INTERNET-DRAFT
draft-irtf-smug-sec-mcast-arch-00.txt

R. Canetti, P-C. Cheng,
D. Pendarakis, J.R. Rao,
P.Rohatgi and D. Saha
IBM

Expires August 1999
February, 1999

An Architecture for Secure Internet Multicast

<draft-irtf-smug-sec-mcast-arch-00.txt>

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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

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

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

ABSTRACT

This document proposes an architecture for secure IP multicast. It identifies the basic components and their functionalities, and specifies how these components interact with each other and with the surrounding systems.

The main design principles followed in developing this architecture are simplicity, flexibility, ease of incorporation within existing systems. In particular, the design attempts to mimic the IPSec architecture, and to re-use existing IPSec mechanisms wherever possible.

The proposed architecture is able to accommodate many of the existing proposals for multicast key management. In this draft, we concentrate on the architectural building blocks required to enable a group member (either a receiver or a sender of data) to use secure IP multicast. Design of the group controller(s) is left to future documents.

Table of Contents

1. Introduction

This document proposes a basic architecture for a secure IP multicast standard. It identifies the basic components and their functionalities, and specifies how these components interact with each other and with the surrounding systems. We start with a brief overview of the security requirements for multicast communication. We then describe some basic design principles followed here. Next, we sketch and motivate the overall design. We also suggest design principles for two main modules: key management and source authentication. Finally, some issues for discussion and related work are presented.

Given the large variance in the characteristics and security requirements of different multicast applications, it is unlikely that a single solution will be adequate for handling all possible applications (see [CP,Q]). Still, we believe that the architecture described here is flexible and general enough to serve as a common ground for most future solutions.

Following the rationale of the IPSec architecture [KEN98c], this document goes beyond making the minimum specifications required for interoperability. It sketches the internal design required of a secure multicast protocol. Such a detailed specification is needed since the overall security of the system often depends on the correct internal operation of all the components.

The current version of this document does not fully specify the internal structure of all the modules. These will be described in subsequent documents.

2. Security requirements from multicast communication

We briefly sketch the salient security requirements. A more detailed discussion appears in [CP].

(a) Group membership control and confidentiality: This means making sure that the group communication is accessible only to legitimate group members. This document concentrates on a "flat" access control structure; that is, all the group communication is accessible to all group members. Hierarchical group membership (as in, say, [HM]) is not addressed here, and can be added at a later stage.

Group membership control is usually implemented by having a group key shared by all group members. All group data communication is encrypted via symmetric encryption using this key. The architecture proposed here follows this approach.

For many applications, group membership is likely to vary over time. It is often required that members leaving a group lose access to future group communication. It is also sometimes required that members joining a group will not gain access to group communication that occurred before they joined.

Note that in order to implement access control it must be possible to authenticate the identity of potential group members. This can be done using public-key certificates of potential members.

(b) Group data authentication: This refers to the process by which a group member is able to verify that the group communication originated from a source within the group. Here the recipient of the communication is not generally able to authenticate the individual source but is able to verify that the communication originated with some member of the group.

We follow the usual approach of implementing group data authentication using a dedicated key shared among all group members. All the communicated data is authenticated via Message Authentication Codes, using the common key.

(c) Individual source authentication: This refers to the process by which group members are able to verify the identity of the sender of

data to the group. The problem of source authentication in multicast is inherently different from the source authentication problem for point-to-point connections since single MAC based solutions (such as those used in IPSec) are not applicable here. (see [CP] for a survey of techniques for achieving this goal).

Other security concerns, like non-repudiability and anonymity, are not addressed here, and will be determined by the specific implementation.

3. Design Goals and Guidelines

Our main design goals are simplicity, flexibility, ease of incorporation within existing systems. To achieve these goals, we follow the guidelines listed here:

(a) Make the design independent from the underlying routing mechanism.

The architecture does not interfere with the routing mechanism of the data. Data packets may be routed via any multicast or unicast routing. For sake of simplicity and modularity of design, we recommend that the key management mechanism assume reliable communication. Yet, no specific mechanism for obtaining reliability is specified. Any reliable multicast or unicast mechanism (e.g., TCP) can be used.

(b) Mimic the IPSec architecture and design as much as possible.

As in IPSEC, we separate the modules that handle data from those that handle key management. Functions such as the generation, distribution and update of cryptographic keys are encapsulated in a key management module. In addition, the key management module is placed in the "application layer" of the communication, and outside the OS kernel. This facilitates application specific operations, such as multiplexing of data for different users and certificate verification. The data handling part lies mostly in the IP layer, using IPSec modules. (Yet, we introduce an additional level of data handling, in the interest of source authentication. See details in the sequel.)

(c) Use existing components wherever possible.

Specifically, we use IPSec's ESP and AH protocols for encrypting and authenticating data. The ESP protocol [KEN98a] is used for exchanging encrypted and authenticated multicast data. If confidentiality is not an issue or if additional authentication beyond what is provided by the ESP authentication mechanism is required (such as authentication of the IP header) then the AH protocol can also be used.

Since IPSec was mostly designed for unicast, there are a few issues that arise when the ESP/AH protocols are used for multicast data. We discuss some of these issues in Section 6.

Multicast-specific packet transformations may be introduced in the future.

(d) Minimize modifications of the OS kernel.

The current architecture is structured so that the new multicast security specific components are kept in the application space and the IPSec components that are currently in the OS kernel can be used

for multicast security without modification. It is expected that with time, some of the multicast specific components may be moved to the kernel and IPSec components already in the kernel would be modified to better support multicast.

(e) Flexibility in the choice of cryptographic algorithms and key management schemes.

By re-using the IPSec design for data transport, we retain the flexibility of using any data encryption and authentication algorithm which can be supported by IPSec.

The new components introduced in this document, namely the Multicast Internet Key Exchange Module (MIKE) and the Source Authentication Module (SAM) provide frameworks which are flexible enough to support most group key management and source authentication schemes.

4. Architecture

This section presents the architecture. We start with a block diagram, and continue to describe the functionalities of the various components, and the data flows.

4.1 Architectural Block Diagram

There are two types of entities involved in Secure Multicast, namely the group members which participate in the Secure Multicast communications and the controller(s) which manage keys for the group or parts of the group and enforce access control. The architecture of the controller(s) depends on the group access control and key management mechanism and is expected to vary widely. Therefore we concentrate on the architecture of a group member where uniformity is desirable and is achievable.

The architecture separates between control and data plane functions. The primary responsibility of control plane functions is to manage group membership and enforce access control privileges. Data plane functions handle multicast data distribution and cryptographic operations such as encryption, decryption, digital signature generation and signature verification.

The block diagram of a secure multicast host is shown below.

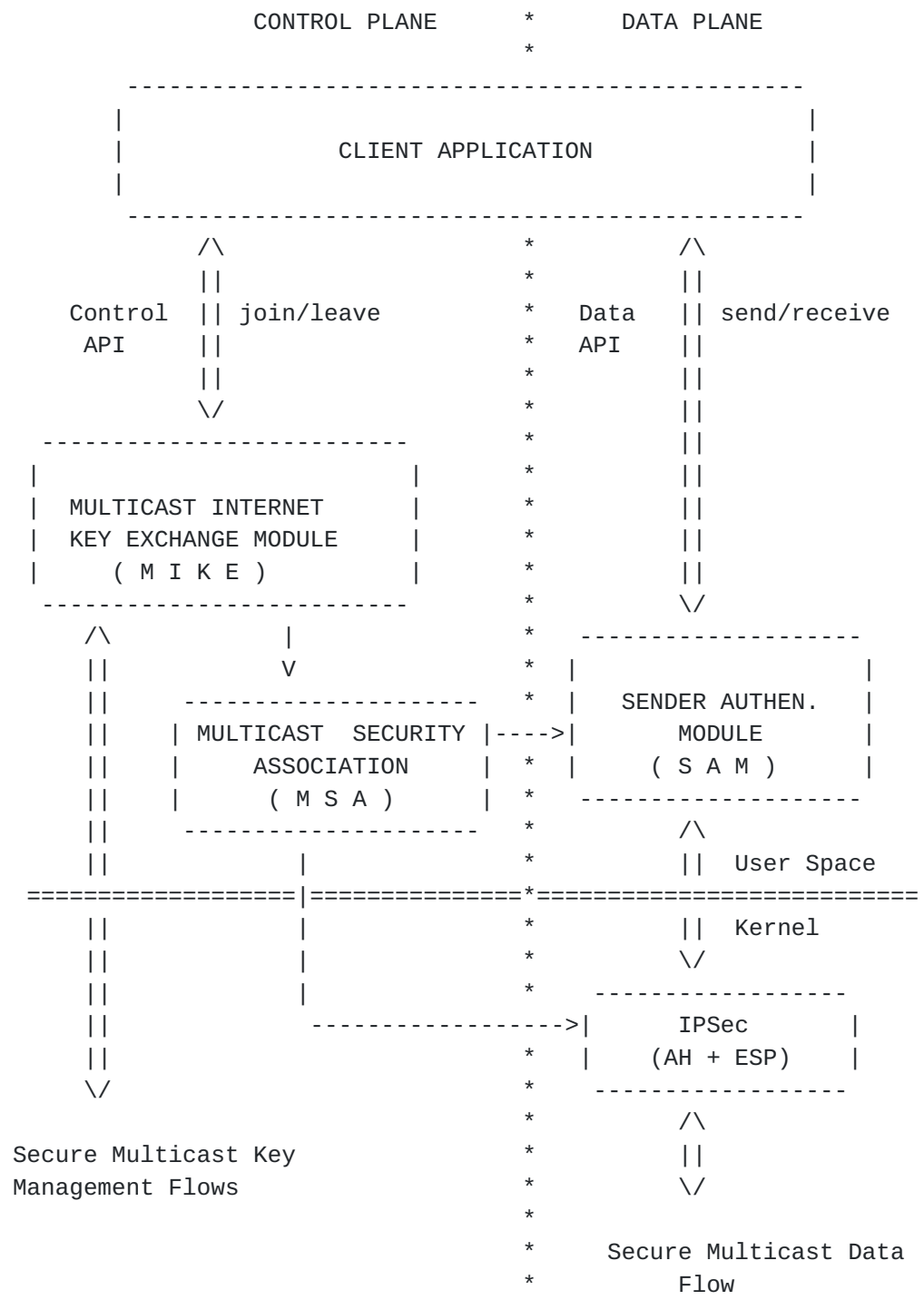

Figure 1: Block diagram of secure multicast framework

4.2. Architecture Overview

We describe the architecture of a member of a secure multicast group. The member could either be a data recipient or a data sender or both. From the application's perspective, the secure multicast framework provides a simple API for using secure multicast. This API is logically partitioned into the Data API which deals with sending and receiving multicast data securely and the Control API which deals with the process of leaving and joining a secure multicast group and the associated access control and key update functions.

4.2.1 Architectural Components

The major architectural components of the Secure Multicast Framework are:

(a) MIKE - Multicast Internet Key Exchange.

This module is responsible for key management and implements the Control API which permits applications to join and leave secure multicast groups. This module resides in the application layer, outside the OS kernel. It interacts with the MIKE modules of the group controller(s), and generates and maintains a Multicast Security Association (MSA) that contains:

(i) Group keys for encryption/decryption and authentication of data (via the AH/ESP modules).

(ii) Signing/Verification keys for source authentication of data by the SAM module, described below.

(iii) Other information regarding the connection, as in an IPSec SA.

In order to make a MIKE specification complete, the MIKE modules within the group controller(s) need to be specified. See more discussion on the design of MIKE in section 5.

An additional functionality of MIKE is periodic updates of the MSA,

whenever group keys or keys used by SAM are changed.

(b) IPSec modules: AH/ESP.

These are the IPSec modules that reside in the OS kernel and deal with encryption/decryption and authentication of data packets. These modules provide Encryption/Decryption and Group Authentication of incoming or outgoing Multicast Data. Data is encrypted with the group key by the sender(s) and decrypted using the same key by receivers. The ESP header remains as defined in the unicast case; the protocol header preceding the ESP header will contain the value 50 in its Protocol (IPv4) or Next Header (IPv6, Extension) field and its destination IP address will be the IP multicast group address, a class D address. Thus, the packet will be forwarded to all members of the group by routers supporting multicast delivery. ESP can be used in conjunction with the ESP authentication option (more on authentication below). In principle, ESP for multicast traffic can be used either in transport or tunnel mode, although transport mode is clearly more adequate in an environment where most participating members are end hosts.

For multicast data authentication different techniques will be used depending on whether group or individual sender authentication is desired. For group authentication, the protocol designed for unicast IP security, namely the IP Authentication Header (AH) [KEN98b] and/or the authentication option within ESP are sufficient. All members share a common, symmetric authentication key which is administered by the group controller and which is used to generate the message authentication code (MAC). The AH header remains as defined in the unicast case; the protocol header preceding the ESP header will contain the value 51 in its Protocol (IPv4) or Next Header (IPv6, Extension) field and its destination IP address will be the IP multicast group address, a class D address. The SPI value is selected, as for ESP, by the controller.

(c) SAM - Source Authentication Module.

This module is responsible for the transformations that enable authenticating the source of received data and possibly for replay protection. Scalable source authentication would typically involve operations that span more than a single packet, both for outgoing and incoming data. (UDP frames are good candidates for a basic unit for authentication.) Thus, it seems reasonable to place this component in a higher layer in the protocol stack, specifically in the application layer above UDP. Another advantage of placing SAM in the application layer is that this way the OS kernel need not be immediately changed.

The internal structure of the SAM depends to a large extent on the source authentication mechanism used. Two basic source authentication mechanisms exist. One is authentication by public key signatures which may be applied on a group of packets via a variety of mechanisms (see [CP]). The other is authentication via symmetric authentication with multiple keys [CGJPMN].

An additional potential functionality of SAM is to provide replay protection for data, in case that the IPSEC replay protection mechanism is turned off because of multiple sender problems. (See Section 6.)

We defer further design details of SAM to future documents.

4.2.2 Data and Control Flows

Typical operation of the system proceeds as follows:

Remark:
Here we assume that the data is sent/received via reliable or unreliable IP multicast. However, the security mechanism described here can be used even when the data is being routed via point-to-point connections, such as TCP.

The member initiates a secure multicast session by invoking the join operation in the Control API which is implemented by MIKE. This enables the member to register in the group as a sender or a receiver or both. Subsequently, the member is able to send and receive datagrams securely to the multicast group using the send/receive functions of the Data API. All group key management, data encryption/decryption and group/source authentication functions are managed by the secure multicast framework and are transparent to the member. If at some later point in time the member decides to leave the secure multicast group then this is done by invoking the leave operation in the Control API. This action eventually results in the termination of the member's ability to securely send/receive messages to the group.

We now outline the data and control flows in more detail.

(a) Control Flows:

- Client Join: The application invokes MIKE to join a multicast group. At a minimum, the application must identify the group that it wishes to join and provide information as to the authentication required (e.g., whether or not source authentication is required and if so, the sources it will trust).

MIKE then performs the process of registering with the group controller(s), sets up a Multicast Security Association (MSA), invokes a standard registration mechanism for the underlying IP multicast group and enables the ESP/AH and SAM modules to start processing data.

The registration process will inevitably include communication with the group controller(s) and this communication will require mechanisms for authentication of the parties as well as confidentiality of the information exchanged. This communication, its authentication and encryption mechanisms should be dealt with within the MIKE module.

At the end of the join process, a Multicast Security Association (MSA) needs to be set up. (See Section 4.2.1 for the proposed contents of the MSA.) The relevant information from the MSA is then pushed to the ESP/AH and SAM modules.

- Key update: Key updates messages are internal MIKE messages and are not part of the high-level architecture. These messages are authenticated and encrypted separately, as specified in MIKE. A special class of key update messages consist of member expulsion messages in which the controller expels the member from the group. The expulsion process is dependent on the specifics of the Key Management Protocol, but should result in the member being cryptographically unable to send/receive messages from the secure multicast group. In this case MIKE module should treat an expulsion message like a member leave request without the need to contact the controller(s).

- Client Leave: First MIKE is invoked to de-register with the group controller(s). Next the Multicast Security Association is deleted (or marked stale). Finally, the host executes the standard procedure for leaving the underlying IP multicast group.

(b) Data Flows

- Sending of data: If source authentication is not needed, then data is transmitted directly via UDP (or a reliable multicast layer) and the IPSec module in the IP layer, with the IP multicast group in the destination address.

If source authentication is needed then the data is first transferred to the SAM. There the data is processed for source verification. (The keys for performing these operations are obtained from the MSA.) Next, the data is directed to the AH/ESP transformations in the kernel. These transformations are executed with the group keys that appear in the MSA. Finally the data packets are sent on the channel in the standard way.

- Receipt of data: Incoming data packets are first being processed by IPSec's AH/ESP transformations in the kernel. There the data is decrypted and group authentication is verified. Next, the data stream is processed by the SAM and source identity is authenticated, if needed. Finally, the data is handed to the calling application.

5. Design Guidelines For MIKE

Although the design of MIKE deserves a separate document, we proceed to suggest some requirements for MIKE, describe an architectural framework that will allow MIKE to meet these requirements and yet be flexible enough to accommodate a variety of group key management techniques. This framework provides an interface for plugging in different group key management modules into MIKE. Next we suggest some design principles for one such pluggable group key management module which we call the Multicast Key Management Module (MKMM).

5.1 Requirements of MIKE

(a) MIKE should support the simple scenario where there is only a single group controller that communicates with all group members. More complex environments where intermediate servers facilitate the communication may also be supported.

(b) MIKE should support having a set of keys (for symmetric encryption and authentication) shared among all group members. In addition, MIKE will help in forwarding the public verification keys of the group controller and of senders in the group, to support source authentication by group members. (Note that these verification keys may be different from the long-term certificates of these parties.)

(c) MIKE will be placed in the application layer of the communication, and outside the OS kernel.

(d) MIKE should be flexible enough to accommodate any reasonable multicast group key management solution.

5.2 Architectural Block Diagram of MIKE

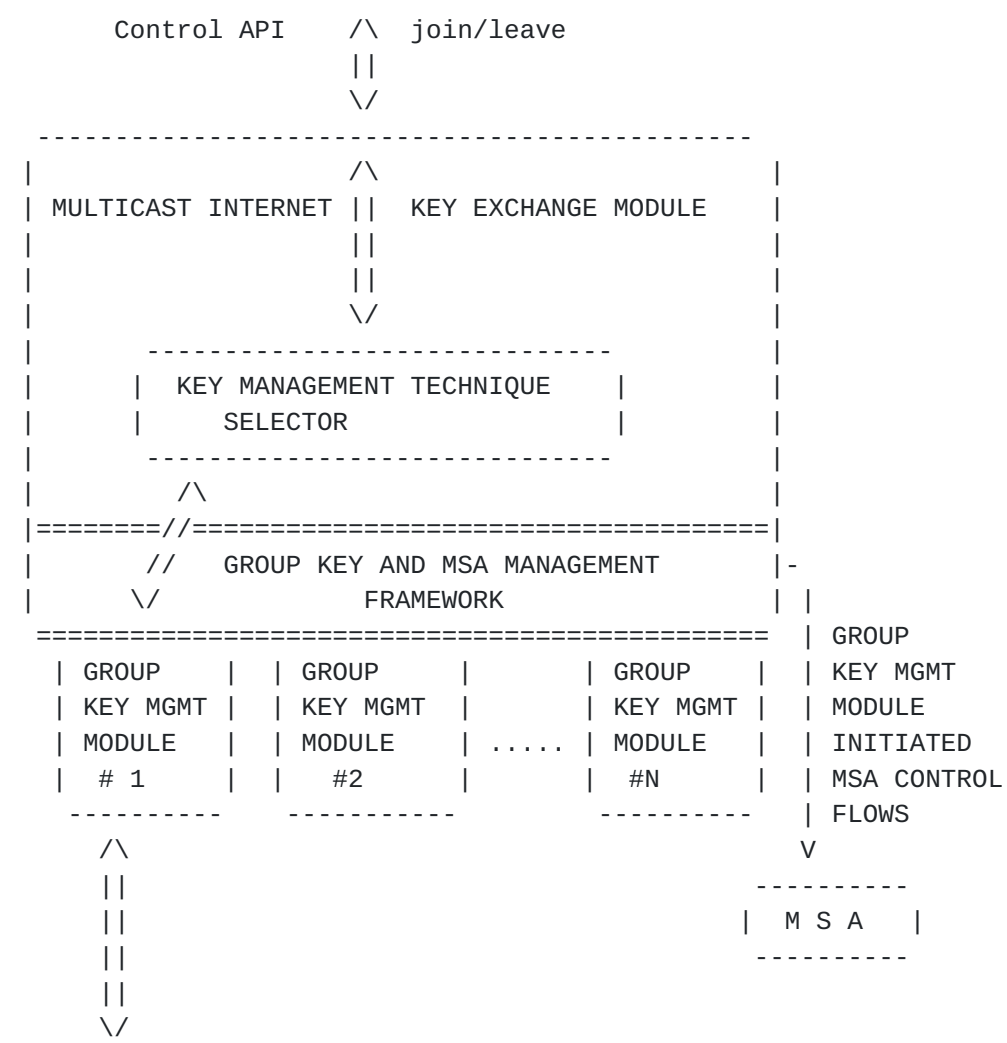

Figure 2: Block diagram of MIKE

5.3 An example Group Key Management Module.

This section suggests a group key management modules that can be plugged into MIKE. We call this suggested module as the Multicast Key Management Module (MKMM).
The main design criterion for MKMM are similar to those for

Secure Multicast Framework, i.e., simplicity and the re-use of existing solutions and standards such as IPSEC and IKE

where possible. This module uses IPSEC to set-up secure channels for all point-to-point between the host and the controller(s). Of course, other solutions that provide secure channels (e.g., SSL or proprietary communication protocols) can be used instead.

(a) Point-to-point communication between a group member and the controller (say, for group registration and de-registration) will be secured via a standard IPSec connection established by IKE. This connection will provide confidentiality as well as authentication of the information exchanged. In particular, the group keys and additional information will be transmitted as DATA in the secure connection. The IPSec SA between a group member and a controller will be short-lived, and will generally NOT last throughout the lifetime of the multicast group.

(b) Key update messages will be transmitted from the group controller to the members using an abstract transportation mechanism, called "Reliable Multicast Shim" (RMS). This abstract mechanism provides reliable multicast, in the sense that any message transmitted via the RMS is guaranteed to reach all group members.

The RMS abstraction can then be implemented via any available reliable multicast mechanism, or alternatively via point-to-point reliable communication (TCP).

(c) Key update messages sent by the controller will have a special format. In particular, they will be authenticated using public-key signatures that are verifiable using a public key that is handed to the members at registration time. MKMM will implement its own signature verification mechanism.

5.4 Architectural Block Diagram of Multicast Key Management Module (MKMM).

The following is the architectural block diagram of MKMM.

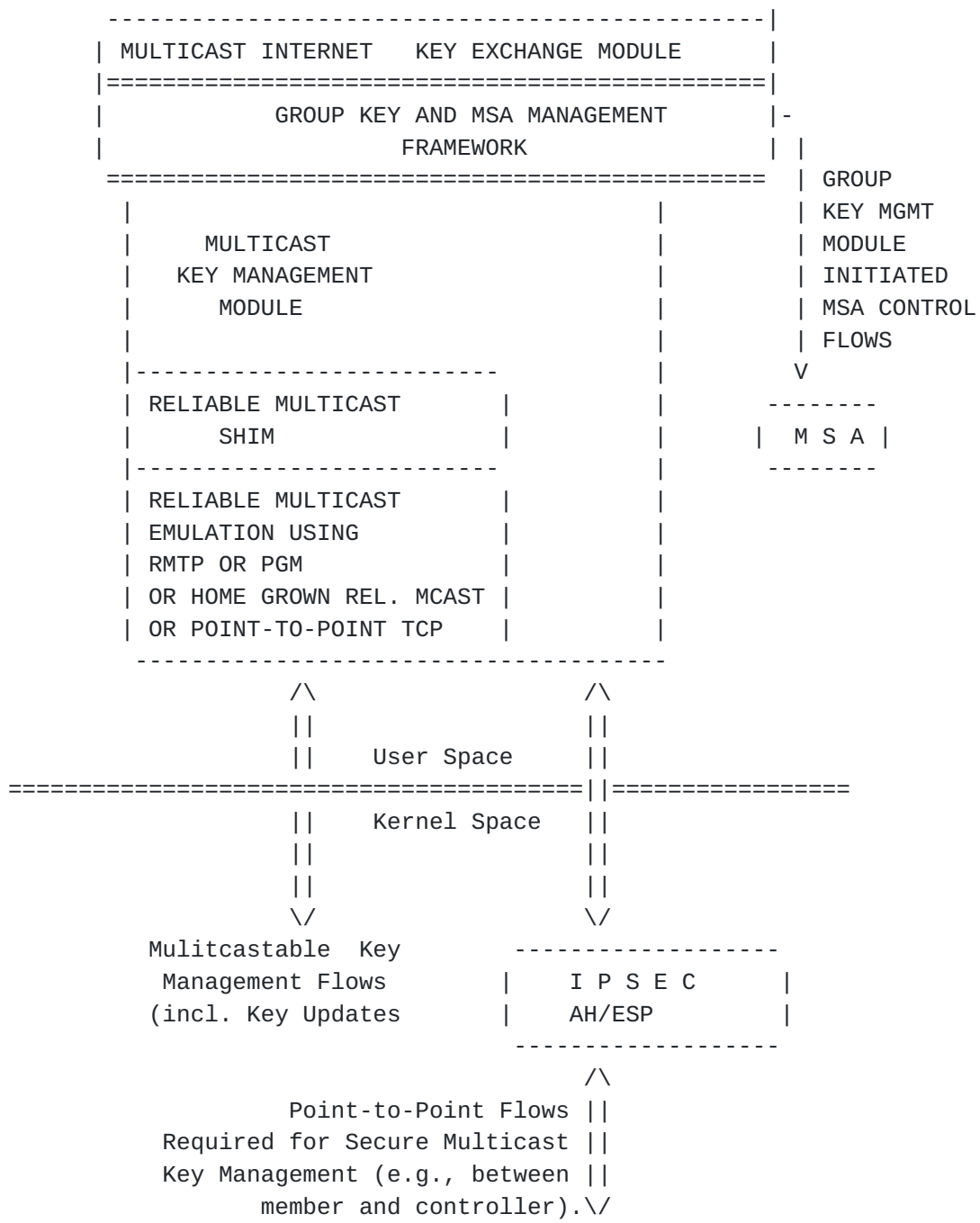

Figure 3: Structure of Multicast Key Management Module.

6. Issues related to the use of IKE and IPSEC

In this section we discuss some issues and apparent problems related to the use of the IPSEC components (IKE, AH, ESP) in a secure multicast protocol. Some of these issues appear to have satisfactory solutions. Other issues are brought up for further discussion.

6.1 Use of IKE

It is clear that group key management is very different from key management for point-to-point communications and therefore the Internet Key Exchange Protocol (IKE) cannot be re-used for group key management. However, it is proposed that the key exchange module for multicast (which we call MIKE, for the Multicast Internet Key Exchange module) should use IKE whenever it needs to establish point-to-point security associations between entities.

Similarly, the problem of source authentication for multicast requires a different solution than the approach used in IPSec. We also propose that the Source Authentication Module (SAM) in this framework use existing standards such as standards for digital signatures and certification wherever possible.

6.2 Identification of Multicast Security Associations

In the Internet Protocol, a Security Association (SA) is uniquely identified by the combination of the destination address, Security Parameter Index (SPI) and the protocol used (e.g., AH, ESP). As stated in the Security Architecture for the Internet Protocol document [KEN98c], the destination address can be either unicast or multicast; the definition of an SA remains the same.

6.3 SPI Assignment

In unicast SAs, in order to avoid potential conflicts of SPI values, receivers are responsible for assignment of the SPI. Since in the multicast case there are multiple destinations, all within the same multicast destination address, such an approach is impractical since it would require coordination by all receivers. Selection by the sender would also be problematic, especially in the case of multiple

group senders.

Within our framework, a reasonable solution to the problem is to utilize the benefits of the centralized controller by requiring that the group controller selects the SPI for each multicast group and communicates it to members, senders and receivers during registration. Selection by the controller guarantees that the SA is uniquely identified by the combination of the SPI value, the multicast group address and the protocol.

As stated in [KEN98c], multiple senders to a multicast group MAY use a single Security Association (and hence Security Parameter Index) for all traffic to that group. In that case, the receiver only knows that the message came from a system knowing the security association data for that multicast group. Multicast traffic MAY also use a separate Security Association (and hence SPI) for each sender to the multicast group, in which case source authentication can also be provided via IPSEC. The assignment of SA's to senders can be done by the group controller.

6.4 Sequence Number Handling and Replay-Prevention

According to the latest ESP and AH drafts, both ESP and AH headers contain a mandatory, monotonically increasing, sequence number field intended to provide anti-replay protection. Processing of the sequence number is at the discretion of the receiver, but the sender MUST always transmit it. The sender's and receiver's counters have to be initialized to 0 when the SA is established and the first packet of that SA will have a sequence number of 1.

In the case of multiple senders using the same security association (and hence the same SPI value) consistency and monotonicity of the sequence number cannot be guaranteed. Hence, as stated in the latest ESP draft, anti-replay service SHOULD NOT be used in a multi-sender environment that employs a single SA. Multicast security implementations should thus ensure that receivers do not perform sequence number processing and verification.

We see two possible solutions to provide anti-replay protection:

(1) Using multiple SAs, one for each sender. (All these SAs may be part of a single MSA.)

(2) Putting anti-replay protection in some higher level module such as SAM. This solution requires application-layer framing of multicast messages.

Further work and debate is needed to decide which solution is best for the first implementations of secure multicast.

6.5 Allowing IPSEC processing of multicast packets

Some current implementations of the IP protocol stack will discard any IP packet with a class D destination address and a "protocol" field that is not UDP. Such implementations need to be changed to support IP-multicast packets protected by IPSEC.

7. Related work

Several recent works suggest mechanisms for IP multicast [HCD,HM,HCM,HD]. (See also the mini-survey in [CP].) Typically, these works concentrate on the key management module (MIKE, in our terminology). Most of these suggest a design that can be incorporated within the architecture proposed here either as a pluggable group key management module under MIKE or in some cases even as a version of MKMM (such as the schemes based on [WHA].) An exception is the [HCD1] design that calls for a hierarchy of group keys for data encryption.

The notion of Multicast Security Association (MSA) is addressed in the IPSec architecture document [KEN98c] and in [HCM], with roughly the same functionality. (Here we introduce the additional functionality of use by SAM.)

8. References:

[CGIMNP] Canetti R., J. Garay, G. Itkis, D. Micciancio, M. Naor, B. Pinkas, "Multicast Security: A Taxonomy and Efficient Authentication", to be presented at INFOCOM '99.

[CP] R. Canetti, B. Pinkas, "A taxonomy of multicast security issues", draft-canetti-secure-multicast-taxonomy-01.txt, Nov. 1998.

[HCD] Hardjono T., Cain B., Doraswamy N., "A Framework for Group Key Management for Multicast Security", internet draft draft-ietf-ipsec-gkmframework-00.txt, July 1998.

[HCM] Hardjono T., Cain B., Monga N., "Intra-Domain Group Key Management Protocol", internet draft, draft-ietf-ipsec-intragkm-00.txt, November 1998.

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

[KEN98a] Stephen Kent, Randall Atkinson, "IP Encapsulating Security Payload (ESP)", Internet Draft draft-ietf-ipsec-esp-v2-06.txt, July 1998.

[KEN98b] Stephen Kent, Randall Atkinson, "IP Authentication Header", Internet Draft draft-ietf-ipsec-auth-header-07.txt, July 1998.

[KEN98c] Stephen Kent, Randall Atkinson, "Security Architecture for the Internet Protocol", Internet Draft draft-ietf-ipsec-arch-sec-07.txt, July 1998.

[Quinn] Bob Quinn, "IP Multicast Applications: Challenges and Solutions", draft-quinn-multicast-apps-00.txt, Nov 1998.

[WHA], D.M. Wallner, E. J. Harder, R. C. Agee, Key Management for Multicast: Issues and Architectures, , internet draft draft-wallner-key-arch-01.txt, September 1998.

9. Authors Addresses

Ran Canetti
EMail: canetti@watson.ibm.com

Pau-Chen Cheng
EMail: pau@watson.ibm.com

Dimitris Pendarakis
EMail: dimitris@watson.ibm.com

J.R. Rao
EMail: jrrao@watson.ibm.com

Pankaj Rohatgi
EMail: rohatgi@watson.ibm.com

Debanjan Saha
EMail: debanjan@us.ibm.com

IBM T.J. Watson Research Center
30 Saw Mill River Road
Hawthorne, NY 10598, USA
Tel: +1-914-784-6692