GSAKMP

Status of this memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress".

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Abstract

This document specifies the Group Secure Association Key Management Protocol (GSAKMP). The GSAKMP provides a security framework for creating and managing cryptographic groups on a network. It provides mechanisms to disseminate group policy and authenticate users, rules to perform access control decisions during group establishment and recovery, capabilities to recover from the compromise of group members, delegation of group security functions, and capabilities to destroy the group. It also generates group keys.

Copyright Notice

Copyright (c) The Internet Society (2003). All Rights Reserved.
List of Figures

1  GSAKMP Ladder Diagram ........................................... 14
2  GSAKMP Ladder Diagram with Cookies ............................... 18
3  GSAKMP Header Format .............................................. 22
4  Generic Payload Header ............................................. 24
5  Policy Token Payload Format ....................................... 25
6  Key Download Payload Format ...................................... 26
7  Rekey Event Payload Format ....................................... 27
8  Identification Payload Format ..................................... 28
9  Certificate Payload Format ........................................ 29
10 Signature Payload Format .......................................... 31
11 Notification Payload Format ....................................... 32
12 Notification Data - Acknowledge Payload Type Format ............ 34
13 Key Creation Payload Format ...................................... 35
14 Nonce Payload Format ............................................. 36
15 GSAKMP State Diagram ............................................ 38
16 B. 1: Rekey Event Header Format ................................ 44
17 B. 2: Rekey Event Packet Data Format ............................ 45
18 B. 3: Key Pack Data Format ....................................... 45
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Request to Join Message Definition</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>Key Download Message Definition</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>Notification Message Definition</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>Cookie Download Message Definition</td>
<td>17</td>
</tr>
<tr>
<td>5</td>
<td>Rekey Event Message Definition</td>
<td>20</td>
</tr>
<tr>
<td>6</td>
<td>Group Identification Types</td>
<td>22</td>
</tr>
<tr>
<td>7</td>
<td>Payload Types</td>
<td>23</td>
</tr>
<tr>
<td>8</td>
<td>Exchange Types</td>
<td>23</td>
</tr>
<tr>
<td>9</td>
<td>Policy Token Types</td>
<td>25</td>
</tr>
<tr>
<td>10</td>
<td>Key Download Data Types</td>
<td>26</td>
</tr>
<tr>
<td>11</td>
<td>Rekey Event Types</td>
<td>28</td>
</tr>
<tr>
<td>12</td>
<td>Identification Types</td>
<td>29</td>
</tr>
<tr>
<td>13</td>
<td>Certificate Payload Types</td>
<td>30</td>
</tr>
<tr>
<td>14</td>
<td>Signature Types</td>
<td>31</td>
</tr>
<tr>
<td>15</td>
<td>Authorization Types</td>
<td>32</td>
</tr>
<tr>
<td>16</td>
<td>Notify Payload Types</td>
<td>33</td>
</tr>
<tr>
<td>17</td>
<td>Notify Payload -- Status Types</td>
<td>34</td>
</tr>
<tr>
<td>18</td>
<td>Acknowledgement Types</td>
<td>35</td>
</tr>
<tr>
<td>19</td>
<td>Types Of Key Creation Information</td>
<td>36</td>
</tr>
<tr>
<td>20</td>
<td>Nonce Types</td>
<td>37</td>
</tr>
<tr>
<td>21</td>
<td>GSAKMP States</td>
<td>39</td>
</tr>
<tr>
<td>22</td>
<td>State Transition Events</td>
<td>40</td>
</tr>
</tbody>
</table>
1 Overview

1.1 GSAKMP Overview

Protecting group information requires the definition of a security policy and the enforcement of that policy by all participating parties. Controlling dissemination of cryptographic key is the primary mechanism to enforce the access control policy. It is the primary purpose of GSAKMP to generate and disseminate a group key in a secure fashion.

GSAKMP separates group security management functions and responsibilities into three major roles: 1) Group Owner, 2) Group Controller Key Server, and 3) Group Member. The Group Owner is responsible for creating the security policy rules for a group and expressing these in the Policy Token. The Group Controller Key Server (GCKS) is responsible for creating and maintaining the keys and enforcing the group policy by granting access to potential Group Members (GM) in accordance with the Policy Token. To enforce a group’s policy the potential Group Members need to have knowledge of the access control policy for the group, an unambiguous identification of any party downloading keys to them, and verifiable chains of authority for key download. In other words, the Group Members need to know who potentially will be in the group and to verify that the key disseminator is authorized to act in that capacity.

In order to establish a Group Secure Association (GSA) to support these activities, the identity of each party in the process MUST be unambiguously asserted and authenticated. It MUST also be verified that each party is authorized, as defined by the Policy Token, to function in his role in the protocol (e.g., GM or GCKS).

The security features of the establishment protocol for the SA include

- Group policy identification
- Group policy dissemination
- GM to GCKS SA establishment to protect data
- Access control checking

GSAKMP provides mechanisms for cryptographic group creation and management. Other protocols may be used in conjunction with GSAKMP to allow various applications to create functional groups according to their application-specific requirements. For example, in a small-scale video conference the organizer might use a session invitation protocol like SIP [RFC2543] to transmit information about the time of the conference, the address of the session, and the formats to be used. For a large-scale video
transmission, the organizer might use a multicast announcement protocol like SAP [RFC2974].

This document describes a useful default set of security algorithms and configurations, Security Suite 1. This suite allows an entire set of algorithms and settings to be described to prospective group members in a concise manner. Other security suites MAY be defined as needed and MAY be disseminated during the out-of-band announcement of a group.

Distributed architectures support large scale cryptographic groups. Secure distributed architectures require authorized delegation of GSA actions to network resources. The fully specified Policy Token is the mechanism to facilitate this authorization. Trasmission of this Policy Token to all joining GMs allows GSAKMP to securely support distributed architectures and multiple data sources.

Many-to-many group communications require multiple data sources. Multiple data sources are supported because the inclusion of a policy token and policy payloads allow group members to review the group access control and authorization parameters. This member review process gives each member (each potential source of data), the ability to determine if the group provides adequate protection for member data.

1.2 Protocol Considerations

IANA has provided GSAKMP port number 3761 in both the UDP and TCP spaces. All implementations MUST use this port assignment in the appropriate manner.

1.3 Document Organization

The remainder of this document is organized as follows: Section 2 presents the terminology and concepts used to present the requirements of this protocol. Section 3 outlines the security considerations with respect to GSAKMP. Section 4 describes the group management life-cycle. Section 5 describes the Security Suite Definition. Section 6 presents the message types and formats used during each phase of the life-cycle. Section 7 defines the state diagram for the protocol.

2 Terminology

The following terminology is used throughout the GSAKMP paper.

Certificate: A data structure used to verifiably bind an identity to a
cryptographic key (e.g., X.509v3).

Compromise Recovery: The act of recovering a secure operating state after detecting that a group member cannot be trusted. This can be accomplished by rekey.

Cryptographic Group: A set of entities sharing or desiring to share a GSA.

Group Controller Key Server (GCKS): A group member with authority to perform critical protocol actions including creating and distributing keys and building and maintaining the rekey structures. As the group evolves, it MAY become desirable to have multiple controllers perform these functions.

Group Member (GM): A Group Member is any entity with access to the group keys. Regardless of how a member becomes a part of the group or how the group is structured, GMs will perform the following actions:

- Authenticate and validate the identities and the authorizations of entities performing security relevant actions
- Accept group keys from the GCKS
- Request group keys from the GCKS
- Enforce the cooperative group policies as stated in the group policy token
- Perform peer review of key management actions
- Manage local key

Group Owner (GO): A Group Owner is the entity authorized for generating and modifying an authenticatable policy token for the group, and notifying the GCKS to start the group.

Group Policy: The Group Policy completely describes the protection mechanisms and security relevant behaviors of the group. This policy MUST be commonly understood and enforced by the group for coherent secure operations.

Group Secure Association (GSA): A GSA is a logical association of users or hosts that share cryptographic key(s). This group may be established to support associations between applications or communication protocols.

Group Traffic Encryption Key (GTEK): The key or keys created for encrypting the group data.
Logical Key Hierarchy (LKH) Array: The group of keys created to facilitate the LKH compromise recovery methodology.

Policy Token: The policy token is a data structure used to disseminate group policy and the mechanisms to enforce it. The policy token is issued and signed by an authorized source. Each member of the group MUST verify the token, meet the group join policy, and enforce the policy of the group, (e.g., encrypt application data with a specific algorithm). The group policy token will contain a variety of information including:

- GSAKMP protocol version
- Key creation method
- Key dissemination policy
- Access control policy
- Group authorization policy
- Compromise recovery policy
- Data protection mechanisms

An example of a policy token is specified in [HCLM00].

Rekey: The act of changing keys within a group.

Subordinate Group Controller Key Server (SGCKS): Any group member having the appropriate processing and trust characteristics as defined in the group policy that has the potential to act as a SGCKS. This will allow the group processing and communication requirements to be distributed equitably throughout the network (e.g., distribute group key). The optional use of GSAKMP with Subordinate Group Controller Key Servers will be documented in a separate paper.

3 Security Considerations

In addition to the specification of GSAKMP itself, the security of an implemented GSAKMP system is affected by supporting factors. These are discussed here.
3.1 Security Assumptions

The following assumptions are made as the basis for the security discussion:

1. GSAKMP assumes its supporting platform can provide the process and data separation services at the appropriate assurance level to support its groups.
2. The key generation function of the cryptographic engine will only generate strong keys.
3. The security of this protocol is critically dependent on the randomness of the randomly chosen parameters. These should be generated by a strong random or properly seeded pseudo-random source.

3.2 GSAKMP Use of Other Protocols

GSAKMP is based upon two (2) existing protocols: ISAKMP [MSST98] and FIPS Pub 196 [FIPS 196]. GSAKMP MAY use Diffie-Hellman key exchange [DH77] for two party key creation and MAY use LKH for rekey capability.

3.2.1 ISAKMP

ISAKMP provides a flexible structure of chained payloads in support of authenticated key exchange and security association management for pairwise communications. GSAKMP expands upon these features to provide policy enforcement features in support of diverse group communications.

3.2.2 FIPS Pub 196

FIPS Pub 196 provides a mutual authentication protocol.

3.2.3 LKH

GSAKMP relies upon a rekey capability, i.e., LKH, to enable group recovery after a compromise.
3.2.4 Diffie-Hellman

GSAKMP MAY rely upon two party key creation mechanisms, i.e., Diffie-Hellman, to protect sensitive data during download.

The information in this section is borrowed heavily from [IKEv2] as this protocol has already worked through this issue and GSAKMP is using the same security considerations for its purposes. This section will contain paraphrased sections of [IKEv2] modified for GSAKMP as appropriate.

The strength of a key derived from a Diffie-Hellman exchange using specific p and g values depends on the inherent strength of the values, the size of the exponent used, and the entropy provided by the random number generator used. Security Suite 1 defined in section 5, based on [IKEv2] Group 2, with a strong random number generator and an exponent no less than 200 bits is sufficient to use for 3DES. An implementation should make note of this conservative estimate when establishing policy and negotiating security parameters.

Note that these limitations are on the Diffie-Hellman values themselves. There is nothing in GSAKMP which prohibits using stronger values nor is there anything which will dilute the strength obtained from stronger values. In fact, the extensible framework of GSAKMP encourages the definition of more Security Suites.

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

3.3 Denial of Service (DoS) Attack

This GSAKMP specification addresses the mitigation for a distributed IP spoofing attack (a subset of possible DoS attacks) in section 4.2.2, Cookies.

3.4 Proof of Trust Hierarchy

As defined by [HCM], security group policy MUST be defined in a verifiable manner. GSAKMP anchors its trust in the creator of the group, the GO.

The Policy Token explicitly defines all the parameters that create a secure verifiable infrastructure. The GSAKMP Policy Token is issued and signed by the GO. The GCKS will verify it and grant access to GMs only if they meet the rules of the Policy Token. The new GMs will accept access only if 1) the token verifies, 2) the GCKS is an authorized disseminator, and 3) the
group mechanisms are acceptable for protecting the GM's data.

4 Group Life Cycle

The management of a cryptographic group follows a life-cycle: group definition, group establishment, and security relevant group maintenance. Group definition involves defining the parameters necessary to support a secure group, including its policy token. Group establishment is the process of granting access to new members. Security relevant group maintenance messages include rekey, policy changes and member deletion. Each of these life-cycle phases is discussed in the following sections.

4.1 Group Definition

A cryptographic group is established to support secure communications among a group of individuals. The activities necessary to create a Policy Token in support of a cryptographic group include

- Determine Access Policy - identify the entities that are authorized to receive the group key.
- Determine Authorization Policy - identify which entities are authorized to perform security relevant actions, including key dissemination, policy creation, and initiation of security management actions.
- Determine Mechanisms - define the algorithms and protocols used by GSAKMP to secure the group.
- Create Group Policy Token - format the policies and mechanisms into a Policy Token and apply the GO signature.

4.2 Group Establishment

4.2.1 Standard Group Establishment

After the out-of-band receipt of a Policy Token, a potential Group Controller Key Server (GCKS) verifies the token and its rules. This entity then establishes that eligibility rules for GCKS creation have been met. The GCKS is then permitted to create any needed group keys and begin to establish the group.

The GSAKMP Ladder Diagram, Figure 1, is presented to illustrate the process...
of establishing a cryptographic group. The left side of the diagram represents the actions of the GCKS. The right side of the diagram represents the actions of the GMs. The components of each message shown in the diagram are presented in sections 4.2.1.1 - 4.2.1.3.

<table>
<thead>
<tr>
<th>CONTROLLER</th>
<th>MESSAGE</th>
<th>MEMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Process RTJ&gt;</td>
<td>!&lt;----------Request to Join-----------&gt;!</td>
<td></td>
</tr>
<tr>
<td></td>
<td>!&lt;---------Key Download---------------&gt;!</td>
<td>!&lt;Process KD&gt;</td>
</tr>
<tr>
<td></td>
<td>!&lt;-----Notification - Ack/Failure--------!</td>
<td></td>
</tr>
<tr>
<td>&lt;Process Notif&gt;</td>
<td>!&lt;========SHARED KEYED GROUP SESSION======&gt;!</td>
<td></td>
</tr>
</tbody>
</table>

Figure 1: GSAKMP Ladder Diagram

To facilitate a well ordered group creation, the GCKS MUST send security policy information to the GM using a group policy token. The group policy token MUST include the group’s identification, group permissions, group join policy, group controller key server identity, group management information, and digital signature of the policy creation authority. This will allow the GM to determine whether group policy is compatible with local policy.

Standard design principles for secure protocols mandate the use of explicit identification of senders and recipients of messages. The signature payload of each message identifies the signer of the message and therefore satisfies the sender requirement. Within the GSAKMP header is a group ID. Because the member may be served by any Key Server within a group, this ID provides sufficient granularity for the recipient ID for the Request to Join message. Other messages sent by the potential member will contain the recipient ID for the GCKS serving that member.

If the GCKS is unable to correctly process the Notification message, the GCKS MUST remove this GM from the group and handle according to policy.

For the following message structure sections, details about payload formats can be found in Section 6.

4.2.1.1 Request to Join

The components of a Request to Join Message are shown in Table 1.

As shown by Figure 1, potential GMs join a group by initiating a request for permission to join. In response, the GCKS accepts or denies the request based on local configuration. <Process RTJ> indicates the GCKS actions...
Table 1: Request to Join Message Definition

<table>
<thead>
<tr>
<th>Message Name</th>
<th>Dissection</th>
<th>Payload Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request to Join (RTJ)</td>
<td>{HDR-GrpID, Key Creation, Nonce_I, [Notification]}</td>
<td>GSAKMP Header, Key Creation, Nonce, Signature, [Certificate], [Notification]</td>
</tr>
<tr>
<td>SigM</td>
<td>: Signature of Group Member</td>
<td></td>
</tr>
<tr>
<td>Cert</td>
<td>: Necessary Certificates, zero or more</td>
<td></td>
</tr>
<tr>
<td>{}SigX</td>
<td>: Indicates minimum fields used in Signature</td>
<td></td>
</tr>
<tr>
<td>[]</td>
<td>: Indicate an optional data item</td>
<td></td>
</tr>
</tbody>
</table>

that will determine if the RTJ will be acted upon. Reasons for refusal of action can include: malformed RTJ, unverifiable RTJ, GCKS processing issues, and state violations. NOTE: At any one time, a GCKS MUST process no more than one (1) valid RTJ from a single GM per group. If the results of the Request to Join message, the GCKS processes the Request to Join. In this procedure, the GCKS will verify the signature on the message to ensure its authenticity. The GCKS MUST used verified and trusted authentication material from a known root. If the message signature does not verify, the session MUST be terminated. If the message signature verifies, then the identity of the sender is extracted from the Signature Payload. This identity information will be used the Key Download message.

The GCKS then confirms that all required payloads are present and properly formatted based upon the mechanisms announced with the group characteristics. If not, the session MUST be terminated.

If the message signature is verified, and the GM passes the GCKS’s access control checks, the GCKS will create and send the Key Download message as described in section 4.2.1.2.

The OPTIONAL Notification payload will be used when the GCKS requires cookies in the Request to Join message. The use of this payload will be defined in section 4.2.2.

4.2.1.2 Key Download

The components of a Key Download Message are shown in Table 2:

The GM receives this message and verifies the following information:

signature on the message to ensure its authenticity, the contents of the nonce responder and combined payloads, the contents of the key creation payload, and verify the identity information. If the message signature,
Table 2: Key Download Message Definition

<table>
<thead>
<tr>
<th>Message Name</th>
<th>Key Download (KD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dissection</td>
<td>{HDR-GrpID, Member ID, Nonce_R, Nonce_C, Key Creation, (Policy Token)<em>, (Key Download)</em>} SigC, [Cert]</td>
</tr>
<tr>
<td>Payload Types</td>
<td>GSAKMP Header, Identification, Nonce, Key Creation, Policy Token, Key Download, Signature, [Certificate]</td>
</tr>
</tbody>
</table>

| SigC               | Signature of Group Controller Key Server                                          |
| Cert               | Necessary Certificates, zero or more                                               |
| {}SigX             | Indicates minimum fields used in Signature                                           |
| []                 | Indicate an optional data item                                                     |
| (data)*            | Indicates encrypted information                                                   |

nonce, or identification information does not verify, the session MUST be terminated. GSAKMP sends a properly authenticated message with a Notification Payload of type NACK to indicate termination.

The Policy Token and Key Download payloads are sent encrypted in the KEK generated by the Key Creation payload information using the mechanisms defined in the group announcement. This guarantees that the sensitive policy and key data for the group and potential rekey data for this individual cannot be read by anyone but the intended recipient.

4.2.1.3 Notification

The components of a Notification Message are shown in Table 3:

Table 3: Notification Message Definition

<table>
<thead>
<tr>
<th>Message Name</th>
<th>Notification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dissection</td>
<td>{HDR-GrpID, Nonce_C, ACK}SigM</td>
</tr>
<tr>
<td>Payload Types</td>
<td>GSAKMP Header, Nonce, Notification, Signature</td>
</tr>
<tr>
<td>SigM</td>
<td>Signature of Group Member</td>
</tr>
<tr>
<td>{}SigX</td>
<td>Indicates minimum fields used in Signature</td>
</tr>
</tbody>
</table>

The GCKS receives the signed notification and will verify the signature on the message to ensure its authenticity and the nonce value. If the message signature or nonce does not verify, the session MUST be terminated.

If the message signature and nonce are verified, then the GCKS and GM have established a GSA.
4.2.2 Cookies - Group Establishment with Denial of Service Protection

This section defines an OPTIONAL capability that MAY be implemented into GSAKMP. The information in this section is borrowed heavily from [IKEv2] as this protocol has already worked through this issue and GSAKMP is copying this concept. This section will contain paraphrased sections of [IKEv2] modified for GSAKMP to define the purpose of Cookies.

An optional Cookie mode is being defined for the GSAKMP to help against DoS attacks.

The term "cookies" originates with Karn and Simpson [RFC 2522] in Photuris, an early proposal for key management with IPsec. The ISAKMP fixed message header includes two eight octet fields titled "cookies". Instead of placing this cookie data in the header, this data is moved into a Notification payload.

An expected attack against GSAKMP is state and CPU exhaustion, where the target GCKS is flooded with Request to Join requests from forged IP addresses. This attack can be made less effective if a GCKS implementation uses minimal CPU and commits no state to the communication until it knows the initiator potential GM can receive packets at the address from which it claims to be sending them. To accomplish this, the GCKS when operating in Cookie mode, SHOULD reject initial Request to Join messages unless they contain a Notification payload of type "cookie". It SHOULD instead send a Cookie Download message as a response to the RTJ and include a cookie in a notify payload of type Cookie_Requested. Potential GMs who receive such responses MUST retry the Request to Join message with the responder GCKS supplied cookie in its notification payload of type Cookie, as defined by the optional Notification payload of the Request to Join Msg as defined in section 4.2.1.1. This initial exchange will then be as shown in Figure 2 with the components of the new message Cookie Download shown in Table 4.

Table 4: Cookie Download Message Definition

<table>
<thead>
<tr>
<th>Message Name</th>
<th>Cookie Download</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dissection</td>
<td>{HDR-GrpID, COOKIE_REQUIRED}</td>
</tr>
<tr>
<td>Payload Types</td>
<td>GSAKMP Header, Notification, Signature</td>
</tr>
</tbody>
</table>

The first two messages do not affect any GM or GCKS state except for communicating the cookie.

A GSAKMP implementation SHOULD implement its GCKS cookie generation in such a way as to not require any saved state to recognize its valid cookie when the second Request to Join message arrives. The exact algorithms and syntax they use to generate cookies does not affect interoperability and hence is not specified here. The following is an example of how an endpoint could use cookies to implement limited DoS protection.
A good way to do this is to set the cookie to be:

Cookie = <SecretVersionNumber> ! Hash(Ni ! IPi ! <secret>)

where <secret> is a randomly generated secret known only to the responder GCKS and periodically changed, Ni is the Nonce value taken from the initiator potential GM, IPi is the supposed IP of the initiator potential GM. <SecretVersionNumber> should be changed whenever <secret> is regenerated. The cookie can be recomputed when the "Request to Join with Cookie Info" arrives and compared to the cookie in the received message. If it matches, the responder GCKS knows that all values have been computed since the last change to <secret> and that IPi MUST be the same as the source address it saw the first time. Incorporating Ni into the hash assures that an attacker who sees only the Cookie_Download message can’t successfully forge a "Request to Join with Cookie Info" message.

If a new value for <secret> is chosen while there are connections in the process of being initialized, a "Request to Join with Cookie Info" might be returned with other than the current <SecretVersionNumber>. The responder GCKS in that case MAY reject the message by sending another response with a new cookie or it MAY keep the old value of <secret> around for a short time and accept cookies computed from either one. The responder GCKS SHOULD NOT accept cookies indefinitely after <secret> is changed, since that would defeat part of the denial of service protection. The responder GCKS SHOULD change the value of <secret> frequently, especially if under attack.
4.3 Group Maintenance

The Group Maintenance phase includes member joins and leaves, group rekey activities, and the management of Rekey events. These activities are presented in the following sections.

4.3.1 Member Joins/Leaves

The addition of group members to a previously established group will closely follow the processing presented in Sections 4.2.1. With the exception of the pure group establishment tasks (e.g., creation of policy token, GTEK, and Rekey array), an entity becomes a GM using the same message exchanges described in Sections 4.2.1.1 through 4.2.1.3.

A member who elects to voluntarily leave the group MUST destroy local copies of the group key.

4.3.2 Rekey Events

A Rekey Event is any action, including compromise report or key expiration, that requires the creation and dissemination of a new group key and/or Rekey information.

Once an event has been identified (as defined in the group security policy token), the GCKS MUST create and send a signed message containing the GTEK and Rekey information to the group.

Each GM who receives this message MUST verify the signature on the message to ensure its authenticity. If the message signature does not verify, the message MUST be discarded. Upon verification the GM will find the appropriate Rekey download packet and decrypt the information with a stored Rekey key(s). If a new Policy Token is distributed with the message, it MUST be encrypted in the old GTEK.

The components of a Rekey Event message are shown in Table 5:

5 Security Suite

The Security Definition Suite 1 MUST be supported. Other security suite definitions MAY be defined in other internet specifications.
Table 5:  Rekey Event Message Definition

<table>
<thead>
<tr>
<th>Message Name</th>
<th>Rekey Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dissection</td>
<td>{HDR-GrpID, ([Policy Token])*, Rekey Array</td>
</tr>
<tr>
<td>Payload Types</td>
<td>GSAKMP Header, [Policy Token], Rekey Event, Signature, [Certificate],</td>
</tr>
<tr>
<td>SigC</td>
<td>Signature of Group Controller Key Server</td>
</tr>
<tr>
<td>Cert</td>
<td>Necessary Certificates, zero or more</td>
</tr>
<tr>
<td>()SigX</td>
<td>Indicates minimum fields used in Signature</td>
</tr>
<tr>
<td>(data)*</td>
<td>Indicates encrypted information</td>
</tr>
<tr>
<td>[]</td>
<td>Indicate an optional data item</td>
</tr>
</tbody>
</table>

5.1 Assumptions

All potential GMS will have enough information available to them to use the correct Security Suite to join the group. This can be accomplished by a well known default suite 'Security Suite 1' or by announcing/posting another suite.

5.2 Definition Suite 1

GSAKMP implementations MUST support the following suite of algorithms and configurations. The following definition of Suite 1 borrows heavily from IKE's Oakley group 2 definition and Oakley itself.

The GSAKMP Suite 1 definition defines all the algorithm and cryptographic definitions required to process group establishment messages. It is important to note that GSAKMP does not negotiate these cryptographic mechanisms. This definition is set by the Group Owner via the Policy Token (passed during the GSAKMP exchange for member verification purposes).

The GSAKMP Suite definition is

Key download encryption algorithm definition:
Algorithm: 3DES
Mode: CBC64
Key Length: 192 bits

Policy Token encryption algorithm definition:
Algorithm: 3DES
Mode: CBC64
Key Length: 192 bits
Policy Token digital signature algorithm is:
  DSS-ASN1- DER
Hash algorithm is:
  SHA-1

Nonce Hash algorithm is:
  SHA-1

The Key Creation definition is:
Algorithm type is Diffie Hellman
MODP group definition
  g:   2
  p:   "FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1"
      "29024E08 8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD"
      "EF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245"
      "E485B576 625E7EC6 F44C42E9 A637ED6B 0BF5CB6 F406B7ED"
      "EE386BBB 5A899FA5 AE9F2411 7C4B1FE6 49286651 ECE65381"
      "FFFFFFFF FFFFFFFF"

NOTE: The p and g values comes from IKE [RFC 2409], section 6.2 Second Oakley Group, and p is 1024 bits long.

The digital signature algorithm is:
  DSS-ASN1- DER
Hash algorithm is:
  SHA-1

6 GSAKMP Payload Structure

A GSAKMP Message is composed of a GSAKMP Header (Section 6.1) followed by at least one GSAKMP Payload. All GSAKMP Payloads are composed of the Generic Payload Header (Section 6.2) followed by the specific payload data. The message is chained by a preceeding payload defining its succeeding payload. The final payload in a message will point to no succeeding payload.

6.1 GSAKMP Header

The GSAKMP Header fields are defined in Figure 3:

Group Identification Type (1 octet) - Table 6 presents the group identification types.

Group Identification Value (variable length) - Indicates the name/title
of the group. The specific format for this field is defined in Section A.1.

Next Payload (1 octet) - Indicates the type of the first payload in the message. The format for each payload is defined in the following sections. Table 7 presents the payload types.

Version (1 octet) - Indicates the version of the GSAKMP protocol in use. The current value is one (1).

Exchange Type (1 octet) - Indicates the type of exchange (also known as the message type). Table 8 presents the exchange type values.

Sequence ID (4 octets) - Group Management replay protection field.
Sequence ID for group management messages. If not a group management message, this value is set to zero (0). When a GCKS/SGCKS is initiated, it MUST generate/select an initial value, not zero (0). For each distinct group management message that this GCKS/SGCKS transmits, this value MUST be incremented. Receivers of this group management message MUST confirm that the value received is greater that the value of the sequence ID received with the last group management message from this GCKS/SGCKS. GCKS/SGCKS MUST also guarantee that this value does not suffer from rollover problems.
Table 7: Payload Types

<table>
<thead>
<tr>
<th>Next_Payload_Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>0</td>
</tr>
<tr>
<td>Policy Token</td>
<td>1</td>
</tr>
<tr>
<td>Key Download Packet</td>
<td>2</td>
</tr>
<tr>
<td>Rekey event</td>
<td>3</td>
</tr>
<tr>
<td>Identification</td>
<td>4</td>
</tr>
<tr>
<td>Reserved</td>
<td>5</td>
</tr>
<tr>
<td>Certificate</td>
<td>6</td>
</tr>
<tr>
<td>Reserved</td>
<td>7</td>
</tr>
<tr>
<td>Signature</td>
<td>8</td>
</tr>
<tr>
<td>Notification</td>
<td>9</td>
</tr>
<tr>
<td>Reserved</td>
<td>10</td>
</tr>
<tr>
<td>Key Creation</td>
<td>11</td>
</tr>
<tr>
<td>Nonce</td>
<td>12</td>
</tr>
<tr>
<td>Reserved</td>
<td>13 - 127</td>
</tr>
<tr>
<td>Private Use</td>
<td>128 -- 255</td>
</tr>
</tbody>
</table>

Table 8: Exchange Types

<table>
<thead>
<tr>
<th>Exchange_Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>0 - 3</td>
</tr>
<tr>
<td>Notification</td>
<td>4</td>
</tr>
<tr>
<td>Rekey Event</td>
<td>5</td>
</tr>
<tr>
<td>Reserved</td>
<td>6</td>
</tr>
<tr>
<td>No Message</td>
<td>7</td>
</tr>
<tr>
<td>Request to Join</td>
<td>8</td>
</tr>
<tr>
<td>Key Download</td>
<td>9</td>
</tr>
<tr>
<td>Cookie Download</td>
<td>10</td>
</tr>
<tr>
<td>Other</td>
<td>11-255</td>
</tr>
</tbody>
</table>
Length (4 octets) - Length of total message (header + payloads) in octets.

6.2 Generic Payload Header

Each GSAKMP payload defined in the following sections begins with a generic header, shown in Figure 4, which provides a payload "chaining" capability and clearly defines the boundaries of a payload. The Generic Payload Header fields are defined as follows:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
| Next Payload | RESERVED | Payload Length |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
```

Figure 4: Generic Payload Header

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 0. This field provides the "chaining" capability. Table 7 identifies the payload types.

RESERVED (1 octet) - Unused, set to 0.

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

6.3 Policy Token Payload

The Policy Token Payload contains group specific information that describes the group security relevant behaviors, access control parameters, and security mechanisms. This information may contain a digital signature(s) to prove authority and integrity of the information. Figure 5 shows the format of the payload.

The Policy Token Payload fields are defined as follows:

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 0.

RESERVED (1 octet) - Unused, set to 0.
Payload Length (2 octets) - Length in octets of the current payload, including the generic payload header.

ID Type (1 octet) - Specifies the type of Policy Token being used. Table 9 identifies the types of policy tokens.

Table 9: Policy Token Types

<table>
<thead>
<tr>
<th>ID_Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group</td>
<td>0</td>
</tr>
<tr>
<td>Auxiliary</td>
<td>1</td>
</tr>
<tr>
<td>Reserved</td>
<td>2-63</td>
</tr>
<tr>
<td>Unassigned</td>
<td>64-255</td>
</tr>
</tbody>
</table>

Policy Token Data (variable length) - Contains Policy Token information. The values for this field are group specific and the format is specified by the ID Type field. The Policy Token format is specified in [HCLM00].

If this payload is encrypted, only the Policy Token Data field is encrypted.

The payload type for the Policy Token Payload is one (1).

6.4 Key Download Payload

The Key Download Payload contains group keys (eg., group keys, initial rekey keys, etc.). These key download payloads can have several security attributes applied to them based upon the security policy of the group. Figure 6 shows the format of the payload.

The security policy of the group dictates that the key download payload MUST be encrypted with a key exchange key (KEK). The type of encryption used is specified in the Policy Token. The group members MUST create the KEK using the key creation method identified in the Key Creation Payload.
The Key Download Payload fields are defined as follows:

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 0.

RESERVED (1 octet) - Unused, set to 0.

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

Key Download Data (variable length) - Contains Key Download information.

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

For each Key Packet, the data format is as follows:

Key Download Data (KDD) Type (1 octet) -- Identifier for the Key Data field of this Key Packet. See Table 10 for the possible values of this field.

Table 10: Key Download Data Types

<table>
<thead>
<tr>
<th>Key Download Data Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>GTEK</td>
<td>0</td>
</tr>
<tr>
<td>Rekey - LKH</td>
<td>1</td>
</tr>
<tr>
<td>Unassigned</td>
<td>2-255</td>
</tr>
</tbody>
</table>

Key Download Length (2 octets) -- Length in octets of the Key Packet data following this field.

Key Packet Data (variable length) -- Contains Key information.
The format of this field is specific depending on the value of the Key Download Data field.

If this payload is encrypted, only the Key Download Data field is encrypted.

For a Key Download Data value of GTEK, the Key Packet Data field format is defined in Section A.2.1.

For a Key Download Data value of Rekey, the Key Packet Data field format is defined in Section A.2.2.

The payload type for the Key Download Packet is two (2).

### 6.5 Rekey Event Payload

The Rekey Event Payload contains multiple keys encrypted in Rekey keys. These Rekey Event payloads can have several security attributes applied to them based upon the security policy of the group. Figure 7 shows the format of the 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        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
!   ID Type     !           Rekey Event Data                    ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
```

**Figure 7: Rekey Event Payload Format**

The Rekey Event Payload fields are defined as follows:

**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 0.

**RESERVED (1 octet)** - Unused, set to 0.

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

**ID Type (1 octet)** - Specifies the type of Rekey Event being used. Table 11 presents the types of Rekey events.

**Rekey Event Data (variable length)** - Contains Rekey Event information.
Table 11: Rekey Event Types

<table>
<thead>
<tr>
<th>ID_Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>0</td>
</tr>
<tr>
<td>Group Recovery</td>
<td>1</td>
</tr>
<tr>
<td>Individual Recovery</td>
<td>2</td>
</tr>
<tr>
<td>Maintenance</td>
<td>3</td>
</tr>
<tr>
<td>Delete Group Key</td>
<td>4</td>
</tr>
<tr>
<td>Unassigned</td>
<td>5-255</td>
</tr>
</tbody>
</table>

The values for this field are group specific and the format is specified by the ID Type field. The format for this field is defined in Section A.3

The Rekey Event payload type is three (3).

6.6 Identification Payload

The Identification Payload contains entity-specific data used to exchange identification information. This information is used to verify the identities of members. Figure 8 shows the format of the Identification Payload.

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
! Next Payload  !   RESERVED    !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
!   ID Type     !            Identification Data                ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
```

Figure 8: Identification Payload Format

The Identification Payload fields are defined as follows:

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 0.

RESERVED (1 octet) - Unused, set to 0.

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

ID Type (1 octet) - Specifies the type of Identification being used. Table 12 identifies the types of identities.

### Table 12: Identification Types

<table>
<thead>
<tr>
<th>ID_Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sender Distinguished Name</td>
<td>0</td>
</tr>
<tr>
<td>Receiver Distinguished Name</td>
<td>1</td>
</tr>
<tr>
<td>Unassigned</td>
<td>2-255</td>
</tr>
</tbody>
</table>

Identification Data (variable length) - Contains identity information. The values for this field are group-specific and the format is specified by the ID Type field. The format for this field is defined in Section A.4.

The payload type for the Identification Payload is four (4).

#### 6.7 Certificate Payload

The Certificate Payload provides a means to transport certificates or other certificate-related information via GSAKMP and can appear in any GSAKMP message. Certificate payloads SHOULD be included in an exchange whenever an appropriate directory service (e.g. Secure DNS [DNSSEC]) is not available to distribute certificates. Figure 9 shows the format of the Certificate 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
! Next Payload !   RESERVED    !         Payload Length        ![+-]
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
! Cert Type                     !    Certificate Data           ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
```

**Figure 9: Certificate Payload Format**

The Certificate Payload fields are defined as follows:

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 0.
RESERVED (1 octet) - Unused, set to 0.

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

Certificate Type (2 octets) - This field indicates the type of certificate or certificate-related information contained in the Certificate Data field. Table 13 presents the types of certificate payloads.

Table 13: Certificate Payload Types

<table>
<thead>
<tr>
<th>Certificate_Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>0</td>
</tr>
<tr>
<td>Reserved</td>
<td>1 - 3</td>
</tr>
<tr>
<td>X.509 Certificate -- Signature - DER Encoding</td>
<td>4</td>
</tr>
<tr>
<td>Reserved</td>
<td>5 -- 65534</td>
</tr>
</tbody>
</table>

Certificate Data (variable length) - Actual encoding of certificate data. The type of certificate is indicated by the Certificate Type/Encoding field.

The payload type for the Certificate Payload is six (6).

6.8 Signature Payload

The Signature Payload contains data generated by the digital signature function. The digital signature covers the Signature Payload Span and the Signature Payload up to but not including the Signature Data Length. Figure 10 shows the format of the Signature Payload.

The Signature Payload fields are defined as follows:

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 0.

RESERVED (1 octet) - Unused, set to 0.

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

Signature Type (2 octets) -- Indicates the type of signature. This is really treated as 2 one byte fields. The first byte is the Signature Type as represented by Table 14. The second byte is the format for the
Figure 10: Signature Payload Format

Signer ID data and is defined by the values in Table 12.

Table 14: Signature Types

<table>
<thead>
<tr>
<th>Signature Type - byte 1</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSS with ASN.1/DER encoding</td>
<td>0</td>
</tr>
<tr>
<td>Other</td>
<td>1-255</td>
</tr>
</tbody>
</table>

Signature Payload Span (4 octets) - Identifies the information included in the signature. The first two octets define the first signature payload. The third and fourth octet define the last payload. The payloads in the message are an ordered sequence beginning at the header, with a value of zero (0). If the signature payload itself is not in the signature span, you MUST still sign over the signature payload up to the signature data.

Signature ID Role (1 octet) -- Specifies the type of Authorization (Role) being used. Refer to Table 15 for the types of authorization (role).

Signature Timestamp (4 octets) -- Date and time that the digital signature was applied. This field contains the time in seconds from the epoch 00:00 GMT 1 January 1970.

Signer ID Length (2 octets) - Length in octets of the Signer’ ID.

Signer ID Data (variable length) -- Data identifying the Signer’s ID (e.g., DN). The format for this field is based on the second byte in the
Table 15: Authorization Types

<table>
<thead>
<tr>
<th>Auth_Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group Controller Key Server</td>
<td>0</td>
</tr>
<tr>
<td>Group and Rekey Controller</td>
<td>1</td>
</tr>
<tr>
<td>Rekey Controller</td>
<td>2</td>
</tr>
<tr>
<td>Subordinate Group Controller Key Server</td>
<td>3</td>
</tr>
<tr>
<td>Subordinate Group and Rekey Controller</td>
<td>4</td>
</tr>
<tr>
<td>Subordinate Rekey Controller</td>
<td>5</td>
</tr>
<tr>
<td>Member ID</td>
<td>6</td>
</tr>
<tr>
<td>Unassigned</td>
<td>7-255</td>
</tr>
</tbody>
</table>

Signature Type field and is shown where that type is defined.

Signature Length (2 octets) -- Length in octets of the Signature Data.

Signature Data (variable length) - Data that results from applying the
digital signature function to the GSAKMP message and/or payload.

The payload type for the Signature Payload is eight (8).

6.9 Notification Payload

The Notification Payload can contain both GSAKMP and group specific data
and is used to transmit informational data, such as error conditions, to
a GSAKMP peer. It is possible to send multiple independant Notification
payloads in a single GSAKMP message. Figure 11 shows the format of the
Notification 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
! Next Payload  !   RESERVED    !        Payload Length         !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
!     Notify Payload Type       !  STATUS TYPE  !               ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
~                       Notification Data                       ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
```

Figure 11: Notification Payload Format

The Notification Payload fields are defined as follows:
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 0.

RESERVED (1 octet)  - Unused, set to 0.

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

Notify Payload Type (2 octets)  - Specifies the type of notification message. Table 16 presents the Notify Payload Types.

<table>
<thead>
<tr>
<th>Information</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>0</td>
</tr>
<tr>
<td>Invalid-Payload-Type</td>
<td>1</td>
</tr>
<tr>
<td>Situation-Not-Supported</td>
<td>2</td>
</tr>
<tr>
<td>Invalid-Major-Version</td>
<td>3</td>
</tr>
<tr>
<td>Invalid-Version</td>
<td>4</td>
</tr>
<tr>
<td>Invalid-Group-ID</td>
<td>5</td>
</tr>
<tr>
<td>Invalid-Sequence-ID</td>
<td>6</td>
</tr>
<tr>
<td>Payload-Malformed</td>
<td>7</td>
</tr>
<tr>
<td>Invalid-Key-Information</td>
<td>8</td>
</tr>
<tr>
<td>Invalid-ID-Information</td>
<td>9</td>
</tr>
<tr>
<td>Invalid-Cert-Encoding</td>
<td>10</td>
</tr>
<tr>
<td>Invalid-Certificate</td>
<td>11</td>
</tr>
<tr>
<td>Cert-Type-Unsupported</td>
<td>12</td>
</tr>
<tr>
<td>Invalid-Cert-Authority</td>
<td>13</td>
</tr>
<tr>
<td>Authentication-Failed</td>
<td>14</td>
</tr>
<tr>
<td>Invalid-Signature</td>
<td>15</td>
</tr>
<tr>
<td>Notify-GSA-Lifetime</td>
<td>16</td>
</tr>
<tr>
<td>Certificate-Unavailable</td>
<td>17</td>
</tr>
<tr>
<td>Unequal-Payload-Lengths</td>
<td>18</td>
</tr>
<tr>
<td>Unauthorized Request</td>
<td>19</td>
</tr>
<tr>
<td>Unable To Take Requested Role</td>
<td>20</td>
</tr>
<tr>
<td>Reserved (future use)</td>
<td>21-22</td>
</tr>
<tr>
<td>Reserved</td>
<td>23</td>
</tr>
<tr>
<td>Acknowledgement</td>
<td>24-25</td>
</tr>
<tr>
<td>Nack</td>
<td>26</td>
</tr>
<tr>
<td>Cookie Required</td>
<td>27</td>
</tr>
<tr>
<td>Cookie</td>
<td>28</td>
</tr>
<tr>
<td>Reserved (future use)</td>
<td>29-8191</td>
</tr>
<tr>
<td>Private Use</td>
<td>8192-16383</td>
</tr>
</tbody>
</table>

Status Type (1 octet)  - Specifies the status of group with respect to originator of notification. Notification information specifies status data and can be used by a process managing a SA database to communicate
with a peer process. For example, a secure front end or security gateway may use the Notify message to synchronize SA communication. Table 17 presents the Notification Payload Status Types.

Table 17: Notify Payload -- Status Types

<table>
<thead>
<tr>
<th>Status</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not connected</td>
<td>0</td>
</tr>
<tr>
<td>Establishing group</td>
<td>1</td>
</tr>
<tr>
<td>Reserved (future use)</td>
<td>2-255</td>
</tr>
</tbody>
</table>

Notification Data (variable length) - Informational or error data transmitted in addition to the Notify Payload Type. Values for this field are Domain of Interpretation (DOI)-specific.

The payload type for the Notification Payload is nine (9).

6.9.1 Notification Data - Acknowledgement (ACK) Payload Type

The data portion of the Notification payload of type ACK serves either for confirmation of correct receipt of the Key Download message, or, when needed, can provide other receipt information when included in a signed message. Figure 12 shows the format of the Notification Data - Acknowledge Payload Type.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
! Ack Type     !       Acknowledgement Data                     ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
```

Figure 12: Notification Data - Acknowledge Payload Type Format

The Notification Data - Acknowledgement Payload Type data fields are defined as follows:

Ack Type (1 octet) - Specifies the type of acknowledgement. Table 18 presents the Notify Acknowledgement Payload Types.

Simple - Data portion null.
Table 18: Acknowledgement Types

<table>
<thead>
<tr>
<th>ACK_Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple</td>
<td>0</td>
</tr>
<tr>
<td>Unassigned</td>
<td>1-255</td>
</tr>
</tbody>
</table>

6.9.2 Notification Data - Cookie Request and Cookie Payload Type

The data portion of the Notification payload of types Cookie_Request and Cookie contain the Cookie value. The value for this field will have been computed by the reponder GCKS and sent to the GM. The GM will take the value received and place AS IS into the Notification payload Notification Data field of type Cookie that is transmitted in the "Request to Join with Cookie Info" back to the GCKS. This value MUST NOT be modified.

The format for this is already described in the discussion on cookies in section 4.2.2.

6.10 Key Creation Payload

The Key Creation Payload contains information used to create key encryption keys. The security attributes for this payload are provided in the Policy Token. Figure 13 shows the format of the 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
! Next Payload  !   RESERVED    !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
!   ID Type     !           Key Creation Data                   ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
```

Figure 13: Key Creation Payload Format

The Key Creation Payload fields are defined as follows:

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 0.

RESERVED (1 octet) - Unused, set to 0.
Payload Length (2 octets) - Length in octets of the current payload, including the generic payload header.

ID Type (1 octet) - Specifies the type of Key Creation being used. Table 19 identifies the types of key download information.

<table>
<thead>
<tr>
<th>ID_Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>Diffie-Hellman</td>
<td>1</td>
</tr>
<tr>
<td>other</td>
<td>2-255</td>
</tr>
</tbody>
</table>

Key Creation Data (variable length) - Contains Key Creation information. The values for this field are group specific and the format is specified by the ID Type field.

The payload type for the Key Creation Packet is eleven (11).

6.11 Nonce Payload

The Nonce Payload contains random data used to guarantee freshness during an exchange and protect against replay attacks. Figure 14 shows the format of the Nonce 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 !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+++
! Nonce Type ! Nonce Data ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+++
```

Figure 14: Nonce Payload Format

The Nonce Payload fields are defined as follows:

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 0.

RESERVED (1 octet) - Unused, set to 0.
Payload Length (2 octets) - Length in octets of the current payload, including the generic payload header.

Nonce Type (1 octet) - Specifies the type of Nonce being used. Table 20 identifies the types of nonces.

Table 20: Nonce Types

<table>
<thead>
<tr>
<th>Nonce_Type</th>
<th>Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Initiator</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Responder</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Combined</td>
<td>3</td>
<td>Hash ( Append (Initiator_Value, Responder_Value) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The hash type comes from the Security Suite Definition.</td>
</tr>
<tr>
<td>Unassigned</td>
<td>4-255</td>
<td></td>
</tr>
</tbody>
</table>

Nonce Data (variable length) - Contains the nonce information. The values for this field are group-specific and the format is specified by the Nonce Type field. If no group-specific information is provided, the minimum length for this field is 4 bytes.

The payload type for the Nonce Payload is twelve (12).
7 GSAKMP State Diagram

Figure 15 presents the states encountered in the use of this protocol. Table 21 defines the states. Table 22 defines the transitions.

![GSAKMP State Diagram](image)

Figure 15: GSAKMP State Diagram
<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idle</td>
<td>GSAKMP Application waiting for input</td>
</tr>
<tr>
<td>Wait for GCKS Event</td>
<td>GCKS up and running, waiting for events</td>
</tr>
<tr>
<td>Wait for Response</td>
<td>GCKS has sent Key Download, waiting for response from GM</td>
</tr>
<tr>
<td>Wait for Group Membership</td>
<td>GM in process of joining group</td>
</tr>
<tr>
<td>Wait for Group Membership Event</td>
<td>GM has group key, waiting for group management messages.</td>
</tr>
</tbody>
</table>
Table 22: State Transition Events

| Transition 1  | Create group command |
| Transition 2  | Receive bad RTJ      |
|               | Receive valid command to change group membership |
|               | Send Compromise message x times |
| Transition 3  | Receive valid RTJ   |
| Transition 4  | Timeout             |
|               | Receive Ack         |
|               | Receive Nack        |
| Transition 5  | Delete group command |
| Transition 1a | Join group command  |
| Transition 2a | Send Ack            |
| Transition 3a | Receipt of group management messages |
| Transition 4a | Delete group command |
| Transition 5a | Time out            |
|               | Msg failure         |
|               | errors              |
This appendix defines the format of all variable length fields that contain multiple items of information.

A.1 GSAKMP Header Fields

Group Identification Value Format - For IPv4, this field is 8 octets long, 4 octets for IPv4 internet address in network byte order, and 4 octets for serial number in network byte order.

A.2 Key Download Payload Fields

A.2.1 GTEK Key Packet Fields

For a Key Download Data value of GTEK, the Key Packet Data field is formatted as follows:

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

Key Creation Date (4 octets) -- This is the time value of when this key data was originally generated. This field contains the time in seconds from the epoch 00:00 GMT 1 January 1970.

Key Expiration Date (4 octets) -- This is the time value of when this key is no longer valid for use. This field contains the time in seconds from the epoch 00:00 GMT 1 January 1970.

Key Handle (4 octets) -- This is the randomly generated value to uniquely identify a key.

Key Data (variable length) -- This is the actual encryption key data, which is dependent on the Key Type algorithm for its format.

A.2.2 Rekey Key Packet Fields

This field is defined within the specific type of rekey scheme used by the group.

For LKH, the format of the this field is defined in Section B.1.
A.3 Rekey Event Payload Fields

This field is defined within the specific type of rekey event scheme used by the group.

For LKH, the format of the this field is defined in Section B.2.

A.4 Identification Payload Fields

Identification Data  - For type Distinguished Name (DN) the format is:

   Serial Number (4 octets)  -- The certificate serial number in network byte order.
   Length (4 octets)  -- Length in octets of the DN Data field.
   DN Data (variable length)  -- The actual DN value.

B APPENDIX B -- LKH Variable Length Payload Field Definitions

This appendix defines the format of all LKH specific variable length fields that contain multiple items of information.

B.1 LKH Rekey Key Packet Fields

When using the Logical Key Hierarchy (LKH) protocol for Rekey operations, the Key Packet Data is assumed to contain LKH Array data of the following format:

   LKH Version (1 octet)  -- Contains the version of the LKH protocol in which the data is formatted. The current value for this field is one (1).
   Leaf ID (2 octets)  -- This is the Leaf Node ID of the LKH sequence contained in this Key Packet Data block. The leftmost leaf node ID value is one (1).
   Number of LKH Keys (2 octets)  -- This value is the number of distinct LKH keys in this sequence.

   For each LKH key in the sequence, the data format is as follows:
LKH ID (2 octets) -- This is the position of this key in the binary tree structure used by LKH. The base node of the binary tree has a value of one (1).

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

Key Creation Date (4 octets) -- This is the time value of when this key data was originally generated. This field contains the time in seconds from the epoch 00:00 GMT 1 January 1970.

Key Expiration Date (4 octets) -- This is the time value of when this key is no longer valid for use. This field contains the time in seconds from the epoch 00:00 GMT 1 January 1970.

Key Handle (4 octets) -- This is the randomly generated value to uniquely identify a key.

Key Data (variable length) -- This is the actual encryption key data, which is dependent on the Key Type algorithm for its format.

B.2 LKH Rekey Packet Data Format Fields

This section defines the format of the Rekey Event Data in the Rekey Event Payload, when using Logical Key Hierarchy (LKH) as the rekeying mechanism.

The Rekey Event Data consists of Rekey Event Header and Rekey Event Packet Data(s). A Packet Data is a complete set of information that an end-user requires to be Rekeyed. Packet Datas are comprised of new Key Packs of types GTEK and Rekey.

B.2.1 Rekey Event Header

The Rekey Event Data Header contains information about the rekey data being transmitted to the group. Figure 16 shows the format for the header.

Group Identification Value (variable length) -- Indicates the name/title of the group to be rekeyed. This is the same format as the Group Identification Value in Section 6.1 GSAKMP Message Header.

Time/Date Stamp (4 octets) -- This is the time stamp when the Rekey Event Data was generated. This field contains the time in seconds from the epoch 00:00 GMT 1 January 1970.

Rekey Type (1 octet) -- This is the Rekey algorithm being used for this
group. This value is token specific. For this appendix, this value is LKH, which has a value of one (1).

Algorithm Version (1 octet) - Indicates the version of the Rekey Type being used. The value at this time is one (1).

# of Rekey Packets (2 octets) - The number of Rekey Packets contained in the Rekey Data.

Rekey Event Packet Data(s) (variable length) - Contains the packets of rekey event information being transmitted.

B.2.2 Rekey Event Packet Data(s)

As defined in the Rekey Event Header, # of Rekey Packets field, multiple pieces of information are sent in a Rekey Event Data. Each end user, will be interested in only one packet of the information sent. Each Packet, will contain all the Key Packs that a user requires. For each Packet, the data following the Security Header fields is encrypted with the key identified in the Security Header. Figure 17 shows the format of each Rekey Event Packet with respect to LKH.

Packet Length (2 octets) - Length in octets of the Rekey Packet, which consists of the # of Key Packs and the Key Pack Data(s).

Security Header: LKH ID (2 octets) - This is the LKH ID of the Rekey Pack that is being used for encryption/decryption. Refer to Section B.1 for the values of this field.

Security Header: Key Handle (4 octets) - This is a randomly generated
value to uniquely identify the key defined by the LKH ID.

# of Key Packs (2 octets) - The number of key packs contained in this Packet Data.

Key Pack Data(s) (variable length) - Contains all the key pack data for this packet.

**B.2.3 Key Pack Data**

Each Key Pack contains all the information about the key. Figure 18 shows the format for each type of key pack.

```
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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
! Pack Type       ! Pack Length                        ! Pack Data
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-!
```

**Figure 18:  B. 3:  Key Pack Data Format**

Pack Type (1 octet) - The type of key in this key pack. Legal values are GTEK (0) and LKH (1).

Pack Length (2 octets) - The length of the Pack Data.

Pack Data (variable length) - The actual data of the key, defined by the key type.
B.2.4 Pack Data Formats

There are 2 legal values for the Pack Type, GTEK and LKH. The formats for each Pack type are defined in this section.

B.2.4.1 GTEK Pack Data

This is data for the new GTEK being sent to the Rekeyed group.

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

Key Creation Date (4 octets) - This is the time value of when this key data was originally generated. This field contains the time in seconds from the epoch 00:00 GMT 1 January 1970.

Key Expiration Date (4 octets) - This is the time value of when this key is no longer valid for use. This field contains the time in seconds from the epoch 00:00 GMT 1 January 1970.

Key Handle (4 octets) - This is the randomly generated value to uniquely identify a key.

Key Data (variable length) - This is the actual encryption key data, which is dependent on the Key Type algorithm for its format.

B.2.4.2 LKH Pack Data

This is the data to fix an Group Member Rekey sequence to recover from a compromise.

LKH ID (2 octets) -- This is the position of this key in the binary tree structure used by LKH. Refer to Section B.1 for the values of this field.

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

Key Creation Date (4 octets) - This is the time value of when this key data was originally generated. This field contains the time in seconds from the epoch 00:00 GMT 1 January 1970.

Key Expiration Date (4 octets) - This is the time value of when this key is no longer valid for use. This field contains the time in seconds from the epoch 00:00 GMT 1 January 1970.
Key Handle (4 octets) - This is the randomly generated value to uniquely identify a key.

Key Data (variable length) - This is the actual encryption key data, which is dependent on the Key Type algorithm for its format.

**B.2.5 Example**

This section will give an example of the data. The data to be transmitted is:

<table>
<thead>
<tr>
<th>GroupID</th>
<th>Date/Time</th>
<th>Rekey Type</th>
<th>Algorithm Ver</th>
<th># of Packets</th>
</tr>
</thead>
<tbody>
<tr>
<td>(GTEK)A, (GTEK, B, E)6, (GTEK, B)F</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This data shows that three packets are being transmitted. Read each packet as:

a) GTEK wrapped in LKH key A
b) GTEK, LKH keys B & E, all wrapped in LKH key 6
c) GTEK and LKH key B, all wrapped in LKH key F

We will show format for all header data, and packet (b).

Definition of values:

- 0xLLLL  - length value
- 0xHHHHHHH#  - handle value
- 0xTTTTTTTC  - creation time
- 0xTTTTTTTE  - expiration time

**GroupID**  - 0xAABBCCDD

- 0x12345678

**Date/Time**  - 0x34574509

**Rekey Type**  - 0x01 (LKH)

**Algorithm Vers**  - 0x01

**# of Packets**  - 0x0003

For Packet (b):

- **Packet Length**  - 0xLLLL
- **Sec HDR:LKH ID**  - 0x0006
- **Sec HDR:Key Handle**  - 0xHHHHHHHH1
- **# of Key Packs**  - 0x0003

**Key Pack 1:**

- **Pack Type**  - 0x00 (GTEK)
- **Pack Length**  - 0xLLLL

  - **Key Type**  - 0x02 (DES3)
  - **Key Creation Date**  - 0xTTTTTTTC
  - **Key Expiration Date**  - 0xTTTTTTTE
Key Handle          - 0xHHHHHHH2
Key Data            - variable, based on key definition

Key Pack 2:
Pack Type   - 0x01 (LKH)
Pack Length - 0xLLLL
  LKH ID              - 0x000B
  Key Type            - 0x02 (DES3)
  Key Creation Date   - 0xTTTTTTTC
  Key Expiration Date - 0xTTTTTTTE
  Key Handle          - 0xHHHHHHH3
  Key Data            - variable, based on key definition

Key Pack 3:
Pack Type   - 0x01 (LKH)
Pack Length - 0xLLLL
  LKH ID              - 0x000E
  Key Type            - 0x02 (DES3)
  Key Creation Date   - 0xTTTTTTTC
  Key Expiration Date - 0xTTTTTTTE
  Key Handle          - 0xHHHHHHH4
  Key Data            - variable, based on key definition

C APPENDIX C -- Change History

C.1 Changes from GSAKMP-01 to GSAKMP-02 February 2003

This specification was based on two earlier versions of GSAKMP drafts, referred to as GSAKMP and GSAKMP-Light. These two specifications were merged to incorporate all information necessary to allow the original GSAKMP-Light specification to stand on its own. The original GSAKMP protocol no longer exists as a standard, it has been subsumed by GSAKMP-Light. GSAKMP-Light is now called GSAKMP.

Major modifications to the specification are


Signature Processing: The signature processing has been modified.
C.2 Changes from GSAKMP-02 to GSAKMP-03 June 2003

1. The specification was modified to confirm that key words are used as defined by RFC2119.

2. The Protocol Considerations section for IANA port number was added.

3. The Cookie section for mitigation of DoS attacks was added.

4. The Protocol State Diagram was added.

D References, Authors Addresses, and Acknowledgements

The following references were used in the preparation of this document.

D.1 References


[DH77], Diffie, W., and M. Hellman, ‘‘New Directions in Cryptography’’, IEEE Transactions on Information Theory, June 1977.


for Message Authentication'', RFC 2104, Informational, February


[RFC 2402] Kent S. and Atkinson, R., ``IP Authentication Header'', RFC 2402,


Security Association and Key Management Protocol (ISAKMP)'', RFC 2408,

[RFC 2409] Harkins D. and Carrel D., ``The Internet Key Exchange (IKE)'',

[IEEEv2], C. Kaufman, ``Internet Key Exchange (IKEv2) Protocol'',
draft-ietf-ipsec-ikev2-o6.txt, March 2003

Internet Society, San Diego, CA, February 2001

[HCLM00] H. Harney, A. Colegrove, P. Lough, U. Meth, ``GSAKMP Token
Specification'', draft-ietf-msec-tokenspec-sec-00.txt

D.2 Authors Addresses

Hugh Harney (point-of-contact)
SPARTA, Inc.
7075 Samuel Morse Drive
Columbia, MD 21046
(410) 872-1515 ext 203
FAX (410) 872-8079

Harney, etal.        draft-ietf-msec-gsakmp-sec-02.txt        [Page 50]
hh@sparta.com

Uri Meth
SPARTA, Inc.
7075 Samuel Morse Drive
Columbia, MD 21046
(410) 872-1515 ext 233
FAX (410) 872-8079
umeth@sparta.com

Andrea Colegrove
SPARTA, Inc.
7075 Samuel Morse Drive
Columbia, MD 21046
(410) 872-1515 ext 232
FAX (410) 872-8079
acc@sparta.com

Angela Schuett
R231 NSA
9800 Savage Rd
Suite 6534
Fort Meade, MD 20755
(301) 688-0850
FAX (301) 688-0255
amschue@tycho.ncsc.mil

Patrick McDaniel
AT&T Labs - Research
A203, Bldg. 103
180 Park Ave.
Florham Park, NJ 07932
Office (973) 360-5721
pdmcdan@research.att.com

Gavin Kenny
LogicaCMG
Keats House
The Office Park
Springfield Drive
Leatherhead, Surrey KT22 7LP, UK
+44 1372 838043
FAX +44 1372 759196
Gavin.IA.Kenny@LogicaCMG.com

Haitham S. Cruickshank
Centre for Communication Systems Research (CCSR)
University of Surrey
Guildford, Surrey GU2 7XH, UK
+44 1483 686007 (indirect 689844)
D.3 Acknowledgements

The following individuals deserve recognition and thanks for their contributions which have greatly improved this protocol: Eric Harder is an author to the Tunneled-GSAKMP, whose concepts are found in GSAKMP as well. Rodney Fleischer, also a Tunneled-GSAKMP author, and Peter Lough were both instrumental in coding a prototype of the GSAKMP software and helped define many areas of the protocol that were vague at best. Andrew McFarland and Gregory Bergren provided critical analysis of early versions of the specification. Ran Canetti analyzed the security of the protocol and provided denial of service suggestions leading to optional "cookie protection".

Document expiration: December 30, 2003