Internet Engineering Task Force (IETF)
Request for Comments: 6407
Obsoletes: 3547
Category: Standards Track
ISSN: 2070-1721

B. Weis
S. Rowles
Cisco Systems
T. Hardjono
MIT

October 2011

# The Group Domain of Interpretation

## Abstract

This document describes the Group Domain of Interpretation (GDOI) protocol specified in RFC 3547. The GDOI provides group key management to support secure group communications according to the architecture specified in RFC 4046. The GDOI manages group security associations, which are used by IPsec and potentially other data security protocols. This document replaces RFC 3547.

## Status of This Memo

## Copyright Notice

Table of Contents

## 1. Introduction

Secure group and multicast applications require a method by which each group member shares common security policy and keying material. This document describes the Group Domain of Interpretation (GDOI), which is an Internet Security Association and Key Management Protocol (ISAMKP) [RFC2408] Domain of Interpretation (DOI), a group key management system. The GDOI distributes security associations (SAs) for IPsec Authentication Header (AH) [RFC4302] and Encapsulating Security Payload (ESP) [RFC4303] protocols and potentially other data security protocols used in group applications. The GDOI uses the group key management model defined in [RFC4046], and described more generally by "The Multicast Group Security Architecture" [RFC3740].

In this group key management model, the GDOI protocol participants are a Group Controller/Key Server (GCKS) and a group member (GM). A group member contacts ("registers with") a GCKS to join the group. During the registration, mutual authentication and authorization are achieved, after which the GCKS distributes current group policy and keying material to the group member over an authenticated and encrypted session. The GCKS may also initiate contact ("rekeys") with group members to provide updates to group policy.

ISAKMP defines two "phases" of negotiation (Section 2.3 of [RFC2408]). A Phase 1 security association provides mutual authentication and authorization, and a security association that is used by the protocol participants to execute a Phase 2 exchange. This document incorporates (i.e., uses but does not redefine) the Phase 1 security association definition from the Internet DOI [RFC2407], [RFC2409]. Although RFCs 2407, 2408, and 2409 were obsoleted by [RFC4306] (and subsequently [RFC5996]), they are used by this document because the protocol definitions remain relevant for ISAKMP protocols other than IKEv2.

The GDOI includes two new Phase 2 ISAKMP exchanges (protocols), as well as necessary new payload definitions to the ISAKMP standard (Section 2.1 of [RFC2408]). These two new protocols are:

1. The GROUPKEY-PULL registration protocol exchange. This exchange uses "pull" behavior since the member initiates the retrieval of these SAs from a GCKS. It is protected by an ISAKMP Phase 1 protocol, as described above. At the culmination of a GROUPKEY-PULL exchange, an authorized group member has received and installed a set of SAs that represent group policy, and it is ready to participate in secure group communications.

2. The GROUPKEY-PUSH rekey protocol exchange. The rekey protocol is a datagram initiated ("pushed") by the GCKS, usually delivered to group members using a IP multicast address. The rekey protocol is an ISAKMP protocol, where cryptographic policy and keying material ("Rekey SA") are included in the group policy distributed by the GCKS in the GROUPKEY-PULL exchange. At the culmination of a GROUPKEY-PUSH exchange, the key server has sent group policy to all authorized group members, allowing receiving group members to participate in secure group communications. If a group management method is included in group policy (as described in Section 7.4), at the conclusion of the GROUPKEY-PUSH exchange, some members of the group may have been de-authorized and no longer able to participate in the secure group communications.

```
      +--------------------------------------------------------------+
      |                                                              |
      |                    +--------------------+                    |
      |            +------>|     GDOI GCKS      |<------+            |
      |            |       +--------------------+       |            |
      |            |                 |                  |            |
      |       GROUPKEY-PULL          |             GROUPKEY-PULL     |
      |         PROTOCOL             |               PROTOCOL        |
      |            |                 |                  |            |
      |            v           GROUPKEY-PUSH            v            |
      |   +-----------------+     PROTOCOL     +-----------------+   |
      |   |                 |        |         |                 |   |
      |   |    GDOI GM(s)   |<-------+-------->|    GDOI GM(S)   |   |
      |   |                 |                  |                 |   |
      |   +-----------------+                  +-----------------+   |
      |            |                                    ^            |
      |            v                                    |            |
      |            +-Data Security Protocol (e.g., ESP)-+            |
      |                                                              |
      +--------------------------------------------------------------+
```

Figure 1. Group Key Management Model

Although the GROUPKEY-PUSH protocol specified by this document can be used to refresh the Rekey SA protecting the GROUPKEY-PUSH protocol, the most common use of GROUPKEY-PUSH is to establish keying material and policy for a data security protocol.

GDOI defines several payload types used to distribute policy and keying material within the GROUPKEY-PULL and GROUPKEY-PUSH protocols: Security Association (SA), SA KEK, SA TEK, Group Associated Policy (GAP), Sequence Number (SEQ), and Key Download (KD). Format and usage of these payloads are defined in later sections of this memo.

In summary, 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 data security protocol SAs, a Rekey SA, and/or other data shared by group members for multicast and groups security applications.

## 1.1. Requirements Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

## 1.2. Terminology

The following key terms are used throughout this document.

Data-Security SA The security policy distributed by a GDOI GCKS describing traffic that is expected to be protected by group members. This document described the distribution of IPsec AH and ESP Data-Security SAs.

Group Controller/Key Server A device that defines group policy and distributes keys for that policy [RFC3740].

Group Member. An authorized member of a secure group, sending and/or receiving IP packets related to the group.

GROUPKEY-PULL. A protocol used by a GDOI group member to request group policy and keying material.

GROUPKEY-PUSH. A protocol used by a GDOI GCKS to distribute updates of group policy and keying material to authorized group members.

Key Encrypting Key. The symmetric cipher key used to protect the GROUPKEY-PUSH message.

Logical Key Hierarchy. A group management method defined in Section 5.4 of [RFC2627].

Rekey SA. The security policy protecting a GROUPKEY-PUSH protocol.

SA Attribute Payload A payload that follows the Security Association payload and that describes group security attributes associated with the security association. SA Attribute payloads include the SAK, SAT, and GAP payloads.

Security Parameter Index An arbitrary value that is used by a receiver to identify a security association such as an IPsec ESP Security Association or a Rekey SA.

Traffic Encryption Key. The symmetric cipher key used to protect a data security protocol (e.g., IPsec ESP).

## 1.3. Acronyms and Abbreviations

The following acronyms and abbreviations are used throughout this document.

| | |
|---|---|
| AH | IP Authentication Header |
| ATD | Activation Time Delay |
| DOI | Domain of Interpretation |
| DTD | Deactivation Time Delay |
| ESP | IP Encapsulating Security Payload |
| GCKS | Group Controller/Key Server |
| GDOI | Group Domain of Interpretation |
| GAP | Group Associated Policy Payload |
| GM | Group Member |
| GSPD | Group Security Policy Database |
| IV | Initialization Vector |
| KD | Key Download Payload |
| KEK | Key Encryption Key |
| LKH | Logical Key Hierarchy |
| SA | Security Association |
| SAK | SA KEK Payload |
| SEQ | Sequence Number Payload |
| SAT | SA TEK Payload |
| SID | Sender-ID |
| SPI | Security Parameter Index |
| SSIV | Sender-Specific IV |
| TEK | Traffic Encryption Key |

TLV Type/Length/Value

TV Type/Value

## 2. GDOI Phase 1 Protocol

The GDOI GROUPKEY-PULL exchange is a Phase 2 protocol that MUST be protected by a Phase 1 protocol. The Phase 1 protocol can be any protocol that provides for the following protections:

- Peer Authentication

- Confidentiality

- 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.

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 7.1 describes how the ISAKMP Phase 1 protocols meet the requirements of a GDOI Phase 1 protocol.

### 2.1. 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.

### 2.2. UDP port

IANA has assigned port 848 for the use of GDOI; this allows for an implementation to use separate ISAKMP implementations to service GDOI and the Internet Key Exchange Protocol (IKE) [RFC5996]. A GCKS SHOULD listen on this port for GROUPKEY-PULL exchanges, and the GCKS MAY use this port to distribute GROUPKEY-PUSH messages. An ISAKMP Phase 1 exchange implementation supporting NAT traversal [RFC3947] MAY move to port 4500 to process the GROUPKEY-PULL exchange.

## 3. GROUPKEY-PULL Exchange

The goal of the GROUPKEY-PULL exchange is to establish a Rekey 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

It is important that a group member explicitly trust entities that it expects to act as a GCKS for a particular group. When no authorization is performed, it is possible for a rogue GDOI participant to perpetrate a man-in-the-middle attack between a group member and a GCKS [MP04]. A group member MUST specifically list each authorized GCKS in its Group Peer Authorization Database (GPAD) [RFC5374]. A group member MUST ensure that the Phase 1 identity of the GCKS is an authorized GCKS.

It is important that a GCKS explicitly authorize group members before providing them with group policy and keying material. A GCKS implementation SHOULD have a method of authorizing group members (e.g., by maintaining an authorization list). When the GCKS performs authorization, it MUST use the Phase 1 identity to authorize the GROUPKEY-PULL request for group policy and keying material.

### 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 ISAKMP HASH payloads [RFC2408] included in GROUPKEY-PULL messages. When using the Phase 1 defined in this document, SKEYID_a is derived according to [RFC2409]. Each GROUPKEY-PULL message hashes a uniquely defined set of values (described below) and includes the result in the HASH payload. 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 "liveness" as well as against replay of a recent GROUPKEY-PULL message. The replay attack is only possible 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 GCKS MUST NOT adjust its internal state (e.g., keeping a record of the GM) until it receives a message with Nr included properly in the HASH payload. This requirement ensures that replays of GDOI messages will not cause the GCKS to change the state of the group until it has confirmation that the initiating group member is live.

```
        Group Member                      GCKS
        ------------                      ----
    (1) HDR*, HASH(1), Ni, ID     -->
    (2)                           <--     HDR*, HASH(2), Nr, SA
    (3) HDR*, HASH(3) [,GAP]      -->
    (4)                           <--     HDR*, HASH(4), [SEQ,] KD

        * Protected by the Phase 1 SA; encryption occurs after HDR
```

Figure 2. GROUPKEY-PULL Exchange

Figure 2 demonstrates the four messages that are part of a GROUPKEY-PULL exchange. HDR is an ISAKMP header payload that uses the Phase 1 cookies and a message identifier (M-ID) as in ISAKMP. Following each HDR is a set of payloads conveying requests (messages 1 and 3 originated by the group member), or group policy and/or keying material (messages 2 and 4 originated by the GCKS).

Hashes are computed in the manner described within [RFC2409]. The HASH computation for each message is unique; it is shown in Figure 2 and below as HASH(n) where (n) represents the GROUPKEY-PULL message number. Each HASH calculation is a pseudo-random function ("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. The GM expects to find its nonce, Ni, in the HASH of a returned message, and the GCKS expects to see its nonce, Nr, in the HASH of a returned message. HASH(2), HASH(3), and HASH(4) also include nonce values previously passed in the protocol (i.e., Ni or Nr minus the payload header). The nonce passed in Ni is represented as Ni_b, and the nonce passed in Nr is represented as Nr_b. The HASH payloads prove that the peer has the Phase 1 secret (SKEYID_a) and the nonce for the exchange identified by message ID, M-ID.

```
     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 [ | GAP ])
     HASH(4) = prf(SKEYID_a, M-ID | Ni_b | Nr_b [ | SEQ ] | KD)
```

In addition to the Nonce and HASH payloads, the GM identifies the group it wishes to join through the ISAKMP ID payload.

The GCKS informs the member of the cryptographic policies of the group in the SA payload, which describes the DOI, KEK, and/or TEK keying material, authentication transforms, and other group policy. Each SPI is 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 Rekey SA, which is not negotiated but downloaded. Each SA TEK attribute contains a SPI as defined in Section 5.5 of this document.

After receiving and parsing the SA payload, the GM responds with an acknowledgement message proving its liveness. It optionally includes a GAP payload requesting resources.

The GCKS informs the GM of the value of the sequence number in the SEQ payload. This sequence number provides anti-replay state associated with a KEK, and its knowledge ensures that the GM will not accept GROUPKEY-PUSH messages sent prior to the GM joining the group. The SEQ payload has no other use and is omitted from the GROUPKEY-PULL exchange when a KEK attribute is not included in the SA payload. When a SEQ payload is included in the GROUPKEY-PULL exchange, it includes the most recently used sequence number for the group. At the conclusion of a GROUPKEY-PULL exchange, the initiating group member MUST NOT accept any rekey message with both the KEK attribute SPI value and a sequence number less than or equal to the one received during the GROUPKEY-PULL exchange. When the first group member initiates a GROUPKEY-PULL exchange, the GCKS provides a Sequence Number of zero, since no GROUPKEY-PUSH messages have yet been sent. Note the sequence number increments only with GROUPKEY-PUSH messages. The GROUPKEY-PULL exchange distributes the current sequence number to the group member. The sequence number resets to a value of one with the usage of a new KEK attribute. Thus, the first packet sent for a given Rekey SA will have a Sequence Number of 1. The sequence number increments with each successive rekey.

The GCKS always returns a KD payload containing keying material to the GM. If a Rekey 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.

### 3.2.1. ISAKMP Header Initialization

Cookies are used in the ISAKMP header to identify a particular GDOI session. The GDOI GROUPKEY-PULL exchange uses cookies according to ISAKMP [RFC2408].

Next Payload identifies an ISAKMP or GDOI payload (see Section 5).

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

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

Flags, Message ID, and Length are according to ISAKMP (Section 3.1 of [RFC2408]). The Commit flag is not useful because there is no synchronization between the GROUPKEY-PULL exchange and the data traffic protected by the policy distributed by the GROUPKEY-PULL exchange.

### 3.3. Group Member Operations

Before a GM contacts the GCKS, it needs to determine the group identifier and acceptable Phase 1 policy via an out-of-band method. Phase 1 is initiated using the GDOI DOI in the SA payload. Once Phase 1 is complete, the GM state machine moves to the GDOI protocol.

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

Upon receipt of the second GDOI message, the GM validates HASH(2), extracts the nonce Nr, and interprets the SA payload (including its SA Attribute payloads) . The SA payload contains policy describing the security protocol and cryptographic protocols used by the group. This policy describes the Rekey SA (if present), Data-Security SAs, and other group policy. If the policy in the SA payload is acceptable to the GM, it continues the protocol. Otherwise, the GM SHOULD tear down the Phase 1 session after notifying the GCKS with an ISAKMP Informational Exchange containing a Delete payload.

When constructing the third GDOI message, it first reviews each Data-Security SA given to it. If any describe the use of a counter mode cipher, the GM determines whether it requires more than one Sender-ID (SID) (see Section 3.5). If so, it requests the required number of Sender-IDs for its exclusive use within the counter mode nonce as described in Section 5.4 of this document. The GM then completes construction of the third GDOI message by creating HASH(3).

Upon receipt of the fourth GDOI message, the GM validates HASH(4).

If the SEQ payload is present, the sequence number included in the SEQ payload asserts the lowest acceptable sequence number present in a future GROUPKEY-PUSH message. But if the KEK associated with this sequence number had been previously installed, due to the asynchronous processing of GROUPKEY-PULL and GROUPKEY-PUSH messages, this sequence number may be lower than the sequence number contained in the most recently received GROUPKEY-PUSH message. In this case, the sequence number value in the SEQ payload MUST be considered stale and ignored.

The GM interprets the KD key packets, where each key packet includes the keying material for SAs distributed in the SA payload. Keying material is matched by comparing the SPI in each key packet to SPI values previously sent in the SA payloads. Once TEKs and policy are matched, the GM provides them to the data security subsystem, and it is ready to send or receive packets matching the TEK policy. If this group has a KEK, the KEK policy and keys are marked as ready for use, and the GM knows to expect a sequence number not less than the one distributed in the SEQ payload. The GM is now ready to receive GROUPKEY-PUSH messages.

If the KD payload included an LKH array of keys, the GM takes the last key in the array as the group KEK. The array is then stored without further processing.

### 3.4. GCKS Operations

The GCKS 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) and extracts the Ni and group identifier in the ID payload. It verifies that its database contains the group information for the group identifier and that the GM is authorized to participate in the group.

The GCKS constructs the second GDOI message, including a nonce Nr, and the policy for the group in an SA payload, followed by SA Attribute payloads (i.e, SA KEK, GAP, and/or SA TEK payloads) according to the GCKS policy. (See Section 5.2.1 for details on how the GCKS chooses which payloads to send.)

Upon receipt of the third GDOI message, the GCKS validates HASH(3). If the message includes a GAP payload, it caches the requests included in that payload for the use of constructing the fourth GDOI message.

The GCKS constructs the fourth GDOI message, including the SEQ payload (if the GCKS sends rekey messages), and the KD payload containing keys corresponding to policy previously sent in the SA TEK and SA KEK payloads. If a group management algorithm is defined as part of group policy, the GCKS will first insert the group member into the group management structure (e.g., a leaf in the LKH tree), and then create an LKH array of keys and include it in the KD payload. The first key in the array is associated with the group member leaf node, followed by each LKH node above it in the tree, culminating with the root node (which is also the KEK). If one or more Data-Security SAs distributed in the SA payload included a counter mode of operation, the GCKS includes at least one SID value in the KD payload, and possibly more depending on a request received in the third GDOI message.

## 3.5. Counter-Modes of Operation

Several new counter-based modes of operation have been specified for ESP (e.g., AES-CTR [RFC3686], AES-GCM [RFC4106], AES-CCM [RFC4309], AES-GMAC [RFC4543]) and AH (e.g., AES-GMAC [RFC4543]). These counter-based modes require that no two senders in the group ever send a packet with the same Initialization Vector (IV) using the same cipher key and mode. This requirement is met in GDOI when the following requirements are met:

- The GCKS distributes a unique key for each Data-Security SA.
- The GCKS uses the method described in [RFC6054], which assigns each sender a portion of the IV space by provisioning each sender with one or more unique SID values.

When at least one Data-Security SA included in the group policy includes a counter-mode, the GCKS automatically allocates and distributes one SID to each group member acting in the role of sender on the Data-Security SA. The SID value is used exclusively by the group member to which it was allocated. The group member uses the same SID for each Data-Security SA specifying the use of a counter-based mode of operation. A GCKS MUST distribute unique keys for each Data-Security SA including a counter-based mode of operation in order to maintain a unique key and nonce usage.

When a group member receives a Data-Security SA in a SA TEK payload for which it is a sender, it can choose to request one or more SID values. Requesting a value of 1 is not necessary since the GCKS will automatically allocate exactly one to the sending group member. A group member MUST request as many SIDs matching the number of encryption modules in which it will be installing the TEKs in the outbound direction. Alternatively, a group member MAY request more than one SID and use them serially. This could be useful when it is anticipated that the group member will exhaust their range of Data-Security SA nonces using a single SID too quickly (e.g., before the time-based policy in the TEK expires).

When group policy includes a counter-based mode of operation, a GCKS SHOULD use the following method to allocate SID values, which ensures that each SID will be allocated to just one group member.

1. A GCKS maintains a SID-counter, which records which SIDs have been allocated. SIDs are allocated sequentially, with the first SID allocated to be zero.

2. Each time a SID is allocated, the current value of the counter is saved and allocated to the group member. The SID-counter is then incremented in preparation for the next allocation.

3. When the GCKS distributes a Data-Security SA specifying a counter-based mode of operation, and a group member is a sender, a group member may request a count of SIDs in a GAP payload. When the GCKS receives this request, it increments the SID-counter once for each requested SID, and distributes each SID value to the group member.

4. A GCKS allocates new SID values for each GROUPKEY-PULL exchange originated by a sender, regardless of whether a group member had previously contacted the GCKS. In this way, the GCKS does not have a requirement of maintaining a record of which SID values it had previously allocated to each group member. More importantly, since the GCKS cannot reliably detect whether the group member had sent data on the current group Data-Security SAs, it does not know which Data-Security counter-mode nonce values a group member has used. By distributing new SID values, the key server ensures that each time a conforming group member installs a Data-Security SA it will use a unique set of counter-based mode nonces.

5. When the SID-counter maintained by the GCKS reaches its final SID value, no more SID values can be distributed. Before distributing any new SID values, the GCKS MUST delete the Data-Security SAs for the group, followed by creation of new Data-Security SAs, and resetting the SID-counter to its initial value.

6. The GCKS SHOULD send a GROUPKEY-PUSH message deleting all Data-Security SAs and the Rekey SA for the group. This will result in the group members initiating a new GROUPKEY-PULL exchange, in which they will receive both new SID values and new Data-Security SAs. The new SID values can safely be used because they are only used with the new Data-Security SAs. Note that deletion of the Rekey SA is necessary to ensure that group members receiving a GROUPKEY-PUSH exchange before the re-register do not inadvertently use their old SIDs with the new Data-Security SAs.

Using the method above, at no time can two group members use the same IV values with the same Data-Security SA key.

## 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 Rekey SA KEK or KEK array, and/or it creates a new Data-Security SA.

```
     GM                    GCKS
     --                    ----
                           <---- HDR*, SEQ, [D,] SA, KD, SIG

     * Protected by the Rekey SA KEK; encryption occurs after HDR
```

Figure 3. GROUPKEY-PUSH Message

HDR is defined below. The SEQ payload is defined in Section 5 ("Payloads"). One or more D (Delete) payloads (further described in Section 5.9) optionally specify the deletion of existing group policy. The SA defines the group policy for replacement Rekey SA and/or Data-Security SAs as described in Section 5, with the KD providing keying material for those SAs.

The SIG payload includes a signature of a hash of the entire GROUPKEY-PUSH message (excepting the SIG payload octets) before it has been encrypted. The HASH is taken over the string 'rekey', the GROUPKEY-PUSH HDR, followed by all payloads preceding the SIG payload. The prefixed string ensures that the signature of the Rekey datagram cannot be used for any other purpose in the GDOI protocol. The SIG payload is created using the signature of the above hash, with the receiver verifying the signature using a public key retrieved in a previous GDOI exchange. The current KEK (also previously distributed in a GROUPKEY-PULL exchange or GROUPKEY-PUSH message) encrypts all the payloads following the GROUPKEY-PUSH HDR. Note: The rationale for this order of operations is given in Section 7.3.5.

If the SA defines the use of a single KEK or an LKH KEK array, KD MUST contain a corresponding KEK or KEK array for a new Rekey SA, which has a new cookie pair. When the KD payload carries a new SA KEK attribute (Section 5.3), a Rekey 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. Note the first packet for the given Rekey SA encrypted with the new KEK attribute will have a Sequence number of 1. 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 5.6).

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. Use of Signature Keys

A signing key should not be used in more than one context (e.g., used for host authentication and also for message authentication). Thus, the GCKS SHOULD NOT use the same key to sign the SIG payload in the GROUPKEY-PUSH message as was used for authentication in the GROUPKEY-PULL exchange.

### 4.2. ISAKMP Header Initialization

Unlike ISAKMP, the cookie pair is completely determined by the GCKS. The cookie pair in the GDOI ISAKMP header identifies the Rekey 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).

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

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

Flags MUST have the Encryption bit set according to Section 3.1 of [RFC2408]. All other bits MUST be set to zero.

Message ID MUST be set to zero.

Length is according to ISAKMP (Section 3.1 of [RFC2408]).

### 4.3. GCKS Operations

GCKS 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 that is 1 greater than the previous rekey datagram. If the message is using the new KEK attribute for the first time, the SEQ is reset to 1 in this message.

An SA payload is then added. This is identical in structure and meaning to an SA payload sent in a GROUPKEY-PULL exchange. If there are changes to the KEK (including due to group members being excluded, 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 KEKs (or a KEK update array) are included. A KEK update array is created by first determining which group members have been excluded, generating new keys as necessary, and then distributing LKH update arrays sufficient to provide the new KEK to remaining group members (see Section 5.4.1 of [RFC2627] for details). TEKs are also sent for each SA_TEK attribute included in the SA payload.

In the penultimate step, the GCKS creates the SIG payload and adds it to the datagram.

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

## 4.4. 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 sequence number in the SEQ payload is validated to ensure that it is greater than the previously received sequence number. The SIG payload is then validated. If the signature fails, the message is discarded.

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 Data-Security SAs being added to the system. If the KD payload includes an LKH update array, the group member compares the LKH ID in each key update packet to the LKH IDs that it holds. If it finds a match, it decrypts the key using the key prior to it in the key array and stores the new key in the LKH key array that it holds. The final decryption yields the new group KEK.

If the SA payload includes one or more Data-Security SAs including a counter-mode of operation and if the receiving group member is a sender for that SA, the group member uses its current SID value with the Data-Security SAs to create counter-mode nonces. If it is a sender and does not hold a current SID value, it MUST NOT install the Data-Security SAs. It MAY initiate a GROUPKEY-PULL exchange to the GCKS in order to obtain a SID value (along with current group policy).

## 5. Payloads and Defined Values

This document specifies use of several ISAKMP payloads, which are defined in accordance with [RFC2408]. The following payloads are used as defined in [RFC2408].

| Next Payload Type | Value |
|---|---|
| Hash Payload (HASH) | 8 |
| Signature (SIG) | 9 |

The following payloads are extended or further specified.

| Next Payload Type | Value |
|---|---|
| Security Association (SA) | 1 |
| Identification (ID) | 5 |
| Nonce (N) | 10 |
| Delete (D) | 12 |

Several payload formats specific to the group security exchanges are required.

| Next Payload Type | Value |
|---|---|
| SA KEK (SAK) | 15 |
| SA TEK (SAT) | 16 |
| Key Download (KD) | 17 |
| Sequence Number (SEQ) | 18 |
| Group Associated Policy (GAP) | 22 |

All multi-octet fields in GDOI payloads representing integers are laid out in big endian order (also known as "most significant byte first" or "network byte order").

All payloads including an ISAKMP Generic Payload Header create a Payload Length field that includes the length of the generic payload header (Section 3.2 of [RFC2408]).

## 5.1. Identification Payload

The Identification payload is defined in [RFC2408]. For the GDOI, it 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 IPv4 or IPv6 multicast address, or may specify a more general identifier, such as one that represents a set of related multicast streams.

When used with the GDOI, the DOI-Specific ID Data field MUST be set to 0.

When used with the GDOI, the ID_KEY_ID ID Type MUST be supported by a conforming implementation and MUST specify a 4-octet group identifier as its value. Implementations MAY also support other ID Types.

## 5.2. Security Association Payload

The Security Association payload is defined in [RFC2408]. For the GDOI, it is used by the GCKS to assert security attributes for both Rekey 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            !
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
```

Figure 4. Security Association Payload

The Security Association payload fields are defined as follows:

- 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 an SA Attribute payload; it MUST be the next payload following the Security Association type payload.

- RESERVED (1 octet) -- MUST be zero.

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

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

- Situation (4 octets) -- MUST be zero.

- SA Attribute Next Payload (2 octets) -- MUST be the code for an SA Attribute payload type. See Section 5.2.1 for a description of which circumstances are required for each payload type to be present.

- RESERVED (2 octets) -- MUST be zero.

### 5.2.1. SA Attribute Payloads

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 payload, zero or one GAP payload, and zero or more SAT payloads, where either one SAK or SAT payload MUST be present. When present, the order of the SA Attribute payloads MUST be: SAK, GAP, and SATs.

This latitude regarding SA Attribute payloads allows various group policies to be accommodated. For example, if the group policy does not require the use of a Rekey 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 Rekey 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 Rekey 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.

A GAP payload allows for the distribution of group-wide policy, such as instructions as to when to activate and deactivate SAs.

## 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                              ~
      !                                                               !
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
      !                           RESERVED2                           !
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
      ~                        KEK Attributes                         ~
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
```

Figure 5. SA KEK Payload

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 GAP payload, SAT payload, or zero to indicate that no SA Attribute payloads follow.

- 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) [PROT-REG] for the GROUPKEY-PUSH 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 ISAKMP 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 MUST be ignored.

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

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

- 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 ISAKMP registry [ISAKMP-REG].

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

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

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

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

- SPI (16 octets) -- Security Parameter Index for the KEK. The SPI is 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.

- RESERVED2 (4 octets) -- MUST be zero. These octets represent fields previously defined but no longer used by GDOI.

- KEK Attributes -- Contains KEK policy attributes associated with the group. The following attributes may be present in a SAK payload. The attributes must follow the format defined in ISAKMP (Section 3.3 of [RFC2408]). 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 |
| RESERVED | 8 | B |
| Unassigned | 9-127 | |
| Private Use | 128-255 | |
| Unassigned | 256-32767 | |

The KEK_ALGORITHM and SIG_ALGORITHM attributes MUST be included; others are OPTIONAL and are included depending on group policy. The KEK_MANAGEMENT_ALGORITHM attribute MUST NOT be included in a GROUPKEY-PULL message, and MUST be ignored if present.

#### 5.3.1. 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 |
| Unassigned | 2-127 |
| Private Use | 128-255 |
| Unassigned | 256-65535 |

##### 5.3.1.1. LKH

This type indicates the group management method described in Section 5.4 of [RFC2627]. A general discussion of LKH operations can also be found in Section 6.3 of "Multicast and Group Security" [HD03]

#### 5.3.2. KEK_ALGORITHM

The KEK_ALGORITHM class specifies the encryption algorithm in which the KEK is used to provide confidentiality for the GROUPKEY-PUSH message. Defined values are specified in the following table. A GDOI implementation MUST abort if it encounters an attribute or capability that it does not understand.

| Algorithm Type | Value |
|---|---|
| RESERVED | 0 |
| KEK_ALG_DES | 1 |
| KEK_ALG_3DES | 2 |
| KEK_ALG_AES | 3 |
| Unassigned | 4-127 |
| Private Use | 128-255 |
| Unassigned | 256-32767 |

If a KEK_MANAGEMENT_ALGORITHM is defined that specifies 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 that are included as part of the management.

#### 5.3.2.1. KEK_ALG_DES

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

#### 5.3.2.2. KEK_ALG_3DES

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

#### 5.3.2.3. KEK_ALG_AES

This type specifies AES as described in [FIPS197]. The mode of operation for AES is CBC as defined in [SP.800-38A].

### 5.3.3. KEK_KEY_LENGTH

The KEK_KEY_LENGTH class specifies the KEK Algorithm key length (in bits). The Group Controller/Key Server (GCKS) adds the KEK_KEY_LENGTH attribute to the SA payload when distributing KEK policy to group members. The group member verifies whether or not it has the capability of using a cipher key of that size. If the cipher definition includes a fixed key length (e.g., KEK_ALG_3DES), the group member can make its decision solely using the KEK_ALGORITHM attribute and does not need the KEK_KEY_LENGTH attribute. Sending the KEK_KEY_LENGTH attribute in the SA payload is OPTIONAL if the KEK cipher has a fixed key length. Also, note that the KEK_KEY_LEN includes only the actual length of the cipher key (the IV length is not included in this attribute).

### 5.3.4. 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 4-octet number defining a valid time period in seconds.

### 5.3.5. SIG_HASH_ALGORITHM

SIG_HASH_ALGORITHM specifies the SIG payload hash algorithm. The following table defines the algorithms for SIG_HASH_ALGORITHM.

| Algorithm Type | Value |
|---|---|
| Reserved | 0 |
| SIG_HASH_MD5 | 1 |
| SIG_HASH_SHA1 | 2 |
| SIG_HASH_SHA256 | 3 |
| SIG_HASH_SHA384 | 4 |
| SIG_HASH_SHA512 | 5 |
| Unassigned | 6-127 |
| Private Use | 128-255 |
| Unassigned | 256-65535 |

The SHA hash algorithms are defined in the Secure Hash Standard [FIPS180-3.2008].

If the SIG_ALGORITHM is SIG_ALG_ECDSA-256, SIG_ALG_ECDSA-384, or SIG_ALG_ECDSA-521, the hash algorithm is implicit in the definition, and SIG_HASH_ALGORITHM is OPTIONAL in a SAK payload.

### 5.3.6. 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 |
| SIG_ALG_ECDSA-256 | 4 |
| SIG_ALG_ECDSA-384 | 5 |
| SIG_ALG_ECDSA-521 | 6 |
| Unassigned | 7-127 |
| Private Use | 128-255 |
| Unassigned | 256-65535 |

#### 5.3.6.1. SIG_ALG_RSA

This algorithm specifies the RSA digital signature algorithm using the EMSA-PKCS1-v1_5 encoding method, as described in [RFC3447].

#### 5.3.6.2. SIG_ALG_DSS

This algorithm specifies the DSS digital signature algorithm as described in Section 4 of [FIPS186-3].

#### 5.3.6.3. SIG_ALG_ECDSS

This algorithm specifies the Elliptic Curve Digital Signature Algorithm as described in Section 5 of [FIPS186-3]. This definition is deprecated in favor of the SIG_ALG_ECDSA family of algorithms.

#### 5.3.6.4. SIG_ALG_ECDSA-256

This algorithm specifies the 256-bit Random ECP Group, as described in [RFC5903]. The format of the signature in the SIG payload MUST be as specified in [RFC4754].

#### 5.3.6.5. SIG_ALG_ECDSA-384

This algorithm specifies the 384-bit Random ECP Group, as described in [RFC5903]. The format of the signature in the SIG payload MUST be as specified in [RFC4754].

#### 5.3.6.6. SIG_ALG_ECDSA-521

This algorithm specifies the 521-bit Random ECP Group, as described in [RFC5903]. The format of the signature in the SIG payload MUST be as specified in [RFC4754].

### 5.3.7. SIG_KEY_LENGTH

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

## 5.4. Group Associated Policy

A GCKS may have group-specific policy that is not distributed in an SA TEK or SA KEK. Some of this policy is relevant to all group members, and some is sender-specific policy for a particular group member. The former can be distributed in either a GROUPKEY-PULL or GROUPKEY-PUSH exchange, whereas the latter MUST only be sent in a GROUPKEY-PULL exchange. Additionally, a group member sometimes has the need to make policy requests for resources of the GCKS in a

GROUPKEY-PULL exchange. GDOI distributes this associated group policy and the policy requests in the Group Associated Policy (GAP) payload.

The GAP payload can be distributed by the GCKS as part of the SA payload. It follows any SA KEK payload and is placed before any SA TEK payloads. In the case that group policy does not include an SA KEK, the SA Attribute Next Payload field in the SA payload MAY indicate the GAP payload.

The GAP payload can be optionally included by a group member in message 3 of the GROUPKEY-PULL exchange in order to make policy requests.

The GAP 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         !
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
    !               Group Associated Policy Attributes              ~
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
```

Figure 6. GAP Payload

The GAP payload fields are defined as follows:

- Next Payload (1 octet) -- Identifies the next payload present in the GROUPKEY-PULL or the GROUPKEY-PUSH message. The only valid next payload type for this message is an SA TEK or zero to indicate there are no more security association attributes.

- RESERVED (1 octet) -- MUST be zero.

- Payload Length (2 octets) -- Length of this payload, including the GAP header and Attributes.

- Group Associated Policy Attributes (variable) -- Contains attributes following the format defined in Section 3.3 of [RFC2408]. 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).

| Attribute Type | Value | Type |
|---|---|---|
| RESERVED | 0 | |
| ACTIVATION_TIME_DELAY | 1 | B |
| DEACTIVATION_TIME_DELAY | 2 | B |
| SENDER_ID_REQUEST | 3 | B |
| Unassigned | 4-127 | |
| Private Use | 128-255 | |
| Unassigned | 256-32767 | |

Several group associated policy attributes are defined in this memo. A GDOI implementation MUST abort if it encounters an attribute or capability that it does not understand. The values for these attributes are included in the IANA Considerations section of this memo.

### 5.4.1. ACTIVATION_TIME_DELAY/DEACTIVATION_TIME_DELAY

Section 4.2.1 of [RFC5374] specifies a key rollover method that requires two values be given it from the group key management protocol. The ACTIVATION_TIME_DELAY attribute allows a GCKS to set the Activation Time Delay (ATD) for SAs generated from TEKs. The ATD defines how long after receiving new SAs that they are to be activated by the GM. The ATD value is in seconds.

The DEACTIVATION_TIME_DELAY allows the GCKS to set the Deactivation Time Delay (DTD) for previously distributed SAs. The DTD defines how long after receiving new SAs that it SHOULD deactivate SAs that are destroyed by the rekey event. The value is in seconds.

The values of ATD and DTD are independent. However, the most effective policy will have the DTD value be the larger value, as this allows new SAs to be activated before older SAs are deactivated. Such a policy ensures that protected group traffic will always flow without interruption.

### 5.4.2. SENDER_ID_REQUEST

The SENDER_ID_REQUEST attribute is used by a group member to request SIDs during the GROUPKEY-PULL message, and includes a count of how many SID values it desires.

### 5.5. 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           ~
    +-+-+-+-+-+-+-+-+                                               ~
    ~                                                               ~
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
```

Figure 7. SA TEK 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 |
| GDOI_PROTO_IPSEC_AH | 2 |
| Unassigned | 3-127 |
| Private Use | 128-255 |

- o TEK Protocol-Specific Payload (variable) -- Payload which describes the attributes specific for the Protocol-ID.

### 5.5.1. GDOI_PROTO_IPSEC_ESP/GDOI_PROTO_IPSEC_AH

The TEK Protocol-Specific payload for ESP and AH 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                  ~
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
```

Figure 8. ESP/AH TEK Payload

The SAT payload fields are defined as follows:

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

- 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 ISAKMP registry [ISAKMP-REG].

- 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 MUST be ignored.

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

- SRC Identification Data (variable length) -- Value, as indicated by the SRC ID Type. Set to 3 octets or zero for multiple-source multicast groups that use a common TEK for all senders.

- 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 ISAKMP registry [ISAKMP-REG].

- DST ID Prot (1 octet) -- Value describing an IP protocol ID (e.g., UDP/TCP) [PROT-REG]. A value of zero means that the DST ID Prot field MUST be ignored.

- 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 MUST be ignored.

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

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

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

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

- RFC 2407 Attributes -- ESP and AH Attributes from Section 4.5 of [RFC2407]. The GDOI supports all IPsec DOI SA Attributes for GDOI_PROTO_IPSEC_ESP and GDOI_PROTO_IPSEC_AH, excluding the Group Description (Section 4.5 of [RFC2407]), which MUST NOT be sent by a GDOI implementation and is ignored by a GDOI implementation if received. The following attributes MUST be supported by an implementation supporting ESP and AH: SA Life Type, SA Life Duration, and Encapsulation Mode. An implementation supporting ESP MUST also support the Authentication Algorithm attribute if the ESP transform includes authentication. The Authentication Algorithm attribute of the IPsec DOI is group authentication in GDOI.

#### 5.5.1.1. New IPsec Security Association Attributes

"Multicast Extensions to the Security Architecture for the Internet Protocol" (RFC 5374) introduces new requirements for a group key management system distributing IPsec policy. It also defines new attributes as part of the Group Security Policy Database (GSPD). These attributes describe policy that a group key management system must convey to a group member in order to support those extensions. The GDOI SA TEK payload distributes IPsec policy using IPsec security association attributes defined in [ISAKMP-REG]. This section defines how GDOI can convey the new attributes as IPsec Security Association Attributes.

#### 5.5.1.1.1. Address Preservation

Applications use the extensions in [RFC5374] to copy the IP addresses into the outer IP header when encapsulating an IP packet as an IPsec tunnel mode packet. This allows an IP multicast packet to continue to be routed as a IP multicast packet. This attribute also provides the necessary policy so that the GDOI group member can appropriately set up the GSPD. The following table defines values for the Address Preservation attribute.

| Address Preservation Type | Value |
|---|---|
| Reserved | 0 |
| None | 1 |
| Source-Only | 2 |
| Destination-Only | 3 |
| Source-and-Destination | 4 |
| Unassigned | 5-61439 |
| Private Use | 61440-65535 |

Depending on group policy, several address preservation methods are possible: no address preservation ("None"), preservation of the original source address ("Source-Only"), preservation of the original destination address ("Destination-Only"), or both addresses ("Source-and-Destination"). If this attribute is not included in a GDOI SA TEK payload provided by a GCKS, then Source-and-Destination address preservation has been defined for the SA TEK.

#### 5.5.1.1.2. SA Direction

Depending on group policy, an IPsec SA created from an SA TEK payload is defined to be in the sending and/or receiving direction. The following table defines values for the SA Direction attribute.

| Name | Value |
|---|---|
| Reserved | 0 |
| Sender-Only | 1 |
| Receiver-Only | 2 |
| Symmetric | 3 |
| Unassigned | 4-61439 |
| Private Use | 61440-65535 |

SA TEK policy used by multiple senders MUST be installed in both the sending and receiving direction ("Symmetric"), whereas SA TEK for a single sender SHOULD be installed in the receiving direction by receivers ("Receiver-Only") and in the sending direction by the sender ("Sender-Only").

An SA TEK payload that does not include the SA Direction attribute is treated as a Symmetric IPsec SA. Note that Symmetric is the only value that can be meaningfully described for an SA TEK distributed in a GROUPKEY-PUSH message. Alternatively, Receiver-Only could be distributed, but group senders would need to be configured to not receive GROUPKEY-PUSH messages in order to retain their role.

### 5.5.2. Other Security Protocols

Besides ESP and AH, 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.

- The Protocol-ID for the particular Security Protocol
- The SPI Size
- The method of SPI generation
- 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 that MAY be shared by more than two entities.

## 5.6. 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.

```
       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                                ~
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
```

Figure 9. Key Download Payload

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 key packets being passed in this data block.

o  Key Packets (variable) -- 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                      ~
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
```

Figure 10. Key Packet

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 |
| SID | 4 |
| Unassigned | 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 a 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.

### 5.6.1. 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 (Section 3.3 of [RFC2408]). 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 |
| Unassigned | 4-127 | |
| Private Use | 128-255 | |
| Unassigned | 256-32767 | |

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 Rekey SA. At least one TEK must be included in each Rekey KD payload. Multiple TEKs may be included if multiple streams associated with the SA are to be rekeyed.

When an algorithm specification specifies the format of the keying material, the value transported in the KD payload for that key is passed according to that specification. The keying material may contain information besides a key. For example, "The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)" [RFC4106] defines a salt value as part of KEYMAT.

#### 5.6.1.1. 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 bits). 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).

#### 5.6.1.2. 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 GM. HMAC-SHA1 keys will consist of 160 bits [RFC2404], and HMAC-MD5 keys will consist of 128 bits [RFC2403]. HMAC-SHA2 and AES-GMAC keys will have a key length equal to the output length of the hash functions [RFC4868] [RFC4543].

#### 5.6.1.3. 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.

### 5.6.2. 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 (Section 3.3 of [RFC2408]). 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 |
| Unassigned | 3-127 | |
| Private Use | 128-255 | |
| Unassigned | 256-32767 | |

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

#### 5.6.2.1. 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 IV, an explicit IV MUST be included in the KEK_ALGORITHM_KEY before the actual key.

#### 5.6.2.2. 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.

### 5.6.3. LKH Download Type

The LKH key packet is comprised of attributes representing different nodes 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 (Section 3.3 of [RFC2408]). 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 |
| Unassigned | 4-127 | |
| Private Use | 128-255 | |
| Unassigned | 256-32767 | |

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

#### 5.6.3.1. 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                          !
 ~                                                               ~
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 11. LKH Download Array

The KEK_LKH attribute fields are defined as follows:

- LKH version (1 octet) -- Version of the LKH data format. Must be one.
- Number of LKH Keys (2 octets) -- This value is the number of distinct LKH keys in this sequence.
- 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                           ~
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 12. LKH Key

- LKH ID (2 octets) -- Identity of the LKH node. A GCKS is free to choose the ID in an implementation-specific manner (e.g., the position of this key in a binary tree structure used by LKH).

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

- RESERVED (1 octet) -- Unused; set to zero.

- Key Creation Date (4 octets) -- Unsigned time value defining a valid time period in seconds representing the number of seconds since 0 hours, 0 minutes, 0 seconds, January 1, 1970, Coordinated Universal Time (UTC), without including leap seconds. [RFC5905]. This is the time 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.

- Key Expiration Date (4 octets) -- Unsigned time value defining a valid time period in seconds representing the number of seconds since 0 hours, 0 minutes, 0 seconds, January 1, 1970, Coordinated Universal Time (UTC), without including leap seconds. [RFC5905]. This is the time when this key is no longer valid for use. A time value of zero indicates that this key does not have an expiration time.

- Key Handle (4 octets) -- Value assigned by the GCKS to uniquely identify a key within an LKH ID. Each new key distributed by the GCKS for this node will have a key handle identity distinct from previous or successive key handles specified for this node.

- o Key Data (variable length) -- Key data, which is dependent on the Key Type algorithm for its format. If the mode of operation for the algorithm requires an IV, an explicit IV MUST be included in the Key Data field prepended to 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.

#### 5.6.3.2. 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 the previous section.

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                          !
       ~                                                               ~
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 13. LKH Update Array

- o LKH version (1 octet) -- Version of the LKH data format. Must be one.

- o Number of LKH Keys (2 octets) -- Number of distinct LKH keys in this sequence.

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

- LKH ID (2 octets) -- Node identifier associated with the key used to encrypt the first LKH Key.

- RESERVED2 (2 octets) -- Unused; set to zero.

- Key Handle (4 octets) -- Value assigned by the GCKS to uniquely identify the key within the LKH ID used to encrypt the first LKH Key.

The LKH Keys are as defined in the previous section. 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.

#### 5.6.3.3. 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.

### 5.6.4. SID Download Type

This attribute is used to download one or more Sender-ID (SID) values for the exclusive use of a group member.

The SID Download Type does not require an SPI. When the KD Type is SID, the SPI Size field MUST be zero, and the SPI field is omitted.

| SID Class | Value | Type |
|---|---|---|
| RESERVED | 0 | |
| NUMBER_OF_SID_BITS | 1 | B |
| SID_VALUE | 2 | V |
| Unassigned | 3-128 | |
| Private Use | 129-255 | |
| Unassigned | 256-32767 | |

Because a SID value is intended for a single group member, the SID Download type MUST NOT be distributed in a GROUPKEY-PUSH message distributed to multiple group members.

#### 5.6.4.1. NUMBER_OF_SID_BITS

The NUMBER_OF_SID_BITS class declares how many bits of the cipher nonce in which to represent a SID value. This value is applied to each SID value distributed in the SID Download.

#### 5.6.4.2. SID_VALUE

The SID_VALUE class declares a single SID value for the exclusive use of the group member. Multiple SID_VALUE attributes MAY be included in a SID Download.

#### 5.6.4.3. Group Member Semantics

The SID_VALUE attribute value distributed to the group member MUST be used by that group member as the SID field portion of the IV for all Data-Security SAs including a counter-based mode of operation distributed by the GCKS as a part of this group.

When the Sender-Specific IV (SSIV) field for any Data-Security SA is exhausted, the group member MUST no longer act as a sender on that SA using its active SID. The group member SHOULD re-register, at which time the GCKS will issue a new SID to the group member, along with either the same Data-Security SAs or replacement ones. The new SID replaces the existing SID used by this group member and also resets the SSIV value to its starting value. A group member MAY re-register prior to the actual exhaustion of the SSIV field to avoid dropping data packets due to the exhaustion of available SSIV values combined with a particular SID value.

GROUPKEY-PUSH message may include Data-Security SAs that are distributed to the group member for the first time. A SID previously issued to the receiving group member is used with counter-based mode of operation Data-Security SAs on which the group member acts as a sender. Because this Data-Security SA has not previously been used for transmission, the SSIV field should be set to its starting value.

#### 5.6.4.4. GCKS Semantics

If any KD payload includes keying material that is associated with a counter-mode of operation, a SID Download Type KD payload containing at least one SID_VALUE attribute MUST be included.

The GCKS MUST NOT send the SID Download Type KD payload as part of a GROUPKEY-PUSH message because distributing the same sender-specific policy to more than one group member will reduce the security of the group.

## 5.7. Sequence Number Payload

The Sequence Number (SEQ) Payload provides an anti-replay protection for GROUPKEY-PUSH messages. Its use is similar to the Sequence Number field defined in the IPsec ESP protocol [RFC4303].

```
       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                          !
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 14. Sequence Number Payload

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. MUST be a value of 8.

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 payload.

## 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 octets.

## 5.9. Delete

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 (Section 3.15 of [RFC2408]) 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:

- The Domain of Interpretation field contains the GDOI DOI.

- The Protocol-ID field contains TEK protocol ID values defined in Section 5.5 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. Thus, if a TEK SA and a KEK SA are to be deleted, their SPI values MUST be sent in different Delete payloads.

There may be circumstances where the GCKS may want to start over with a clean slate. If the administrator is no longer confident in the integrity of the group, the GCKS can signal deletion of all policy of a particular TEK protocol by sending a TEK with an SPI value equal to zero in the delete payload. For example, if the GCKS wishes to remove all the KEKs and all the TEKs in the group, the GCKS SHOULD send a delete payload with an SPI of zero and a Protocol-ID of a TEK Protocol-ID value, followed by another delete payload with an SPI value of zero and Protocol-ID of zero, indicating that the KEK SA should be deleted.

# 6. Algorithm Selection

For GDOI implementations to interoperate, they must support one or more security algorithms in common. This section specifies the security algorithm implementation requirements for standards-conformant GDOI implementations. In all cases, the choices are intended to maintain at least 112 bits of security [SP.800-131].

Algorithms not referenced in this section MAY be used.

### 6.1. KEK

These tables list the algorithm selections for values related to the KEK.

| Requirement | KEK Management Algorithm |
|---|---|
| SHOULD | LKH |

| Requirement | KEK Algorithm (notes) |
|---|---|
| MUST | KEK_ALG_AES with 128-bit keys |
| SHOULD NOT | KEK_ALG_DES (1) |

| Requirement | KEK Signature Hash Algorithm (notes) |
|---|---|
| MUST | SIG_HASH_SHA256 |
| SHOULD | SIG_HASH_SHA1 (2) |
| SHOULD NOT | SIG_HASH_MD5 (3) |

| Requirement | KEK Signature Algorithm (notes) |
|---|---|
| MUST | SIG_ALG_RSA with 2048-bit keys |

Notes:

(1) DES, with its small key size and corresponding security strength, is of questionable security for general use

(2) The use of SIG_HASH_SHA1 as a signature hash algorithm used with GROUPKEY-PUSH messages remains safe at the time of this writing, and it is a widely deployed signature hash algorithm.

(3) Although a real weakness with second preimage resistance with MD5 has not been found at the time of this writing, the security strength of MD5 has been shown to be rapidly declining over time, and its use should be understood and carefully weighed.

### 6.2. TEK

The following table lists the requirements for Security Protocol support for an implementation.

| Requirement | KEK Management Algorithm |
|---|---|
| MUST | GDOI_PROTO_IPSEC_ESP |

## 7. Security Considerations

GDOI is a security association (SA) management protocol for groups of senders and receivers. This protocol performs authentication of communicating protocol participants (Group Member, Group Controller/ Key Server). It provides confidentiality of key management messages, and it provides source authentication of those messages. GDOI includes defenses against man-in-middle, connection hijacking, replay, reflection, and denial-of-service (DoS) attacks on unsecured networks. GDOI assumes the network is not secure and may be under the complete control of an attacker.

GDOI assumes that the group members and GCKS are 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. A GDOI entity compromised by an attacker may reveal the secrets necessary to eavesdrop on group traffic and/or take the identity of a group sender, so host security measures mitigating unauthorized access are of the utmost importance. The latter threat could be mitigated by using source origin authentication in the Data-Security SAs (e.g., the use of RSA signatures [RFC4359] or TESLA [RFC4082]). The choice of Data-Security SAs is a matter of group policy and is not within the scope of this memo.

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

### 7.1. ISAKMP Phase 1

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, such as identity exposure, absence of unidirectional authentication, or stateful cookies [PK01].

### 7.1.1. Authentication

Authentication is provided via the mechanisms defined in [RFC2409], namely pre-shared keys or public key encryption.

### 7.1.2. 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 sent in the GROUPKEY-PULL protocol.

### 7.1.3. 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.

### 7.1.4. 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.

### 7.1.5. Denial-of-Service Protection

A DoS 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 DoS 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.

## 7.2. 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.

### 7.2.1. 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.

### 7.2.2. Confidentiality

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

### 7.2.3. 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.

### 7.2.4. Replay Protection

A GROUPKEY-PULL message identifies its messages using a cookie pair from the Phase 1 exchange that precedes it. A GROUPKEY-PULL message with invalid cookies will be discarded. Therefore, GDOI messages that are not associated with a current GDOI session will be discarded without further processing.

Replayed GDOI messages that are associated with a current GDOI session will be decrypted and authenticated. The M-ID in the HDR identifies a session. Replayed packets will be processed according to the state machine of that session. Packets not matching that state machine will be discarded without processing.

### 7.2.5. Denial-of-Service Protection

GCKS implementations SHOULD keep a record of recently received GROUPKEY-PULL messages (e.g., a hash of the packet) and reject messages that have already been processed. This provides DoS and replay protection of previously sent messages. An implementation MAY choose to rate-limit the receipt of GDOI messages in order to mitigate overloading its computational resources.

The GCKS SHOULD 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, SID-counter) until it receives the third message in the exchange with a valid HASH payload including its own nonce.

#### 7.2.6. Authorization

A GCKS implementation SHOULD maintain an authorization list of authorized group members. A group member MUST specifically list each authorized GCKS in its Group Peer Authorization Database (GPAD) [RFC5374].

### 7.3. 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 message provides an efficient rekey and group membership adjustment capability.

#### 7.3.1. Authentication

The GROUPKEY-PULL exchange distributes 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. This digital signature provides source authentication for the message. Thus, GDOI protects the GCKS from impersonation in group environments.

#### 7.3.2. Confidentiality

The GCKS encrypts the GROUPKEY-PUSH message with an encryption key that was distributed in the GROUPKEY-PULL exchange or a previous GROUPKEY-PUSH exchange. The encryption key may be a simple KEK or the result of a group management method (e.g., LKH) calculation.

#### 7.3.3. 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.

#### 7.3.4. 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 discard sequence numbers associated with the current KEK SPI that have the same or lower value as the most recently received replay number.

Implementations SHOULD keep a record (e.g., a hash value) of recently received GROUPKEY-PUSH messages and reject duplicate messages prior to performing cryptographic operations. This enables an early discard of the replayed messages.

### 7.3.5. 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 DoS 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 DoS 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 DoS 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 most recently received sequence number. Only when the sequence number is valid (i.e., it is a larger value than previously received) is the digital signature verified and the message further processed. Thus, in order for a DoS 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 DoS attack.

## 7.4. Forward and Backward Access Control

Through GROUPKEY-PUSH, the GDOI supports group management methods such as LKH (Section 5.4 of [RFC2627]) 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). The concepts "forward access control" and "backward access control" have also been described as "perfect forward security" and "perfect backward security", respectively, in the literature [RFC2627].

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

### 7.4.1. Forward Access Control Requirements

When group membership is altered using a group management algorithm, new Data-Security SAs are usually also needed. New SAs 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 Data-Security SAs MUST NOT be included in a GROUPKEY-PUSH message that changes group membership. This is required because the new Data-Security SAs are not protected with the new KEK. Instead, two sequential GROUPKEY-PUSH messages must be sent by the GCKS; the first changing the KEK, and the second (protected with the new KEK) distributing the new Data-Security SAs.

Note that in the above sequence, although the new KEK can effectively deny access to the group to some group members, they will be able to view the new KEK policy. If forward access control policy for the group includes keeping the KEK policy secret as well as the KEK itself secret, then two GROUPKEY-PUSH messages changing the KEK must occur before the new Data-Security SAs are transmitted.

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.

### 7.4.2. Backward Access Control Requirements

If backward access control is a desired property of the group, a new member MUST NOT be given Data-Security SAs that were used prior to its joining the group. This can be accomplished if the GCKS provides only the Rekey SA to the new member in a GROUPKEY-PULL exchange, followed by a GROUPKEY-PUSH message that both deletes current Data-Security SAs and provides new replacement Data-Security SAs. The new group member will effectively join the group at such time as the existing members begin sending on the Data-Security SAs.

If there is a possibility that the new group member has stored GROUPKEY-PUSH messages delivered prior to joining the group, then the above procedure is not sufficient. In this case, to achieve backward access control, the GCKS needs to return a new Rekey SA to the group member in a GROUPKEY-PULL exchange rather than the existing one. The GCKS would subsequently deliver two GROUPKEY-PUSH messages. The first, intended for existing group members, distributes the new Rekey SA to existing members. The GCKS would then deliver the second GROUPKEY-PUSH message using the new Rekey SA that both deletes current Data-Security SAs and provides new replacement Data-Security SAs. Both preexisting and new members would process the second GROUPKEY-PUSH message, and all would be able to communicate using the new Data-Security SAs.

### 7.5. Derivation of Keying Material

A GCKS distributes keying material associated with Data-Security SAs and the Rekey SA. Because these security associations are used by a set of group members, this keying material is not related to any pair-wise connection, and there is no requirement in "The Multicast Group Security Architecture" [RFC3740] for group members to permute group keying material. Because the GCKS is solely responsible for the generation of the keying material, the GCKS MUST derive the keying material using a strong random number generator. Because there are no interoperability concerns with key generation, no method is prescribed in GDOI.

## 8. IANA Considerations

### 8.1. Additions to Current Registries

The GDOI KEK Attribute named SIG_HASH_ALGORITHM [GDOI-REG] has been assigned several new Algorithm Type values from the RESERVED space to represent the SHA-256, SHA-384, and SHA-512 hash algorithms as defined in [FIPS180-3.2008]. The new algorithm names are SIG_HASH_SHA256, SIG_HASH_SHA384, and SIG_HASH_SHA512, respectively, and have the values of 3, 4, and 5, respectively.

The GDOI KEK Attribute named SIG_ALGORITHM [GDOI-REG] has been assigned several new Algorithm Type values from the RESERVED space to represent the SIG_ALG_ECDSA-256, SIG_ALG_ECDSA-384, and SIG_ALG_ECDSA-521 signature algorithms. The Algorithm Types values are 4, 5, and 6, respectively.

A new GDOI SA TEK type Protocol-ID type [GDOI-REG] has been assigned from the RESERVED space. The new algorithm ID is called GDOI_PROTO_IPSEC_AH, refers to the IPsec AH encapsulation, and has a value of 2.

A new Next Payload Type [ISAKMP-REG] has been assigned. The new type is called "SA Group Associated Policy (GAP)" and has a value of 22.

A new Key Download Type Section 5.6 has been assigned. The new type is called "SID" and has a value of 4.

## 8.2. New Registries

A new registry identifying the possible values of GAP Payload Policy Attributes (of the form described in Section 3.3 of [RFC2408]) has been created in the GDOI Payloads registry [GDOI-REG]. This memo defines the following values for this registry:

| Attribute Type | Value | Type |
|---|---|---|
| RESERVED | 0 | |
| ACTIVATION_TIME_DELAY | 1 | B |
| DEACTIVATION_TIME_DELAY | 2 | B |
| SENDER_ID_REQUEST | 3 | B |
| Unassigned | 4-127 | |
| Private Use | 128-255 | |
| Unassigned | 256-32767 | |

The registration procedure is Standards Action. The terms Standards Action and Private Use are to be applied as defined in [RFC5226].

A new IPsec Security Association Attribute [ISAKMP-REG] defining the preservation of IP addresses has been registered. The attribute class is called "Address Preservation", and it is a Basic type. The following rules apply to define the values of the attribute:

| Name | Value |
|---|---|
| Reserved | 0 |
| None | 1 |
| Source-Only | 2 |
| Destination-Only | 3 |
| Source-and-Destination | 4 |
| Unassigned | 5-61439 |
| Private Use | 61440-65535 |

The registration procedure is Standards Action. The terms Standards Action and Private Use are to be applied as defined in [RFC5226].

A new IPsec Security Association Attribute [ISAKMP-REG] defining the SA direction has been created. The attribute class is called "SA Direction", and it is a Basic type. The following rules apply to define the values of the attribute:

| Name | Value |
|---|---|
| Reserved | 0 |
| Sender-Only | 1 |
| Receiver-Only | 2 |
| Symmetric | 3 |
| Unassigned | 4-61439 |
| Private Use | 61440-65535 |

The registration procedure is Standards Action. terms Standards Action and Private Use are to be applied as defined in [RFC5226].

When the SID "Key Download Type" (described in the previous section) has a set of attributes, the attributes must follow the format defined in ISAKMP (Section 3.3 of [RFC2408]). In the table, attributes defined as TV are marked as Basic (B); attributes defined as TLV are marked as Variable (V).

| SID Class | Value | Type |
|---|---|---|
| RESERVED | 0 | |
| NUMBER_OF_SID_BITS | 1 | B |
| SID_VALUE | 2 | V |
| Unassigned | 3-128 | |
| Private Use | 129-255 | |
| Unassigned | 256-32767 | |

The registration procedure is Standards Action. terms Standards Action and Private Use are to be applied as defined in [RFC5226].

### 8.3. Cleanup of Existing Registries

Several existing GDOI Payloads registries do not use the terms in RFC 5226 and/or do not describe the entire range of possible values. The following sections correct these registries. The terms Standards Action, Unassigned, and Private Use are to be applied as defined in [RFC5226].

#### 8.3.1. Pop Algorithm

The registration procedure is Standards Action. Values 4-27 are designated Unassigned. Values 256-32767 have been added and are designated Unassigned.

### 8.3.2. KEK Attributes

The registration procedure is Standards Action. Values 9-127 have been added and are designated Unassigned. Values 128-255 have been added and are designated Private Use. Values 256-32767 have been added and are designated Unassigned.

### 8.3.3. KEK_MANAGEMENT_ALGORITHM

The registration procedure is Standards Action. Values 2-127 are designated Unassigned. Values 128-255 have been added and designated Private Use. Values 256-65535 have been added and are designated Unassigned.

### 8.3.4. KEK_ALGORITHM

The registration procedure is Standards Action. Values 4-127 are designated Unassigned. Values 256-65535 have been added and are designated Unassigned.

### 8.3.5. SIG_HASH_ALGORITHM

The registration procedure is Standards Action. Values 6-127 are designated Unassigned. Values 256-65535 have been added and are designated Unassigned.

### 8.3.6. SIG_ALGORITHM

The registration procedure is Standards Action. Values 7-127 are designated Unassigned. Values 256-65535 have been added and are designated Unassigned.

### 8.3.7. SA TEK Payload Values

The registration procedure is Standards Action. Values 3-127 are designated Unassigned.

### 8.3.8. Key Download Types

The registration procedure is Standards Action. Values 5-127 are designated Unassigned.

### 8.3.9. TEK Download Type

The registration procedure is Standards Action. Values 4-127 have been added and are designated Unassigned. Values 128-255 have been added and are designated Private Use. Values 256-32767 have been added and are designated Unassigned.

### 8.3.10. KEK Download Type

The registration procedure is Standards Action. Values 3-127 are designated Unassigned. Values 128-255 have been added and are designated Private Use. Values 256-32767 have been added and are designated Unassigned.

### 8.3.11. LKH Download Type

The registration procedure is Standards Action. Values 4-127 are designated Unassigned. Values 256-32767 have been added and are designated Unassigned.

## 9. Acknowledgements

This memo replaces RFC 3547, and the authors wish to thank Mark Baugher and Hugh Harney for their extensive contributions that led to this newer specification of GDOI.

The authors are grateful to Catherine Meadows for her careful review and suggestions for mitigating the man-in-the-middle attack she had previously identified. Yoav Nir, Vincent Roca, Sean Turner, and Elwyn Davies provided many useful technical and editorial comments and suggestions for improvement.

## 10. References

### 10.1. Normative References

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

[RFC2403] Madson, C. and R. Glenn, "The Use of HMAC-MD5-96 within ESP and AH", RFC 2403, November 1998.

[RFC2404] Madson, C. and R. Glenn, "The Use of HMAC-SHA-1-96 within ESP and AH", RFC 2404, November 1998.

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

[RFC2408] Maughan, D., Schneider, M., and M. Schertler, "Internet Security Association and Key Management Protocol (ISAKMP)", RFC 2408, November 1998.

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

[RFC2627] Wallner, D., Harder, E., and R. Agee, "Key Management for Multicast: Issues and Architectures", RFC 2627, June 1999.

[RFC3447] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, February 2003.

[RFC4302] Kent, S., "IP Authentication Header", RFC 4302, December 2005.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.

[RFC4754] Fu, D. and J. Solinas, "IKE and IKEv2 Authentication Using the Elliptic Curve Digital Signature Algorithm (ECDSA)", RFC 4754, January 2007.

[RFC4868] Kelly, S. and S. Frankel, "Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec", RFC 4868, May 2007.

[RFC5374] Weis, B., Gross, G., and D. Ignjatic, "Multicast Extensions to the Security Architecture for the Internet Protocol", RFC 5374, November 2008.

[RFC5903] Fu, D. and J. Solinas, "Elliptic Curve Groups modulo a Prime (ECP Groups) for IKE and IKEv2", RFC 5903, June 2010.

[RFC6054] McGrew, D. and B. Weis, "Using Counter Modes with Encapsulating Security Payload (ESP) and Authentication Header (AH) to Protect Group Traffic", RFC 6054, November 2010.

## 10.2. Informative References

[FIPS180-3.2008]
National Institute of Standards and Technology, "Secure Hash Standard", FIPS PUB 180-3, October 2008, <http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf>.

[FIPS186-3] "Digital Signature Standard (DSS)", United States of America, National Institute of Science and Technology, Federal Information Processing Standard (FIPS) 186-2, June 2009.

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

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

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

[GDOI-REG] Internet Assigned Numbers Authority, "Group Domain of Interpretation (GDOI) Payload Type Values", IANA Registry, December 2004, <http://www.iana.org/assignments/gdoi-payloads>.

[HD03] Hardjono, T. and L. Dondeti, "Multicast and Group Security", Artech House Computer Security Series, ISBN 1-58053-342-6, 2003.

[ISAKMP-REG] "'Magic Numbers' for ISAKMP Protocol", <http://www.iana.org/assignments/isakmp-registry>.

[MP04] Meadows, C. and D. Pavlovic, "Deriving, Attacking, and Defending the GDOI Protocol", European Symposium on Research in Computer Security (ESORICS) 2004, pp. 53-72, September 2004.

[NNL] Naor, D., Noal, M., and J. Lotspiech, "Revocation and Tracing Schemes for Stateless Receivers", Advances in Cryptology, Crypto '01, Springer-Verlag LNCS 2139, 2001, pp. 41-62, 2001, <http://www.iacr.org/archive/crypto2001/21390040.pdf>.

[OFT] Sherman, A. and D. McGrew, "Key Establishment in Large Dynamic Groups Using One-Way Function Trees", IEEE Transactions on Software Engineering, Vol. 29, Issue 5, pp. 444-458, May 2003, <http://ieeexplore.ieee.org/search/freesrchabstract.jsp?tp=&arnumber=1199073>.

[PK01] Perlman, R. and C. Kaufman, "Analysis of the IPsec Key Exchange Standard", Enabling Technologies: Infrastructure for Collaborative Enterprises, WET ICE 2001, Proceedings. Tenth IEEE International Workshops on IEEE Transactions on Software Engineering, pp. 150-156, June 2001, <http://ieeexplore.ieee.org/search/freesrchabstract.jsp?tp=&arnumber=953405>.

[PROT-REG] "Assigned Internet Protocol Numbers", <http://www.iana.org/assignments/protocol-numbers/>.

[RFC3686] Housley, R., "Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP)", RFC 3686, January 2004.

[RFC3740] Hardjono, T. and B. Weis, "The Multicast Group Security Architecture", RFC 3740, March 2004.

[RFC3947] Kivinen, T., Swander, B., Huttunen, A., and V. Volpe, "Negotiation of NAT-Traversal in the IKE", RFC 3947, January 2005.

[RFC4046] Baugher, M., Canetti, R., Dondeti, L., and F. Lindholm, "Multicast Security (MSEC) Group Key Management Architecture", RFC 4046, April 2005.

[RFC4082] Perrig, A., Song, D., Canetti, R., Tygar, J., and B. Briscoe, "Timed Efficient Stream Loss-Tolerant Authentication (TESLA): Multicast Source Authentication Transform Introduction", RFC 4082, June 2005.

[RFC4106] Viega, J. and D. McGrew, "The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)", RFC 4106, June 2005.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[RFC4306] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, December 2005.

[RFC4309] Housley, R., "Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP)", RFC 4309, December 2005.

[RFC4359] Weis, B., "The Use of RSA/SHA-1 Signatures within Encapsulating Security Payload (ESP) and Authentication Header (AH)", RFC 4359, January 2006.

[RFC4543] McGrew, D. and J. Viega, "The Use of Galois Message Authentication Code (GMAC) in IPsec ESP and AH", RFC 4543, May 2006.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5905] Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.

[RFC5996] Kaufman, C., Hoffman, P., Nir, Y., and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5996, September 2010.

[SP.800-131] Barker, E. and A. Roginsky, "Recommendation for the Transitioning of Cryptographic Algorithms and Key Lengths", United States of America, National Institute of Science and Technology, DRAFT NIST Special Publication 800-131, June 2010.

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

## Appendix A. GDOI Applications

GDOI can be used to distribute keys for several secure multicast applications, where different applications have different key management requirements. This section outlines two examples of ways that GDOI can be used. Other examples can be found in Section 10 of [HD03].

A simple application is secure delivery of periodic multicast content over an organization's IP network, perhaps a multicast video broadcast. Assuming the content delivery time frame is bounded and the group membership is not expected to change over time, there is no need for group policy to include a GROUPKEY-PUSH exchange, and there is no need for the GCKS to distribute a Rekey SA. Thus, the GDOI GCKS may only need to distribute a single set of Data-Security SAs to protect the time-bounded broadcast.

In contrast, a persistent IP multicast application (e.g., stock-ticker delivery service) may have many group members, where the group membership changes over time. A periodic change of Data-Security SAs may be desirable, and the potential for change in group membership requires the use of a group management method enabling de-authorization of group members. The GDOI GCKS will distribute the current set of Data-Security SAs and a Rekey SA to registering group members. It will then use regularly scheduled GROUPKEY-PUSH exchanges to deliver the new SAs for the group. Additionally, the group membership on the GCKS may be frequently adjusted, which will result in a GROUPKEY-PUSH exchange that delivers new Rekey SAs protected by a group management method. Each GROUPKEY-PUSH may include Data-Security SAs and/or a Rekey SA.

In each example, the relevant policy is defined on the GCKS and relayed to group members using the GROUPKEY-PULL and/or GROUPKEY-PUSH protocols. Specific policy choices configured by the GCKS administrator depend on each application.

## Appendix B. Significant Changes from RFC 3547

The following significant changes have been made from RFC 3547.

o The Proof of Possession (POP) payload was removed from the GROUPKEY-PULL exchange. It provided an alternate form of authorization, but its use was underspecified. Furthermore, Meadows and Pavlovic [MP04] discussed a man-in-the-middle attack on the POP authorization method, which would require changes to its semantics. No known implementation of RFC 3547 supported the

POP payload, so it was removed. Removal of the POP payload obviated the need for the CERT payload in that exchange, and it was removed as well.

- The Key Exchange payloads (KE_I, KE_R) were removed from the GROUPKEY-PULL exchange. However, the specification for computing keying material for the additional encryption function in RFC 3547 is faulty. Furthermore, it has been observed that because the GDOI registration message uses strong ciphers and provides authenticated encryption, additional encryption of the keying material in a GDOI registration message provides negligible value. Therefore, the use of KE payloads is deprecated in this memo.

- The Certificate Payload (CERT) was removed from the GROUPKEY-PUSH exchange. The use of this payload was underspecified. In all known use cases, the public key used to verify the GROUPKEY-PUSH payload is distributed directly from the key server as part of the GROUPKEY-PULL exchange.

- Supported cryptographic algorithms were changed to meet current guidance. Implementations are required to support AES with 128-bit keys to encrypt the rekey message and support SHA-256 for cryptographic signatures. The use of DES is deprecated.

- New protocol support for AH.

- New protocol definitions were added to conform to the most recent "Security Architecture for the Internet Protocol" [RFC4301] and the "Multicast Extensions to the Security Architecture for the Internet Protocol" [RFC5374]. This includes addition of the GAP payload.

- New protocol definitions and semantics were added to support "Using Counter Modes with Encapsulating Security Payload (ESP) and Authentication Header (AH) to Protect Group Traffic" [RFC6054].

- Specification to IANA was added to better clarify the use of the GDOI Payloads registry.

## Authors' Addresses

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

Phone: +1-408-526-4796
EMail: bew@cisco.com

Sheela Rowles
Cisco Systems
170 W. Tasman Drive
San Jose, California  95134-1706
USA

Phone: +1-408-527-7677
EMail: sheela@cisco.com

Thomas Hardjono
MIT
77 Massachusetts Ave.
Cambridge, Massachusetts  02139
USA

Phone: +1-781-729-9559
EMail: hardjono@mit.edu