Internet Draft
IETF MSEC WG
Expires: December 08, 2004
Category: Informational

Mark Baugher (Cisco)
Ran Canetti (IBM)
Lakshminath Dondeti (Nortel)
Fredrik Lindholm (Ericsson)

June 09, 2004

MSEC Group Key Management Architecture

<draft-ietf-msec-gkmarch-08.txt>

Status of this Memo

Abstract

This document defines the common architecture for Multicast Security (MSEC) key management protocols that support a variety of application, transport, and network layer security protocols. It also defines the group security association (GSA), and describes the key management protocols that help establish a GSA. The framework and guidelines described in this document allow for a modular and flexible design of group key management protocols for a variety of different settings that are specialized to applications needs. MSEC key management protocols may be used to facilitate secure one-to-many, many-to-many, or one-to-one communication.

Comments on this document should be sent to msec@securemulticast.org.

Table of Contents

1.0 Introduction: Purpose of this Document

Group and multicast applications have diverse requirements in IP networks [TAXONOMY]. Their key management requirements - briefly reviewed in Section 2.0 - include support for internetwork, transport, and application-layer protocols. Some applications may achieve simpler operation by running key-management messaging over a pre-established secure channel (e.g., TLS, IPsec). Other security protocols may benefit from a key management protocol that can run over already deployed session initiation or management protocol (e.g., SIP or RTSP). Finally, some may benefit from a light-weight key management protocol that finishes in fewest round trips. For these reasons, different application, transport, and internetwork-layer data security protocols (e.g., SRTP [RFC3711] and IPsec [RFC2401]) may benefit from using different group key management systems. The purpose of this document is to define a common architecture and design for group key-management protocols for internet, transport, and application services.

The common architecture for group key management is called the MSEC key management architecture and is based on the group control or key server model developed in GKMP [RFC2094] and assumed by group key management algorithms such as LKH [RFC2627], OFT [OFT], and MARKS [MARKS]. There are other approaches that are not considered in this architecture such as the highly distributed Cliques group key management protocol [CLIQUES] and broadcast key management schemes [FN93, Wool]. MSEC (Multicast Security) key management may in fact be complementary to other group key management designs, but these are not considered in this document. The integration of MSEC group key management with Cliques, broadcast key management and other group key systems is not considered in this document.

Indeed, key-management protocols are difficult to design and validate. The common architecture described in this document eases this burden by defining common abstractions and overall design that can be specialized for different uses.

This document builds on and extends the Group Key Management Building Block document of the IRTF SMuG research group [GKMBB] and is part of the MSEC document roadmap. The MSEC architecture [MSEC-Arch] is a reference for a complete multicast or group security architecture, of which key management is a component.

The rest of this document is organized as follows. Section 2 discusses the security, performance and architectural requirements for a group key management protocol. Section 3 presents the overall architectural design principles. Section 4 describes the registration

protocol in detail and Section 5 does the same for rekey protocol. Section 6 considers the interface to the Group Security Association

(GSA). Section 7 reviews the scalability issues for group key management protocols and Section 8 discusses security considerations.

2.0 Requirements of a Group Key Management Protocol

A group key management protocol supports protected communication between members of a secure group. A secure group is a collection of principals, called members, who may be senders, receivers or both receivers and senders to other members of the group. (Note that group membership may vary over time.) A group key management protocol helps to ensure that only members of a secure group gain access to group data (by gaining access to group keys) and can authenticate group data. The goal of a group key management protocol is to provide legitimate group members with the up-to-date cryptographic state they need for their secrecy and authenticity requirements.

Multicast applications, such as video broadcast and multicast file transfer, typically have the following key-management requirements (see also [TAXONOMY]). Note that the list is neither applicable to all applications, nor exhaustive.

1. The group members receive security associations including encryption keys, authentication/integrity keys, cryptographic policy that describes the keys, and attributes such as an index for referencing the security association (SA) or particular objects contained in the SA.

2. In addition to the policy associated with group keys, the group owner or the Group Controller and Key Server (GCKS) may define and enforce group membership, key management, data security and other policies that may or may not be communicated to the membership-at-large.

3. Keys will have a predetermined lifetime and may be periodically refreshed.

4. Key material should be delivered securely to members of the group so that they are secret, integrity-protected and can be verified as coming from an authorized source.

5. The key-management protocol should be secure against replay attacks and Denial of Service(DoS) attacks (see the Security Considerations section of this memo).

6. The protocol should facilitate addition and removal of group members so that members who are added may optionally be denied access to the key material used before they joined the group, and that removed members lose access to the key material

following their departure.

7. The protocol should support a scalable group rekey operation without unicast exchanges between members and a group controller/key server, to avoid overwhelming a GCKS managing a large group.

8. The protocol should be compatible with the infrastructure and performance needs of the data-security application, such as IPsec security protocols, AH and ESP, and/or application-layer security protocols, such as SRTP.

9. The key management protocol should offer a framework for replacing or renewing transforms, authorization infrastructure and authentication systems.

10. The key management protocol should be secure against collusion among excluded members and non-members. Specifically, collusion must not result in attackers gaining any additional group secrets than each of them individually are privy to. In other words, combining the knowledge of the colluding entities must not result in revealing additional group secrets.

12. The key management protocol should provide a mechanism to securely recover from a compromise of some or all of the key material.

13. Key management protocols may need to address real-world deployment issues such as NAT-traversal and may need to interface with legacy authentication mechanisms already deployed.

In contrast to typical unicast key and SA negotiation protocols such as TLS and IKE, group key management protocols provide SA and key download capability. This feature may be useful for point-to-point communication as well. Thus, a group key management protocol may also be useful to unicast applications. In other words, group key management protocols may be used for protecting multicast communications, or unicast communications between members of a secure group. Secure sub-group communication is also plausible using the group SA.

There are other requirements for small group operation where there will be many senders or in which all members may potentially be senders. In this case, the group setup time may need to be optimized to support a small, highly interactive group environment [RFC2627].

The current key management architecture covers secure communication in large single-sender groups, such as source-specific multicast groups. Scalable operation to a range of group sizes is also a

desirable feature, and a better group key management protocol will

support large, single-sender groups as well as groups that have many senders. It may be that no single key management protocol can satisfy the scalability requirements of all group-security applications.

In addition to these requirements, it is useful to emphasize two non-requirements, namely, technical protection measures (TPM) [TPM] and broadcast key management. TPM are used for such things as copy protection by preventing the user of a device to get easy access to the group keys. There is no reason why a group key management protocol cannot be used in an environment where the keys are kept in a tamper-resistant store using various types of hardware or software to implement TPM. However, for simplicity, the MSEC key management architecture described in this document considers design for technical protection measures out of scope.

The second non-requirement is broadcast key management where there is no back channel [FN93, JKKV94] or where the device is not on a network, such as a digital videodisk player. We assume IP network operation where there is two-way communication, however asymmetric, and that authenticated key-exchange procedures can be used for member registration. It is possible that broadcast applications can make use of a one-way Internet group key management protocol message, and a one-way rekey message as described below.

3.0 Overall Design of the Group Key Management Architecture

This section describes the overall structure of a group key management protocol. The design is based upon a group controller model [RFC2093, RFC2094, RFC2627, OFT, GSAKMP, and RFC3547] with a single group owner as the root-of-trust. The group owner designates a group controller for member registration and GSA rekeying.

3.1 Overview

The main goal of a group key management protocol is to securely provide the group members with an up-to-date security association (SA), which contains the needed information for securing group communication (i.e., the group data). We call this SA the Data SA. In order to obtain this goal, the Group Key Management Architecture defines the following protocols.

(1) Registration protocol.
    ======================
This is a unicast protocol between the group controller/key server (GCKS) and a joining group member. In this protocol, the GCKS and joining member mutually authenticate each other. If the authentication succeeds and the GCKS finds that the joining member is authorized, then the GCKS supplies the joining member with the

following information:

(a) Sufficient information to initialize the Data SA within the joining member. This information is given only in the case that the group security policy calls for initializing the Data SA at registration, instead of or in addition to as part of the rekey protocol.

(b) Sufficient information to initialize a Rekey SA within the joining member (see more details about this SA below). This information is given only in case that the group security policy calls for using a rekey protocol.

The registration protocol must ensure that the transfer of information from GCKS to member is done in an authenticated and confidential manner over a security association. We call this SA the Registration SA. A complementary de-registration protocol serves to explicitly remove Registration SA state. Members may choose to delete Registration SA state on their own volition.

(2) Rekey protocol.
==========

A GCKS may periodically update or change the Data SA, by sending rekey information to the group members. Rekey messages may result from group membership changes, change in group security policy, the creation of new traffic-protection keys (TPKs, see next section) for the particular group, or from key expiration. Rekey messages are protected by the Rekey SA, which is initialized in the registration protocol. They contain information for updating the Rekey SA and/or the Data SA. Rekey messages can be sent via multicast to group members or unicast from the GCKS to a particular group member.

Note that there are other means for managing (e.g. expiring or refreshing) the Data SA without interaction between the GCKS and the members. For example in MARKS [MARKS], the GCKS pre-determines TPKs for different periods in the lifetime of the secure group and distributes keys to members based on their membership periods. Alternative schemes such as the GCKS disbanding the secure group and starting a new group with a new Data SA are also possible, although this type of operation is typically limited to small groups.

Rekey messages are authenticated using one of the two following options:

- The first option is to use source authentication [TAXONOMY], that is to enable each group member to verify that a rekey message originates with the GCKS and none other.

- The second option is to use only group-based authentication using a symmetric key. Members can only be assured that the

rekey messages originated within the group. Therefore, this is

applicable only when all members of the group are trusted not to impersonate the GCKS. Group authentication for rekey messages is typically used when public-key cryptography is not suitable for the particular group.

The rekey protocol ensures that all members receive the rekey information in a timely manner. In addition, the rekey protocol specifies mechanisms for the parties to contact the GCKS and re-synch in case that their keys expired and an updated key has not yet been received. The rekey protocol for large-scale groups offers mechanisms to avoid implosion problems and ensure the needed reliability in its delivery of keying material.

Rekey messages are protected by a Rekey SA, which is established by the registration protocol, and updated using rekey protocol. When a member leaves the group, it destroys its local copy of the GSA. Use of a de-registration message may be an efficient mechanisms for a member to inform the GCKS that it has destroyed the SAs, or is about to destroy them. Such a message may prompt the GCKS to cryptographically remove the member from the group (i.e., to prevent the member from having access to future group communication). In large-scale multicast applications, however, de-registration has the potential to cause implosion at the GCKS.

3.2 Detailed Description of the GKM Architecture

Figure 1 depicts the overall design of a GKM protocol. Each group member, sender or receiver, uses the registration protocol to get authorized, authenticated access to a particular Group, its policies, and its keys. The two types of group keys are the key encryption keys (KEKs) and the traffic encryption keys (TEKs). For group authentication of rekey messages or data, key integrity keys or traffic integrity keys may be used as well. We use the term protection keys to refer to both integrity keys and the encryption keys. For example, the term traffic protection key (TPK) is used to denote the combination of a TEK and a traffic integrity key, or key material used to generate them.

The KEK may be a single key that protects the rekey message, typically containing a new Rekey SA (containing a KEK) and/or Data SA (containing a TEK). A Rekey SA may also contain a vector of keys that are part of a group key membership algorithm [RFC2627, OFT, TAXONOMY, SD1, SD2]. The TPKs are used by the data security protocol to protect streams, files, or other data sent and received by the data security protocol. Thus the registration protocol and/or the rekey protocol establish the KEK(s) and/or the TPKs.

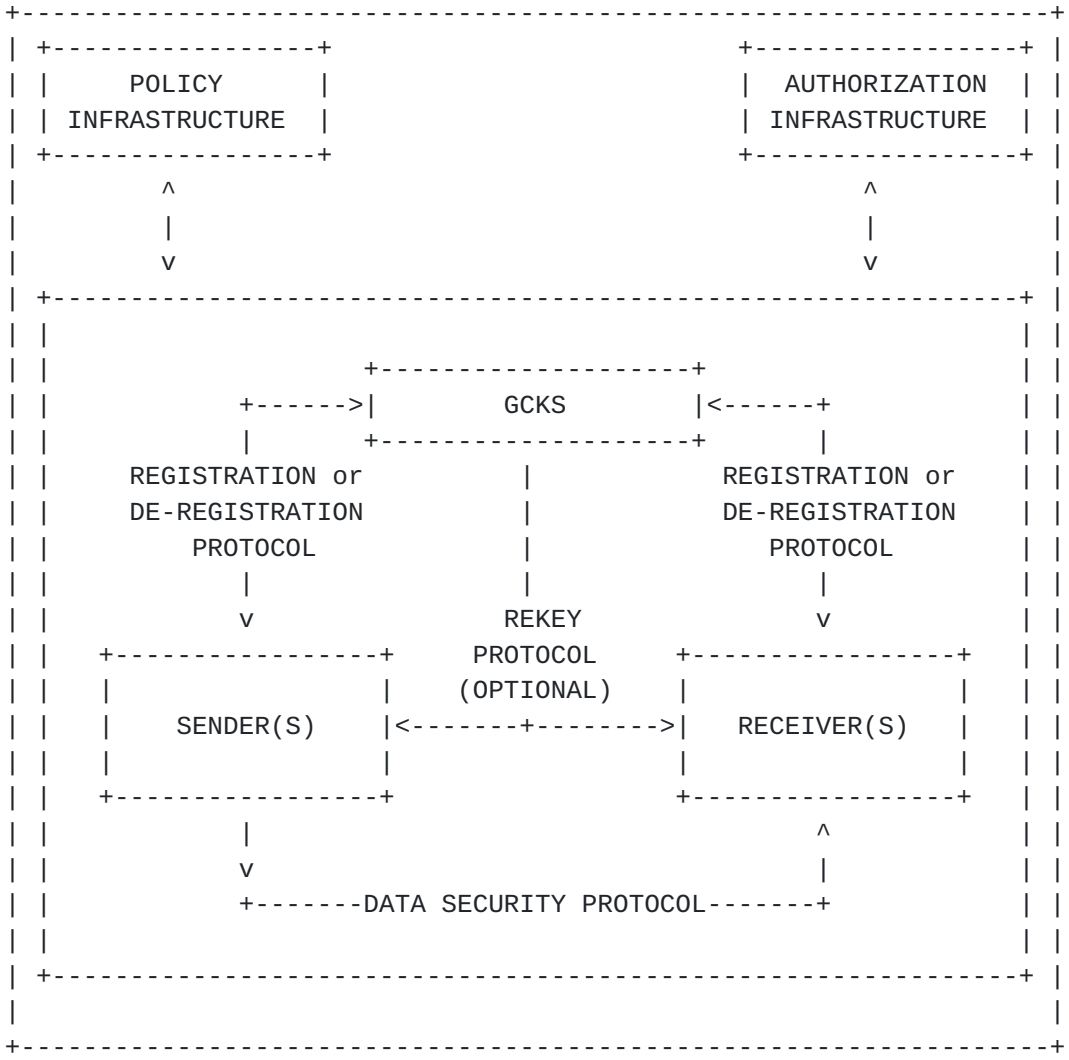

FIGURE 1: Group Security Association Model

There are a few, distinct outcomes to a successful registration Protocol exchange.

- If the GCKS uses rekey messages, then the admitted member receives the Rekey SA. The Rekey SA contains the groups rekey policy (note that not all of the policy need to be revealed to members), and at least a group KEK. In addition, the GCKS may send a group key integrity key, and if the group uses a group key management algorithm, a set of KEKs (or key material used to derive the KEKs) according to the particular algorithm.

- If rekey messages are not used for the Group, then the admitted member will receive TPKs (as part of the Data

Security SAs) that are passed to the members Data Security Protocol (as IKE does for IPsec).

- The GCKS may pass one or more TPKs to the member even if rekey messages are used, for efficiency reasons according to group policy.

The GCKS creates the KEK and TPKs and downloads them to each member - as the KEK and TPKs are common to the entire group. The GCKS is a separate, logical entity that performs member authentication and authorization according to the group policy that is set by the group owner. The GCKS may present a credential to the group member that is signed by the group owner so the member can check the GCKSs authorization. The GCKS, which may be co-located with a member or be a separate physical entity, runs the rekey protocol to push rekey messages containing refreshed KEKs, new TPKs, and/or refreshed TPKs to members. Note that some group key management algorithms refresh any of the KEKs (potentially), whereas others only refresh the group KEK.

Alternatively, the sender may forward rekey messages on behalf of the GCKS when it uses a credential mechanism that supports delegation. Thus, it is possible for the sender (or other members) to source keying material - TPKs encrypted in the Group KEK - as it sources multicast or unicast data. As mentioned above, the rekey message can be sent using unicast or multicast delivery. Upon receipt of a TPK (as part of a Data SA) from a rekey message or a registration protocol exchange, the members group key management functional block will provide the new or updated security association (SA) to the data security protocol to protect the data sent from sender to receiver.

The Data SA protects the data sent on the arc labeled DATA SECURITY PROTOCOL shown in Figure 1. A second SA, the Rekey SA, is optionally established by the key-management protocol for rekey messages as shown in Figure 1 by the arc labeled REKEY PROTOCOL. The rekey message is optional because all keys, KEKs and TPKs, can be delivered by the registration protocol exchanges shown in Figure 1, and those keys may not need to be updated. The registration protocol is protected by a third, unicast, SA between the GCKS and each member; this is called the Registration SA. There may be no need for the Registration SA to remain in place after the completion of the registration protocol exchanges. The de-registration protocol may be used when explicit teardown of the SA is desirable (such as when a phone call or conference terminates). The three SAs compose the GSA. Only one SA is optional and that is the Rekey SA.

Figure 1 shows two blocks that are external to the group key management protocol: The policy and authorization infrastructures are discussed in Section 6.1. The Multicast Security Architecture document further clarifies the SAs and their use as part of the

complete architecture of a multicast security solution [MSEC-Arch].

3.3 Properties of the Design

The design of Section 3.2 achieves scalable operation by (1) allowing the de-coupling of authenticated key exchange in a registration protocol from a rekey protocol, (2) allowing the rekey protocol to use unicast push or multicast distribution of group and data keys as an option, (3) allowing all keys to be obtained by the unicast registration protocol, and (4) delegating the functionality of the GCKS among multiple entities, i.e., to permit distributed operation of the GCKS.

High-capacity operation is obtained by (1) amortizing computationally-expensive asymmetric cryptography over multiple data keys used by data security protocols, (2) supporting multicast distribution of symmetric group and data keys, and (3) supporting key revocation algorithms such as LKH [RFC2627, OFT, SDR] that allow members to be added or removed at logarithmic rather than linear space/time complexity. The registration protocol may use asymmetric cryptography to authenticate joining members and optionally establish the group KEK. Asymmetric cryptography such as Diffie-Hellman key agreement and/or digital signatures are amortized over the life of the group KEK: A Data SA can be established without the use of asymmetric cryptography - the TPKs are simply encrypted in the symmetric KEK and sent unicast or multicast in the rekey protocol.

The design of the registration and rekey protocols is flexible. The registration protocol establishes either a Rekey SA or one or more Data SAs or both types of SAs. At least one of the SAs is present (otherwise, there is no purpose to the Registration SA). The Rekey SA may update the Rekey SA, or establish or update one or more Data SAs. Individual protocols or configurations may take advantage of this flexibility for efficient operation.

3.4 Group Key Management Block Diagram

In the block diagram of Figure 2, group key management protocols run between a GCKS and member principal to establish a Group Security Association (GSA). The GSA consists of a Data SA, an optional Rekey SA, and a Registration SA. The GCKS may use a delegated principal, such as the sender, which has a delegation credential signed by the GCKS. The Member of Figure 2 may be a sender or receiver of multicast or unicast data. There are two functional blocks in Figure 2 labeled GKM, and there are two arcs between them depicting the group key-management registration (reg) and rekey (rek) protocols. The message exchanges are the GSA establishment protocols, which are the registration protocol and the rekey protocol described above.

Figure 2 shows that a complete group-key management functional

specification includes much more than the message exchange. Some of these functional blocks and the arcs between them are peculiar to an

operating system (OS) or vendor product, such as vendor specifications for products that support updates to the IPsec Security Association Database (SAD) and Security Policy Database (SPD) [RFC2367]. Various vendors also define the functions and interface of credential stores, CRED in Figure 2.

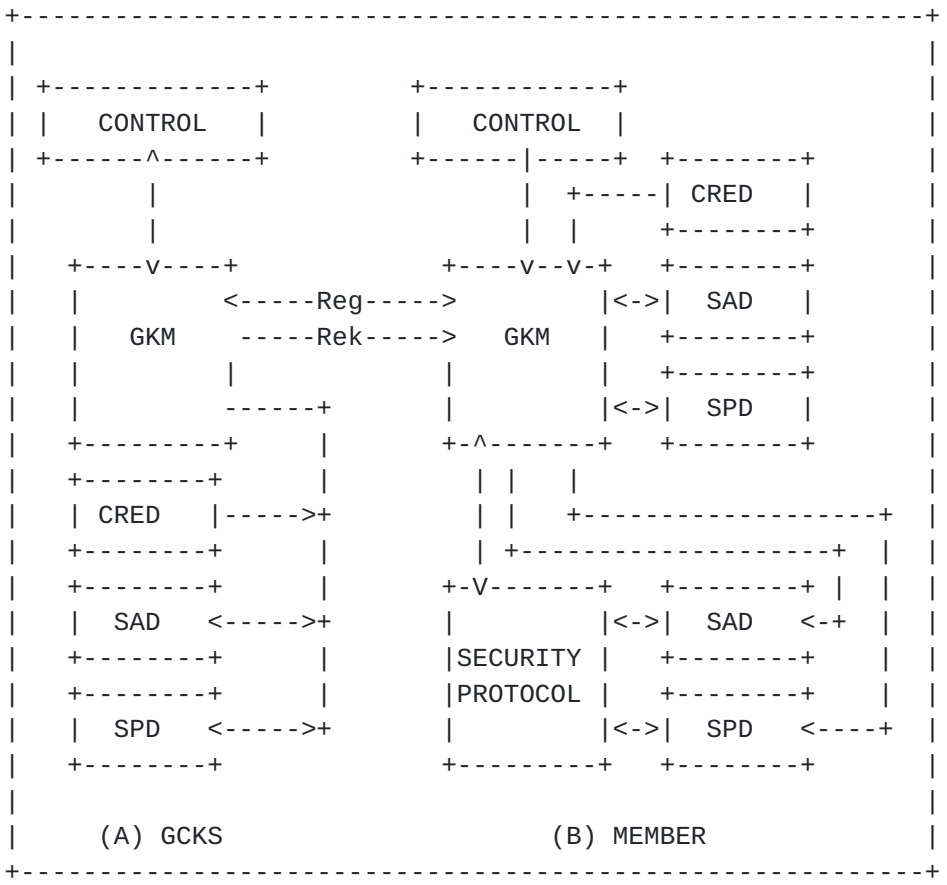

Figure 2: Group key management block diagram for a host computer

The CONTROL function directs the GCKS to establish a group, admit a member, or remove a member, or it directs a member to join or leave a group. CONTROL includes authorization, which is subject to group policy [GSPT], but how this is done is specific to the GCKS implementation. For large-scale multicast sessions, CONTROL could perform session announcement functions to inform a potential group member that it may join a group or receive group data (e.g. a stream of file transfer protected by a data security protocol). Announcements notify group members to establish multicast SAs in advance of secure multicast data transmission. Session Description Protocol (SDP) is one form that the announcements might take [RFC2327]. The announcement function may be implemented in a session-directory tool, an electronic program guide (EPG), or by other means. The Data Security or the announcement function directs

group key management using an application-programming interface (API), which is peculiar to the host OS in its specifics. A generic

API for group key management is for further study, but this function is necessary to allow Group (KEK) and Data (TPKs) key establishment to be done in a way that is scalable to the particular application. A GCKS application program will use the API to initiate the procedures to establish SAs on behalf of a Security Protocol in which members join secure groups and receive keys for streams, files or other data.

The goal of the exchanges is to establish a GSA through updates to the SAD of a key-management implementation and particular Security Protocol. The data security protocol of Figure 2 may span internetwork and application layers or operate at the internetwork layer, such as AH and ESP.

4.0 Registration protocol

The design of the registration protocol is flexible, and can support different application scenarios. The chosen registration protocol solution reflects the specific requirements of specific scenarios. In principle, it is possible to base a registration protocol on any secure-channel protocol, such as IPsec and TLS, which is the case in tunneled GSAKMP [tGSAKMP]. GDOI [RFC3547] reuses IKE Phase 1 as the secure channel to download Rekey and/or Data SAs. Other protocols, such as MIKEY and GSAKMP, use authenticated Diffie-Hellman exchanges similar to IKE Phase 1, but specifically tailored for key download to achieve efficient operation. We discuss the design of a registration protocol in detail in the rest of this section.

4.1 Registration protocol via Piggybacking or Protocol Reuse

Some registration protocols need to tunnel through a data-signaling protocol to take advantage of already existing security functionality, and/or to optimize the total session setup time. For example, a telephone call has strict bounds for delay in setup time. It is not feasible to run security exchanges in parallel with call setup since the latter often resolves the address: Call setup must complete before the caller knows the address of the callee. In this case, it may be advantageous to tunnel the key exchange procedures inside call establishment [H.235, MIKEY] so both can complete (or fail, see below) at the same time.

The registration protocol has different requirements depending on the particular integration/tunneling approach. These requirements are not necessarily security requirements, but will have an impact on the chosen security solution. For example, the security association will certainly fail if the call setup fails in the case of IP telephony.

Conversely, the registration protocol imposes requirements on the protocol that tunnels it. In the case of IP telephony, the call setup usually will fail when the security association is not successfully established. In the case of video-on-demand, protocols such as RTSP that convey key management data will fail when a needed security association cannot be established.

Both GDOI and MIKEY use this approach, but in different ways. MIKEY can be tunneled in SIP and RTSP. It takes advantage of the session information contained in these protocols and the possibility to optimize the setup time for the registration procedure. SIP requires that a tunneled protocol must use at most one roundtrip (i.e. two messages). This is also desirable requirement from RTSP as well.

The GDOI approach takes advantage of the already defined ISAKMP phase 1 exchange [RFC2409], and extends the phase 2 exchange for the registration. The advantage here is the reuse of a successfully deployed protocol and the code base, where the defined phase 2 exchange is protected by the SA created by phase 1. GDOI also inherits other functionality of the ISAKMP, and thus it is readily suitable for running IPsec protocols over IP multicast services.

4.2 Properties of Alternative registration Exchange Types

The required design properties of a registration protocol have different tradeoffs. A protocol that provides perfect forward secrecy and identity protection trades performance or efficiency for better security, while a protocol that completes in one or two messages may trade security functionality (e.g. identity protection) for efficiency.

Replay protection generally uses either a timestamp or a sequence number. The first requires synchronized clocks, while the latter requires that it is possible to keep state. In a timestamp-based protocol, a replay cache is needed to store the authenticated messages (or the hashes of the messages) received within the allowable clock skew. The size of the replay cache depends on the number of authenticated messages received during the allowable clock skew. During a DoS attack, the replay cache might become overloaded. One solution is to over-provision the replay cache. However, this may lead to a large replay cache. Another solution is to let the allowable clock skew be changed dynamically during runtime. During a suspected DoS attack, the allowable clock skew is decreased so that the replay cache becomes manageable.

A challenge-response mechanism (using Nonces) obviates the need for synchronized clocks for replay protection when the exchange uses

three or more messages [MVV].

Additional security functions become possible as the number of allowable messages in the registration protocol increase. ISAKMP offers identity protection, for example, as part of a six-message exchange. With additional security features, however, comes added complexity: Identity protection, for example, not only requires additional messages, but may result in DoS vulnerabilities since authentication is performed in a late stage of the exchange after resources already have been devoted.

In all cases, there are tradeoffs with the number of message exchanged, the desired security services, and the amount of infrastructure that is needed to support the group key management service. Whereas protocols that use two or even one-message setup have low latency and computation requirements, they may require more infrastructure such as secure time or offer less security such as the absence of identity protection. What tradeoffs are acceptable and what are not is very much dictated by the application and application environment.

4.3 Infrastructure for Alternative registration Exchange Types

The registration protocol may need external infrastructures to be able to handle authentication and authorization, replay protection, protocol-run integrity, and potentially other security services such as secure, synchronized clocks. For example, authentication and authorization may need a PKI deployment (with either authorization-based certificates or a separate management for this) or may be handled by using AAA infrastructure. Replay protection using timestamps requires an external infrastructure or protocol for clock synchronization.

However, external infrastructures may not always be needed, if for example pre-shared keys are used for authentication and authorization; this may be the case if the subscription base is relatively small. In a conversational multimedia scenario (e.g., a VoIP call between two or more people), it may very well be the end user who handles the authorization by manually accepting/rejecting the incoming calls. Thus, infrastructure support may not be required in that case.

4.4 De-registration Exchange

The session-establishment protocol (e.g., SIP, RTSP) that conveys a registration exchange often has a session-disestablishment protocol such as RTSP TEARDOWN [RFC2326] or SIP BYE [RFC2543]. The session-disestablishment exchange between endpoints offers an opportunity to signal the end of the GSA state at the endpoints. This exchange

need only be a uni-directional notification by one side that the GSA is to be destroyed. For authentication of this notification, we may

use a proof-of-possession of the group key(s) by one side to the other. Some applications benefit from acknowledgement in a mutual, two-message exchange signaling disestablishment of the GSA concomitant with disestablishment of the session, e.g., RTSP or SIP session. In this case, a two-way proof-of-possession might serve for mutual acknowledgement of the GSA disestablishment.

5.0 Rekey protocol

The group rekey protocol is for transport of keys and SAs between a GCKS and the members of a secure communications group. The GCKS sends rekey messages to update a Rekey SA, or initialize/update a Data SA or both. Rekey messages are protected by a Rekey SA. The GCKS may update the Rekey SA when group membership changes or when KEKs or TPKs expire. Recall that KEKs correspond to a Rekey SA and TPKs correspond to a Data SA.

The following are some desirable properties of the rekey protocol:

- Rekey protocol ensures that all members receive the rekey information in a timely manner.
- Rekey protocol specifies mechanisms for the parties involved, to contact the GCKS and re-sync when their keys expire and no updates have been received.
- Rekey protocol avoids implosion problems and ensures the needed reliability in delivering Rekey information.

We further note that the rekey protocol is primarily responsible for scalability of the group key management architecture. Hence it is imperative that we provide the above listed properties in a scalable manner. Note that solutions exist in the literature (both IETF standards and research articles) for parts of the problem. For instance, the rekey protocol may use a scalable group key management algorithm (GKMA) to reduce the number of keys sent in a rekey message. Examples of a GKMA include LKH, OFT, Subset difference based schemes etc.

5.1 Goals of the rekey protocol

The goals of the rekey protocol are:

- to synchronize a GSA
- to provide privacy and (symmetric or asymmetric) authentication, replay protection and DoS protection

- efficient rekeying after changes in group membership, or when keys (KEKs) expire,
- reliable delivery of rekey messages,
- provide methods for members to recover from an out-of-sync GSA,
- high throughput and low latency, and
- to use IP Multicast or multi-unicast.

We identify several major issues in the design of a rekey protocol:

1. rekey message format
2. reliable transport of rekey messages
3. implosion
4. recovery from out-of-sync GSA
5. incorporating GKMAs in rekey messages
6. interoperability of GKMAs

Note that for a GCKS to successfully rekey a group, it is not sufficient that rekey protocol implementations interoperate. We also need to ensure that the GKMA also interoperates, i.e., standards versions of group key management algorithms, such as LKH, OFT, subset difference and others need to be used.

In the rest of this section we discuss these topics in detail.

5.2 Rekey message Transport and Protection

Rekey messages contain Rekey and/or Data SAs along with KEKs and TPKs. These messages need to be confidential, authenticated, and protected against replay and DoS attacks. They are sent via multicast or multi-unicast from the GCKS to the members.

Rekey messages are encrypted with the Group KEK for confidentiality. When used in conjunction with a GKMA, portions of the rekey message are first encrypted with the appropriate KEKs as specified by the GKMA. The GCKS authenticates rekey messages using either a MAC - computed using the group Authentication key - or a digital signature. In both cases, a sequence number is included in computation of the MAC or the signature to protect against replay attacks.

When group authentication is provided - with a symmetric key - rekey messages are vulnerable to attacks by other members of the group. Rekey messages are digitally signed when group members do not trust each other. When asymmetric authentication is used, members receiving rekey messages are vulnerable to DoS attacks. An external adversary may send a bogus rekey message, which a member cannot identify until after it performs an expensive digital signature operation. To protect against such an attack, a MAC may be sent as part of the rekey message. Members verify the signature only upon successful verification of the MAC.

Rekey messages contain group key updates corresponding to a single [RFC2627, OFT] or multiple membership changes [SD, BatchRekey] and may contain group key initialization messages [OFT].

5.3 Reliable Transport of rekey messages

The GCKS needs to ensure that all members have the current Data Security and Rekey SAs. Otherwise, authorized members may be inadvertently excluded from receiving group communications. Thus, the GCKS needs to use a rekey algorithm that is inherently reliable or employ some reliable transport mechanism to send rekey messages.

There are two dimensions to the problem: Messages that update group keys may be lost in transit or may be missed by a host when it is offline. LKH and OFT group key management algorithms rely on past history of updates being received by the host. If the host goes offline, it will need to resynchronize its group-key state when it comes online; this may require a unicast exchange with the GCKS. The Subset Difference algorithm, however, conveys all the needed state in its rekey messages and does not need members to be always online, nor keeping state. Subset difference algorithm does not require a backchannel and can operate on a broadcast network. If a rekey message is lost in transmission, subset difference algorithm cannot decrypt messages encrypted with the TPK sent via the lost rekey message. There are self-healing GKMAs proposed in the literature that allow a member to recover lost rekey messages, as long as rekey messages before and after the lost rekey message are received.

Rekey messages are typically short (for single membership change as well as for small groups) which makes it easy to design a reliable delivery protocol. On the other hand, the security requirements may add an additional dimension to address. Also there are some special cases where membership changes are processed as a batch, which reduces the frequency of rekey messages, but increases their size. Furthermore, among all the KEKs sent in a rekey message, as many as half the members need only a single KEK. We may take

advantage of these properties in designing a rekey message(s) and a protocol for their reliable delivery.

Three categories of solutions have been proposed:

1. Repeatedly transmit the rekey message: Recall that in many cases rekey messages translate to only one or two IP packets.

2. Use an existing reliable multicast protocol/infrastructure

3. Use FEC for encoding rekey packets (with NACKs as feedback) [BatchRekey]

Note that for small messages, category 3 is essentially the same as category 1.

The group member might be out of synchrony with the GCKS if it receives a rekey message having a sequence number that is more than one greater than the last sequence number processed. This is one means by which the GCKS member detects that it has missed a rekey message. Alternatively, the data-security application might detect that it is using an out-of-date key and notifies the group key management module of this condition. What action the GCKS member takes is a matter of group policy: The GCKS member should log the condition and may contact the GCKS to re-run the re-registration protocol to obtain a fresh group key. The group policy needs to take into account boundary conditions, such as re-ordered rekey messages when rekeying is so frequent that two messages might get reordered in an IP network. The group key policy also needs to take into account the potential for denial of service attacks where an attacker delays or deletes a rekey message in order to force a subnetwork or subset of the members to synchronously contact the GCKS.

If a group member becomes out-of-synch with the GSA then it should re-register with the GCKS. However, in many cases there are other, simpler methods for re-synching with the group:

- The member can open a simple, unprotected connection (say, TCP) with the GCKS and obtain the current (or several recent) rekey messages. Note that there is no need for authentication or encryption here, since the rekey message is already signed and is anyway multicasted in the clear. One may think that this opens the GCKS to DoS attacks by many bogus such requests. But this does not seem to worsen the situation: in fact, bombarding the GCKS with bogus resynch requests would be much more problematic.

- The GCKS can post the rekey messages on some public site (say, web site) and the out-of-synch memeber can obtain the rekey messages from that site.

It is suggested that the GCKS always provide all three ways of resynching (i.e., re-registration, simple TCP, and public posting). This way, it is up to the member to choose how to resynch; it also avoids adding yet another field to the policy token [GSPT]. Alternatively, a policy token may contain a field specifying one or more methods supported for resynchronization of a GSA.

5.4 State-of-the-art on Reliable Multicast Infrastructure

The rekey message may be sent using reliable multicast. There are multiple types of reliable multicast protocols and products, which have different properties. However, there are no standard reliable multicast protocols at the present time. Thus, this document makes no recommendation for use of a particular reliable multicast protocol or set of protocols for the purposes group key management. The suitability of NAK-based, ACK-based or other reliable multicast methods are determined by the particular needs of the group key management application and environment. In the future, group key management protocols may choose to use particular standards-based approaches that meet the needs of the particular application. A secure announcement facility is needed to signal the use of a reliable multicast protocol, which must be specified as part of group policy. The reliable multicast announcement and policy specification, however, can only follow the establishment of reliable multicast standards and are not considered further in this document.

Today, the several MSEC group key management protocols support sequencing of the rekey messages through a sequence number, which is authenticated along with the rekey message. A sender of rekey messages may re-transmit multiple copies of the message provided that they have the same sequence number. Thus, re-sending the message is a rudimentary means of overcoming loss along the network path. A member who receives the rekey message will check the sequence number to detect duplicate and missing rekey messages. The member receiver will discard duplicate messages that it receives. Large rekey messages, such as those that contain LKH or OFT tree structures, might benefit from transport-layer FEC when standard methods are available in the future. It is unlikely that forward error correction (FEC) methods will benefit rekey messages that are short and fit within a single message. In this case, FEC degenerates to simple retransmission of the message.

5.5 Implosion

Implosion may occur due to one of two reasons. First, recall that one of the goals of the rekey protocol is to synchronize a GSA. When a rekey or Data SA expires, members may contact the GCKS for an

update. If all or even many members contact the GCKS at about the

same time, the GCKS cannot handle all those messages. We refer to this as an out-of-sync implosion.

The second case is in the reliable delivery of rekey messages. Reliable multicast protocols use feedback (NACK or ACK) to determine which packets must be retransmitted. Packet losses may result in many members sending NACKs to the GCKS. We refer to this as feedback implosion.

The implosion problem has been studied extensively in the context of reliable multicasting. Some of the proposed solutions viz., feedback suppression and aggregation, might be useful in the GKM context as well.

Members may wait for a random time before sending an out-of-sync or feedback message. Meanwhile, members might receive the key updates they need and therefore will not send a feedback message.

An alternative solution is to have the members contact one of several registration servers when they are out-of-sync. This requires GSA synchronization between the multiple registration servers.

Feedback aggregation and local recovery employed by some reliable multicast protocols are not easily adaptable to transport of rekey messages. There are authentication issues to address in aggregation. Local recovery is more complex in that members need to establish SAs with the local repair server (Any member of the group or a subordinate GCKS might serve as a repair server. Repair servers may be responsible for resending rekey messages).

Members may use the group SA, more specifically the Rekey SA, to authenticate requests sent to the repair server; however, replay protection requires maintaining state at members as well as repair servers. Authentication of repair requests is to protect against DoS attacks. Note also that an out-of-sync member may use an expired Rekey SA to authenticate repair requests, which requires repair servers to accept messages protected by old SAs.

Alternatively, a simple mechanism may be employed to achieve local repair efficiently. Each member receives a set of local repair server addresses as part of group operation policy information. When a member does not receive a rekey message, it can send a "retransmit replay message(s) with sequence number n and higher" to one of the local repair servers. The repair server can do one of two things: ignore the request if it is busy, or retransmit the requested rekey messages as received from the GCKS. The repair server, which is also another member may choose to serve only m requests in a given time period (i.e., rate limits responses) or per a given rekey message.

Rate limiting the requests and responses protects the repair servers

as well other members of the group from being vulnerable to DoS attacks.

5.6 Issues in Incorporating Group Key Management Algorithms

Group key management algorithms make Rekeying scalable. Large group Rekeying without employing GKMAs is prohibitively expensive.

First we list some requirements to consider in selecting a GKMA:

- Protection against Collusion: Members (or non members) should not be able to collaborate to deduce keys that they are not privileged to (following the GKMA key distribution rules).

- Forward access control: Ensure that departing members cannot get access to future group data.

- Backward access control: Ensure that joining members cannot decrypt past data.

5.7 Stateless, Stateful, and Self-healing Rekeying Algorithms

We classify group key management algorithms into three categories, viz., stateful, stateless, and self-healing algorithms.

Stateful algorithms [RFC2627, OFT] use KEKs from past rekeying instances to encrypt (protect) KEKS corresponding to the current and future rekeying instances. The main disadvantage in these schemes is that if a member were offline or otherwise fails to receive KEKs from a past rekeying instance, it may no longer be able to synchronize its GSA even though it can receive KEKs from all future rekeying instances. The only solution is to contact the GCKS explicitly for resynchronization. Note that the KEKs for the first rekeying instance are protected by the Registration SA. Recall that communication in that phase is one to one, and therefore it is easy to ensure reliable delivery.

Stateless GKMAs [SD1, SD2] encrypt rekey messages with KEKs sent during the registration protocol. Since rekey messages are independent of any past rekey messages (i.e. not protected by KEKs therein), a member may go offline, but continue to be able to decipher future communications. However, they offer no mechanisms to recover past rekeying messages. Stateless rekeying may be relatively inefficient, particularly for immediate (in contrast to batch) rekeying in highly dynamic groups.

In self-healing schemes [Self-healing], a member can reconstruct a lost rekey message, as long as it receives some past rekey messages and some future rekey messages.

5.8 Interoperability of a GKMA

Most GKMA specifications do not specify packet formats although any group key management algorithms need to, for the purposes of interoperability. In particular there are several alternative ways to managing key trees and numbering nodes within key trees. The following information is generally needed during initialization of a Rekey SA or included with each GKMA packet.

- o GKMA name (e.g. LKH, OFT, Subset difference)
- o GKMA version number (implementation specific). Version may imply several things such as the degree of a key tree, proprietary enhancements, and qualify another field such as a key id.
- o Number of keys or Largest ID
- o Version specific data
- o Per key information
  - Key ID
  - Key lifetime (creation/expiration data)
  - Encrypted key
  - Encryption key's ID (optional)

Key IDs may change in some implementations in which case we need to send:

- o List of <old id, new id>

6.0 Group Security Association

The GKM Architecture defines the interfaces between the registration, Rekey, and data security protocols in terms of the Security Associations (SAs) of those protocols. By isolating these protocols behind a uniform interface, the architecture allows implementations to use protocols best suited to their needs. For example, a rekey protocol for a small group could use multiple unicast transmissions with symmetric authentication, while that for a large group could use IP Multicast with packet-level Forward Error Correction and source authentication.

The Group Key Management Architecture provides an interface between

the security protocols and the group SA (GSA), which consists of

three SAs, viz., Registration SA, Rekey SA and Data SA. The Rekey SA is optional. There are two cases in defining the relationships between the three SAs. In both cases, the Registration SA protects the registration protocol.

In Case 1, group key management is done WITHOUT using a Rekey SA. The registration protocol initializes and updates one or more Data SAs (having TPKs to protect files or streams). Each Data SA corresponds to a single group - and a group may have more than one Data SA.

In Case 2, group key management is done WITH a Rekey SA to protect the rekey protocol. The registration protocol initializes the Rekey SAs (one or more) as well as zero or more Data SAs upon successful completion. When a Data SA is not initialized in the registration protocol, this is done in the rekey protocol. The rekey protocol updates Rekey SA(s) AND establishes Data SA(s).

6.1 Group policy

Group policy is described in detail in the Group Security Policy Token document [GSPT]. Group policy can be distributed through group announcements, key management protocols, and other out-of-band means (e.g., via a web page). The group key management protocol carries cryptographic policies of the SAs and keys it establishes as well as additional policies for the secure operation of the group.

The acceptable cryptographic policies for the registration protocol, which may run over TLS [TLS], IPsec, or IKE, are not conveyed in the group key-management protocol since they precede any of the key management exchanges. Thus, a security policy repository having some access protocol may need to be queried prior to key-management session establishment to determine the initial cryptographic policies for that establishment. This document assumes the existence of such a repository and protocol for GCKS and member policy queries. Thus group security policy will be represented in a policy repository and accessible using a policy protocol. Policy distribution may be a push or a pull operation.

The group key management architecture assumes that the following group-policy information may be externally managed, e.g., by the content owner, group conference administrator or group owner.

- Identity of the Group owner, and authentication method, and delegation method for identifying a GCKS for the group
- Group GCKS, authentication method, and delegation method for any subordinate GCKSs for the group
- Group membership rules or list and authentication method

There are also two additional policy-related requirements external to group key management.

- There is an authentication and authorization infrastructure such as X.509 [RFC 2459], SPKI [RFC 2693], or a pre-shared key scheme in accordance with the group policy for a particular group.

- There is an announcement mechanism for secure groups and events that operates according to group policy for a particular group.

Group policy determines how the registration and rekey protocols initialize or update Rekey and Data SAs. The following sections describe potential information sent by the GCKS for the Rekey and Data SAs. A member needs to have the information specified in the next sections to establish Rekey and Data SAs.

6.2 Contents of the Rekey SA

The Rekey SA protects the rekey protocol. It contains cryptographic policy, Group Identity and Security Parameter Index (SPI) [RFC2401] to uniquely identify an SA, replay protection information, and key protection keys.

6.2.1 Rekey SA Policy

The GROUP KEY MANAGEMENT ALGORITHM represents the group key revocation algorithm that enforces forward and backward access control. Examples of key revocation algorithms include LKH, LKH+, OFT, OFC and Subset Difference [RFC2627, OFT, TAXONOMY, SDR]. The key revocation algorithm could also be NULL. In that case, the Rekey SA contains only one KEK, which serves as the group KEK. The rekey messages initialize or update Data SAs as usual. But, the Rekey SA itself can be updated (group KEK can be Rekeyed) when members join or the KEK is about to expire. Leave Rekeying is done by re-initializing the Rekey SA through the rekey protocol.

The KEK ENCRYPTION ALGORITHM uses a standard encryption algorithm such as 3DES or AES. The KEK KEY LENGTH is also specified.

The AUTHENTICATION ALGORITHM uses digital signatures for GCKS authentication (since all shared secrets are known to some or all members of the group), or some symmetric secret in computing MACs for group authentication. Symmetric authentication provides weaker authentication in that any group member can impersonate a particular source. The AUTHENTICATION KEY LENGTH is also be specified.

The CONTROL GROUP ADDRESS is used for multicast transmission of rekey messages. This information is sent over the control channel such as in an ANNOUNCEMENT protocol or call setup message. The degree to

which the control group address is protected is a matter of group policy.

The REKEY SERVER ADDRESS allows the registration server to be a different entity from the server used for Rekey, such as for future invocations of the registration and rekey protocols. If the registration server and the Rekey server are two different entities, the registration server sends the Rekey servers address as part of the Rekey SA.

6.2.2 Group Identity

The Group identity accompanies the SA (payload) information as an identifier if the specific group key management protocol allows multiple groups to be initialized in a single invocation of the registration protocol or multiple groups to be updated in a single rekey message. It is often much simpler to restrict each registration invocation to a single group; no such restriction is necessary. There is always a need to identify the group when establishing a Rekey SA either implicitly through an SPI or explicitly as an SA parameter.

6.2.3 KEKs

Corresponding to the key management algorithm, the Rekey SA contains one or more KEKs. The GCKS holds the key encrypting keys of the group, while the members receive keys following the specification of the key-management algorithm. When there are multiple KEKs for a group (as in an LKH tree), each KEK needs to be associated with a Key ID, which is used to identify the key needed to decrypt it. Each KEK has a LIFETIME associated with it, after which the KEK expires.

6.2.4 Authentication Key

The GCKS provides a symmetric or public key for authentication of its rekey messages. Symmetric-key authentication is appropriate only when all group members can be trusted not to impersonate the GCKS. The architecture does not rule out methods for deriving symmetric authentication keys at the member [RFC2409] rather than being pushed from the GCKS.

6.2.5 Replay Protection

Rekey messages need to be protected from replay/reflection attacks. Sequence numbers are used for this purpose and the Rekey SA (or protocol) contains this information.

6.2.6 Security Parameter Index (SPI)

The tuple <Group identity, SPI > uniquely identifies a Rekey SA. The SPI changes each time the KEKs change.

6.3 Contents of the Data SA

The GCKS specifies the data security protocol used for secure transmission of data from sender(s) to receiving members. Examples of data security protocols include IPsec ESP [RFC 2401] and SRTP [RFC 3711]. While the content of each of these protocols is out of the scope of this document, we list the information sent by the registration protocol (or the rekey protocol) to initialize or update the Data SA.

6.3.1 Group Identity

The Group identity accompanies SA information when Data SAs are initialized or Rekeyed for multiple groups in a single invocation of the registration protocol or in a single rekey message.

6.3.2 Source Identity

The SA includes source identity information when the group owner chooses to reveal Source identity to authorized members only. A public channel such as announcement protocol is only appropriate when there is no need to protect source or group identities.

6.3.3 Traffic Protection Keys

Irrespective of the data security protocol used, the GCKS supplies the TEKs or information to derive TEKs, used for data encryption.

6.3.4 Data Authentication Keys

Depending on the data-authentication method used by the data security protocol, group key management may pass one or more keys, functions (e.g., TESLA [TESLA]), or other parameters used for authenticating streams or files.

6.3.5 Sequence Numbers

The GCKS passes sequence numbers when needed by the data security protocol, for SA synchronization and replay protection.

6.3.6 Security Parameter Index (SPI)

The GCKS may provide an identifier as part of the Data SA contents for data security protocols that use an SPI or similar mechanism to identify an SA or keys within an SA.

6.3.7 Data SA policy

The Data SA parameters are specific to the data security protocol but generally include encryption algorithm and parameters, the source authentication algorithm and parameters, the group authentication algorithm and parameters, and/or replay protection information.

7.0 Scalability Considerations

The area of group communications is quite diverse. In teleconferencing, a multipoint control unit (MCU) may be used to aggregate a number of teleconferencing members into a single session; MCUs may be hierarchically organized as well. A loosely coupled teleconferencing session [RFC3550] has no central controller but is fully distributed and end-to-end. Teleconferencing sessions tend to have at most dozens of participants whereas video broadcast, which uses multicast communications, and media on demand, which uses unicast, are large-scale groups numbering hundreds to millions of participants.

As described in the Requirements section above, the group key management architecture supports multicast applications with a single sender. The architecture described in this paper supports large-scale operation through the following features.

1. There is no need for a unicast exchange to provide data keys to a security protocol for members who have previously-registered in the particular group; data keys can be pushed in the rekey protocol.

2. The registration and rekey protocols are separable to allow flexibility in how members receive group secrets. A group can use a smart-card based system in place of the registration protocol, for example, to allow the rekey protocol to be used with no back channel for broadcast applications such as television conditional access systems.

3. The registration and rekey protocols support new keys, algorithms, authentication mechanisms and authorization infrastructures in the architecture. When the authorization infrastructure supports delegation, as in X.509 and SPKI, the GCKS function can be distributed as shown in Figure 3.

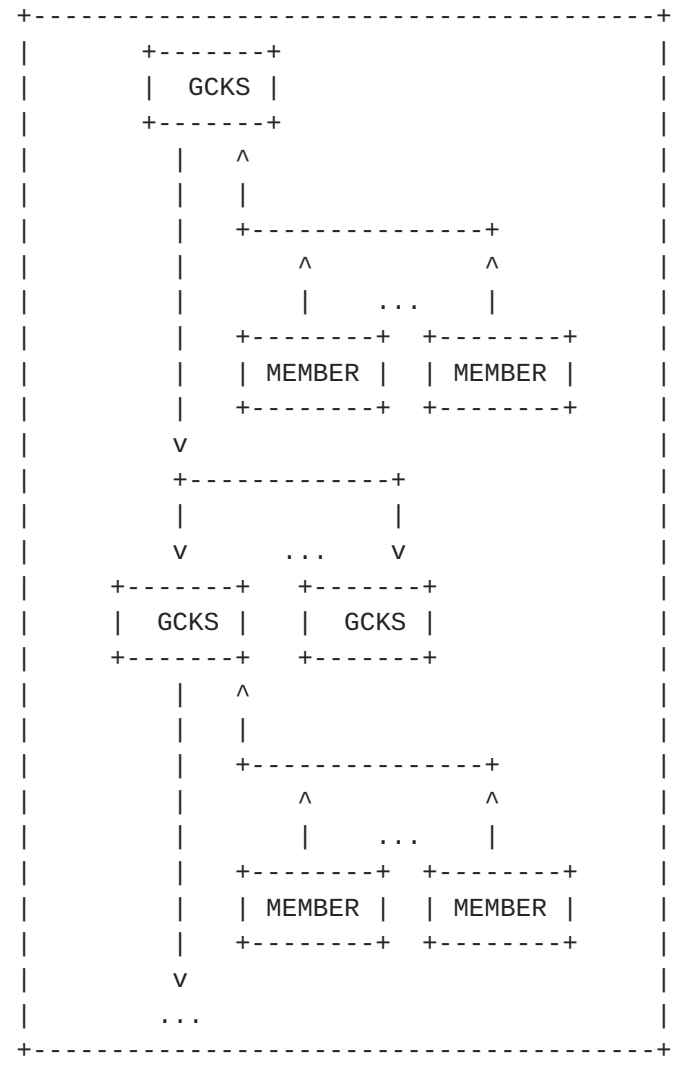

Figure 3: Hierarchically-organized Key Distribution

The first feature in the list allows fast keying of data security protocols when the member already belongs to the group. While this is realistic for subscriber groups and customers of service providers who offer content events, it may be too restrictive for applications that allow member enrollment at the time of the event. The MSEC group key management architecture suggests hierarchically organized key distribution to handle potential mass simultaneous registration requests. The Figure 3 configuration may be needed when conventional clustering and load-balancing solutions of a central GCKS site cannot meet customer requirements. Unlike conventional caching and content-distribution networks, however, the configuration shown in Figure 3 has additional security ramifications for physical security of a GCKS.

More analysis and work needs to be done on the protocol instantiations of the group key management architecture to determine

how effectively and securely the architecture can support large-scale multicast applications. In addition to being as secure as

pairwise key management against man-in-the-middle, replay, and reflection attacks, group key management protocols have additional security needs. Unlike pairwise key management, group key management needs to be secure against attacks not only by non-members but by members who may attempt to impersonate a GCKS or disrupt the operation of a GCKS. Thus, secure groups need to converge to a common group key under the conditions of members attacking the group, joining and leaving the group, and being evicted from the group. Group key management protocols also need to be robust when denial of service attacks or network partitions lead to large numbers of synchronized requests. An instantiation of group key management, therefore, needs to consider how GCKS operation might be distributed across multiple GCKS as designated by the group owner to serve keys on behalf of a designated GCKS. GSAKMP [GSAKMP] protocol uses the policy token and allows designating some of the members as subordinate GCKSs to address this scalability issue.

8.0 Security Considerations

This memo describes MSEC key management architecture. This architecture will be instantiated in one or more group key management protocols, which must be protected against man-in-the-middle, connection hijacking, replay or reflection of past messages, and denial of service attacks.

Authenticated key exchange [STS, SKEME, RFC2408, RFC2412, RFC2409] techniques limit the effects of man-in-the-middle and connection-hijacking attacks. Sequence numbers and low-computation message authentication techniques can be effective against replay and reflection attacks. Cookies [RFC2522], when properly implemented, provide an efficient means to reduce the effects of denial of service attacks.

This memo does not address attacks against key management or security protocol implementations such as so-called type attacks that aim to disrupt an implementation by such means as buffer overflow. The focus of this memo is on securing the protocol, not an implementation of the protocol.

While classical techniques of authenticated key exchange can be applied to group key management, new problems arise with the sharing of secrets among a group of members: Group secrets may be disclosed by a member of the group and group senders may be impersonated by other members of the group. Key management messages from the GCKS should not be authenticated using shared symmetric secrets unless all members of the group can be trusted not to impersonate the GCKS or

each other. Similarly, members who disclose group secrets undermine the security of the entire group. group owners and GCKS

administrators must be aware of these inherent limitations of group key management.

Another limitation of group key management is policy complexity: Whereas peer-to-peer security policy is an intersection of the policy of the individual peers, a group owner sets group security policy externally in secure groups. This document assumes there is no negotiation of cryptographic or other security parameters in group key management. Group security policy, therefore, poses new risks to members who send and receive data from secure groups. Security administrators, GCKS operators, and users need to determine minimal acceptable levels of security (e.g., authentication and admission policy of the group, key lengths, cryptographic algorithms and protocols used etc.) when joining secure groups.

Given the limitations and risks of group security, the security of the group key management registration protocol should be as good as the base protocols on which it is developed such as IKE, IPsec, TLS, or SSL. The particular instantiations of this Group Key Management architecture must ensure that the high standards for authenticated key exchange are preserved in their protocol specifications, which will be Internet standards-track documents that are subject to review, analysis and testing.

The second protocol, the group key management rekey protocol, is new and has unknown risks associated with it. The source-authentication risks described above are obviated by the use of public-key cryptography. The use of multicast delivery may raise additional security issues such as reliability, implosion, and denial of service attacks based upon the use of multicast. The rekey protocol specification needs to offer secure solutions to these problems. Each instantiation of the rekey protocol, such as the GSAKMP Rekey or the GDOI Groupkey-push operations, need to validate the security of their Rekey specifications.

Novelty and complexity are the biggest risks to group key management protocols. Much more analysis and experience are needed to ensure that the architecture described in this document can provide a well-articulate standard for security and risks of group key management.

9.0 Acknowledgments

The GKM Building Block [GKMBB) I-D by SMuG was a precursor to this document; thanks to Thomas Hardjono and Hugh Harney for their efforts. During the course of preparing this document, Andrea Colegrove, Brian Weis, George Gross and several others in MSEC WG and GSEC and SMuG research groups provided valuable comments that

helped improve this document. The authors appreciate their contributions to this document.

10.0 References and Bibliography

[BatchRekey] Yang, Y. R., et al., Reliable Group Rekeying: Design and Performance Analysis, in Proc. of ACM SIGCOMM, San Diego, CA, August 2001.

[CLIQUES] M. Steiner, G. Tsudik and M. Waidner, CLIQUES: A New Approach to Group Key Agreement, IEEE ICDCS 97, May 1997

[FN93] A. Fiat, M. Naor, Broadcast Encryption, Advances in Cryptology - CRYPTO 93 Proceedings, Lecture Notes in Computer Science, Vol. 773, 1994, pp. 480 -- 491.

[GSAKMP] H.Harney, A.Colegrove, E.Harder, U.Meth, R.Fleischer, Group Secure Association Key Management Protocol, http://www.ietf.org/proceedings/03mar/I-D/draft-ietf-msec-gsakmp-sec-01.txt, February 2003, Work in Progress.

[GSPT] Hardjono, T., H. Harney, P. McDaniel, A. Colegrove, and P. Dinsmore, The MSEC Group Security Policy Token, draft-ietf-msec-gspt-02.txt, August 2003, Work in Progress.

[H.235] ITU, Security and encryption for H-Series (H.323 and other H.245-based) multimedia terminals, ITU-T Recommendation H.235 Version 3, 2001, Work in progress.

[JKKV94] M. Just, E. Kranakis, D. Krizanc, P. van Oorschot, On Key Distribution via True Broadcasting. In Proceedings of 2nd ACM Conference on Computer and Communications Security, November 1994, pp. 81--88.

[MARKS] Briscoe, B., MARKS: Zero Side Effect Multicast Key Management Using Arbitrarily Revealed Key Sequences, in Proc. of First International Workshop on Networked Group Communication (NGC), Pisa, Italy, November 1999.

[MIKEY] J. Arkko, E. Carrara, F. Lindholm, M. Naslund, and K. Norrman, MIKEY: Multimedia Internet KEYing, Internet Draft, http://www.ietf.org/proceedings/03mar/I-D/draft-ietf-msec-mikey-06.txt, February 2003, Work in progress.

[MSEC-Arch] Hardjono, T., and B. Weis, The Multicast Group Security Architecture, RFC 3740 (Informational), March 2004.

[MVV] A.J.Menzes, P.C.van Oorschot, S.A. Vanstone, Handbook of Applied Cryptography, CRC Press, 1996.

[OFT] Balenson, D., D. McGrew, and A. Sherman, Key Management for Large Dynamic Groups: One-Way Function Trees and Amortized Initialization, draft-irtf-smug-groupkeymgmt-oft-00.txt, IRTF, August 2000, Work in progress.

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

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

[RFC2326] Schulzrinee, H., A. Rao, and R. Lanphier, Real Time Streaming Protocol (RTSP), RFC 2326 (Proposed Standard), April 1998.

[RFC2327] Handley, M., and V. Jacobson, SDP: Session Description Protocol, RFC 2327 (Proposed Standard), April 1998.

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

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

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

[RFC2408] Maughan, D., et al., Internet Security Association and Key Management Protocol (ISAKMP), RFC 2408 (proposed standard), November 1998.

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

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

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

[RFC2543] Handley, M., et. al., SIP: Session Initiation Protocol, RFC 2543 (Proposed Standard), March 1999.

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

[RFC3547] M. Baugher, T. Hardjono, H. Harney, B. Weis, The Group Domain of Interpretation, RFC 3547 (Proposed Standard), July 2003.

[RFC3550] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, RTP: A Transport Protocol for Real-Time Applications, RFC 3550 (Proposed Standard), July 2003.

[RFC3711] Baugher, M., et. al., The Secure Real Time Transport Protocol, RFC 3711 (Proposed Standard), March 2004.

[SD1] Naor, D., M. Naor, and J. Lotspiech, Revocation and Tracing Schemes for Stateless Receivers, in Advances in Cryptology - CRYPTO, Santa Barbara, CA: Springer-Verlag Inc., LNCS 2139, August 2001.

[SD2] Moni Naor and Benny Pinkas, Efficient Trace and Revoke Schemes, In Proceedings of Financial Cryptography 2000, Anguilla, British West Indies, February 2000.

[Self-Healing] Staddon, J., et. al., Self-healing Key Distribution with Revocation, In proceedings of the 2002 IEEE Symposium on Security and Privacy, Oakland, CA, May 2002.

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

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

[TAXONOMY] R. Canetti et al, Multicast Security: A taxonomy and some Efficient Constructions, IEEE INFOCOM, 1999.

[TESLA-INFO] Perrig, A., R. Canetti, D. Song, D. Tygar, and B. Briscoe, TESLA: Multicast Source Authentication Transform Introduction, http://www.ietf.org/proceedings/03mar/I-D/draft-ietf-msec-tesla-intro-01.txt, October 2002, Work in Progress.

[TESLA-SPEC] Perrig, A., R. Canetti, and Whillock, TESLA: Multicast Source Authentication Transform Specification, http://www.ietf.org/proceedings/03mar/I-D/draft-ietf-msec-tesla-spec-00.txt, April 2002, Work in Progress.

[tGSAKMP] Harney, H., et. al., Tunneled Group Secure Association Key Management Protocol, http://www.ietf.org/internet-drafts/draft-ietf-msec-tgsakmp-00.txt, May 2003, Work in Progress.

[TPM] D.S. Marks, B.H. Turnbull, Technical protection measures: The intersection of technology, law, and commercial licenses, Workshop on Implementation Issues of the WIPO Copyright Treaty (WCT) and the WIPO Performances and Phonograms Treaty (WPPT), World Intellectual Property Organization, Geneva, December 6 and 7, 1999

(http://www.wipo.org/eng/meetings/1999/wct_wppt/pdf/imp99_3.pdf).

[Wool] Wool. A., Key Management for Encrypted broadcast, 5th ACM Conference on Computer and Communications Security, San Francisco, CA, Nov. 1998.

11.0 Authors' Addresses

Mark Baugher
Cisco Systems
5510 SW Orchid St.
Portland, OR 97219, USA
+1 408-853-4418
mbaugher@cisco.com

Ran Canetti
IBM Research
30 Saw Mill River Road
Hawthorne, NY 10532, USA
+1 914-784-7076
canetti@watson.ibm.com

Lakshminath R. Dondeti
Nortel Networks
600 Technology Park Drive
Billerica, MA 01821, USA
+1 978-288-6406
ldondeti@nortelnetworks.com

Fredrik Lindholm
Ericsson Research
SE-16480 Stockholm, Sweden
+46 8 58531705
fredrik.lindholm@ericsson.com

Full Copyright Statement

Intellectual Property

pertain to the implementation or use of the technology described in

Additional Acknowledgements

Funding for the RFC Editor function is currently provided by the Internet Society.