashes the string "rekey"
followed by the key management message already formed. The hash is
signed, placed in a SIG payload and added to the datagram.

Lastly, the payloads following the HDR are encrypted using the
current KEK encryption key. The datagram can now be sent.

4.8. Group Member Operations

   A group member receiving the GROUPKEY-PUSH datagram matches the
   cookie pair in the ISAKMP HDR to an existing SA. The message is
   decrypted, and the form of the datagram is validated. This weeds out
   obvious ill-formed messages (which may be sent as part of a Denial of
   Service attack on the group).

   The signature of the decrypted message is then validated, possibly
   using the CERT payload if it is included.

   The sequence number in the SEQ payload is validated to ensure that it
   is greater than the previously received sequence number, and that it
   fits within a window of acceptable values.

   The SA and KD payloads are processed which results in a new GDOI
   Rekey SA (if the SA payload included an SA_KEK attribute) and/or new
   IPsec SAs being added to the system.

5. Payloads and Defined Values

   This document specifies use of several ISAKMP payloads, which are
   defined in accordance with RFC2408. The following payloads are
   extended or further specified.

           Next Payload Type   Value
           -----------------   -----
           Security Association (SA)   1
           Identification (ID)   5
           Nonce (N)   10

   Several new payload formats are required in the group security
   exchanges.

           Next Payload Type   Value
           -----------------   -----
           SA KEK Payload (SAK)   15
           SA TEK Payload (SAT)   16
           Key Download (KD)   17
           Sequence Number (SEQ)   18
           Proof of Possession (POP)   19

5.1. Identification Payload

   The Identification Payload is used to identify a group identity that
   will later be associated with Security Associations for the group. A
   group identity may map to a specific IP multicast group, or may
   specify a more general identifier, such as one that represents a set
   of related multicast streams.

   The Identification Payload is defined as follows:

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   ! Next Payload !   RESERVED    !          Payload Length        !
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   !  ID Type     !                     RESERVE2                   !
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   ~                     Identification Data                       ~
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
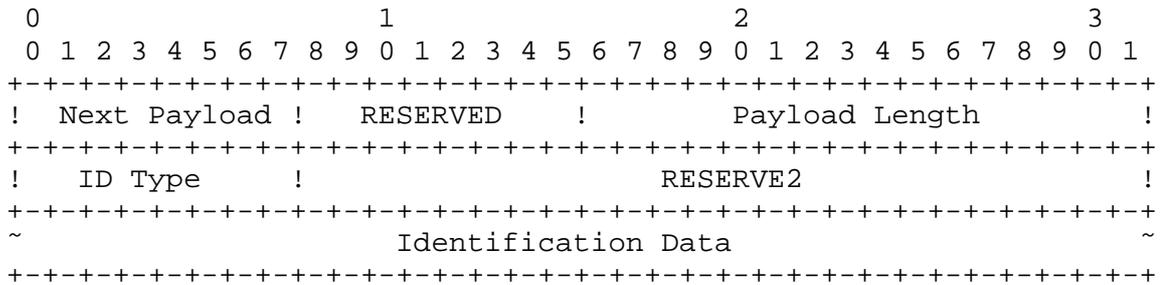
   The Identification Payload fields are defined as follows:

   o Next Payload (1 octet) -- Identifier for the payload type of
      the next payload in the message. If the current payload is the
      last in the message, this field will be zero (0).

   o RESERVED (1 octet) -- Unused, must be zero (0).

   o Payload Length (2 octets) -- Length, in octets, of the
      identification data, including the generic header.

   o Identification Type (1 octet) -- Value describing the identity
      information found in the Identification Data field.

   o RESERVED2 (2 octets) -- Unused, must be zero (0).

   o Identification Data (variable length) -- Value, as indicated by
      the Identification Type.

5.1.1. Identification Type Values

   The following table lists the assigned values for the Identification
   Type field found in the Identification Payload.

            ID Type   Value
            -------   -----
            RESERVED   0 - 10
            ID_KEY_ID   11
            RESERVED   12 - 127
            Private Use   128 - 255

5.1.1.1. ID_KEY_ID

   In the context of a GDOI ID payload, ID_KEY_ID specifies a four
   (4)-octet group identifier.

5.2. Security Association Payload

   The Security Association payload is defined in RFC 2408. For the
   GDOI, it is used by the GCKS to assert security attributes for both
   Re-key and Data-security SAs.

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
   ! Next Payload  !   RESERVED    !          Payload Length       !
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   !                              DOI                              !
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
   !                           Situation                          !
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
   ! SA Attribute Next Payload     !            RESERVED2          !
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
```

   The Security Association Payload fields are defined as follows:

      o Next Payload (1 octet) -- Identifies the next payload for the
         GROUPKEY-PULL or the GROUPKEY-PUSH message as defined above.
         The next payload MUST NOT be a SAK Payload or SAT Payload type,
         but the next non-Security Association type payload.

      o RESERVED (1 octet) -- Must be zero.

      o Payload Length (2 octets) -- Is the octet length of the current
         payload including the generic header and all TEK and KEK
         payloads.

        o DOI (4 octets) -- Is the GDOI, which is value 2.

        o Situation (4 octets) -- Must be zero.

        o SA Attribute Next Payload (1 octet) -- Must be either a SAK
           Payload or a SAT Payload. See section 5.2.1 for a description
           of which circumstances are required for each payload type to be
           present.

        o RESERVED (2 octets) -- Must be zero.

5.2.1. Payloads following the SA payload

   Payloads that define specific security association attributes for the
   KEK and/or TEKs used by the group MUST follow the SA payload. How
   many of each payload is dependent upon the group policy. There may
   be zero or one SAK Payloads, and zero or more SAT Payloads, where
   either one SAK or SAT payload MUST be present.

   This latitude allows various group policies to be accommodated. For
   example if the group policy does not require the use of a Re-key SA,
   the GCKS would not need to send an SA KEK attribute to the group
   member since all SA updates would be performed using the Registration
   SA. Alternatively, group policy might use a Re-key SA but choose to
   download a KEK to the group member only as part of the Registration
   SA. Therefore, the KEK policy (in the SA KEK attribute) would not be
   necessary as part of the Re-key SA message SA payload.

   Specifying multiple SATs allows multiple sessions to be part of the
   same group and multiple streams to be associated with a session
   (e.g., video, audio, and text) but each with individual security
   association policy.

5.3. SA KEK payload

   The SA KEK (SAK) payload contains security attributes for the KEK
   method for a group and parameters specific to the GROUPKEY-PULL
   operation. The source and destination identities describe the
   identities used for the GROUPKEY-PULL datagram.

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
   ! Next Payload  !   RESERVED    !           Payload Length      !
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
   !    Protocol   ! SRC ID Type   !           SRC ID Port         !
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
   !SRC ID Data Len!          SRC Identification Data              ~
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
   ! DST ID Type   !          DST ID Port          !DST ID Data Len!
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
   !                     DST Identification Data                   ~
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
   !                                                               !
   ~                             SPI                               ~
   !                                                               !
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
   !          POP Algorithm        !        POP Key Length         !
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
   ~                         KEK Attributes                        ~
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
```
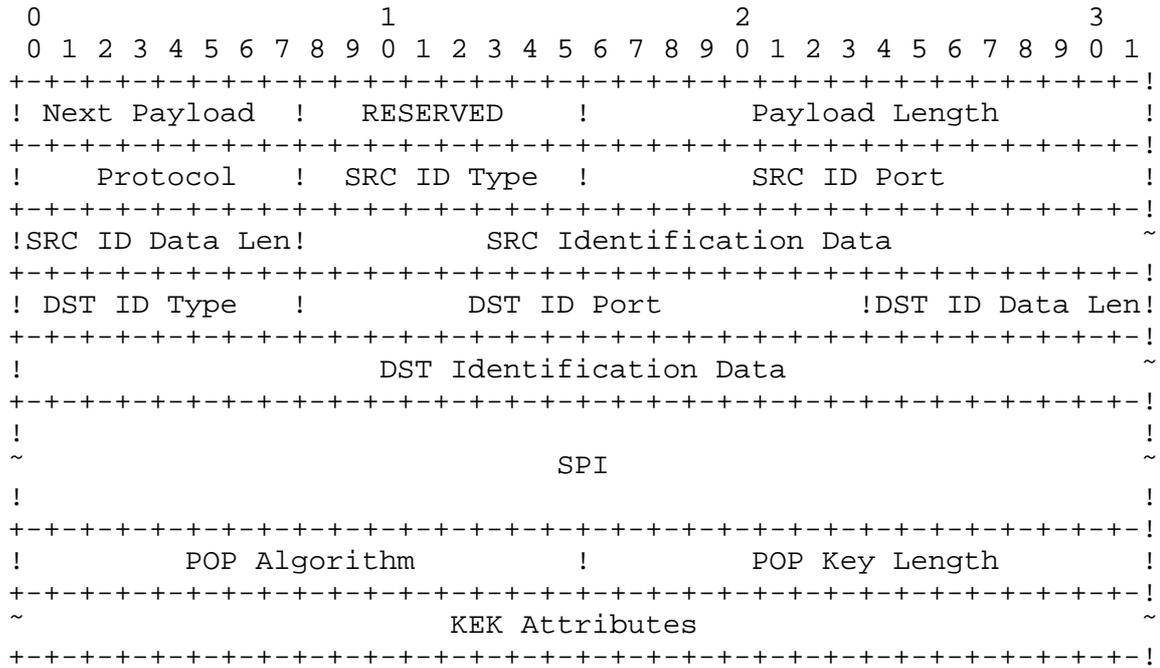
The SAK Payload fields are defined as follows:

   o Next Payload (1 octet) -- Identifies the next payload for the
      GROUPKEY-PULL or the GROUPKEY-PUSH message. The only valid
      next payload types for this message are a SAT Payload or zero
      to indicate there is no SA TEK payload.

   o RESERVED (1 octet) -- Must be zero.

   o Payload Length (2 octets) -- Length of this payload, including
      the KEK attributes.

   o Protocol (1 octet) -- Value describing an IP protocol ID (e.g.,
      UDP/TCP) for the rekey datagram.

   o SRC ID Type (1 octet) -- Value describing the identity
      information found in the SRC Identification Data field.
      Defined values are specified by the IPSEC Identification Type
      section in the IANA isakmpd-registry [ISAKMP-REG].

   o SRC ID Port (2 octets) -- Value specifying a port associated
      with the source Id. A value of zero means that the SRC ID Port
      field should be ignored.

   o SRC ID Data Len (1 octet) -- Value specifying the length of the
      SRC Identification Data field.

   o SRC Identification Data (variable length) -- Value, as
      indicated by the SRC ID Type.

   o DST ID Type (1 octet) -- Value describing the identity
      information found in the DST Identification Data field.
      Defined values are specified by the IPSEC Identification Type
      section in the IANA isakmpd-registry [ISAKMP-REG].

   o DST ID Prot (1 octet) -- Value describing an IP protocol ID
      (e.g., UDP/TCP).

   o DST ID Port (2 octets) -- Value specifying a port associated
      with the source Id.

   o DST ID Data Len (1 octet) -- Value specifying the length of the
      DST Identification Data field.

   o DST Identification Data (variable length) -- Value, as
      indicated by the DST ID Type.

   o SPI (16 octets) -- Security Parameter Index for the KEK. The
      SPI must be the ISAKMP Header cookie pair where the first 8
      octets become the "Initiator Cookie" field of the GROUPKEY-PUSH
      message ISAKMP HDR, and the second 8 octets become the
      "Responder Cookie" in the same HDR. As described above, these
      cookies are assigned by the GCKS.

   o POP Algorithm (2 octets) -- The POP payload algorithm. Defined
      values are specified in the following table. If no POP
      algorithm is defined by the KEK policy this field must be zero.

              Algorithm Type   Value
              --------------   -----
              RESERVED   0
              POP_ALG_RSA   1
              POP_ALG_DSS   2
              POP_ALG_ECDSS   3
              RESERVED   4-127
              Private Use   128-255

   o POP Key Length (2 octets) -- Length of the POP payload key. If
      no POP algorithm is defined in the KEK policy, this field must
      be zero.

      o KEK Attributes -- Contains KEK policy attributes associated
         with the group. The following sections describe the possible
         attributes. Any or all attributes may be optional, depending on
         the group policy.

5.3.1. KEK Attributes

   The following attributes may be present in a SAK Payload. The
   attributes must follow the format defined in ISAKMP [RFC2408] section
   3.3. In the table, attributes that are defined as TV are marked as
   Basic (B); attributes that are defined as TLV are marked as Variable
   (V).

            ID Class   Value   Type
            --------   -----   ----
            RESERVED   0
            KEK_MANAGEMENT_ALGORITHM   1   B
            KEK_ALGORITHM   2   B
            KEK_KEY_LENGTH   3   B
            KEK_KEY_LIFETIME   4   V
            SIG_HASH_ALGORITHM   5   B
            SIG_ALGORITHM   6   B
            SIG_KEY_LENGTH   7   B
            KE_OAKLEY_GROUP   8   B

   The following attributes may only be included in a GROUPKEY-PULL
   message: KEK_MANAGEMENT_ALGORITHM, KE_OAKLEY_GROUP.

5.3.2. KEK_MANAGEMENT_ALGORITHM

   The KEK_MANAGEMENT_ALGORITHM class specifies the group KEK management
   algorithm used to provide forward or backward access control (i.e.,
   used to exclude group members). Defined values are specified in the
   following table.

            KEK Management Type   Value
            -------------------   -----
            RESERVED   0
            LKH   1
            RESERVED   2-127
            Private Use   128-255

5.3.3. KEK_ALGORITHM

   The KEK_ALGORITHM class specifies the encryption algorithm using with
   the KEK. Defined values are specified in the following table.

                  Algorithm Type   Value
                  --------------   -----
                  RESERVED   0
                  KEK_ALG_DES   1
                  KEK_ALG_3DES   2
                  KEK_ALG_AES   3
                  RESERVED   4-127
                  Private Use   128-255

   A GDOI implementation MUST support the KEK_ALG_3DES algorithm
   attribute.

   If a KEK_MANAGEMENT_ALGORITHM is defined which defines multiple keys
   (e.g., LKH), and if the management algorithm does not specify the
   algorithm for those keys, then the algorithm defined by the
   KEK_ALGORITHM attribute MUST be used for all keys which are included
   as part of the management.

5.3.3.1. KEK_ALG_DES

   This algorithm specifies DES using the Cipher Block Chaining (CBC)
   mode as described in [FIPS81].

5.3.3.2. KEK_ALG_3DES

   This algorithm specifies 3DES using three independent keys as
   described in "Keying Option 1" in [FIPS46-3].

5.3.3.3. KEK_ALG_AES

   This algorithm specifies AES as described in [FIPS197]. The mode of
   operation for AES is Cipher Block Chaining (CBC) as recommended in
   [AES-MODES].

5.3.4. KEK_KEY_LENGTH

   The KEK_KEY_LENGTH class specifies the KEK Algorithm key length (in
   bits).

5.3.5. KEK_KEY_LIFETIME

   The KEK_KEY_LIFETIME class specifies the maximum time for which the
   KEK is valid. The GCKS may refresh the KEK at any time before the
   end of the valid period. The value is a four (4) octet number
   defining a valid time period in seconds.

5.3.6. SIG_HASH_ALGORITHM

   SIG_HASH_ALGORITHM specifies the SIG payload hash algorithm. The
   following tables define the algorithms for SIG_HASH_ALGORITHM.

```
            Algorithm Type  Value
            --------------  -----
            RESERVED          0
            SIG_HASH_MD5      1
            SIG_HASH_SHA1     2
            RESERVED        3-127
            Private Use    128-255
```

   SIG_HASH_ALGORITHM is not required if the SIG_ALGORITHM is
   SIG_ALG_DSS or SIG_ALG_ECDSS, which imply SIG_HASH_SHA1.

5.3.7. SIG_ALGORITHM

   The SIG_ALGORITHM class specifies the SIG payload signature
   algorithm. Defined values are specified in the following table.

```
            Algorithm Type  Value
            --------------  -----
            RESERVED          0
            SIG_ALG_RSA       1
            SIG_ALG_DSS       2
            SIG_ALG_ECDSS     3
            RESERVED        4-127
            Private Use    128-255
```

   A GDOI implementation MUST support the following algorithm attribute:
   SIG_ALG_RSA.

5.3.7.1. SIG_ALG_RSA

   This algorithm specifies the RSA digital signature algorithm as
   described in [RSA].

5.3.7.2. SIG_ALG_DSS

   This algorithm specifies the DSS digital signature algorithm as
   described in [FIPS186-2].

5.3.7.3. SIG_ALG_ECDSS

   This algorithm specifies the Elliptic Curve digital signature
   algorithm as described in [FIPS186-2].

5.3.8. SIG_KEY_LENGTH

   The SIG_KEY_LENGTH class specifies the length of the SIG payload key.

5.3.9. KE_OAKLEY_GROUP

   The KE_OAKLEY_GROUP class defines the OAKLEY Group used to compute
   the PFS secret in the optional KE payload of the GDOI GROUPKEY-PULL
   exchange. This attribute uses the values assigned to Group
   Definitions in the IANA IPsec-registry [IPSEC-REG].

5.4. SA TEK Payload

   The SA TEK (SAT) payload contains security attributes for a single
   TEK associated with a group.
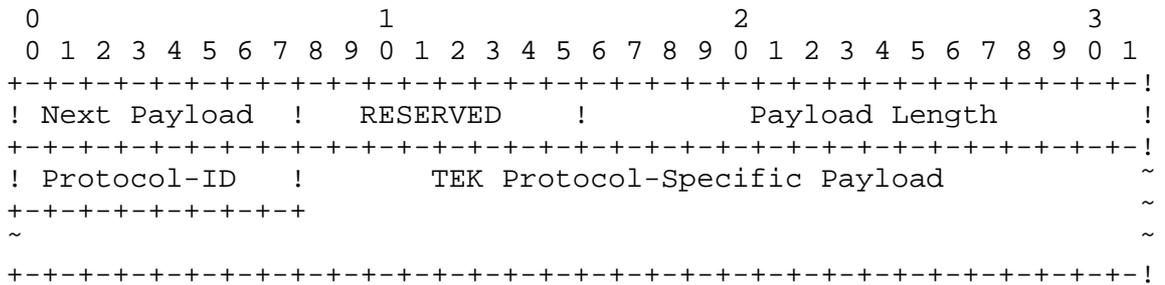
```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
   ! Next Payload  !   RESERVED    !         Payload Length        !
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
   ! Protocol-ID   !      TEK Protocol-Specific Payload            ~
   +-+-+-+-+-+-+-+-+                                               ~
   ~                                                               ~
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
```

   The SAT Payload fields are defined as follows:

   o Next Payload (1 octet) -- Identifies the next payload for the
      GROUPKEY-PULL or the GROUPKEY-PUSH message. The only valid
      next payload types for this message are another SAT Payload or
      zero to indicate there are no more security association
      attributes.

   o RESERVED (1 octet) -- Must be zero.

   o Payload Length (2 octets) -- Length of this payload, including
      the TEK Protocol-Specific Payload.

   o Protocol-ID (1 octet) -- Value specifying the Security
      Protocol. The following table defines values for the Security
      Protocol

        Protocol ID   Value
        -----------   -----
        RESERVED   0
        GDOI_PROTO_IPSEC_ESP   1
        RESERVED   2-127
        Private Use   128-255
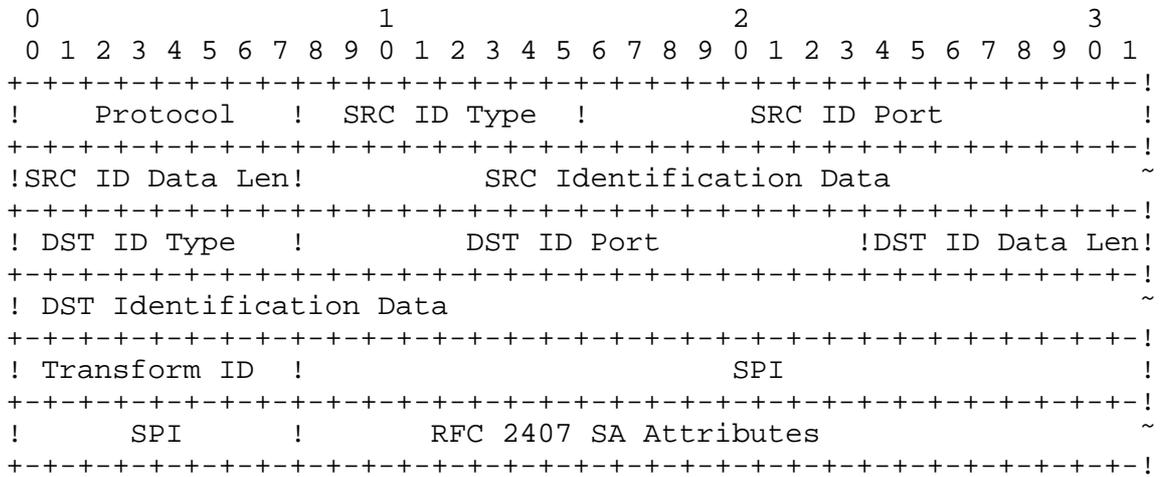
   o TEK Protocol-Specific Payload (variable) -- Payload which
      describes the attributes specific for the Protocol-ID.

5.4.1. PROTO_IPSEC_ESP

   The TEK Protocol-Specific payload for ESP is as follows:

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
!    Protocol   ! SRC ID Type  !          SRC ID Port          !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
!SRC ID Data Len!          SRC Identification Data             ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
! DST ID Type   !          DST ID Port          !DST ID Data Len!
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
! DST Identification Data                                      ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
! Transform ID  !                    SPI                       !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
!     SPI       !        RFC 2407 SA Attributes                ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
```

   The SAT Payload fields are defined as follows:

   o Protocol (1 octet) -- Value describing an IP protocol ID (e.g.,
      UDP/TCP). A value of zero means that the Protocol field should
      be ignored.

   o SRC ID Type (1 octet) -- Value describing the identity
      information found in the SRC Identification Data field.
      Defined values are specified by the IPSEC Identification Type
      section in the IANA isakmpd-registry [ISAKMP-REG].

   o SRC ID Port (2 octets) -- Value specifying a port associated
      with the source Id. A value of zero means that the SRC ID Port
      field should be ignored.

   o SRC ID Data Len (1 octet) -- Value specifying the length of the
      SRC Identification Data field.

   o SRC Identification Data (variable length) -- Value, as
      indicated by the SRC ID Type. Set to three bytes of zero for
      multiple-source multicast groups that use a common TEK for all
      senders.

   o DST ID Type (1 octet) -- Value describing the identity
      information found in the DST Identification Data field.
      Defined values are specified by the IPSEC Identification Type
      section in the IANA isakmpd-registry [ISAKMP-REG].

   o DST ID Prot (1 octet) -- Value describing an IP protocol ID
      (e.g., UDP/TCP). A value of zero means that the DST Id Prot
      field should be ignored.

   o DST ID Port (2 octets) -- Value specifying a port associated
      with the source Id. A value of zero means that the DST ID Port
      field should be ignored.

   o DST ID Data Len (1 octet) -- Value specifying the length of the
      DST Identification Data field.

   o DST Identification Data (variable length) -- Value, as
      indicated by the DST ID Type.

   o Transform ID (1 octet) -- Value specifying which ESP transform
      is to be used. The list of valid values is defined in the
      IPSEC ESP Transform Identifiers section of the IANA
      isakmpd-registry [ISAKMP-REG].

   o SPI (4 octets) -- Security Parameter Index for ESP.

   o RFC 2407 Attributes -- ESP Attributes from RFC 2407 Section
      4.5. The GDOI supports all IPSEC DOI SA Attributes for
      PROTO_IPSEC_ESP excluding the Group Description [RFC2407,
      section 4.5], which MUST NOT be sent by a GDOI implementation
      and is ignored by a GDOI implementation if received. All
      mandatory IPSEC DOI attributes are mandatory in GDOI
      PROTO_IPSEC_ESP. The Authentication Algorithm attribute of the
      IPSEC DOI is group authentication in GDOI.

5.4.2. Other Security Protocols

   Besides ESP, GDOI should serve to establish SAs for secure groups
   needed by other Security Protocols that operate at the transport,
   application, and internetwork layers. These other Security
   Protocols, however, are in the process of being developed or do not
   yet exist.

   The following information needs to be provided for a Security
   Protocol to the GDOI.

      o The Protocol-ID for the particular Security Protocol
      o The SPI Size
      o The method of SPI generation
      o The transforms, attributes and keys needed by the Security
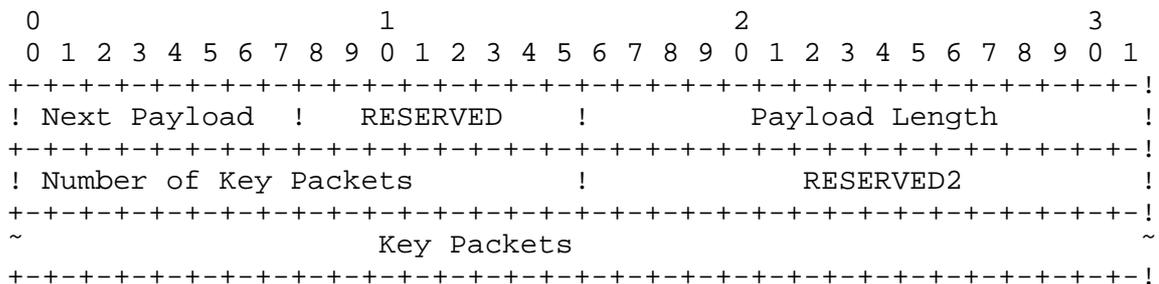   Protocol

   All Security Protocols must provide the information in the bulleted
   list above to guide the GDOI specification for that protocol.
   Definitions for the support of those Security Protocols in GDOI will
   be specified in separate documents.

   A Security Protocol MAY protect traffic at any level of the network
   stack. However, in all cases applications of the Security Protocol
   MUST protect traffic which MAY be shared by more than two entities.

5.5. Key Download Payload

   The Key Download Payload contains group keys for the group specified
   in the SA Payload. These key download payloads can have several
   security attributes applied to them based upon the security policy of
   the group as defined by the associated SA Payload.

   When included as part of the Re-key SA with an optional KE payload,
   The Key Download Payload will be xor'ed with the new Diffie-Hellman
   shared secret. The xor operation will begin at the "Number of Key
   Packets" field.

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
   ! Next Payload  !   RESERVED    !            Payload Length      !
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
   ! Number of Key Packets         !            RESERVED2          !
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
   ~                          Key Packets                          ~
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
```

The Key Download Payload fields are defined as follows:

o Next Payload (1 octet) -- Identifier for the payload type of
   the next payload in the message. If the current payload is the
   last in the message, then this field will be zero.

o RESERVED (1 octet) -- Unused, set to zero.

o Payload Length (2 octets) -- Length in octets of the current
   payload, including the generic payload header.

o Number of Key Packets (2 octets) -- Contains the total number
   of both TEK and Rekey arrays being passed in this data block.

o Key Packets
   Several types of key packets are defined. Each Key Packet has
   the following format.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
!   KD Type     !   RESERVED    !            KD Length          !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
!   SPI Size    !                   SPI (variable)              ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
~                    Key Packet Attributes                      ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
```

o Key Download (KD) Type (1 octet) -- Identifier for the Key Data
   field of this Key Packet.

```
          Key Download Type      Value
          -----------------      -----
          RESERVED                 0
          TEK                      1
          KEK                      2
          LKH                      3
          RESERVED               4-127
          Private Use           128-255
```

"KEK" is a single key whereas LKH is an array of key-encrypting keys.

o RESERVED (1 octet) -- Unused, set to zero.

o Key Download Length (2 octets) -- Length in octets of the Key
   Packet data, including the Key Packet header.

    o SPI Size (1 octet) -- Value specifying the length in octets of
       the SPI as defined by the Protocol-Id.

    o SPI (variable length) -- Security Parameter Index which matches
       a SPI previously sent in an SAK or SAT Payload.

    o Key Packet Attributes (variable length) -- Contains Key
       information. The format of this field is specific to the value
       of the KD Type field. The following sections describe the
       format of each KD Type.

5.5.1. TEK Download Type

   The following attributes may be present in a TEK Download Type.
   Exactly one attribute matching each type sent in the SAT payload MUST
   be present. The attributes must follow the format defined in ISAKMP
   [RFC2408] section 3.3. In the table, attributes defined as TV are
   marked as Basic (B); attributes defined as TLV are marked as Variable
   (V).

              TEK Class   Value   Type
              ---------   -----   ----
              RESERVED   0
              TEK_ALGORITHM_KEY   1   V
              TEK_INTEGRITY_KEY   2   V
              TEK_SOURCE_AUTH_KEY   3   V

   If no TEK key packets are included in a Registration KD payload, the
   group member can expect to receive the TEK as part of a Re-key SA.
   At least one TEK must be included in each Re-key KD payload.
   Multiple TEKs may be included if multiple streams associated with the
   SA are to be rekeyed.

5.5.1.1. TEK_ALGORITHM_KEY

   The TEK_ALGORITHM_KEY class declares that the encryption key for this
   SPI is contained as the Key Packet Attribute. The encryption
   algorithm that will use this key was specified in the SAT payload.

   In the case that the algorithm requires multiple keys (e.g., 3DES),
   all keys will be included in one attribute.

   DES keys will consist of 64 bits (the 56 key bits with parity bit).
   Triple DES keys will be specified as a single 192 bit attribute
   (including parity bits) in the order that the keys are to be used for
   encryption (e.g., DES_KEY1, DES_KEY2, DES_KEY3).

5.5.1.2. TEK_INTEGRITY_KEY

   The TEK_INTEGRITY_KEY class declares that the integrity key for this
   SPI is contained as the Key Packet Attribute. The integrity
   algorithm that will use this key was specified in the SAT payload.
   Thus, GDOI assumes that both the symmetric encryption and integrity
   keys are pushed to the member. SHA keys will consist of 160 bits,
   and MD5 keys will consist of 128 bits.

5.5.1.3. TEK_SOURCE_AUTH_KEY

   The TEK_SOURCE_AUTH_KEY class declares that the source authentication
   key for this SPI is contained in the Key Packet Attribute. The
   source authentication algorithm that will use this key was specified
   in the SAT payload.

5.5.2. KEK Download Type

   The following attributes may be present in a KEK Download Type.
   Exactly one attribute matching each type sent in the SAK payload MUST
   be present. The attributes must follow the format defined in ISAKMP
   [RFC2408] section 3.3. In the table, attributes defined as TV are
   marked as Basic (B); attributes defined as TLV are marked as Variable
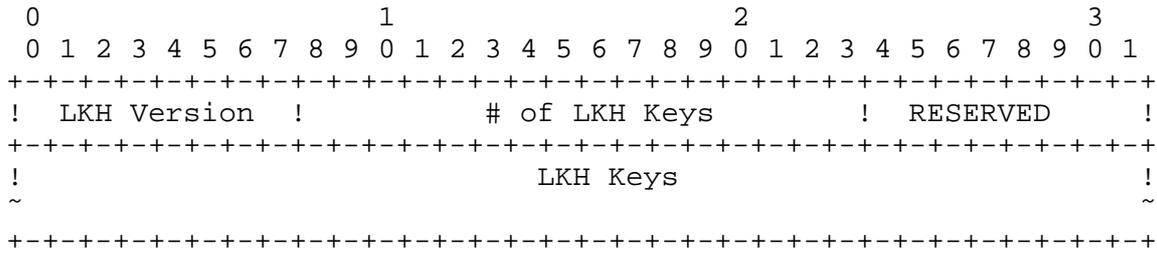   (V).

          KEK Class   Value   Type
          ---------   -----   ----
          RESERVED   0
          KEK_ALGORITHM_KEY   1   V
          SIG_ALGORITHM_KEY   2   V

   If the KEK key packet is included, there MUST be only one present in
   the KD payload.

5.5.2.1. KEK_ALGORITHM_KEY

   The KEK_ALGORITHM_KEY class declares the encryption key for this SPI
   is contained in the Key Packet Attribute. The encryption algorithm
   that will use this key was specified in the SAK payload.

   If the mode of operation for the algorithm requires an Initialization
   Vector (IV), an explicit IV MUST be included in the KEK_ALGORITHM_KEY
   before the actual key.

5.5.2.2. SIG_ALGORITHM_KEY

   The SIG_ALGORITHM_KEY class declares that the public key for this SPI
   is contained in the Key Packet Attribute, which may be useful when no
   public key infrastructure is available. The signature algorithm that
   will use this key was specified in the SAK payload.

5.5.3. LKH Download Type

   The LKH key packet is comprised of attributes representing different
   leaves in the LKH key tree.

   The following attributes are used to pass an LKH KEK array in the KD
   payload. The attributes must follow the format defined in ISAKMP
   [RFC2408] section 3.3. In the table, attributes defined as TV are
   marked as Basic (B); attributes defined as TLV are marked as Variable
   (V).

KEK Class	Value	Type
RESERVED	0	
LKH_DOWNLOAD_ARRAY	1	V
LKH_UPDATE_ARRAY	2	V
SIG_ALGORITHM_KEY	3	V
RESERVED	4-127	
Private Use	128-255	

   If an LKH key packet is included in the KD payload, there must be
   only one present.

5.5.3.1. LKH_DOWNLOAD_ARRAY

   This attribute is used to download a set of keys to a group member.
   It MUST NOT be included in a GROUPKEY-PUSH message KD payload if the
   GROUPKEY-PUSH is sent to more than the group member. If an
   LKH_DOWNLOAD_ARRAY attribute is included in a KD payload, there must
   be only one present.

   This attribute consists of a header block, followed by one or more
   LKH keys.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! LKH Version  !         # of LKH Keys        ! RESERVED      !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                          LKH Keys                            !
~                                                              ~
                                                               
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The KEK_LKH attribute fields are defined as follows:

   o LKH version (1 octet) -- Contains the version of the LKH
      protocol which the data is formatted in. Must be one.

   o Number of LKH Keys (2 octets) -- This value is the number of
      distinct LKH keys in this sequence.

   o RESERVED (1 octet) -- Unused, set to zero. Each LKH Key is
      defined as follows:

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!           LKH ID            !   Key Type   !   RESERVED    !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                       Key Creation Date                      !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                       Key expiration Date                    !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                          Key Handle                          !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                              !
~                          Key Data                            ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
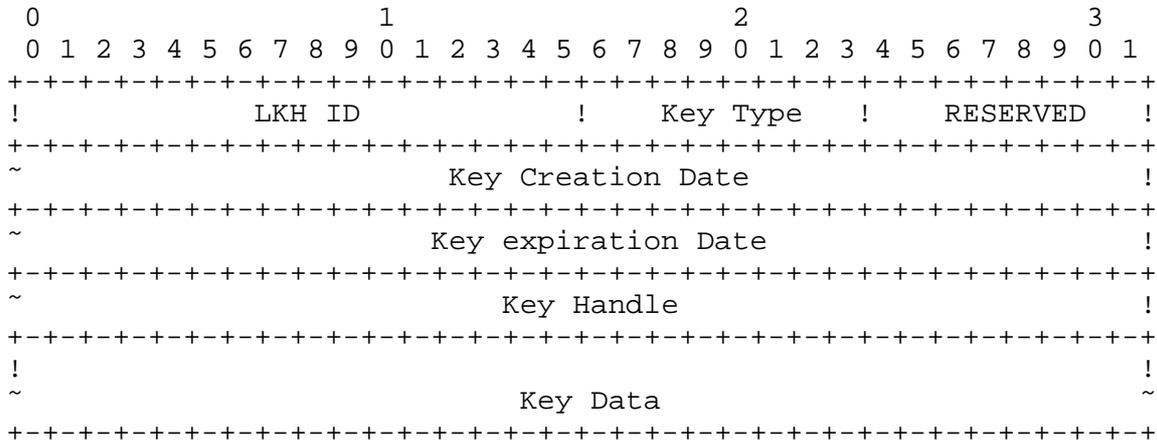
   o LKH ID (2 octets) -- This is the position of this key in the
      binary tree structure used by LKH.

   o Key Type (1 octet) -- This is the encryption algorithm for
      which this key data is to be used. This value is specified in
      Section 5.3.3.

   o RESERVED (1 octet) -- Unused, set to zero.

   o Key Creation Date (4 octets) -- This is the time value of when
      this key data was originally generated. A time value of zero
      indicates that there is no time before which this key is not
      valid.

        o Key Expiration Date (4 octets) -- This is the time value of
           when this key is no longer valid for use. A time value of zero
           indicates that this key does not have an expiration time.

        o Key Handle (4 octets) -- This is the randomly generated value
           to uniquely identify a key within an LKH ID.

        o Key Data (variable length) -- This is the actual encryption key
           data, which is dependent on the Key Type algorithm for its
           format. If the mode of operation for the algorithm requires an
           Initialization Vector (IV), an explicit IV MUST be included in
           the Key Data field before the actual key.

   The Key Creation Date and Key expiration Dates MAY be zero. This is
   necessary in the case where time synchronization within the group is
   not possible.

   The first LKH Key structure in an LKH_DOWNLOAD_ARRAY attribute
   contains the Leaf identifier and key for the group member. The rest
   of the LKH Key structures contain keys along the path of the key tree
   in order from the leaf, culminating in the group KEK.

5.5.3.2. LKH_UPDATE_ARRAY

   This attribute is used to update the keys for a group. It is most
   likely to be included in a GROUPKEY-PUSH message KD payload to rekey
   the entire group. This attribute consists of a header block,
   followed by one or more LKH keys, as defined in Section 5.5.3.1

   There may be any number of UPDATE_ARRAY attributes included in a KD
   payload.
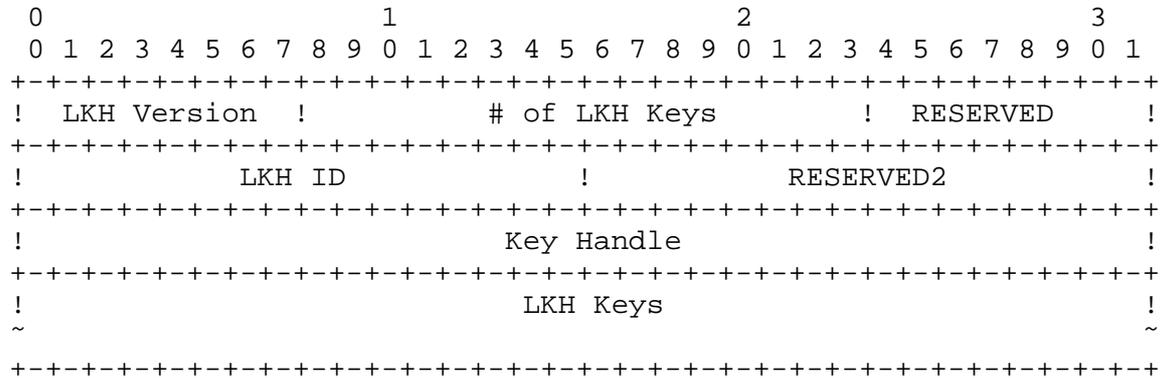
```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   ! LKH Version  !        # of LKH Keys       ! RESERVED     !
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   !          LKH ID          !          RESERVED2          !
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   !                          Key Handle                          !
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   !                          LKH Keys                            !
   ~                                                              ~
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

        o LKH version (1 octet) -- Contains the version of the LKH
           protocol which the data is formatted in. Must be one.

        o Number of LKH Keys (2 octets) -- This value is the number of
           distinct LKH keys in this sequence.

        o RESERVED (1 octet) -- Unused, set to zero.

        o LKH ID (2 octets) -- This is the node identifier associated
           with the key used to encrypt the first LKH Key.

        o RESERVED2 (2 octets) -- Unused, set to zero.

        o Key Handle (4 octets) -- This is the value to uniquely identify
           the key within the LKH ID which was used to encrypt the first
           LKH key.

   The LKH Keys are as defined in Section 5.5.3.1. The LKH Key
   structures contain keys along the path of the key tree in order from
   the LKH ID found in the LKH_UPDATE_ARRAY header, culminating in the
   group KEK. The Key Data field of each LKH Key is encrypted with the
   LKH key preceding it in the LKH_UPDATE_ARRAY attribute. The first
   LKH Key is encrypted under the key defined by the LKH ID and Key
   Handle found in the LKH_UPDATE_ARRAY header.

5.5.3.3. SIG_ALGORITHM_KEY

   The SIG_ALGORITHM_KEY class declares that the public key for this SPI
   is contained in the Key Packet Attribute, which may be useful when no
   public key infrastructure is available. The signature algorithm that
   will use this key was specified in the SAK payload.

5.6. Sequence Number Payload

   The Sequence Number Payload (SEQ) provides an anti-replay protection
   for GROUPKEY-PUSH messages. Its use is similar to the Sequence
   Number field defined in the IPsec ESP protocol [RFC2406].

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   ! Next Payload  !   RESERVED    !         Payload Length        !
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   !                        Sequence Number                        !
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
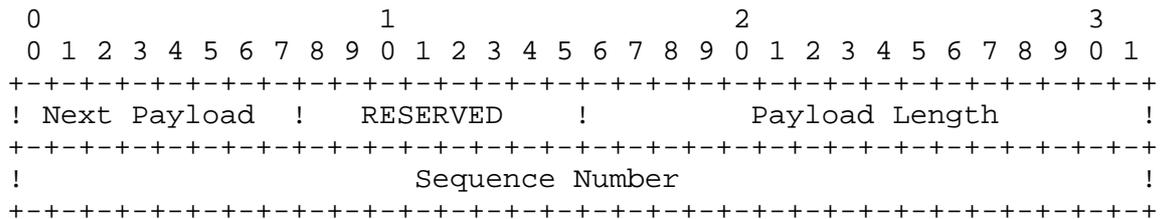
   The Sequence Number Payload fields are defined as follows:

        o Next Payload (1 octet) -- Identifier for the payload type of
           the next payload in the message. If the current payload is the
           last in the message, then this field will be zero.

   o RESERVED (1 octet) -- Unused, set to zero.

   o Payload Length (2 octets) -- Length in octets of the current
      payload, including the generic payload header.

   o Sequence Number (4 octets) -- This field contains a
      monotonically increasing counter value for the group. It is
      initialized to zero by the GCKS, and incremented in each
      subsequently-transmitted message. Thus the first packet sent
      for a given Rekey SA will have a Sequence Number of 1. The
      GDOI implementation keeps a sequence counter as an attribute
      for the Rekey SA and increments the counter upon receipt of a
      GROUPKEY-PUSH message. The current value of the sequence
      number must be transmitted to group members as a part of the
      Registration SA SA payload. A group member must keep a sliding
      receive window. The window must be treated as in the ESP
      protocol [RFC2406] Section 3.4.3.

5.7. Proof of Possession

   The Proof of Possession Payload is used as part of group membership
   authorization during a GDOI exchange. The Proof of Possession
   Payload is identical to an ISAKMP SIG payload. However, the usage is
   entirely different.

   The GCKS, GCKS delegate or member signs a hash of the following
   values:
        POP_HASH = hash("pop" | Ni | Nr)
   Where hash() is the hash function used with the signature.

   The "pop" prefix ensures that the signature of the POP payload cannot
   be used for any other purpose in the GDOI protocol.

5.8. Nonce

   The data portion of the Nonce payload (i.e., Ni_b and Nr_b included
   in the HASHs) MUST be a value between 8 and 128 bytes.

6. Security Considerations

   GDOI is a security association (SA) management protocol for groups of
   senders and receivers. Unlike a data security protocol, SA
   management includes a key establishment protocol to securely
   establish keys at communication endpoints. This protocol performs
   entity authentication of the GDOI member or Group Controller/Key
   Server (GCKS), it provides confidentiality of key management
   messages, and it provides source authentication of those messages.
   This protocol also uses best-known practices for defense against

man-in-middle, connection hijacking, replay, reflection, and
denial-of-service (DOS) attacks on unsecured networks [STS, RFC2522,
SKEME]. GDOI assumes the network is not secure and may be under the
complete control of an attacker.

GDOI assumes that the host computer is secure even though the network
is insecure. GDOI ultimately establishes keys among members of a
group, which MUST be trusted to use those keys in an authorized
manner according to group policy. The security of GDOI, therefore,
is as good as the degree to which group members can be trusted to
protect authenticators, encryption keys, decryption keys, and message
authentication keys.

There are three phases of GDOI as described in this document: an
ISAKMP Phase 1 protocol, a new exchange called GROUPKEY-PULL which is
protected by the ISAKMP Phase 1 protocol, and a new message called
GROUPKEY-PUSH. Each phase is considered separately below.

6.1. ISAKMP Phase 1

As described in this document, GDOI uses the Phase 1 exchanges
defined in [RFC2409] to protect the GROUPKEY-PULL exchange.
Therefore all security properties and considerations of those
exchanges (as noted in [RFC2409]) are relevant for GDOI.

GDOI may inherit the problems of its ancestor protocols [FS00], such
as identity exposure, absence of unidirectional authentication, or
stateful cookies [PK01]. GDOI could benefit, however, from
improvements to its ancestor protocols just as it benefits from years
of experience and work embodied in those protocols. To reap the
benefits of future IKE improvements, however, GDOI would need to be
revised in a future standards-track RFC, which is beyond the scope of
this specification.

6.1.1. Authentication

Authentication is provided via the mechanisms defined in [RFC2409],
namely Pre-Shared Keys or Public Key encryption.

6.1.2. Confidentiality

Confidentiality is achieved in Phase 1 through a Diffie-Hellman
exchange that provides keying material, and through negotiation of
encryption transforms.

The Phase 1 protocol will be protecting encryption and integrity keys
sent in the GROUPKEY-PULL protocol. The strength of the encryption
used for Phase 1 SHOULD exceed that of the keys send in the
GROUPKEY-PULL protocol.

6.1.3. Man-in-the-Middle Attack Protection

A successful man-in-the-middle or connection-hijacking attack foils
entity authentication of one or more of the communicating entities
during key establishment. GDOI relies on Phase 1 authentication to
defeat man-in-the-middle attacks.

6.1.4. Replay/Reflection Attack Protection

In a replay/reflection attack, an attacker captures messages between
GDOI entities and subsequently forwards them to a GDOI entity.
Replay and reflection attacks seek to gain information from a
subsequent GDOI message response or seek to disrupt the operation of
a GDOI member or GCKS entity. GDOI relies on the Phase 1 nonce
mechanism in combination with a hash-based message authentication
code to protect against the replay or reflection of previous key
management messages.

6.1.5. Denial of Service Protection

A denial of service attacker sends messages to a GDOI entity to cause
that entity to perform unneeded message authentication operations.
GDOI uses the Phase 1 cookie mechanism to identify spurious messages
prior to cryptographic hash processing. This is a "weak" form of
denial of service protection in that the GDOI entity must check for
good cookies, which can be successfully imitated by a sophisticated
attacker. The Phase 1 cookie mechanism is stateful, and commits
memory resources for cookies, but stateless cookies are a better
defense against denial of service attacks.

6.2. GROUPKEY-PULL Exchange

The GROUPKEY-PULL exchange allows a group member to request SAs and
keys from a GCKS. It runs as a "phase 2" protocol under protection
of the Phase 1 security association.

6.2.1. Authentication

Peer authentication is not required in the GROUPKEY-PULL protocol.
It is running in the context of the Phase 1 protocol, which has
previously authenticated the identity of the peer.

Message authentication is provided by HASH payloads in each message,
where the HASH is defined to be over SKEYID_a (derived in the Phase 1
exchange), the ISAKMP Message-ID, and all payloads in the message.
Because only the two endpoints of the exchange know the SKEYID_a
value, this provides confidence that the peer sent the message.

6.2.2. Confidentiality

Confidentiality is provided by the Phase 1 security association,
after the manner described in [RFC2409].

6.2.3. Man-in-the-Middle Attack Protection

Message authentication (described above) includes a secret known only
to the group member and GCKS when constructing a HASH payload. This
prevents man-in-the-middle and connection-hijacking attacks because
an attacker would not be able to change the message undetected.

6.2.4. Replay/Reflection Attack Protection

Nonces provide freshness of the GROUPKEY-PULL exchange. The group
member and GCKS exchange nonce values first two messages. These
nonces are included in subsequent HASH payload calculations. The
Group member and GCKS MUST NOT perform any computationally expensive
tasks before receiving a HASH with its own nonce included. The GCKS
MUST NOT update the group management state (e.g., LKH key tree) until
it receives the third message in the exchange with a valid HASH
payload including its own nonce.

Implementations SHOULD keep a record of recently received
GROUPKEY-PULL messages and reject messages that have already been
processed. This enables an early discard of the replayed messages.

6.2.5. Denial of Service Protection

A GROUPKEY-PULL message identifies its messages using a cookie pair
from the Phase 1 exchange that precedes it. The cookies provide a
weak form of denial of service protection as described above, in the
sense that a GROUPKEY-PULL message with invalid cookies will be
discarded.

The replay protection mechanisms described above provide the basis
for denial of service protection.

6.2.6. Authorization

   The CERT payload in a GROUPKEY-PULL exchange allows a group member or
   GCKS to submit a certificate containing authorization attributes to
   the peer as well as identifying a public/private key pair. The
   GROUPKEY-PULL POP payload enables authorization to be accomplished
   where the authorization infrastructure is different than the
   GROUPKEY-PULL authentication infrastructure by proving that it is in
   possession of the private key.

6.3. GROUPKEY-PUSH Exchange

   The GROUPKEY-PUSH exchange is a single message that allows a GCKS to
   send SAs and keys to group members. This is likely to be sent to all
   members using an IP multicast group. This provides an efficient
   rekey and group membership adjustment capability.

6.3.1. Authentication

   The GROUPKEY-PULL exchange identifies a public key that is used for
   message authentication. The GROUPKEY-PUSH message is digitally
   signed using the corresponding private key held by the GCKS or its
   delegate. This digital signature provides source authentication for
   the message. Thus, GDOI protects the GCKS from impersonation in
   group environments.

6.3.2. Confidentiality

   The GCKS encrypts the GROUPKEY-PUSH message with an encryption key
   that was established by the GROUPKEY-PULL exchange.

6.3.3. Man-in-the-Middle Attack Protection

   This combination of confidentiality and message authentication
   services protects the GROUPKEY-PUSH message from man-in-middle and
   connection-hijacking attacks.

6.3.4. Replay/Reflection Attack Protection

   The GROUPKEY-PUSH message includes a monotonically increasing
   sequence number to protect against replay and reflection attacks. A
   group member will recognize a replayed message by comparing the
   sequence number to a sliding window, in the same manner as the ESP
   protocol uses sequence numbers.

   Implementations SHOULD keep a record of recently received
   GROUPKEY-PUSH messages and reject duplicate messages. This enables
   an early discard of the replayed messages.

6.3.5. Denial of Service Protection

   A cookie pair identifies the security association for the
   GROUPKEY-PUSH message. The cookies thus serve as a weak form of
   denial-of-service protection for the GROUPKEY-PUSH message.

   The digital signature used for message authentication has a much
   greater computational cost than a message authentication code and
   could amplify the effects of a denial of service attack on GDOI
   members who process GROUPKEY-PUSH messages. The added cost of
   digital signatures is justified by the need to prevent GCKS
   impersonation: If a shared symmetric key were used for
   GROUPKEY-PUSH message authentication, then GCKS source authentication
   would be impossible and any member would be capable of GCKS
   impersonation.

   The potential of the digital signature amplifying a denial of service
   attack is mitigated by the order of operations a group member takes,
   where the least expensive cryptographic operation is performed first.
   The group member first decrypts the message using a symmetric cipher.
   If it is a validly formed message then the sequence number is checked
   against the replay window. Only if the sequence number is valid is
   the digital signature verified. Thus in order for a denial of
   service attack to be mounted, an attacker would need to know both the
   symmetric encryption key used for confidentiality, and a valid
   sequence number. Generally speaking this means only current group
   members can effectively deploy a denial of service attack.

6.3.6. Forward Access Control

   If a group management algorithm (such as LKH) is used, forward access
   control may not be ensured in some cases. This can happen if some
   group members are denied access to the group in the same
   GROUPKEY-PUSH message as new policy and TEKs are delivered to the
   group. As discussed in Section 4.2.1, forward access control can be
   maintained by sending multiple GROUPKEY-PUSH messages, where the
   group membership changes are sent from the GCKS separate from the new
   policy and TEKs.

7. IANA Considerations

7.1. ISAKMP DOI

   An ISAKMP DOI number is needed to identify an SA payload as a GDOI SA
   payload. The IANA has assigned the value 2 to represent GDOI.

7.2. Payload Types

   The present document defines new ISAKMP Next Payload types. See
   Section 5.0 for the payloads defined in this document, including the
   Next Payload values defined by the IANA to identify these payloads.

7.3. New Name spaces

   The present document describes many new name spaces for use in the
   GDOI payloads. Those may be found in subsections under Section 5.0.
   A new GDOI registry has been created for these name spaces.

   Portions of name spaces marked "RESERVED" are reserved for IANA
   allocation. New values MUST be added due to a Standards Action as
   defined in [RFC2434].

   Portions of name spaces marked "Private Use" may be allocated by
   implementations for their own purposes.

7.4. UDP Port

   The IANA has assigned port 848 for use by GDOI.

8. Intellectual Property Rights Statement

   The IETF takes no position regarding the validity or scope of any
   intellectual property or other rights that might be claimed to
   pertain to the implementation or use of the technology described in
   this document or the extent to which any license under such rights
   might or might not be available; neither does it represent that it
   has made any effort to identify any such rights. Information on the
   IETF's procedures with respect to rights in standards-track and
   standards-related documentation can be found in BCP-11. Copies of
   claims of rights made available for publication and any assurances of
   licenses to be made available, or the result of an attempt made to
   obtain a general license or permission for the use of such
   proprietary rights by implementors or users of this specification can
   be obtained from the IETF Secretariat.

   The IETF invites any interested party to bring to its attention any
   copyrights, patents or patent applications, or other proprietary
   rights which may cover technology that may be required to practice
   this standard. Please address the information to the IETF Executive
   Director.

9. Acknowledgements

   The authors thank Ran Canetti, Cathy Meadows, Andrea Colegrove, and
   Lakshminath Dondeti. Ran has advised the authors on secure group
   cryptography, which has led to changes in the exchanges and payload
   definitions. Cathy identified several problems in previous versions
   of this document, including a replay attack against the proof of
   possession exchange, as well as several man-in-the-middle attacks.
   Andrea contributed to the group policy section of this document.
   Lakshminath identified several protocol issues that needed further
   specification and helped to resolve them.

10. References

10.1. Normative References

   [AES-MODES]   "Recommendation for Block Cipher Modes of Operation",
                 United States of American, National Institute of Science
                 and Technology, NIST Special Publication 800-38A 2001
                 Edition, December 2001.

   [FIPS46-3]   "Data Encryption Standard (DES)", United States of
                 American, National Institute of Science and Technology,
                 Federal Information Processing Standard (FIPS) 46-3,
                 October 1999.

   [FIPS81]   "DES Modes of Operation", United States of American,
                 National Institute of Science and Technology, Federal
                 Information Processing Standard (FIPS) 81, December
                 1980.

   [FIPS186-2]   "Digital Signature Standard (DSS)", United States of
                 American, National Institute of Science and Technology,
                 Federal Information Processing Standard (FIPS) 186-2,
                 January 2000.

   [FIPS197]   "Advanced Encryption Standard (AES)", United States of
                 American, National Institute of Science and Technology,
                 Federal Information Processing Standard (FIPS) 197,
                 November 2001.

   [IPSEC-REG]   http://www.iana.org/assignments/ipsec-registry

   [ISAKMP-REG] http://www.iana.org/assignments/isakmp-registry

   [RFC2119]   Bradner, S., "Key words for use in RFCs to Indicate
                 Requirement Level", BCP 14, RFC 2119, March 1997.

   [RFC2401]   Kent, S. and R. Atkinson, "Security Architecture for the
                 Internet Protocol", RFC 2401, November 1998

   [RFC2406]   Kent, S. and R. Atkinson, "IP Encapsulating Security
                 Payload (ESP)", RFC 2406, November 1998.

   [RFC2407]   Piper, D., "The Internet IP Domain of Interpretation for
                 ISAKMP", RFC 2407, November 1998.

   [RFC2408]   Maughan, D., Shertler, M., Schneider, M. and J. Turner,
                 "Internet Security Association and Key Management
                 Protocol", RFC 2408, November 1998.

   [RFC2409]   Harkins, D. and D. Carrel, "The Internet Key Exchange
                 (IKE)", RFC 2409, November 1998.

   [RFC2412]   Orman, H., "The OAKLEY Key Determination Protocol", RFC
                 2412, November 1998.

   [RFC2434]   Narten, T. and H. Alvestrand, "Guidelines for Writing an
                 IANA Considerations Section in RFCs", BCP 26, RFC 2434,
                 October 1998.

   [RFC2522]   Karn, P. and W. Simpson, "Photuris: Session-Key
                 Management Protocol", RFC 2522, March 1999.

   [RFC2627]   Wallner, D., Harder, E. and R. Agee, "Key Management for
                 Multicast: Issues and Architectures", RFC 2627,
                 September 1998.

   [RSA]   RSA Laboratories, "PKCS #1 v2.0: RSA Encryption
                 Standard", October 1998.

10.2. Informative References

   [FS00]   N. Ferguson and B. Schneier, "A Cryptographic Evaluation
                 of IPsec, CounterPane",
                 http://www.counterpane.com/ipsec.html.

   [GKMARCH]   M. Baugher, R. Canetti, L. Dondeti, F. Lindholm, "Group
                 Key Management Architecture", Work in Progress.

   [IKEv2]   D. Harkins, et. al., "Proposal for the IKEv2 protocol",
                 Work In Progress.

   [KINK]   M. Thomas, J. Vilhuber, "Kerberized Internet Negotiation
                 of Keys (KINK)", Work in Progress.

    [NNL]   D. Naor, M. Naor and J. Lotspiech, "Revocation and
                    Tracing Schemes for Stateless Receivers", Advances in
                    Cryptology, Crypto '01, Springer-Verlag LNCS 2139, 2001,
                    pp. 41-62. A full version of the paper appears in
                    http://www.wisdom.weizmann.ac.il/~naor/.

    [OFT]   D. Mcgrew and A. Sherman, "Key Establishment in Large
                    Dynamic Groups Using One-Way Function Trees", Manuscript
                    submitted to IEEE Transactions on Software Engineering.
                    A full version of the paper
                    appears in http://download.nai.com/products/media/nai/
                    misc/oft052098.ps, 1998

    [PK01]   R.Perlman, C.Kaufman, "Analysis of the IPsec Key
                    Exchange Standard", WET-ICE conference, 2001.
                    http://sec.femto.org/wetice-2001/papers/radia-paper.pdf

    [RFC2093]   Harney, H., and C. Muckenhirn, "Group Key Management
                    Protocol (GKMP) Specification," RFC 2093, July 1997.

    [RFC2094]   Harney, H. and C. Muckenhirn, "Group Key Management
                    Protocol (GKMP) Architecture," RFC 2094, July 1997.

    [RFC2367]   McDonald, D., Metz, C. and B. Phan, "PF_KEY Key
                    Management API, Version 2", RFC 2367, July 1998.

    [RFC3550]   Schulzrinne, H., Casner, S., Jacobson, V. and R.
                    Frederick, "RTP: A Transport Protocol for Real-Time
                    Applications", RFC 3550, June 2003.

    [SKEME]   H. Krawczyk, "SKEME: A Versatile Secure Key Exchange
                    Mechanism for Internet", ISOC Secure Networks and
                    Distributed Systems Symposium, San Diego, 1996.

    [STS]   Diffie, P. van Oorschot, M. J. Wiener, "Authentication
                    and Authenticated Key Exchanges, Designs, Codes and
                    Cryptography", 2, 107-125 (1992), Kluwer Academic
                    Publishers.

Appendix A: Alternate GDOI Phase 1 protocols

   This section describes a manner in which other protocols could be
   used as GDOI Phase 1 protocols in place of the ISAKMP Phase 1
   protocol. However, they are not specified as a part of this
   document. A separate document MUST be written in order for another
   protocol to be used as a GDOI Phase 1 protocol.

   Other possible phase 1 protocols are also described in [GKMARCH].

   Any GDOI phase 1 protocol MUST satisfy the requirements specified in
   Section 2 of this document.

A.1. IKEv2 Phase 1 protocol

   Version 2 of the IKE protocol (IKEv2) is a work in progress [IKEv2].
   That protocol seeks to simplify the IKE Phase 1 and Phase 2
   protocols, and improve the security of the IKE protocol. An IKEv2
   Phase 1 negotiates an IPSEC SA during phase 1, which was not possible
   in IKE. However, IKEv2 also defines a phase 2 protocol. The phase 2
   protocol is protected by the Phase 1, similar in concept to how IKE
   Quick Mode is protected by the IKE Phase 1 protocols in [RFC2409].

   IKEv2 may not include a DOI value in the SA payload. However, since
   GDOI uses a unique port, choice of a phase 2 protocol in the SA
   payload using a GDOI value is not necessary. It is expected that an
   IKEv2 Phase 1 protocol definition could be run on the GDOI port. The
   SA payload in the protocol would be specific to GDOI, or omitted if
   not needed at all.

   The GROUPKEY-PULL protocol would follow the IKEv2 Phase 1 protocol in
   the same manner as described in this document.

A.2. KINK Protocol

   A work in progress [KINK] has defined a method of encapsulating an
   IKE Quick Mode [RFC2409] encapsulated in Kerberos KRB_AP_REQ and
   KRB_AP_REP payloads. KINK provides a low-latency, computationally
   inexpensive, easily managed, and cryptographically sound method of
   setting up IPSec security associations.

   The KINK message format includes a GDOI field in the KINK header.
   The [KINK] document defines the DOI for the IPSEC DOI.

   A new DOI for KINK could be defined which would encapsulate a
   GROUPKEY-PULL exchange in the Kerberos KRB_AP_REQ and KRB_AP_REP
   payloads. As such, GDOI would benefit from the computational
   efficiencies of KINK.

Authors' Addresses

    Mark Baugher
    Cisco Systems
    5510 SW Orchid Street
    Portland, OR 97219, USA

    Phone: (503) 245-4543
    EMail: mbaugher@cisco.com

    Thomas Hardjono
    VeriSign
    401 Edgewater Place, Suite 280
    Wakefield, MA 01880

    Phone: 781-245-6996
    EMail: thardjono@verisign.com

    Hugh Harney
    Sparta
    9861 Broken Land Parkway
    Columbia, MD 21046

    Phone: (410) 381-9400 x203
    EMail: hh@sparta.com

    Brian Weis
    Cisco Systems
    170 W. Tasman Drive,
    San Jose, CA 95134-1706, USA

    Phone: (408) 526-4796
    EMail: bew@cisco.com

Full Copyright Statement

Acknowledgement