Secure Multicast Communication

TODOs/Questions

- Can the current membership list version can be attached to the selection process messages? Todd?
- IPv6 has keys be sender oriented. Are we going to still be compatible?
- Need to limit lifetime of keys. We don't need to put in explicit time-outs on the group keys, do we?
- When can we discard a group key? Should be based on message stability.
- What happens if a new DES key is installed at the time a new member is joining?
- How to pick the one time identifiers? Need a good pseudo-random generator, as well as a way to seed it.
- Some packets in the group key management algorithms do not need DES encoding if the group doesn't care about confidentiality. Note this/figure out how to integrate this. Should probably also be part of the SAID description.

1. Introduction and Overview

A number of current Internet standards and emerging standards, including IPv6, (TODO: Put in name of HTTP standard)SHTTP and PEM, address the needs of security in the Internet. However, no standards or established systems adequately address the notion of security for multicast groups. IPV6, for example, has made provisions for the use of its authentication and encryption features with multicast traffic, but concede that no scaleable solution exists today for efficient key management and user non-repudiation in these settings. In addition, to our knowledge, no one has adequately addressed the security needs for a reliable multicast protocol. The semantics of security in reliable group communication protocols are necessarily different than unicast protocols. To solve these problems, this document proposes a system, called Secure Multicast Communication (SMC), which provides user authentication, data integrity and data privacy, in a reliable multicast framework. This protocol, which is specifically designed to work with the Reliable Multicast Protocol (RMP), is designed to interoperate with IPV6 and other emerging Internet standards, but does not rely on them. In this way, secure reliable multicast communication can be developed and used immediately, and then integrated with IPV6 as it comes into widespread use.

Following IPV6 and other systems, SMC recognizes three main types of security concerns. The first is data integrity and authentication, which verifies that a packet was sent by a given sender, and that the received packet is the same as the packet that was sent. In addition, authentication may provide non-repudiation, which verifies that a given sender sent a packet, even if that sender decides to deny this fact. The second category is confidentiality, which keeps data private from everyone except the intended destinations. The final category is proof against traffic analysis attacks, which analyze the patterns of encrypted traffic. Following the lead of IPV6, SMC supports data integrity and authentication, as well as data confidentiality, without providing defenses against traffic analysis.

SMC offers four specific, independent security features tailored for group communication, as well as a scaleable algorithm for group key management. The first three features are group authentication, individual authentication with non-repudiation, and data confidentiality. In addition to these three features, SMC offers a novel feature called secure virtual synchrony, which handles group membership changes in such a way that a sender has secure knowledge of which group members currently have the shared group key.

The first feature SMC offers is group authentication. Group authentication is virtually identical to the IPv6 Authentication Header. It uses a secure hash function (MD5 by default) and a symmetric key (DES CBC by default) to sign a RMP packet. The DES key is shared among all of the group members. This allows the recipients to verify that some member of the group sent the packet, and to securely verify that the contents of the packet have not been corrupted since it was sent. This feature requires that all group members be trusted, as any one could corrupt a packet sent by another. Of the four features, this one has the least overhead in computation time.

Individual authentication uses a secure hash function (MD5 by default) and an asymmetric public key protocol (RSA by default) to sign a packet. Individual authentication provides data integrity as well as verification of the identity of the sender of a packet. Unlike group authentication, it identifies the individual member who sent the packet rather than identifying that it came from someone in the group. This feature also provides non-repudiation of packets, since no other user can forge a packet's signature. This feature is moderately more computationally intensive than is group authentication, because it uses RSA to sign the hash instead of DES.

Data confidentiality is provided by encrypting entire packets with a shared symmetric key, and pre-pending a clear-text header on the encrypted packet. This is almost identical to the IPv6 Encapsulating Security Payload (ESP) feature. This provides privacy of data and headers from intruders which do not have the shared group key. All three of these features are very straightforward, and are similar to other systems.

TODO: Describe SVS (briefly) as well as the group key management protocol.

1.1 Design Goals

In designing SMC, a number of key design goals have been followed. These include the following.

- SMC was designed to require as few modifications to the RMP protocol definition and implementation as possible. Instead, a separate SMC layer is defined which implements the necessary security. A small set of callback functions allow the RMP layer to call functions in the SMC layer when necessary.

- SMC recognizes two classes of group members--active members which can change the group membership, and passive members which can not. Both groups can send and receive messages.

- All of the features of SMC are designed to be independent of each other, so that users can pick which ones they need.

- SMC is designed to be as compatible as possible with IPv6 and other emerging standards. To this end, many of the features of SMC can be implemented either by IPv6 or by the SMC layer.

- SMC is designed to be able to provide additional optional group security functions, such as secure virtual synchrony and group key management, on top of other systems such as IPv6.

- SMC is designed to be flexible enough to work with most all encryption algorithms and key distribution systems.

1.2 Default Encryption Algorithms

SMC stipulates a default set of encryption algorithms--MD5, DES, and RSA--that must be supported by each implementation. For simplicity, the rest of this document will at times refer to these specific algorithms instead of the general class of algorithms each one represents. In all cases, other algorithms can be added to SMC implementations that perform the same functionality in a different way.

Following the lead of IPv6, the default algorithm for secure message hashing is MD5. This produces a 128 bit hash of a message. SMC implementations may also choose to use the Secure Hash Algorithm, which produces a 160 bit key, or another similar function.

The default symmetric key encryption algorithm is DES used in the CBC mode. The default DES key length is 128 bits, although 64 bit keys may also be used. With 128 bit keys, the triple-DES algorithm is used by default.

The default public key encryption algorithm is RSA. The default key length for RSA is 512 bits. This is secure enough for most users. However, users may also select to use 1024 bit keys for RSA.

1.3 SMC as Part of the Extended RMP Architecture

SMC, when used with RMP, is provided through a separate secure messaging layer. In the extended RMP architecture, this layer sits directly above RMP, and below the messaging layer. Other layers, such as the hierarchical groups layer, may sit on top of the messaging layer. The applications then sit on top of these layers. These core layers communicate synchronously, while the applications and network/operating system communicate asynchronously with these layers via shared queues. This principle of

separation allows security to be added to RMP with a minimum number of changes to the RMP code and protocol definition. The only changes required are that the RMP layer has to be able to pass packets to the Security layer for encryption and decryption, and it has to be able to pass List Change Requests up to the Security layer for validation of the requested membership change. This architecture is shown in figure 1.

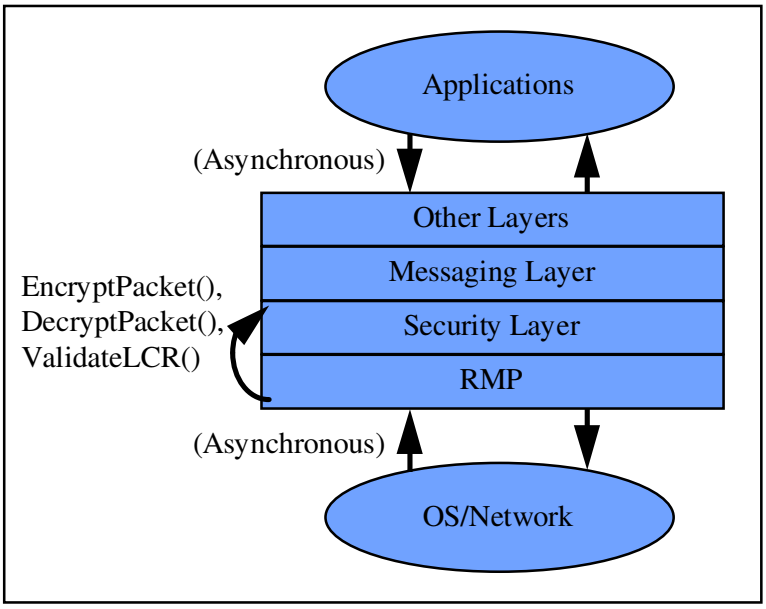


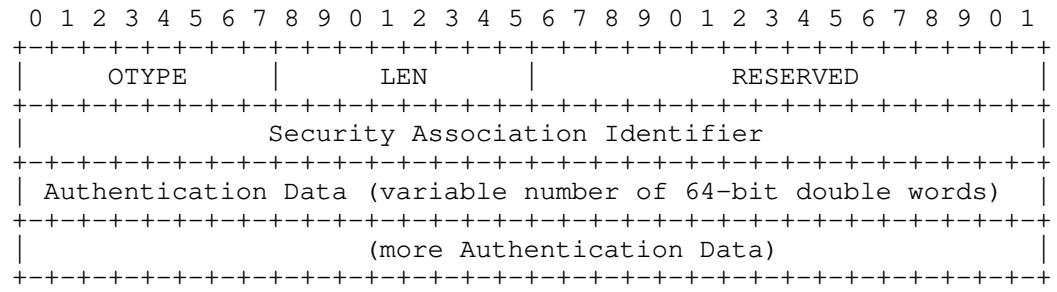
Figure 1: Extended RMP Architecture

2. SMC Features

The security features of group authentication, group encryption, and individual signatures are provided in RMP through the use of variable length options in the packet header, similar to those used in the current IP protocols. These options come between the RMP fixed header (which is the same for all RMP packets) and the RMP variable header. The LEN field of the RMP Fixed Header holds the total length of all options. The first byte of each option defines the type of option. RMP defines an authentication option and an encryption option, for use with SMC.

2.1 Group Authentication and Individual Signatures

Group authentication and individual signatures are both implemented with the Authentication option. Both of these options guarantee both authentication of the sender, as well as integrity of the packet. The group authentication feature guarantees that a member of the group sent the data in this packet, and that the data has not been changed since it was sent. The individual signature feature provides the same functionality, but verifies the actual user that sent the packet. Individual signatures also provide non-repudiation, so that the signer of a message can be securely identified, even if the signer attempts to disavow signing the message. This authentication option is designed to be as compatible as possible with IPv6. The packet format for this option is as follows.



OTYPE: Option Type field. Value of 01 for Authentication option.

LEN: Length of this option, in 32-bit words.

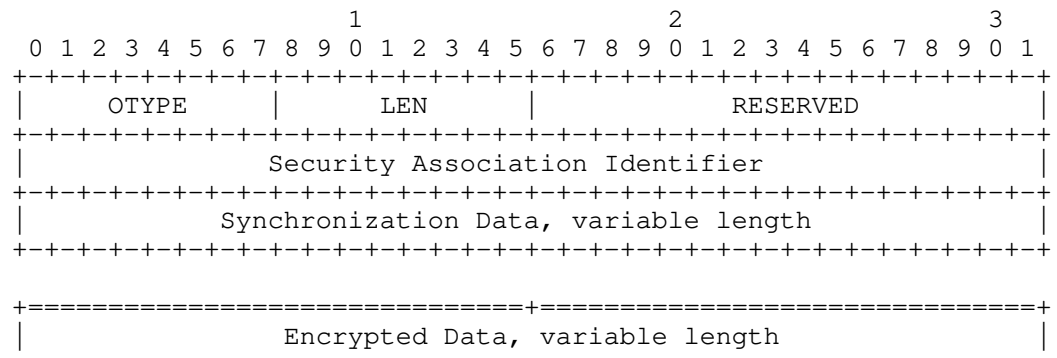
RESERVED: Must be set to 0x0000.

Security Association Identifier (SAID): This is compatible with the IPv6 Security Association Identifier. This is a 32-bit pseudo-random identifier, which identifies the security association for this packet. As in IPv6, if no SAID has been established, the value of this field shall be 0x00000000. The SAID values of 0x00000001 through 0x000000FF are reserved. Each SAID will correspond to a set of shared group keys. When a new set of shared keys is sent to the group, the SAID will also be changed. In this way, a packet's authentication and encryption features are self-identifying.

Authentication Data: The actual message digest. This algorithms used, as well as the identifier for the group keys used, can be derived from the SAID. By default, this field will be equal to an encrypted 128-bit MD5 secure hash of the rest of the RMP packet after this option. By default, group authentication will use a shared DES key to encrypt the MD5 hash. For individual signatures, the default encryption will be RSA. For more details, see the section on default encryption algorithms.

2.2 Group Encryption (Encapsulating Security Payload)

To keep data and headers private, SMC uses data encryption using a shared group key. This is designed to be as compatible as possible with the IPv6 Encapsulating Security Payload standard. The fields of this option are transmitted in plain text. Everything from the end of this option to the end of the packet is encrypted using the key associated with the SAID. The packet format for this option follows.



OTYPE: Option Type field. Value of 02 for Group Encryption option.

LEN: The length of this option, in 32-bit words.

RESERVED: Must be set to 0x0000.

Security Association Identifier (SAID): This is compatible with the IPv6 Security Association Identifier. This is a 32-bit pseudo-random identifier, which identifies the security association for this packet. As in IPv6, if no SAID has been established, the value of this field shall be 0x00000000. The SAID values of 0x00000001 through 0x000000FF are reserved. Each SAID will correspond to a set of shared group keys. When a new set of shared keys is sent to the group, the SAID will also be changed. In this way, a packet's authentication and encryption features are self-identifying.

Synchronization Data: By default, the length of this field is set to zero. This field is only used with those encryption algorithms which require a cryptographic synchronization for each field. The length of this field is derived from the LEN field of this option.

Encrypted Data: All data after the group encryption option is encrypted.

TODO: Should we use the reserved field as a length field?

2.3 Secure Virtual Synchrony

By default, both group authentication and data confidentiality require that a sender trust not only the current group members, but also all of the previous and future group members which have received or will receive a copy of the shared group key. For many applications, this requires an inordinate amount of trust, because although the sender can know which users are currently in the group, it does not know which users have received the DES key in the past or will receive the DES key in the near future. To alleviate this, SMC provides a novel feature called secure virtual synchrony (SVS). RMP provides an optional feature called virtual synchrony, defined originally by Ken Birman in the ISIS system [REFERENCE]. Virtual synchrony in RMP causes messages and group membership changes to be delivered to all users in a consistent total order. This allows users to combine the efficiency of using an asynchronous system with the ease of programming a synchronous system. RMP provides this in an extremely efficient way, with throughput to a group of N destinations approximately equal to that of a single TCP/IP connection between the sender and the slowest receiver. In addition, the latency for these totally ordered packets is at most two times that of TCP/IP, and often approaches the latency of TCP/IP.

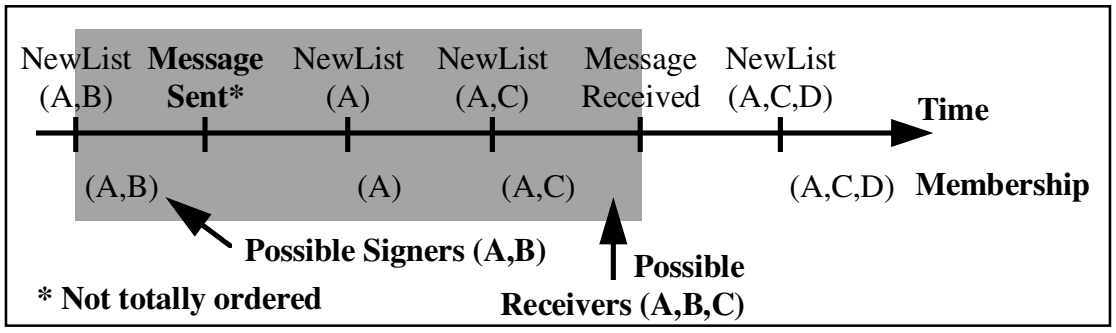


Figure 1: Secure Virtual Synchrony

Secure virtual synchrony changes the group membership key each time a member leaves or joins the group. The new group membership key is securely updated to just the members of the new group list. This requires a combination of virtual synchrony, secure membership changes, and an efficient re-keying of the group key each time that the membership changes. Because of this, secure virtual synchrony only works with a reliable multicast protocol that provides virtual synchrony, and a group key management algorithm such as the one proposed in this paper.

The major advantage of secure virtual synchrony is that it enables a group member to determine exactly what members of the group could have held the key to decrypt, encrypt, or sign a message. This feature is based on the fact that all totally ordered messages and membership changes are delivered in the same total order at all sites. By receiving a copy of its own packet, a sender knows at what time it was delivered relative to other packets and membership changes. Along with this, secure virtual synchrony guarantees three things for totally ordered packets.

1. Each membership list that is generated corresponds to exactly one group key.
2. Only the members of a group list will posses the key corresponding to that list.
3. New members will be able to decrypt and authenticate all packets delivered after the New List adding them to the group is delivered.
4. Any packet encrypted and/or signed after a new list has been received, will use the shared group key for that new list.

Property (3) is ensured by re-encrypting packets, which a new member should be able to decrypt but can not, with the new group key. This is critical for virtual synchrony, but means there is no single membership list that corresponds to the group of members which can hold the key for a given packet. Instead, the set of members who can decrypt a message correspond to the union of the membership lists that exist between when the message was sent and when it was delivered. The set of members who could have encrypted or group signed a message corresponds to the membership list associated with the key the message was encrypted or signed with.

Figure 1 shows an example of how this works for a site (A) sending a message. All of the events shown, except the point when the message is sent from A, are totally

ordered at all sites. Before the message is sent, the membership consists of A and B. After the message is sent, but before it is delivered at all sites, two membership changes occur. Member C joins the list, and member B leaves the list. After the message is delivered, member D joins the group.

Say that site A encrypted its message, and wishes to determine which sites could have decrypted it. In this example, because of property (3), C can receive the message, but D can not. Virtual synchrony dictates that B should not receive the message, since it has left the group before the message is delivered. However, the message was encrypted with the group key which corresponded to the membership list of (A,B). Because of this, B will be able to decrypt the message if it intercepts a copy of it. So, in this example, A, B, and C could have decrypted the message.

Secure virtual synchrony also allows a receiver to determine which users could have signed or encrypted a group message. In this example, this is A and B. Because of property (3), a packet may be re-encrypted or re-signed and passed to a new member. In this case, the original sender of a packet may no longer be in the group list corresponding to a received packet. However, the new member knows that a member of that new list also signed or encrypted the packet, which should be adequate. So, in this example, if B had sent the packet instead of A, C would have received a copy of it encrypted by A instead of by B.

When used with the group key management protocol, below, the use of SVS is selected by the member who has last re-keyed the group. Each re-key operation (including the initial keying of the group) associates the set of group keys with a Security Association Identifier (SAID). The SAID is embedded in packets sent with SMC, and allows the members to find out which keys and which encryption algorithms are being used for a given packet. In addition to this, the descriptor referred to by the SAID also contains a flag which denotes whether or not the group is using SVS.

3. Group Key Management

In addition to these four security features, SMC also supports a new scaleable group key management protocol. This protocol, like the rest of SMC, is a hybrid system, combining public key and symmetric encryption algorithms. This system assumes that an external public key distribution facility exists, such as a manual key entry system, a hierarchical certificate distribution system, or a PGP type network of trust. One promising candidate for this system is the work under way in the IETF to add public key distribution to DNS, but SMC is designed to work with any system. Extra key distribution modules are added on to the SMC system to interface with these key distribution systems.

Like IPv6, SMC is designed to support a variety of encryption algorithms, while mandating default algorithms for wider compatibility. This is also important since the encryption algorithms (such as DES and RSA) are under export restrictions. To accommodate these restrictions, interfaces to SMC are clearly defined which enable SMC

to be shipped without any encryption algorithms, yet easily be linked together with implementations of encryption algorithms.

SMC provides a scaleable group key management protocol which can also provide secure virtual synchrony when used with RMP. This protocol assumes that a system for the secure distribution of individual public keys already exists. Additional modules to SMC implementations will interface to these systems. All of the features of SMC, with the exception of secure virtual synchrony, can be used without this protocol.

A key feature of the group key management protocol is that it recognizes two classes of group members--active and passive users. Active users are allowed to make changes to the group membership--adding or removing other members from the group. Passive users can join the group if added by an active member, but can not themselves add other members. In general, active members are trusted more than passive members. For client/server systems, the trusted servers can be made active members, and the clients passive members. This difference in capabilities is enforced by sharing a private RSA key among the active members. This key is required to sign all group membership changes. This feature provides secure group access control, which is necessary for any group key management protocol.

The group key management protocol is session oriented. When used with RMP, each session corresponds to an RMP group. The first member to create a session is designated as an active member, and it creates both a DES key and a RSA pair of keys, which are used for this session. After this, when a new member requests to join the session, it can request to be added as either an active or as a passive member. One of the current active members will handle this request, and approve it or reject it. This requires two steps--authentication and validation. The authentication step verifies to the active member handling the request that the new member is who she says she is, and verifies to the new member that the active member is who he says he is. Once the new user is authenticated, the active member calls the application program to determine if this new user has the right to join this group as an active or passive member. If the application layer validates the new user, it is added. Otherwise, a reject message is sent back to the new member and the rest of the group. When a new member is added, it receives the session DES key and a copy of the session public RSA key. If it joins as an active member, it also receives a copy of the private RSA key.

For additional security, an active member may at times choose to re-key the group. When this occurs, the active member chooses three new keys for the group--a new symmetric key and a new public/private key pair. The active member securely sends them just to the appropriate current members of the group. Re-keying a group invalidates the keys of users who have left the group. This is important because you often can not trust a user after it leaves the group. An example of why this can be a problem is core dumps. On some systems, a core dump may be readable by more users than just the user of the process which crashed. If this is the case, an intruder could conceivably cause a fatal error to occur for an application, have it core dump, and then pull the group key out of the core file. In addition, re-keying a group keeps messages which were encrypted with the old key

secure from new members which join after both the re-key operation and after the message has been delivered. Re-keying a group can be done at an arbitrary time, and it is done automatically as part of secure virtual synchrony each time the membership list changes.

A member of the group is identified in one of two ways. It has a unique ID which is assigned to it by RMP. This consists of its RMP process ID concatenated with its (optional) client number, which is assigned by the messaging layer. This identifier, which we will call its RMP ID, is only valid for a single RMP session. In addition to the RMP ID, the user will also have its own unique ID which is permanently associated with its public key. We will call this ID the User ID. Both of these identifiers are used when a new member joins the group. The User ID is used to find the public key for a user and authenticate it, and the RMP ID is used to verify the RMP process ID of the joining member. (TODO: Change this to make it compatible with IPv6).

As an alternative to using a public key distribution system for each client, some client/server applications may be willing to use a password scheme instead. This approach assumes that the clients already have a copy of the server's public key, or will easily be able to get one through a certificate system or other key distribution mechanism (such as that used by Netscape's secure version of http). This allows the client to both authenticate the server and to send private messages to it. A client may have registered its ID and a password with this server on a previous occasion, which the server has verified through other channels. For example, a mail order company could require a user to call in with his or her credit card number, and would assign the user a password and User ID at this time. In this case, this password, encrypted with the server's public key for privacy, can be used to authenticate the client. To make sure the password does not fall into the wrong hands, the client must first authenticate the server before sending her password. This scheme is also supported by Secure RMP. While not as secure as a correctly implemented public key distribution scheme, this policy is much easier to implement in cases where there are many untrusted clients and only a relatively few trusted servers. This system is somewhat similar to that used both by Kerberos and by Netscape. When used in this way, the servers will normally correspond to active members, and the clients will normally correspond to passive members.

RMP supports three types of group members: servers, clients, and listeners. Servers are full members of the group, and receive all messages sent to the group. Servers can also be thought of as peers. Clients do not receive messages sent to the group, but can send messages (typically requests) to the group, and receive unicast replies from one or more members of the group. Listeners can not send to the group, but can receive messages sent to the group with best effort semantics. Because listeners are not guaranteed to receive all messages, groups of listeners scale better in unreliable networks than do groups of servers. A listener may also be a client.

With secure virtual synchrony, SMC can only support servers, because the set of users which hold the group keys must be securely known at all times. When secure virtual synchrony is not being used, all three classes of members can be supported. In this case,

when the group is re-keyed, all of the clients and listeners in the group may not receive the new key. In this case, they will have to re-authenticate themselves to an active member in order to get the new keys.

4. Group Key Management Algorithms

These algorithms are defined in the framework of RMP, although they will work with some other multicast protocols. For secure virtual synchrony, all packets are sent with quality of service (QoS) equal to Totally Ordered, unless otherwise noted. This causes all data packets and membership changes to be delivered to all group members in the same order.

4.1 Notations and Naming Conventions

The algorithms of SMC are described using a set of notations below. In addition, a set of standard naming conventions for group entities is followed.

4.1.1 Notations

A→B(L): M	A sends message M to layer L of B. L can be either S for the security layer or R for the RMP layer.
A_R	The RMP ID for process A
A_U	The User ID for process A
$\{M\}^{Key}$	Message M is encrypted with Key
PubA	The public key of A (using RSA)
SecA	The secret (private) key of A (using RSA)
PubG	The public group key, used to authenticate membership changes
SecG	The secret group key, used to sign membership changes
DES	The group's shared DES key
PubG'	A new public group key, which replaces PubG
SecG'	A new secret group key, which replaces SecG
DES'	A new shared group DES key, which replaces DES
I_A	A random, 64 bit, one time identifier, generated by A
LCR(Op)	A RMP List Change Request packet, sent to the group, requesting that the operation Op be performed to the group membership list
NewList	A RMP New List packet, sent to the group, updating the membership list
"Text"	A constant identifier for the packet or operation type
JoinType	The requested membership type. A combination of {active or passive} plus {server, client, or listener}.

4.1.2 Naming Conventions

A	A process attempting to join or leave a group
B	An active member of the group, performing a membership change
G	The group of secure communicating processes
T	The member of the group which is the token site for a packet

4.2 Adding a New Member

New members act as clients in joining the group. They send requests to the security layer of RMP to add them to the group. If the group is using secure virtual synchrony, they can only be added as servers. Otherwise, they specify which type of member they want to be--server, client, or listener. After requesting to be added, a two-way authentication is performed between one of the active members and the new member. If this step succeeds and they both validate each other, the active member then adds the new member to the group and gives her the appropriate group keys. When called for, DES encryption and group authentication is provided through the standard SMC features described above.

Step 1: Authenticate A to B, and B to A

A→G(S): A_R, I_A This message is sent from the non member (A) to the security layer of the group, using a non-member data packet. It requests that A be added in to the group. One of the active sites will send back a reply to A. The algorithm for deciding which active member performs this is given in the next section. If this operation occurs during a reformation of the group, it may not succeed. For this reason, it should be retried if the protocol does not terminate correctly. At the end of this step, nothing is securely known to either party. The RMP ID of A is actually sent as a part of the RMP packet, and does not have to be duplicated by the security layer.

B→A(S): B_R, B_U, $\{I_A, I_B\}^{SecB}$ This message is a non-member reply sent from B to the security layer of A. When this message is received, A has authenticated B. By returning I_A, encrypted with the private key of B, A knows that B signed the packet. Again, the RMP ID of B is actually sent as a part of the RMP packet, and does not have to be duplicated by the security layer. An intruder can corrupt either or both of B's identifiers, but this can only serve to halt the protocol, not to stop it. Replay attacks are prevented through the use of the one time identifier I_A. Integrity of I_B is maintained by encrypting it with SecB. Note that neither I_A nor I_B are kept private from intruders.

A→G(S): B_R, $\{I_B$, JoinType$\}^{SecA}$

A→G(S): B_R, $\{I_B$, A_U, Password, PubA, JoinType$\}^{PubB}$ There are two forms for this message, depending on whether the public key distribution system supports clients (passive members) or not. This message is a non-member data packet sent from A to the security layer of the group, but specifically addressed to B by the first field. Because this field is sent in the clear, it can be corrupted. However, this can only serve to halt the protocol, not to corrupt it. This packet could also be a unicast sent directly to B, but with RMP it is easier to multicast it to the group. In both types, the JoinType field specifies whether A wishes to be added as an active or as a passive member, and whether A wishes to be added as a RMP server, client, or listener. This field is used by the application layer in its decision on whether or not

A should be allowed to join the group. If secure virtual synchrony is being used, A will be rejected if it requests to be added as a client or as a listener.

The first version of the packet uses A's secret key to authenticate it to B. Replay attacks are prevented because I_B is a one time identifier. The second version of the packet sends A's password and user identifier to authenticate it instead. In this case, A must also send a copy of its public key for use when the shared group key is changed. Again, replay attacks are foiled by using a one time identifier.

Step 2: Update the group with A's status

B→G(S): $\{A_U, \text{Reject}\}^{SecG}$

B→G(S): $\{A_U, \text{PubA}\}^{SecG}$ This message is sent to the security layer of the group, encrypted with the current group RSA private key. This ensures that the sender is a valid active member and guarantees the integrity of the message, but does not keep it private from intruders. The first version of the packet notifies the group that A's attempt at joining the group has been rejected. The second version notifies the group that B has cleared A to join the group, and gives each member of the group A's user identifier and A's public key. If the group is using message signatures and the active members are not fully trusted, then each member of the group should verify PubA with the key authority, to make sure B has not corrupted it.

Step 3: Add A to the group membership list (RMP level)

This step is specific only to RMP. When a group member is added to RMP, it normally sends a List Change Request (LCR) packet to the group, and is answered with a NewList packet which adds it in to the group. When the security layer is used, the active member performing the add also sends the LCR packet.

B→G(R): $\{\text{LCR(Add A)}\}^{DES}$ This List Change Request packet is sent from the security layer of B to the RMP layer of the group. It requests that A be added to the group. B does this on behalf of A to ensure that the membership list is coherent with the set of people who have the DES key. Otherwise, A could decrypt group messages without the senders knowing it was doing this, which would violate secure virtual synchrony. When the RMP layer of each site receives this, it will use the DecryptPacket() callback to decode the packet. After this occurs, the member of the group (designated here as C) which currently holds the token will be expected to send the NewList packet adding this member to the list.

T→G(R): $\{\text{NewList}\}^{DES}$ This New List packet contains the new membership list for the group, encoded with the shared key. Before this packet can be sent, T must first use the ValidateLCR() callback to the security layer to verify that this new member can be added to the group. This callback checks that a valid step 2 packet has been received, validating the addition of A to the group. If this callback rejects the new member (which indicates that B is corrupted or not an active member), T

instead sends out a Reject code in the NewList. Because the packets B sends out are delivered in source order as well as total order, T should always receive the step 2 packet before the List Change Request packet.

Step 4: Update A with the group keys

$B \rightarrow A(S)$: $\{\{I_B\}^{SecB}, DES\}^{PubA}$ {PubG, SAID, SAID Description}^{DES}
$B \rightarrow A(S)$: $\{\{I_B\}^{SecB}, DES, PriG\}^{PubA}$ {PubG, SAID, SAID Description}^{DES}

This is a non-member reply, unicast from B to A. The first field verifies to A that B has sent this message, so that A does not start sending confidential data out with a false DES key. Replay attacks are prevented because this is a one time identifier. The rest of the fields in the first encryption block are group keys, encrypted with A's public key for privacy. The first version is used if A is a passive member, and the second version is used if A is an active member. The second encryption block contains the other information about the group that A needs. This includes the public RSA group key, the SAID identifier for these group keys, and the description of the encryption algorithms that are being used for this group. Among other things, this includes whether or not the group is using secure virtual synchrony.

4.3 Adding a New Member with Secure Virtual Synchrony

If secure virtual synchrony is being used, there are some small modifications that must be made to the above algorithms. The group must be re-keyed after the new list packet is delivered, and the new member must be sent the new keys instead of the old ones. The specific changes follow.

Step 3: Add A to the group membership list (RMP level)

If secure virtual synchrony is being used, the old DES key is used instead of the new one. After the new member is added in with the new list, RMP expects the new member to issue the next ACK for this ring. The new member can not issue this until it receives and authenticates the new list. However, the new list is encrypted or signed with the old DES key. Because of this, the new member will momentarily block the progress of the ring. When the re-key operation completes, the old token site (which sent out the new list) will have the new key. This site periodically sends out the new list, encrypted with the current key, until it sees the ACK from the new member. Once the old token site gets the new key, it will start sending the new list out with this new key. A will then get the new list, join the group, get copies of any other messages it has missed (requested with NAKs, and sent to it with the new key), and send out the ACK. While a bit complicated and inefficient, this is necessary because the re-key operation can not be done until after the new list is sent around. This could be made more efficient by modifying the RMP code some, but the authors felt simplicity and a clean separation of layers was more important. Performance will not usually be seriously affected by this, because of the RMP flow control mechanisms.

Step 4: Update B with the group DES key

If secure virtual synchrony is being used, B sends A the new group keys from the re-key operation below, instead of the old group keys.

Step 5: Re-key the group

B→G(S): Re-key group message This step is only performed if the group is using secure virtual synchrony. B generates a new set of three keys, and re-keys the group. The re-key algorithm is given below.

4.4 Re-keying a Group

Any active member can re-key the group at any time. This can be used to implement a maximum lifetime on any group key. In addition, if the group is using SVS, each membership change entails a group re-key operation.

Step 1: Re-key the group

B→G(S): {{SecureHash}^{SecG}, PubG', {{SecG'}^{PubG}}$^{DES'}$, SAID', SAID' Description, *KeyList*}^{DES}
***KeyList* = ((C,{DES'}^{PubC}), (D,{DES'}^{PubD}), ...) or**
***KeyList* = ((C,{DES'}^{PubC},C1, C2, ...), (D,{DES'}^{PubD}, D1, D2, ...), ...)**

This packet re-keys the group. As soon as the new keys are received at a member, the member should start using them for any new encryption or authentication operations it initiates. Each set of keys are identified using a plain-text ID, and this ID is included in each authentication and encryption operation, to identify which key should be used by a member in decrypting or verifying a message.

The first argument is the signature for the packet, using the old private group key. This is done using the standard individual authentication feature, and verifies that this packet was sent by an active member. The entire packet is encrypted with the DES key for the group, using the standard data confidentiality function. The second argument is the new public key for the group. The integrity of this key is guaranteed by the signature of the packet, and kept private by the DES encryption of the entire packet. The third argument holds the new private group key. It is encrypted with the old public key to keep it private from passive members, and it is also encrypted with the new DES key, to keep it private from any members that hold the old RSA key but have been removed from this list. The last two arguments are the new SAID for the group, and the new description of the encryption parameters for the group.

The KeyList updates the new DES key to all of the members of the current group list. There are two versions of this list. The first version has one entry for each member of the new group, except for the sender. Each entry holds the RMP ID of that member, along with the new DES key encrypted with the public key of that member for privacy. Encrypting each DES key with an RSA key is a time-consuming operation, which may be impractical for large groups. The second

version of the KeyList alleviates this by dividing the work of this encoding among multiple members. Each entry in this key list, in addition to naming a member and providing a new key for that member, also includes a list of RMP identifiers this member must update the new key to. In a group of N members using this feature, the burden to each site can easily be reduced by a factor of $\sqrt{N}$, by splitting the re-key message into $\sqrt{N}$ pieces. If desired, these second level members could also break their sub-list into more pieces, by using the second form of the KeyList themselves.

4.5 Leaving a Group (without SVS)

If secure virtual synchrony is not being used, leaving a group is straightforward. Due to the nature of failure detection and recovery, any member of the group can remove any other. This is required in order to remove failed group members. While this requires the group to trust passive members to not remove other members arbitrarily, this is reasonable given the other trust placed in these members.

Step 1: A requests to leave the group

A→G(R): {LCR(Remove A)}^{DES} This List Change Request packet is sent from A to the group, requesting that it actually leave the group. The current token site (C) will respond to this.

Step 2: The member is removed from the list

T→G(R): {NewList}^{DES} This NewList packet removes A from the list.

4.6 Leaving a Group (with SVS)

If secure virtual synchrony is being used, the group must be re-keyed before the new list packet is delivered. In this case, Step 1 above is replaced with the following.

Step 1: Re-key the group and request that the member be removed

A→G(S): {"Leave", A}^{DES} With secure virtual synchrony, the group keys need to be changed at all sites before the member is removed. Because of this, an active member must perform the remove operation on behalf of the member which is leaving. When the group receives this message, one of the active members (B) will elect to handle this.

B→G(S): Re-key group message This step is only performed if the group is using secure virtual synchrony. B generates a new set of three keys, and re-keys the group. The re-key algorithm is given above. After the NewList packet (in step 2) is received by a group member, that member must use the new key for all messages. If part of the re-keying operation is delegated to some of the other sites, the new key may be received at some sites after the NewList packet. In this

case, these members must delay sending out any data packets until they have received the new key.

B→G(R): {LCR(Remove A)}^{DES} This List Change Request packet is sent from B to the group, requesting that A actually leave the group. The current token site (C) will respond to this.

4.7 Group Reformation Due to Failures and Partitions (with SVS)

RMP allows groups to continue operation even when some of the sites fail. It does this by allowing members to detect that other members have not responded for an extended period of time. When this occurs, the non-responsive members are declared to have failed, and they are removed from the group. This operation is unchanged if secure virtual synchrony is not used, but if SVS is used the group must be re-keyed.

SVS requires that once a NewList packet is delivered at a site, the security layer must use the new group keys that correspond to the NewList. In the case of failures, the NewList packet will be delivered before the security layer is able to re-key the group. Because of this, the security layer will block all outgoing data packets until it receives the new key for the group. Compared to the delay involved in the reformation algorithm, this delay is comparatively small.

Step 1: New group is formed with reformation algorithm

(no message) One of the active members of the group will become B and re-key the group. Until this happens, none of the group members can send out any data packets.

Step 2: Re-key the group

B→G(R): Re-key group message This step is only performed if the group is using secure virtual synchrony. B generates a new set of three keys, and re-keys the group. The re-key algorithm is given above. Once the group members receive the new keys, they can start sending data packets again.

4.8 Selecting an Active Member to Handle Membership Changes

In the above algorithms, when a site requests to join a group, or leave a group that is using secure virtual synchrony, a single active member should respond to this request. This primary site, which is designated as site B in the above algorithms, is selected with the following algorithm.

The first member to create a group is an active member. This member is designated as the primary member (B) until it leaves the group, until it requests that it no longer be the primary member, or until a failure reformation occurs. If B requests to leave the group, and the group is using secure virtual synchrony, B must first request that another site become the primary member, so that this other site can perform the re-key

operation. This request is a simple message to the security layer, encrypted with the group DES key, requesting that the selection algorithm be run.

The selection algorithm is very simple. Whenever a new primary member must be selected, all of the active members send out a totally ordered packet requesting that they be the new primary member. The first request that is delivered to the members elects the new primary member. If a group partition occurs, then the failure reformation algorithm will cause a new selection process to be run in each partition. In order to make sure that old selection messages don't interfere with a newer selection process (say due to a group partition), each selection message includes the current membership list version.

5. Performance

64 bits in DES takes 31 microseconds on a 33 MHz 486. This equates to 260 KB/sec, or 2 Mbps. RSA is around 100 times slower than DES, so it will take 3 milliseconds for each 64 bit block. This means that each DES key will take either 3 milliseconds or 6 milliseconds to encrypt. On faster machines, this will take correspondingly less time. For example, on a 120 MHz Pentium processor, a 64-bit RSA block can be encoded in around 400 microseconds.

The table below shows the extra latency that will be incurred due to the encryption for the group key management algorithms. DES overhead is not included because it is not significant. For the shared re-key operation, each of the 10 members encrypts the DES key for 10 other members. In this case, the latency is equal to the overhead for the primary member plus one secondary member, as the secondary members perform their operations in parallel.

Operation	# of RSA blocks	486-33	Pentium-120
Add passive member, no SVS	17	51 ms	68 ms
Add active member, no SVS	25	75 ms	10 ms
Add member, no SVS, password	39	117 ms	16 ms
Re-key operation with 2 members	14	42 ms	5.6 ms
Re-key operation with 10 members	30	90 ms	12 ms
Re-key operation with 100 members (Shared among 10 members)	30-primary plus 22-secondary	156 ms	21 ms

Figure 3: Encryption overhead for membership operations