BBN Technical Memorandum No. 1242

Local Group Keying Protocol

Job No. 11871003

July 12, 2000

Prepared for:

Defense Advanced Research Projects Agency
Information Technology Office
3701 North Fairfax Drive
Arlington, VA 22203-1714

Prepared by:

BBN Technologies
10 Moulton Street
Cambridge, MA 02138
Local Group Keying Protocol

M. Condell (mcondell@bn.com) T. Strayer (strayer@bn.com) M. Fredette (fredette@bn.com)
A. Snoeren (snoeren@bn.com) C. Partridge (craig@bn.com) I. Castineyra (isidro@bn.com)

$Id: LGKP.sgml v 1.7 1999/10/27 13:51:26 snoeren Exp$

Abstract

This document specifies a mechanism for a network node to discover all of its neighbors one hop away, and to establish a single group key that the node can use with those neighbors. Once a neighbor is discovered using beacons in a join procedure, point-to-point Security Associations (SAs) are established using IKE [RFC2409]. The node’s Peer Group Key is then communicated over these SAs to the neighbor, bringing the neighbor into the node’s Peer Group. It is expected that the Peer Group Key will be a symmetric key that the node will use to communicate with its Peer Group using IPSec [RFC2401] over broadcast or multicast.

1. Introduction

The Local Group Keying Protocol (LGKP) establishes a group key between a node and its immediate peers — that is, all other nodes one hop away. It is expected that the group key is a symmetric key that the node will use to communicate with those peers using IPSec [RFC2401] over broadcast or multicast.

Normally with IPSec, all keying material is established by IKE, the Internet Key Exchange protocol [RFC2409], but IKE is only capable of negotiating keys for point-to-point communications and cannot be used in a broadcast or multicast setting.

There has been some work done in the IETF and IRTF on mechanisms to provide keys for multicast communications [RFC2627, RFC2904, HaHa99, CCP99, HaCaMc99, HaCaDe99]. However, most of these mechanisms are either heavyweight or do not cleanly allow members to join or be removed from groups. The LGKP applies many of the design ideas in these documents while taking advantage of a fairly simple environment to produce a simple and clean group keying protocol.

In LGKP, there are two steps required for group key exchange: group joining and group keying. First, a node must discover each of its peers and, upon discovery, establish point-to-point Security Associations (SAs) with each peer. This discovery process is the Join Procedure, which builds a list of peers to be keyed. In the second step, the node provides each unkeyed peer with the group key. This is called the Keying Procedure.

The remainder of this document provides an overview of the protocol, followed by a detailed
description of the protocol and its operation. Keywords "MUST", "MUST NOT", "REQUIRED", "SHOULD", "SHOULD NOT" and "MAY" that appear in this document are to be interpreted as described in [RFC2119].

1.1 Status of this Memo

Discussion and suggestions for improvement are requested. Please refer to the current edition of the "IAB Official Protocol Standards" for the standardization state and status of this protocol. Distribution of this memo is unlimited.

1.2 Definitions

Node

Routers and workstations are examples of nodes. Nodes are connected to other nodes by links.

Link

A link is a transmission medium that connects two nodes and allows bidirectional or unidirectional communication.

Neighbor

One node is a neighbor of another if there is at least one unidirectional link from the first node to the second, and the link is no longer than one hop.

Peer

Two nodes are peers if each is a neighbor of the other and, therefore, each node can send a message to the other in exactly one hop.

Peer Group Key

The Peer Group Key is the keying material a node wants to distribute to its peers.

1.3 The Group Controller and its Peer Group

In LGKP, a node is the Group Controller for the group consisting of its peers, or Peer Group. The relationship between Group Controller and Peer Group is one-way: within a group, the Group Controller is the only initiator, and the Peer Group members are only responders. This is shown below.

```
+-----+ +-----+ +-----+
| Peer | | Peer | | Peer |
+-----+ +-----+ +-----+
    |     |     |
+---------------------+ | Group Controller |
    |     |     |
+-----+ +-----+ +-----+
| Peer | | Peer | | Peer |
+-----+ +-----+ +-----+
```

The Group Controller chooses the Peer Group Key it wants to distribute to its peers.

Note that any node may execute LGKP as the Group Controller for its Peer Group, establishing parallel one-way relationships with those peers that are the controllers of the other groups the node belongs to. The focus of this document, however, is how LGKP distributes a key within a single group.
1.4 Addresses

Network nodes (especially routers) typically have multiple network interfaces and multiple network addresses, where each address maps to a single interface, and each interface has at least one address. For the purposes of LGKP, an interface will be known by only one of its addresses; this address SHOULD be the primary address associated with the interface.

Throughout this document, the terms address and IP address are used for both IP version 4 (IPv4) and IP version 6 (IPv6) network addresses.

Between any two peers there are at least two addresses of note, and sometimes there are four. When the peers share a bidirectional link, as shown below, there are two addresses, a and b.

```
+---------+       +---------+
|         |       |         |
| Peer 1  |   a    | Peer 2  |
+---------+       +---------+
```

Address a is the address of Peer 1’s interface to the link, and address b is the address of Peer 2’s interface to the link. Peer 1 sends to Peer 2 by addressing IP packets to address b, and Peer 2 sends to Peer 1 by addressing IP packets to address a.

However, if the two peers share two unidirectional links, they are still peers, but they send to each other on different links:

```
+---------+       +---------+
|         |       |         |
|         |   c    |         |
| Peer 3  |       | Peer 4  |
+---------+       +---------+
```

```
+---------+       +---------+
|         |       |         |
|         |   d    |         |
|         |       |         |
```

Now there are four addresses. Peer 3’s outgoing address is c and incoming address is e, and Peer 4’s incoming address is d and outgoing address is f.

From one node’s perspective, these four addresses are known as the following: Outgoing Address, Peer’s Incoming Address, Peer’s Outgoing Address, and Incoming Address. For bidirectional links, the Incoming Address and the Outgoing Address are the same, and so are the Peer’s Incoming Address and Peer’s Outgoing Address. Referring to the figures above:

<table>
<thead>
<tr>
<th>node</th>
<th>Peer 1</th>
<th>Peer 2</th>
<th>Peer 3</th>
<th>Peer 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outgoing Address</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>f</td>
</tr>
<tr>
<td>Peer’s Incoming Address</td>
<td>b</td>
<td>a</td>
<td>d</td>
<td>e</td>
</tr>
<tr>
<td>Peer’s Outgoing Address</td>
<td>b</td>
<td>a</td>
<td>f</td>
<td>c</td>
</tr>
<tr>
<td>Incoming Address</td>
<td>a</td>
<td>b</td>
<td>e</td>
<td>d</td>
</tr>
</tbody>
</table>

2. Protocol Overview

LGKP has two main procedures: the Join Procedure and the Keying Procedure. The Join Procedure periodically sends an LGKP Join Message as a beacon to its neighbors. Neighbors hear these beacons and, in their own beacons, indicate that the neighbor was heard. A Group Controller first builds a list of nodes it can hear, then from that, builds a list of peers that it can both hear and send to. This is the goal of the Join Procedure, and the lists are called, respectively, the Heard List and the Pre-Key Group.

Once a Group Controller "discovers" a peer, the Keying Procedure requests the establishment of point-to-point SAs using IKE. When these are established, the Peer Group Key is given to the peer using an LGKP Keying Message. After the peer acknowledges receipt of the key, the peer moves from the Pre-Key Group to a list called the Peer Group.
A Group Controller keeps track of the neighbors from whom it has heard Join Messages. When a Group Controller has not heard from a neighbor for some timer-controller period, records for that neighbor are purged from the Heard List. If that neighbor is also a peer, records for that peer are purged from the Pre-Key Group and Peer Group. To do this, every entry in the Heard List, Pre-Key Group, or Peer Group has associated with it a boolean called heard-from. This boolean is initialized to false, and set to true when a Join Message arrives from the neighbor or peer.

LGKP requires both authentication and encryption for various aspects of the protocol. For authentication of Join Messages, we specify the use of public-key cryptography. For authentication and encryption of Keying Messages, we assume the use of IPsec.

2.1 Heard List

The Heard List is the list of all neighbors from whom a node has received a Join Message. The information that is placed into the Heard List is the IP address of the sender of the Join Message (called the Peer Address), and the interface on which this Join Message was received (called the Interface Address). Each record in the Heard List is a tuple:

<peerAddr, IFAddr>

2.2 Pre-Key Group and Peer Group

The Pre-Key Group is the list of all peers of a Group Controller that have not yet completed the Keying Procedure. The Peer Group is the list of all peers of a Group Controller that have completed the Keying Procedure. The information that represents a peer within the Pre-Key Group and Peer Group is the Outgoing Address, the Peer's Incoming Address, the Peer's Outgoing Address, and the Incoming Address. Each record in the Pre-Key Group and Peer Group is a 4-tuple:

<OutAddr, PeerInAddr, PeerOutAddr, InAddr>

If the two peers are connected by a single bidirectional link, the PeerInAddr and PeerOutAddr values may be the same, and the InAddr and OutAddr values may be the same (they will be the same if there are no aliases on these interfaces).

2.3 Message Transport

LGKP does not have its own IP protocol number, nor do we specify that it be used with any particular high-level protocol on any particular port. Instead, LGKP is meant to be run by an application over a datagram transport protocol and ports appropriate for that application.

Because Join Messages are sent before security associations are established, they are broadcast or multicast "in the clear" over the application's transport. This is shown below:

```
+----------------------------------+
| IP Header | Transport Header | LGKP Join Message |
+----------------------------------+
```

Once the point-to-point SAs are established using IKE, the Keying Messages are unicast using IPsec's Encapsulating Security Payload (ESP) [RFC2406], followed by the application's transport:

```
+----------------------------------+
| IP Header | ESP | Transport Header | LGKP Keying Message |
+----------------------------------+
```

Note that Keying Messages MUST be encrypted to protected the confidentiality of the Peer Group Key, so ESP MUST NOT be used with NULL encryption.
3. Join Procedure

In order for a Group Controller to build its Peer Group, it periodically broadcasts or multicasts a Join Message on all of its interfaces. When a node receives a Join Message, the receiving node notes whom the Join Message came from by adding the sending node’s information to its Heard List.

Simply receiving a Join Message does not establish a peering relationship -- peers must have bidirectional communication, even if that bidirectional communication is accomplished by two unidirectional links. A node must hear from a node that has heard from it before a peering relationship exists.

When a node sends a Join Message, the message carries the node’s Heard List information as an indication of all of the neighbors that the node can hear. Consequently, a node may eventually see a Join Message that contains one or more of its own addresses, indicating that the node has been heard by the neighbor that sent the Join Message. This is shown below:

```
Node 1                      Node 2

Join Message (heard no one) -----------------> Receive Join Message
Add Node 1 to Heard List

Receive Join Message
Add Node 2 to Heard List
Add Node 2 to Pre-Key Group
```

When a node receives a Join Message from a neighbor and finds one of its interface addresses in the list of heard addresses carried in the Join Message, the receiving node knows that it has a peer, and the peer is added to the Pre-Key Group.

The receiving node installs policies into the IPSec Security Policy Database (SPD) requiring an SA with the peer for Keying Messages. The figure below shows the continuation of the Join Message exchanges:

```
Node 1                      Node 2

Join Message (heard no one) -----------------> Receive Join Message
Add Node 1 to Heard List

Receive Join Message
Add Node 2 to Heard List
Add Node 2 to Pre-Key Group
Install policies in SPD

Receive Join Message
Add Node 1 to Pre-Key Group
Install policies in SPD
```

After IKE establishes the needed SAs as directed by the SPD, they are used to protect the Keying Messages carrying the Peer Group Key.

3.1 Join Message Format

The Join Message is used as a periodic beacon. It carries information identifying who sent the message (more specifically, the address of the interface from which this message was sent), as well as a list of all of the beacons the sender has heard. The Join Messages cannot use IPSec
authentication since security associations have not yet been established, so the Join Messages carry a public-key signature and a certificate as authentication credentials.

The Join Message format is given below:

```
  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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
|Version | Resvd | AddrType | NumAddr |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
                  | SenderAddr |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
                  | HeardAddr |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
                  | CertType | Reserved | CertReqLength |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
                  | Certificate Request |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
                  | SigType | Reserved | SigLength |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
                  | Signature |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
                  | Reserved | CertLength |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
                  | Certificate |
```

All fields larger than one octet MUST be in network byte order (big-endian format).

Version (4 bits)
The `Version` field value indicates the version of LGKP in use. The version described by this document is 0, and is used in all LGKP messages. This field will be changed if a LGKP message format should change. If a Join Message arrives with an unexpected version number, the message MUST be discarded.

Reserved (4, 8 or 16 bits)
The `Reserved` fields are reserved for future use. The value for all `Reserved` fields for this version of the LGKP MUST be set to 0.

AddrType (8 bits)
The `AddrType` field value specifies the type of the addresses carried in the `SenderAddr` and `HeardAddr` fields. All addresses in the `SenderAddr` and `HeardAddr` fields are of the same type. The values for the `AddrType` field are given by the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Address Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>IPv4</td>
</tr>
<tr>
<td>0x01</td>
<td>IPv6</td>
</tr>
</tbody>
</table>

NumAddr (16 bits)
The `NumAddr` field value specifies the number of entries in the `HeardAddr` field.

SenderAddr (32 or 128 bits)
The `SenderAddr` field contains the address of the interface on which this message is sent. The address type is that described in `AddrType` field.

HeardAddr (variable)
The `HeardAddr` field is an array of pairs of IP addresses. This is a copy of the sending node's Heard List, and the entries in the `HeardAddr` array have the same fields as the Heard List, that is, `PeerAddr` and `IFAddr`. All heard addresses from the Heard List MUST be listed in the `HeardAddr` field.
CertType (8 bits)
The CertType field value specifies the type of certificate or certificate-related information in the Certificate Request and Certificate fields in this message. The values for the CertType field are the same as those defined for the Certificate Encoding field in [RFC2408] or its successor.

CertReqLength (16 bits)
The CertReqLength field value specifies the length of the Certificate Request field in bytes. Padding is not included in this length.

Certificate Request (variable)
The Certificate Request field holds information the receiver SHOULD use to select certificates to use in its own Join message. The type of information is indicated by the CertType field. If the Certificate Request field does not end on a 4-octet boundary, the field MUST be padded at the end with zero bytes to align on the next 4-octet boundary.

As an example, in an X.509 infrastructure this field SHOULD contain the Distinguished Name encoding of the Issuer Name of an X.509 certificate authority acceptable to the sender of this Join message.

SigType (8 bits)
The SigType field value specifies the type of signature used to sign this message. The values for the SigType field are given by the following table:

<table>
<thead>
<tr>
<th>value</th>
<th>Address Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>RESERVED</td>
</tr>
<tr>
<td>0x01</td>
<td>RSA-PKCS1</td>
</tr>
<tr>
<td>0x02</td>
<td>DSA-SHA1</td>
</tr>
</tbody>
</table>

Future values can be added to this list without perturbing the version number.

SigLength (16 bits)
The SigLength field value specifies the length of the Signature field in bytes. Padding is not included in this length.

Signature (variable)
The Signature field contains the signature over the message from the beginning up to and including the SigType field, using the algorithms specified in the SigType field. If the Signature field does not end on a 4-octet boundary, the field MUST be padded at the end with zero bytes to align on the next 4-octet boundary.

CertLength (16 bits)
The CertLength field value specifies the length of the Certificate field in bytes. Padding is not included in this length.

Certificate (variable)
The Certificate field holds the certificate information that the sender is offering to establish its public key and its credentials for joining the group. The type of certificate information is indicated by the CertType field. If the Certificate Request field does not end on a 4-octet boundary, the field MUST be padded at the end with zero bytes to align on the next 4-octet boundary.

As an example, in an X.509 infrastructure this field SHOULD contain the X.509 public-key certificate for the node, and the X.509 attribute certificate authorizing the node to join the group.
3.2 Join Message Transmission

A node MUST emit a Join Message on each interface upon startup, and periodically after that. Join Messages are broadcast or multicast, as determined by the application, to reach all prospective members of the group. The Message Send Event, described in the Periodic Events section, controls the frequency of sending Join Messages.

When creating a Join message, the node MUST do the following for each interface:

1. Set the AddrType value appropriately for the addresses being used in the SenderAddr and HeardAddr fields of the Join Message. All addresses must be of the same type.
2. Set the SenderAddr to the IP address of the interface on which the Join Message is being sent.
3. Set NumAddr value to the number of address pairs in the Heard List.
4. Copy each of the IP addresses in the Heard List followed by the IP address of the interface on which it was heard into the HeardAddr field of the Join Message.
5. Set the CertType value for the node's certificate.
6. Set the CertKeyLength value for the node's certificate request.
7. Copy any certificate request information into the Certificate Request field of the Join Message.
8. Set the SigType appropriately for the signature that will be computed.
9. Sign the message and copy the signature and its length into the Signature and SigLength fields, respectively, of the Join Message.
10. Set the CertLength value for the node's certificate.
12. Broadcast or multicast the Join Message (as determined by the application), with its TTL set to 1 and on the interface associated with the SenderAddr.

3.3 Join Message Processing

When a node receives a Join message, it MUST do the following:

1. Verify the signature in the Signature field of the message. If the signature does not verify, drop the message. Otherwise, continue.
2. Verify that the node is authorized to join the group. This SHOULD involve verifying certificate information in the Certificate field. If the node is not authorized, drop the message. Otherwise, continue.
3. Create a Heard List entry, called rcd, that holds the following information from the Join Message:

   rcd.PeerAddr = SenderAddr of the Join Message
   rcd.IFAddr = the interface on which the Join Message was received

4. Compare rcd with each entry in the Heard List. If there is a match, check if the SenderAddr matches the PeerOutAddr field of an entry in either the Pre-Key Group or the Peer Group. If so, set the heard-from boolean for this peer to true, then drop the message. Otherwise, continue.
5. If not already in the Heard List, add rcvd to the Heard List, and set the heard-from boolean for rcvd to true.

6. If the Join Message’s NumAddr field is 0, or if none of this node’s interface addresses is in the Join Message’s HandAddr, then stop processing this Join Message since there is not yet a peering relationship with the sender. Otherwise, continue.

At least one of this node’s interface addresses appears in the PeerAddr field of an entry in the Join Message’s HandAddr field.

7. For each interface on this node with a network address that matches a PeerAddr, get that entry from the HandAddr field. Call this entry heard, and do the following:

1. Add an Outgoing Policy called outpolicy to the IPsec SPD to require an ESP SA for Keying Messages sent to the new peer:

   ```
   outpolicy.source = heard.PeerAddr
   outpolicy.destination = heard.IPAddr
   outpolicy.protocol = the transport protocol used by LGKP
   outpolicy.port = the transport protocol port used by LGKP Keying Messages
   outpolicy.useProtocol = IPsec ESP
   ```

   NULL encryption MUST NOT be used. The SA lifetime (also known as the SA duration) value SHOULD NOT be set to infinity. This entry may be a duplicate from another Join Message; a good implementation of IPsec will handle the duplication or give an appropriate return code.

2. Add an Incoming Policy called inpolicy to the IPsec SPD to require an ESP SA for Keying Messages received from the new peer:

   ```
   inpolicy.source = rcvd.PeerAddr
   inpolicy.destination = rcvd.IPAddr
   inpolicy.protocol = the transport protocol used by LGKP
   inpolicy.port = the transport protocol port used by LGKP Keying Messages
   inpolicy.useProtocol = IPsec ESP
   ```

   NULL encryption MUST NOT be used. The SA lifetime (also known as the SA duration) value SHOULD NOT be set to infinity. Again, this entry may be a duplicate from another Join Message.

3. If the insertion of the policies is successful, add an entry (call it prekey) representing this peer to the Pre-Key Group:

   ```
   prekey.OutAddr = heard.PeerAddr
   prekey.PeerInAddr = heard.IPAddr
   prekey.PeerOutAddr = rcvd.PeerAddr
   prekey.InAddr = rcvd.IPAddr
   ```

4. Initiate an IKE exchange to set up an SA for the Outgoing Policy that was just entered. This MAY either be done directly or implicitly; waiting until a Keying Message is sent will implicitly initiate the establishment of the SA. (Implicit initiation will only establish those security associations that are needed.)

   Note that most IKE implementations may not work properly when peers have asymmetric links between them. Specifically, they may need modification to handle the case where prekey.PeerInAddr is not the same as prekey.PeerOutAddr.
3.4 Join Message Exchange Example

Consider the following relationship between two peers:

```
+---------- a ----------+    +---------- b ----------+
|                     |    |                     |
| Peer 1          >    | Peer 2          |
|                  +    +                     +
| +<-----------------> |            +-----------------> |
| c                  | d                     |
```

Here this is a one-way link from $a$ to $b$, and a two-way link between $c$ and $d$.

In the diagram, let HL be the Heard List and PK be the Pre-Key Group. Entries in HL are of the form $<PeerAddr, IEPAddr>$, and entries in PK are of the form $<OutAddr, PeerInAddr, PeerOutAddr, InAddr>$. Let $heard$ be the variable as defined in the Join Message Processing section.

The result of this exchange of Join Messages is that both Node 1 and Node 2 have two entries in their Pre-Key Groups. This is because there are really two peering relationships between Node 1 and Node 2: out $a$ to $b$ and back from $d$ to $c$, and out $c$ to $d$ and back from $d$ to $c$. The Join Procedure is just trying to find any one peering relationship — getting two will not harm the Keying Procedure at all. Recognizing that $c$ and $d$ form a bidirectional link can help make the use of resources more efficient, but it does not matter at all which link the Keying Procedure uses from Node 1 to Node 2.
4. Keying Procedure

The Keying Procedure defines how the Group Controller distributes the Peer Group Key to its peers. Once given to a peer, the Peer Group Key might be an IPsec key used to authenticate multicast transmissions from the Group Controller to the peer; however, the exact contents of the Peer Group Key and how it is used are at the discretion of the application.

When a Group Controller is initializing and before it has started the Keying Procedure with any peer, it MUST generate the initial Peer Group Key. The Group Controller also MAY periodically generate a new Peer Group Key and rekey the peers in its Peer Group. The Rekeying Timer is used to govern when peers in the Peer Group need to be rekeyed.

The Pre-Key Group is built using the Join Procedure described above. The Keying Procedure is begun for each peer as that peer joins the group. Once a peer completes the Keying Procedure, it is moved from the Group Controller’s Pre-Key Group into the Group Controller’s Peer Group.

Keying Messages MUST be protected using IPsec’s ESP. NULL encryption MUST NOT be used. The use of ESP MUST be required by the SPD entries created by the Join Procedure.

4.1 Keying Message Format

The Keying Message is used by the Group Controller to communicate the Peer Group Key to the members of the Pre-Key Group. The Keying Message format is given below:

```
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----------------------------------------------+
| Version | Flags | AddrType | KeyLength |
+-----------------------------------------------+
|        |       |          |            |
+-----------------------------------------------+
| SenderAddr |
+-----------------------------------------------+
|        |
+-----------------------------------------------+
| RespondFromAddr |
+-----------------------------------------------+
|        |
+-----------------------------------------------+
| RespondToAddr |
+-----------------------------------------------+
|        |
+-----------------------------------------------+
| Key |
+-----------------------------------------------+
```

All fields larger than one octet MUST be in network byte order (big-endian format).

Version (4 bits)

The Version field value indicates the version of LGKP in use. The version described by this document is 0, and is used in all LGKP messages. This field will be changed if a LGKP message format should change. If a Keying Message arrives with an unexpected version number, the message MUST be discarded.

Flags (4 bits)

The Flags field value specifies conditions in the sender of the Keying Message. One flag is defined:

```
value | Address Type
-----------------------------------------------
0x1   | KACK
```

KACK is the Key Acknowledge bit. This bit MUST only be set by a peer, and it declares that the peer has received the Peer Group Key.

AddrType (8 bits)

The AddrType field value specifies the type of the addresses carried in the SenderAddr, RespondFromAddr, and RespondToAddr field. The values for the AddrType field are given by the following table:
value | Address Type
---|---
0x00 | IPv4
0x01 | IPv6

SenderAddr (32 or 128 bits)
The SenderAddr field contains the address of the sender of this Keying Message. The address type is that described in AddrType field.

RespondFromAddr (32 or 128 bits)
The RespondFromAddr field contains the address from which the receiver should send a response Keying Message. The address type is that described in AddrType field. This field MUST be all zero bytes if KACK is set in the flags field.

RespondToAddr (32 or 128 bits)
The RespondToAddr field contains the address to which the receiver should send a response Keying Message. The address type is that described in AddrType field. This field MUST be all zero bytes if KACK is set in the flags field.

KeyLength (16 bits)
The KeyLength field value specifies the length of the Key field in bytes. Any padding is not included in this length. If KACK is set in the Flags field, the KeyLength field MUST be set to zero.

Key (variable)
The Key field contains the Peer Group Key being distributed. If the key is not aligned at the 4 octet boundary the field MUST be padded at the end with zero bytes to align on the next 32-bit boundary. If KACK is set in the Flags field, the Key field MUST be empty. If KACK is not set in the Flags field, the Key field MUST be filled.

### 4.2 Keying Message Transmission

The Join Procedure is considered complete for one peer when the Group Controller has placed that peer into the Pre-Keying Group, and the two peers have installed policies into their SPDs calling for point-to-point SAs to protect the Keying Messages that will pass between them.

When the Message Send Event occurs (see "Periodic Events" below), in addition to sending Join Messages on all of its interfaces, a Group Controller MUST unicast a Keying Message carrying the Peer Group Key to each member of its Pre-Key Group. All of the Keying Messages MUST be sent after all of the Join Messages have been sent.

Once a peer receives a Keying Message with the Peer Group Key, the peer MUST respond to the Group Controller with a Keying Message with the KACK bit set.

Note that, due to asymmetric links, the interface on which a Keying Message is sent may not necessarily be the interface on which the response Keying Message is received.

A Keying Message containing the Peer Group Key is created for every entry in the Pre-Key Group when the Message Send Event occurs. These entries represent peers that have not yet received or acknowledged the Peer Group Key. For each entry prekey in the Pre-Key Group, the Group Controller MUST transmit a Keying Message as follows:

1. Zero the Flags field.
2. Set the AddrType value appropriately for the address being used in the SenderAddr, RespondFromAddr, and RespondToAddr fields.
5. Set RespondToAddr to prekey.InAddr.
6. Unicast the Keying Message to the peer using the ESP SA that was set up with the Join Procedure. The message MUST be sent out the interface with address prekey.OutAddr and to the address prekey.PeerInAddr. The sender MUST use the ESP SA to protect the confidentiality of the Key.

A node sending Keying Messages for each Pre-Key Group entry prekey MAY suppress a Keying Message if two prekey entries have the same values in their OutAddr and PeerInAddr fields (that is, the Keying Messages would be sent to the same place from the same interface address). Suppression reduces the number of Keying Messages sent, and also may reduce the number of SAs created by IKE if the SA is created only on demand. The best choice, if available, is to use a Pre-Key Group entry that uses a symmetric link.

Upon receipt of a Keying Message containing a Peer Group Key, a peer transmits a Keying Message in response to the Group Controller as follows:

1. Set KACK in the Flags field.
2. Set the AddrType value appropriately for the address being used in the SenderAddr, RespondFromAddr, and RespondToAddr fields.
3. Copy the RespondFromAddr field from the original message to the SenderAddr field in this response.
4. Zero the RespondFromAddr, RespondToAddr, and KeyLength fields.
5. Omit any Key field.
6. Unicast the Keying Message to the peer using the ESP SA that was set up with the Join Procedure. The message MUST be sent out the interface whose address is RespondFromAddr from the original message, to the address RespondToAddr from the original message. The sender MUST use the ESP SA to protect the confidentiality of the Key.

4.3 Keying Message Processing

IPsec performs decryption and authentication of received Keying Messages.

When a peer receives a Keying Message without the KACK bit set, the Keying Procedure REQUIRES that the peer do the following:

- Pass the key information from the Key field to the application.
- Unicast a Keying Message with KACK set to the node that sent the Keying Message, as described above in Keying Message Transmission.

If the KACK bit is set in a received Keying Message, then the Keying Message is acknowledging that the peer has received the Peer Group Key, and the peer can be moved from the Pre-Key Group to the Peer Group. The Keying Procedure REQUIRES that the receiving node do the following:

- For each peer in the Pre-Key Group where peer.PeerOutAddr matches the received Keying Message’s SenderAddr, move peer to the Peer Group.

A received Keying Message MUST NOT be ignored, even if the information contained within has already been received, and the peer is already in the Peer Group.
4.4 Keying Message Exchange Example

A normal Keying Message exchange is shown below:

<table>
<thead>
<tr>
<th>Group Controller (GC)</th>
<th>Peer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peer is in Pre-Key Group</td>
<td>Keying Message ————&gt; Receive GC’s Key</td>
</tr>
<tr>
<td>Move Peer to Peer Group ————&gt; Keying Message (KACK)</td>
<td></td>
</tr>
</tbody>
</table>

5. Group Operations

Rekeying periodically replaces the security material with fresh material. For this, the Rekeying Timer is used. Independently from rekeying, peers sometimes need to be removed from a node’s records. There are two ways for a peer or potential peer to be removed from the state lists kept by LGKP. The first occurs when a peer has not been heard from for at least the duration of time from one Purge Event to the next. The second happens when a Certificate Revocation List (CRL) or equivalent is issued.

All of the group operations rely on the Periodic Events described in the next section.

5.1 Rekeying the Peer Group

Periodically, the Group Controller may generate a new Peer Group Key. The Rekeying and the Post-Rekeying Events control how rekeying is done.

Note that there is a relationship between the SA lifetime (also known as the SA duration) value for SAs using the Peer Group Key and the how often the Rekeying Event is scheduled. The SA lifetime MUST be longer than the period of the Rekeying Event period plus one Periodic Timer period.

5.2 Leaving the Peer Group

Peers are eventually removed from the Heard List, Pre-Key Group, or Peer Group when no Join Messages have been received from that peer after some amount of time according to the rules of the Purge Event.

5.3 Removing a Peer

When a CRL (or an equivalent instrument) is issued, a Group Controller MUST check the certificates offered by the nodes in its Heard List, Pre-Key Group, and Peer Group against the CRL. If the certificates of any nodes have been revoked, the corresponding entries MUST be removed from these lists. If a revoked certificate has been cached, that certificate MUST be removed from the cache. The Rekey Event MUST be scheduled for the next expiration of the Periodic Timer, and the Post-Rekeying Event MUST be scheduled for the next expiration after that.

When the CRL revokes the Group Controller’s own certificate, and the Group Controller has an alternate, the Group Controller MUST start using the alternate certificate, and the Rekey and Post-Rekey Events MUST be scheduled as above. If there are no alternate certificates left, then the Group Controller MUST cease LGKP operations until a valid certificate is obtained.

6. Periodic Events

The Periodic Timer is the only timer in LGKP. This single timer drives four events: the Message Send, Purging, Rekeying, and Post-Rekeying events. Each event is specified as a multiple of the Periodic Timer, called a factor. The Message Send Event has a factor of one; all other events
SHOULD have tunable factors but MUST obey the boundaries set forth in the descriptions below. The period of the Periodic Timer SHOULD be a tunable parameter, but MUST NOT be set longer than 30 seconds, and MUST have some degree of jitter (that is, the period MUST NOT be constant). The Periodic Timer MUST be longer than the amount of time it takes to install keying information into the SPD, and for IKE to establish an SA.

The following is a description of the events triggered by the Periodic Timer:

Message Send Event

The Message Send Event occurs at each expiration of the Periodic Timer. It controls the periodic emission of Join Messages on each of the Group Controller’s interfaces, and of Keying Messages for each peer in the Pre-Key Group. All Join Messages MUST be sent before any Keying Messages are sent. If scheduled for the same expiration of the Periodic Timer, the Post-Rekey Event MUST occur before the Message Send Event.

Purge Event

The Purge Event controls how long a peer remains in the Peer Group without hearing a Join Message from that peer. The factor for the Purge Event MUST NOT be less than 6. When the Periodic Timer expires and a Purge Event is scheduled, any entries in the Peer Group, the Pre-Key Group, and the Heard List that do not have the heard-from boolean set to true are removed from the lists. The heard-from boolean for all remaining entries on all lists is set to false.

Rekeying Event

The Rekeying Event controls how long a Peer Group Key is valid. The factor for the Rekeying Event MUST be greater than two. A factor of 0 SHOULD be used to turn off the Rekeying Timer, although turning off rekeying is not advised. When the Periodic Timer expires and a Rekeying Event is scheduled, all peers in the Peer Group are copied to the Pre-Key Group and a new Peer Group Key and SPI are generated. This new keying material MAY be installed into the SPD, but it is not yet used for authenticating new multicast messages sent from the Group Controller. The Rekeying Event MUST be scheduled before the Message Send Event. The old keying material will continue to be valid until the Post-Rekey Event.

Post-Rekeying Event

The Post-Rekeying Event controls when a node switches over to a new key. The Post-Rekeying Event occurs one Periodic Timer expiration after the Rekeying Timer, so its factor MUST be the same as the Rekeying Event, but it is offset from the Rekeying Event by one period. When the Periodic Timer expires, and the Post-Rekeying Event is scheduled, the Group Controller MUST inform the SAD to use the new keying material.

7. An Example Use

In the preceding sections, little has been said about the makeup of the Peer Group Key. This is intentional, because the contents of the Peer Group Key are of no importance to LGKP, and are application-specific. LGKP merely distributes this keying information.

That said, this section describes how an application might use LGKP to distribute information for another IPSec ESP security association. The SA will be used to authenticate messages using IPSec’s ESP protocol with NULL encryption. This is a routing application, and a Group Controller reaches its peer group using the all-routers multicast address, as allocated by IANA for use in IGMP (the address is 224.0.0.1 in IPv4).

First, the application generates the material for the Peer Group Key, which will be passed as the
Key field in LGKP Keying Messages. The application gives the following structure to this field:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Reserved</td>
<td>AddrType</td>
<td>LifeType</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>AppSenderAddr</td>
<td>:</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>AppSenderSPI</td>
<td>:</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>Lifetime</td>
<td>:</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>AlgType</td>
<td>AppKeyLength</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>AppKey</td>
<td>:</td>
<td>:</td>
<td></td>
</tr>
</tbody>
</table>

All fields larger than one octet are in network byte order (big-endian format).

Reserved (8 bits)
The Reserved fields are reserved for future use.

AddrType (8 bits)
The AddrType field value specifies the type of the address carried in the AppSender-Addr field. The values for the AddrType field are given by the following table:

<table>
<thead>
<tr>
<th>value</th>
<th>Address Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>IPv4</td>
</tr>
<tr>
<td>0x01</td>
<td>IPv6</td>
</tr>
</tbody>
</table>

LifeType (16 bits)
The LifeType field value specifies the granularity of the value in the Lifetime field, as defined in [RFC2407] or its successor.

AppSenderAddr (32 or 128 bits)
The AppSenderAddr field contains the address that the application will be multicasting authenticated messages from.

AppSenderSPI (32 bits)
The AppSenderSPI field holds the IPSec Security Parameter Index (SPI) that the sending application associates with the SA described in this message.

Lifetime (32 bits)
The Lifetime field value specifies the time-to-live for the SA for the given LifeType.

AlgType (16 bits)
The AlgType field value specifies the authentication algorithm with which the AppKey MUST be used. The values for the AlgType are given by the following table:

<table>
<thead>
<tr>
<th>value</th>
<th>Address Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>RESERVED</td>
</tr>
<tr>
<td>1</td>
<td>HMAC-MD5</td>
</tr>
<tr>
<td>2</td>
<td>HMAC-SHA</td>
</tr>
<tr>
<td>3</td>
<td>DES-MAC</td>
</tr>
<tr>
<td>4</td>
<td>KDFK</td>
</tr>
</tbody>
</table>

These values are defined in [RFC2407] and MUST mirror any updates to the values in that document.
AppKeyLength (16 bits)

The AppKeyLength field value specifies the length of the AppKey field in bytes. Any padding is not included in this length.

AppKey (variable)

The Key field contains the real SA keying material. If the key is not aligned at the 4 octet boundary the field MUST be padded at the end with zero bytes to align on the next 32-bit boundary.

Then, LGKP is given this Peer Group Key, the list of local interfaces, and is told to issue Join Messages to the all-routers address.

As LGKP runs on its own, the application installs this SA locally, with a policy that requires its use for the application's transmissions to the all-routers address. Application packets are then wrapped in this ESP SA when multicast. Peers that receive these packets and hold the sender's SA can authenticate them, otherwise they are dropped.

In this example, the application has distributed a symmetric group key that, when used, provides group authentication but not sender authentication. That is, a message authenticated with the key proves that a node which knows the key sent it (i.e., any node in the group), but not which node sent it. However, the node which should be sending with the key could detect the problem and report it.

Most notable about this example is that the sender has chosen the SPI that it will use when sending to its peers. It is unusual in an IPsec environment to have a sender choose an SPI; normal unicast IPsec operations have the receiver choose the SPI, because [RFC2401] says that the 3-tuple

<SPI, receiver, protocol>

is what uniquely identifies the SA in the network.

However, in this multicast setting with multiple receivers, it would be difficult or wrong to get all receivers to agree on one SPI. So, we have the sender pick it, but it may conflict with one already chosen by one of the receivers for one of its unicast SAs.

The rest of the solution is to have the IPsec implementations of the nodes in this example replace receiver with sender in the tuple when looking up an SA, but only when the packet's destination was a broadcast or multicast address. Finally, a sender-picked SPI must also yield a

<SPI, sender, protocol>

3-tuple that uniquely identifies the SA in the network.

This small twist arguably corrects a problem with [RFC2401]. Curiously, this change also enables the ultimate multicast "one or more senders, one or more receivers" case, where a single shared key is negotiated for all to use. Each sender picks its own SPI to associate with the shared SA and communicates it to the group, and receivers' tuple-to-SA mappings become many-to-one.

8. Security Considerations

An important security consideration in LGKP involves how the Pre-Key Group is maintained. Each entry in this group calls for IKE to establish point-to-point SAs with a host.

Normally, only peers that are both authorized to join the group (per the certificates in their Join Messages) and up (because Join Messages have been heard from them recently) are ever in the Pre-Key Group. However, a Join Message from a peer can be replayed. When the peer is up, this is unimportant, since the peer will want to join the group and IKE will be able to establish the point-to-point SAs with it.

When the peer is actually down, its replayed Join messages will prevent it from being purged
from the Pre-Key Group, and IKE, triggered by the Keying Procedure, will repeatedly attempt (and fail) to establish point-to-point SAs, possibly consuming computation resources.

We believe that this problem is best handled on a per-implementation basis. Some implementations may choose to ignore this problem. Others may choose to implement a "penalty box" for peers that remain in the Pre-Key Group for too long without moving to the Peer Group.

We did not want to expand Join Messages to include replay protection with some notion of timeliness. This would require nodes to generate and validate signatures on Join Messages much more often (i.e., it would preclude nodes from caching their own signatures and the signatures of their neighbors.)

A peer entry can always be definitively removed from all Pre-Key Groups by issuing a CRL (or an equivalent instrument) revoking the peer's key or authority to be a group member.

Another security concern is whether or not a group is rekeyed when a peer leaves the group (either because it was Purged, or a CRL revoked its group membership credentials). This is application dependent. An application that distributes a key for confidentiality should rekey one that distributes a key for authentication may choose not to. LGKP implementations SHOULD allow the application to trigger a Rekeying event at any time.

Finally, the point-to-point SAs negotiated by IKE to protect the Keying Messages must be of a certain quality. They MUST be ESP SAs using an encryption algorithm at least as strong as the algorithm the application intends to ultimately use with the Peer Group Key. The ESP NULL encryption algorithm MUST NOT be used. The point-to-point SAs SHOULD have a lifetime or duration that is less than infinity.

It is also possible that the Join Procedure will add redundant IPSec policies and point-to-point SAs. This can happen if a peer crashes and then comes back up before the Group Controller can purge that peer. It is assumed that the IPSec implementation will use the lifetime value or some other appropriate means to purge the SPD and SAD of old policies and SAs.

9. References


10. Authors’ Address

BBN Technologies
10 Moulton St
Cambridge MA 02138
## CONTENTS

1. Introduction ........................................................................................................... 1  
   1.1 Status of this Memo ............................................................................................. 2  
   1.2 Definitions .......................................................................................................... 2  
   1.3 The Group Controller and its Peer Group .......................................................... 2  
   1.4 Addresses ............................................................................................................ 3  

2. Protocol Overview ................................................................................................... 3  
   2.1 Heard List ............................................................................................................ 4  
   2.2 Pre-Key Group and Peer Group .......................................................................... 4  
   2.3 Message Transport ............................................................................................. 4  

3. Join Procedure ......................................................................................................... 5  
   3.1 Join Message Format ......................................................................................... 5  
   3.2 Join Message Transmission ............................................................................... 8  
   3.3 Join Message Processing ................................................................................... 8  
   3.4 Join Message Exchange Example ..................................................................... 10  

4. Keying Procedure ..................................................................................................... 11  
   4.1 Keying Message Format .................................................................................... 11  
   4.2 Keying Message Transmission ......................................................................... 12  
   4.3 Keying Message Processing ............................................................................... 13  
   4.4 Keying Message Exchange Example .................................................................. 14  

5. Group Operations ..................................................................................................... 14  
   5.1 Rekeying the Peer Group .................................................................................. 14  
   5.2 Leaving the Peer Group ..................................................................................... 14  
   5.3 Removing a Peer ............................................................................................... 14  

6. Periodic Events ........................................................................................................ 14  

7. An Example Use ....................................................................................................... 15  

8. Security Considerations ......................................................................................... 17  

9. References .............................................................................................................. 18  

10. Authors’ Address ................................................................................................... 19