Key Management in IP Multicast

Petri Jokela
Helsinki University of Technology

petri.jokela@nomadiclab.com

ABSTRACT
The IP networking was originally designed to operate in point-to-point way. However, when the same data is delivered to multiple receivers unicast is not very efficient because the data is multiplied already at the original source. In the late 1980's, first data transmission protocol was introduced that provided data multiplication in the routers optimizing the amount of data. During 1990's, when the success story of the Internet began, the need for security emerged also in multicast. Since then work on different security issues has been going on, related to source and receiver host authentication, data integrity and encryption. This paper describes the current status of the multicast key management work in the IETF, discusses missing features, and proposes an enhancement to an existing protocol.

1.INTRODUCTION

1.1 Multicast technology

IP networks were originally designed to deliver data between two nodes connected to the network. This model of transferring data is not optimal when the same data is transferred from one source to multiple recipients. When unicast is used, the source has to open a unicast connection to each of the recipients. The same data will be transmitted multiple times over the same links.

In late 1980's, this problem was tackled with a new proposal; IP multicast [1][2]. This proposal defined a method to deliver data from multiple sources to all hosts that were willing to receive the same data, allowing also data to be multiplied closer to the destination hosts. This saves considerable amounts of network resources, especially when the number of receivers is high.

The introduced Any Source Multicast (ASM) system didn't give any possibility to control the source of the data, thus anyone could send data to the same multicast group (i.e. to a specific multicast IP address defining a group). The Source Specific Multicast (SSM) [6] protocol was introduced to give a solution to this problem; the data channel is identified using both the destination multicast address and the source IP address of the data packet. Basically only one node could send to the multicast channel.

The described SSM is not a bullet-proof system. An attacker can fake the source address of an IP packet and use the IP address of the real source node's address instead of its own. Now these packets can be injected in the transmission and they are delivered to the receivers who are unable to verify if the sending node is the correct one.

The SAM Research Group in the Internet Research Task Force (IRTF) is currently working on research issues related to multicast development. The IP multicast has not become a very popular technology even though there seems to be a need for multicast applications. There are, not necessarily only technical, problems to get all IP routers to support multicast traffic. If all routers do not support multicast, it cannot be used in cases when the traffic should cross this area not supporting multicast. SAM RG is trying to find solutions to these kinds of problems. There are ideas presented that propose application and overlay multicast systems, as well as hybrid models where multicast is partly handled by IP multicast and partly by other, higher layer solutions.

1.2 Multicast security issues

While the number of Internet users have grown, it has also pushed the work on IP security. The security consists of authenticating the multicast stream source node, providing integrity protection for data, and providing secrecy for data.

One solution provided by Internet Engineering Task force (IETF) for the source node authentication is the Timed Efficient Stream Loss-Tolerant Authentication (TESLA) [12] protocol. In TESLA, the receiver can authenticate each packet's source node and verify the integrity of the data.

The data has to be protected against modifications while it is transferred. Even though the data can be plain text and readable by others, the receiver has to be sure that the data is not modified during transmission.

If the nature of the data is such that it must not be revealed to parties not allowed to see it, it has to be encrypted. This data secrecy is one part of security considered in multicast.

Setting up security associations between two nodes is easy and there exists multiple protocols that can be used to do this. However, when multicast transmission is protected it is not feasible to build separate security associations between each of the receivers and the source node. In this case the properties of multicast technology could not be used, but all data would be delivered using unicast connections. New methods for building multicast security associations are required and the IETF Multicast Security (MSEC) working group (WG) has been working on these issues. The currently used method is to share a data encryption key between all participating nodes and use a centralized control point for handling the key distribution as well as client access. The entity handling this is called the Group Control Key Server (GCKS).

The nature of multicast applications vary and depending on their nature different security protocols may be needed and they may be used in different ways. Multicast groups can be short-lived or long-lived with either stable set of clients or all the time changing set of receiving members. Groups can be large or small. The actual data may have different kinds of requirements for protection. For example a multicast sending from a sports event doesn't necessarily suffer if it can be viewed a short time without encryption, but the same cannot happen in a multicasted corporate meeting session.

Security for data transmission can be provided on different layers, e.g. on application layer. In this document, however, we concentrate on IP layer security.

The rest of the paper is organized as follows. In section 2, the current multicast security architecture is presented. Section 3 describes the current status of key exchange protocols that have been standardized in the Internet Engineering Task Force (IETF). In section 4, the Host Identity Protocol (HIP) as well as the HIP registration protocol are shortly presented. Section 5 identifies possible ways to improve the current protocols. Section 6 presents a new way to use HIP as the registration protocol for Group Domain of Interpretation (GDOI) key management protocol, and finally section 7 identifies some items for future work.

2. MULTICAST SECURITY ARCHITECTURE

2.1 History

Multicast technologies have been under standardization since late 1980's in the Internet Engineering Task Force (IETF). During the years new standards have been introduced and the work has been going on developing more efficient and secure solutions.

In 1996, RFC1949, "Scalable Multicast Key Distribution" defined the first IETF standardized version for multicast key management. The basic assumptions were that unicast keying mechanisms cannot be used; each of the receivers cannot negotiate a security association with the source node individually. Thus the decision was to select only one security association that is shared with all participants.

Currently there are few working groups creating multicast related standards; the PIM Working Group (WG) is developing IP multicast data transmission technologies, the RMT WG has focus on reliable multicast transmission protocols and the MSEC WG concentrates on security issues in multicast technology. In addition, SAM Research Group in the Internet Research Task Force (IRTF) is working on more research oriented protocols for trying to find new solutions that would help deployment of multicast also in cases when IP multicast is not supported everywhere.

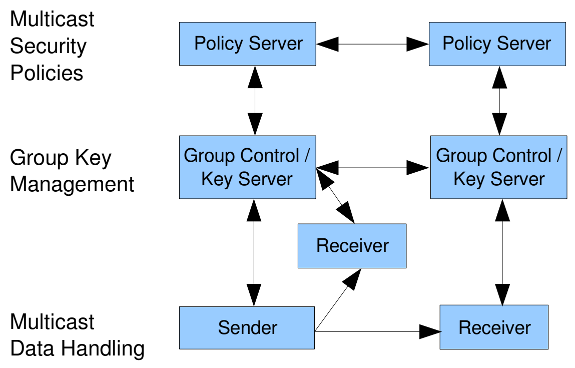


Figure 1: Distributed Multicast Security Reference Framework

The initial goal of the MSEC WG was to provide a security architecture for multicast, define group key and policy management architectures, as well as to specify protocols for source origin authentication, group key management, and group policy management. The first target is to create such protocols for sessions with single source and a large number of recipients. Future work will include other types of sessions.

2.2 Security Architecture

In IETF the multicast security architecture is based on a centralized function for providing both the source and receiver nodes authentication and authorization as well as the distribution of the required keying material. Introducing functions that are separated from the source node and providing the possibility to distribute these functions provide a scalable environment for authentication, authorization and key management.

In [4] the current high level architecture for multicast security is introduced. The basic architecture consists of few different functions; Policy Server (PS), Group Control (GC) and Key Server (KS), receiver, and sender. The PS as well as GCKS functions may be distributed to achieve more efficient operation in the multicast key handling. Figure 1 shows the model with distributed PS and GCKS entities.

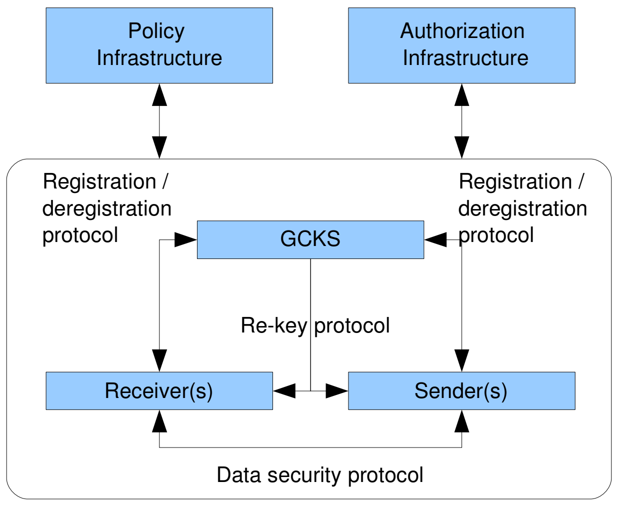
The architecture specification defines three different areas: multicast data handling, group key management, and multicast security policies.

The data handling covers the data encryption and source origin authentication and integrity (or possibly only group authentication with the restriction that if integrity is required all group members have to be trusted.)

The multicast security policies is responsible for creating and managing the policies specific for a certain multicast group. The PS, implementing the required functionality, interacts with the GCKS entity.

In group key management area the GCKS handles functions related to end-host authentication and key management. The end-host may be either the receiver or sender in the multicast group. The two functions in this entity are not necessarily implemented in the same place. End-hosts are authenticated and authorized by the GC function, and they both have to interact with the KS for receiving initial keys as well as new keys when re-keying for the group is necessary.

The security architecture is further developed in [5]. The specification defines the framework for registration protocol used between end-hosts and GCKS, re-key protocol used for managing keys at end-hosts, and a Group Security Association (GSA) defining the data security properties between the sender and receiver.



Policy Infrastructure
Authorization Infrastructure
Registration / deregistration protocol
GCKS
Registration / deregistration protocol
Re-key protocol
Receiver(s)
Sender(s)
Data security protocol

Figure 2: Group Security Association Model

Figure 2 shows the associations defined in [5]. The registration association is established between the end-host and the GCKS. Both receivers and senders have to register to the GCKS. Using the registration association protocol the end-host is registered as a participant in a multicast session. The GC function both authenticates the end-user and verifies if it is authorized to receiver the stream or to send to the stream. After positive authorization, the KS can provide required keying information to the host.

The re-keying association is used to provide new keying information for the end-hosts. Re-keying may be done periodically to avoid revealing too much information for an attacker to do statistical analysis on keys, and in cases when there are changes in the set of multicast receivers or senders.

Data security association is the one that is used to protect transmitted data between the source and destination nodes. KS defines the keys required for data protection, as well as the Security Parameter Index (SPI) that must be used for this SA. The IPsec [18] specification defines the issues that have to be handled at the IPsec client to support ESP and AH for multicast.

The Group Security Association is the combination of all security related parameters as well as needed keys for all three types of security associations.

3. IETF MULTICAST KEY MANAGEMENT

The protocols described in this section use the architecture described in the previous section. All of these protocols have been specified in the IETF and define a method for cryptographic key management for multicast protocols.

These protocols do not define any data transmission related protocols. They are defined in different specifications, like Protocol Independent Multicast – Sparse Mode (PIM-SM) in [11]. It is assumed that the multicast tree building happens after the end-host authentication and authorization has been done and initial GSA has been set up at the client side.

Multicast key management is currently being defined in IETF MSEC working group. The original target of the WG was to create protocols for systems that have single source and large number of receivers. During the time the WG has existed, three key management protocols have been defined. They are slightly different from each other providing different properties and they are targeted to be used in slightly different scenarios.

The GSAKMP protocol defines a protocol controlling everything from policy management to re-keying. MIKEY protocol has a slightly different target; it is designed to be used in heterogeneous networks, providing more efficient ways for key management than other defined protocols. The Group Domain of Interpretation protocol uses the ISAKMP phase 1 negotiation as the authentication protocol and for setting up a secure connection between a receiver and the GCKS system. The GDOI itself defines the phase 2 messages.

3.1 Logical Key Hierarchy

One of the early multicasting architecture specification, “Key Management for Multicast: Issues and Architectures”, [3], was published in 1999. It describes early visions of the key management architecture and defines also a Logical Key Hiearchy (LKH) where Key Encryption Keys (KEK) are organized in a hierarchical manner optimizing re-keying operations. The document doesn't define a specific protocol for key management, but it can be used together with the GSAKMP and GDOI protocols, defined in the following subsections.

When there is no hierarchy, we can either deliver new Traffic Encryption Keys (TEK) to each of the nodes using the point-to-point connection between the server and the client, or then we can use one KEK, shared by all nodes, to encrypt the delivered TEK. When using only point-to-point connection the new data encryption key has to be delivered when a new node joins or an old one leaves. If we use a KEK, we still need to do the encryption operation and sending operation to each of the nodes, because the joined or leaved node is not allowed to have access to any future or past KEK encrypted data key packets, and we have to deliver a new KEK to each of the clients.

When a multicast group contains a hundred thousand nodes, the number of encryption operations and message transmissions would be huge without hierarchy. By adding few levels of hierarchy to keys, a significant amount of cryptographic operations, as well as message transmissions can be saved when compared to non-hierarchical models (see Table 1).

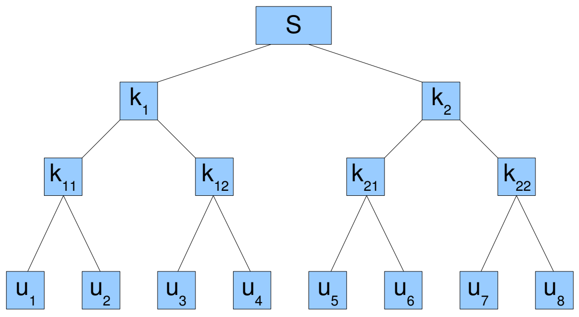


Figure 3: LKH model

The LKH model, as depicted in Figure 3, shows the keys in the LKH hierarchical model. Each node u_n has the traffic encryption key TEK, and all keys that are on the path towards the server S and a host based key encryption key, e.g. host u3 maintains TEK, k_1, k_{12}, and k_{u3}. When a node leaves from the system, e.g. node u_4, part of the keys need to be updated. First, the TEK cannot be used any longer to protect data and also keys known to the leaving node (k_1, k_{12}) have to be updated.

For nodes 5 – 8, the key k2 can be used to encrypt the new TEK. Thus a single encryption and a single message is enough to update that side of the graph. For nodes 1 and 2 the k_{11} can be used to encrypt the new k_1 and TEK. For node 3, the k_{u3} is used to update TEK, k_1, and k_{12}. After updates, node u_4 has no access to future key information.

Table 1 [3] shows the performance of the LKH system in case of re-keying. The degree describes how many underlying keys there are under one parent key, storage the number of keys required to be stored by a user, and re-key transmissions the number of messages that are needed to be transmitted from the key server. The first (single) assumes that each key is transmitted in separate messages, and the second (multi) the case when multiple keys are packed into one message.

Users	*Degree*	*Storage per user*	*Re-key transmissions (single)*	*Re-key transmissions (multi)*
8	2	4	5	3
9	3	3	5	4
16	2	5	7	4
2048	2	12	21	11
2187	3	8	20	14
131072	2	18	33	17
177147	3	12	32	22

Table 1: Efficiency of LKH

3.2 Key Management Protocols

In the following subsections, the multicast key management protocols that have been defined in the IETF are presented.

3.2.1 Group Security Association Key Management Protocol (GSAKMP)

The GSAKMP [10] defines methods for policy distribution, policy enforcement, key distribution, and key management. The following roles are identified:

- Group owner (GO) – responsible for creating policies
- Group Controller / Key Server (GCKS) – responsible for authenticating Group Members, enforcing policies, distributing and managing keys
- S-GCKS – same responsibilities as GCKS in case the functions of GCKS have been distributed
- Group Member (GM) – responsible for verifying that all security related actions are authorized and to use group keys properly.

There are three different configurations defined in the GSAKMP: Only one GM is the sender (default), all GMs can send (this function has to be supported in all implementations), or a subset of GMs are senders (implementations can optionally support this mode).

The GSAKMP supports distributed GCKS operations to enhance scalability. Distributed operations can also provide better GM management locally when the S-GCKS can be better aware of the GMs that it can serve. This can be case e.g. in a corporate network.

The assumption in GSAKMP is that there is one or more trustworthy PKI (Public Key Infrastructure) for the group. The PKI is used for creating and verifying security policy rules. The GO public key must be known by all the GMs.

When a GO creates a new multicast group, it initiates the process by creating a Policy Token (PT) describing the rules for access control and authorizations for a group. The token is signed by the GO. The token contains:

- Identification for the PT and group
- Access Control rules defining who can have access to the group keys
- Authorization rules describing who can be a S-GCKS
- Mechanisms for handling security. E.g. in IPsec this could include configuration data for Security Policy Database / Security Association Database
- Source authentication of the PT to the GO

Once the PT is created and signed, the GO will send it to the GCKS. The GCKS verifies the signature of the PT and checks if it is itself valid to act as a GCKS for this group based on the information in the PT.

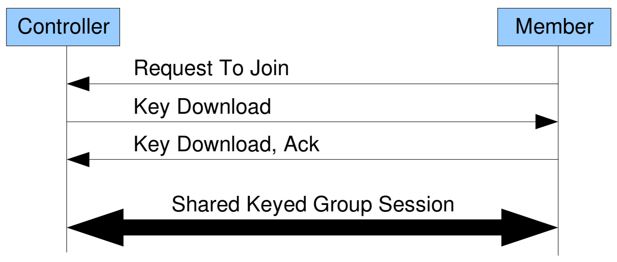


Figure 4: GM registration in GSAKMP

After the group has been formed, each GM registers with the GCKS (see Figure 4). When the GCKS receives the request to join the group it has to verify the signature of the GM for determining its identity and the authorization information for joining the group. Once these requirements are met, the GCKS can send key download message to the GM. The GM in turn has to verify that the GCKS is a valid key management party in this group. Using the keying information received the GM can setup required security associations for re-keying and data.

The negotiation is slightly different when the GM is not going to send data to the group, but will act only as a receiver. In that case there is no Request To Join message, but the Key Download is pushed to the GM immediately after the registration protocol has been run.

To support better scalability, the distributed operation of the GCKS is supported. After joining the group, the GM may become a S-GCKS, providing GCKS services locally if the policy in the PT allows that and the GM is capable for this role. The S-GCKS can manage the group control functions and key encryption keys, but the traffic encryption key is always handled by the root GCKS.

Re-keying is needed when GMs join or leave. The re-keying operation goes always via the GO. Each node change is sent to the GO which creates a new PT. The new PT is distributed among all GCKS nodes. When a GCKS or S-GCKS receives a new PT, it must verify if there have been changes among its own GMs. If not, the new PT is distributed as described in the LKH using a group key. However, if there have been changes the new PT has to be delivered for each of the clients using their host-based keys.

The re-keying operation requires that IP multicast tree for data has been set up or that there exists some other method for distributing the new keying information, such as an overlay network.

3.2.2 Multimedia Internet Keying

MIKEY defines key management functions. The sender can act also as the GCKS, so that a separate GCKS node is not necessarily needed. The GC functionality, authenticating the user, is done during an initial key exchange with signed messages.

The MIKEY protocol has been defined especially for real-time applications. It can be used both for point-to-point communication and group communication.

Requirements for protecting real-time data, transferred using RTP, had been rising in IETF. To support the key management for Secure RTP (SRTP), the MIKEY protocol was designed. This protocol tries to provide a lower latency, better usage over heterogeneous networks, and better performance for small interactive groups.

For distributing traffic encryption keys, the MIKEY protocol uses either pre-shared keys, public key encrypted key transfer, or Diffie-Hellman (only for peer-to-peer connections).

Whe public key or pre-shared key method is used, the Traffic Encryption Key Generation Key (TGK) is a shared information between all hosts participating the session. In case of Diffie-Hellman key sharing each client establishes a point-to-point connection with the source (or separate GCKS node). In the latter case, the TGK is different for each client – GCKS pair.

Authentication in MIKEY is based either on pre-configuration, for example in case of pre-shared keys. Another possibility is to use certificates signed by a trusted CA.

Re-keying in MIKEY is not actually defined and it is defined that the MIKEY protocol is run again when re-keying is needed. However, extensions have been done to MIKEY to allow multicast re-keying in certain environments, e.g. in Multimedia Broadcast / Multicast Service (MBMS) [17].

3.2.3 The Group Domain of Interpretation

The GDOI [8] specification describes a ISAKMP [7] domain of interpretation for group key management. The ISAKMP specification, however, has been obsoleted by “Internet Key Exchange (IKEv2) Protocol” specification [13], but in relation to GDOI, we still use the ISAKMP specification terminology.

ISAKMP defines a two-phase negotiation for establishing secure connections. The ISAKMP phase 1 negotiation (e.g. IKE [13]) defines security association creation between two nodes, including end-host authentication. The GDOI is targeted to take advantage of the ISAKMP phase 1 and leave the initial authentication and secure connection setup for that protocol. During the phase 1 the client host establishes a secure unicast connection with the GCKS and the phase 2 negotiation, specified in GDOI, is run over this established secure connection.

The phase 1 negotiation establishes the so called registration association between the end-host and the GCKS. In GDOI specification the phase 2 negotiation is defined and during the negotiation a re-key and/or data security associations are created. The host authentication is based either on pre-shared keys or public key cryptography as defined in IKE [20].

The GC maintains authorization information and it may use either the identity provided during the phase 1 negotiation, or a separate identity information provided in a certificate during the phase 2 negotiation. The certificate has to be provided by the group owner.

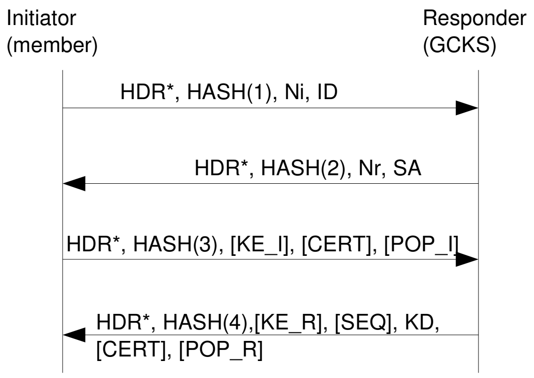

Initiator (member)
Responder (GCKS)
HDR*, HASH(1), Ni, ID
HDR*, HASH(2), Nr, SA
HDR*, HASH(3), [KE_I], [CERT], [POP_I]
HDR*, HASH(4),[KE_R], [SEQ], KD, [CERT], [POP_R]

Figure 5: GDOI GROUPKEY_PULL exchange

The GDOI specification specifies two formats of key transmissions: GROUPKEY_PULL and GROUPKEY_PUSH. The GROUPKEY_PULL is initiated by the client node and it is used to retrieve the TEKs and/or KEK (or KEK array, if LKH is used).

Both hosts generate nonce values that they expect to receive in the following messages included in the hash calculations. This ensures that the messages are not replayed, but are freshly generated by the sender. The hash algorithm is selected during the registration negotiation (e.g. SHA-1).

In Figure 5 the GROUPKEY_PULL message exchange is shown. Messages one to three are used to provide a proof of liveness, and protect against replay attacks. HASHes include nonce values provided by both the initiator and responder, except HASH(1) which contains only the nonce of the Initiator. In message four, the actual keying information is transmitted in the KD parameter.

In message 1, the Initiator sends the GDOI message, containing the message header (HDR*), nonce value selected by the Initiator, and the ID defines the multicast group that the client wants to join. The hash field contains a hash calculated over the SKEYID_a (a shared secret negotiated during the registration phase), message id from the header, nonce, and multicast group identifier. With the hash the hosts can prove that they know the shared secret (SKEYID_a) from the registration phase.

The message 2 contains Responder's nonce and security policy information, for example Security Parameter Index (SPI). No key information is transferred at this phase. The hash contains both nonces, message id, and SA information.

Message 3 finalizes the liveness check. If perfect forward secrecy is desired, the optional KE_I parameter is included in the message containing Diffie-Hellman parameters. The responder generates its Diffie-Hellman parameters (KE_R) and the KD parameter in message 4, containing the actual TEK and KEKs for the multicast session, is XORed with the newly generated shared secret.

It is also possible that the Initiator uses some other identity for the multicast session than the one provided during the registration. In this case it includes the optional certificate in the CERT parameter giving the identity (public key) used for this multicast session. If CERT is included, also the Proof of Possession (POP) payload has to be added, containing the signature verifying that the sender maintains also the private key corresponding to the public key in the CERT.

The GROUPKEY_PUSH can be used by the GCKS to update either the Re-key SA or the Data SA. This message is initiated by the GCKS and it forces the key update on needed SAs. The PUSH message is a single message containing security association information, including all needed KEKs and TEKs, an optional certificate for verification of the signature, and a signature.

4. HOST IDENTITY PROTOCOL

The following subsections define the Host Identity Protocol (HIP) basics as well as the registration extension defined for HIP for supporting service registrations during the base HIP negotiation.

4.1 Host Identities

The Internet hosts have traditionally been identified using the IP address. The problem with this identification is that the IP address describes also the location of the host. This is a problem for example when the end-host is mobile. A host changing the location changes the IP address (locator) but the identity of the host still remains the same and should not be changed.

In Host Identity Protocol [14] the location information is separated from the host identity information by introducing a new name space for host identities. Each host generates a public private key pair for host identification purposes. The host identity (HI) is the public key of the key pair.

To support existing APIs, a HIT is used as a representation of the HI. The HIT is 128-bit hash of the HI. Actually, the HIT contains (at HIP's experimental phase) a 28-bit prefix to separate it from actual IPv6 addresses and a 100-bit hash over the HI.

The HIT can be given to applications that are resolving the address information of a HIP host. The HI layer creates a dynamic mapping between the HI of the peer host and its IP address. Thus, when the application opens a socket towards the peer node, it uses the HIT of the peer host. The HI layer is responsible for mapping the HIT information in the packet to the actual IP address information for routing purposes.

For IPv4 applications, a local scope identifier (LSI) is introduced. This is, as the name says, only local scope and the peer host is not using the same value. The LSI is a 32-bit long value. The mapping on the HI layer is done similarly as in case of HITs.

In addition to the security, separation of locator and identity information provides an easy way to handle mobility and multihoming. While the upper layers do not see the IP address at all and the association between the host identity and the IP address is dynamic, the mapping can be easily changed when hosts move and change IP addresses.

4.2 Base exchange

HIP defines a four-way handshake, a so called base exchange (BEX). During the BEX hosts mutually authenticate themselves using public key cryptography and generate keying material using Diffie-Hellman method. The keying material can be used for various purposes, for example to ESP Security Association to protect data traffic between hosts.

In Figure 6, the HIP base exchange is presented. The roles of the nodes are Initiator and Responder. The Initiator is the host that sends the first HIP message, thus being the connection initializer. The I1 message is basically just a hello message telling that a HIP negotiation is requested. The packet is not signed nor protected using any other means.

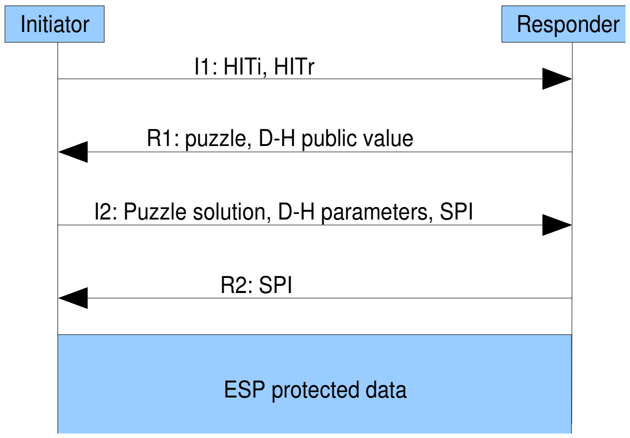

Initiator
Responder
I1: HITi, HITr
R1: puzzle, D-H public value
I2: Puzzle solution, D-H parameters, SPI
R2: SPI
ESP protected data

Figure 6: Host Identity Protocol: Base Exchange

The Responder will respond with an R1 packet. The packet can be precalculated and sent directly to the Initiator. The packet contains the Diffie-Hellman values for the Responder and a puzzle that the Initiator has to solve before the negotiation can be finalized. At this phase the responder doesn't create any state, it can safely forget that someone has tried to initialize a HIP negotiation. The puzzle is created so that at the verifciation phase the Responder can get the needed information without storing too much data.

The Initiator solves the puzzle, selects its own Diffie-Hellman public value, creates an I2 packet and sends it to the Responder. The Initiator has already the common keying material and it could basically build the ESP SA already, except that the SPI value for the Responder is still missing.

When the Responder receives the I2 packet, it verifies the puzzle and checks the signature in the packet. If everything is ok, it can create the ESP SA. It has to select its SPI value, which it will insert in the created R2 packet and send to the Initiator. Initiator, once it receives the R2 packet, finalizes the security association.

R1 and I2 packets contain signatures that are used to verify that the sender “owns” the public key (HI) that it has delivered. Thus after the negotiation the hosts can be sure about the identity of the peer host.

4.3 HIP Registration Protocol

The HIP Registration protocol provides means for a HIP host to register to a service. The service can be provided by a server or by a middle-box. The document does not define how the service is detected; it is assumed that the service detection is handled using other protocols.

The registration protocol defines new parameters that are used for transferring service inforamtion between nodes. First, the service providing node informs the client about services that it is willing to provide for this client. In the next phase, the client sends a registration request containing service that it wants to use, and finally the service providing node confirms the selection. The registration is either included in the HIP BEX messages where the R1 message contains list of provided services, I2 contains the selected service, and finally R2 contains the confirmation (See Figure 7, where bolded parameters are registration parameters). It is also possible to use UPDATE message exchange when there exists already an association between the client and the service providing node. In this case, the service information parameters are included in UPDATE message exhange.

5. POSSIBLE IMPROVEMENTS

The current multicast keying methods are heavily tied to the GCKS system. Currently there does not exist any suitable negotiation for creating group keys between participants especially when the number of participants is very large thus a centralized method is needed for efficient operation.

In the presented methods, the data encryption is based on shared keys, i.e. each participating node, being sender or receiver in the same multicast group, shares the same data encryption key. This means that a malicious node, participating in the communication as a receiver could inject data packets to the tree which are even correctly encrypted. Spoofing the source address the SSM multicast tree would route packets correctly to recipients.

Although the hierarchy in keys is logical, a mapping to physical locations might provide benefits in optimizing the data route. In such case, the node mobility should be taken into account also in the key management, not only in data routing in the multicast tree. When a client moves, it affects similarly as the client would leave from the network and join in the new place; re-keying should be done both on the old and on the new path. However, this has not been considered in the current key management systems, but there is research going on in this area.

The building of registration association has not been defined well in the specifications. The GDOI, however, has specified that the ISAKMP Phase 1 will be used, but others omit this definition.

In the next section we describe how HIP can be used as the registration protocol for the GDOI. While the

6. USING HIP WITH GDOI

When a HIP host joins a GDOI key management system, the host initializes a HIP BEX with the GCKS. During the negotiation, the GCKS authenticates the client using public key cryptography.

If needed, the I2 message can be added to contain the certificate for authorization or authentication purposes if needed by the system. Currently, the HIP base specification [14] specifies the CERT parameter to be used for transmitting certificates, but the usage of the parameter has not been defined.

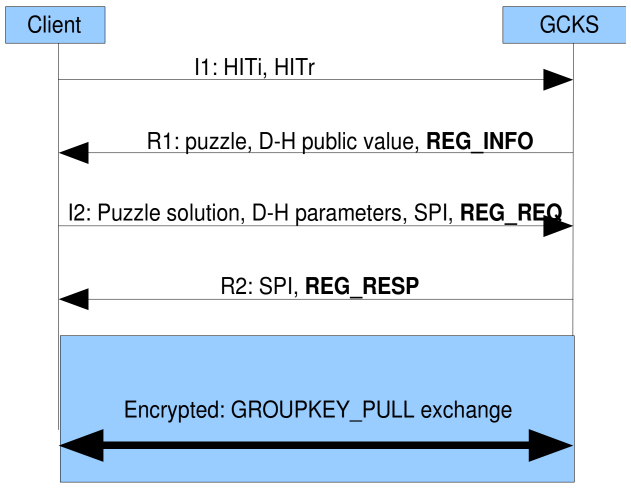

Client
GCKS
I1: HITi, HITr
R1: puzzle, D-H public value, REG_INFO
I2: Puzzle solution, D-H parameters, SPI, REG_REQ
R2: SPI, REG_RESP
Encrypted: GROUPKEY_PULL exchange

Figure 7: HIP Multicast Registration

In GDOI, it is defined that the GDOI information is included in the Phase 1 negotiation. However, in HIP we can define a new service for the Multicast GCKS. While the GCKS provides this new service, the HIP service registration protocol [15] can be used by the client to register as a user to this new service.

In Figure 7 the registration related parameters have been added to the BEX. In R1 the GCKS advertizes the multicast GCKS service that it is providing. In I2 the client registers to the GCKS with REG_REQ parameter included in the packet. The parameter can contain e.g. the multicast channel that the client want's to receive. The R2 finalizes the multicast GCKS service registration.

After the BEX, hosts share a common keying material from where they can retrieve keys when needed. From this material, the SKEYID_a, a shared key used for calculating GDOI message hash values, is retrieved to be used in the GDOI GROUPKEY_PULL and GROUPKEY_PUSH exchanges.

When the HIP node leaves the multicast session, it creates a CLOSE message and sends it to the GCKS. After this the registration association can be tore down and the GCKS can initiate re-keying with other nodes.

The proposed solution requires that HIP is installed on end-hosts, as well as on the GCKS node. It is possible that all clients do not have to support HIP, if the GCKS is able to support also other registration protocols. The amount of required modifications in the GCKS node, in addition to HIP implementation, is not huge; the GCKS has to map the identity information provided during the registration to the identity provided in the access control list. Also the handling of common keying material has to be handled so that the material generated during the HIP BEX can be used also for purposes of the GDOI (e.g. SKEYID_a for calculating required hashes).

7. FUTURE WORK

In this paper we described the current multicast keying environment specified in the IETF. Three different key exchange protocols have been defined currently, GSAKMP, MIKEY, and GDOI. Each of these protocols have been designed for slightly different target systems in mind.

We proposed to use the HIP as the registration protocol for the GDOI protocol, providing host authentication, possible authorization information in form of a certificate, and creation of a security association (registration association) between the client node and the multicast group control and key server function. In addition to the GDOI, HIP could be used also for GSAKMP and MIKEY protocols as the registration protocol. This requires some work for determining all the required parameters that may have to be included in HIP to support these protocols.

8. ABBREVIATIONS

AH	Authentication Header
API	Application Programming Interface
BEX	HIP Base Exchange
D-H	Diffie-Hellman
ESP	Encapsulating Security Payload
GC	Group Controller
GCKS	Both GC and KS functionality
GDOI	Group Domain of Interpretation
GM	Group Member
GO	Group Owner
GSA	Group Security Association
GSAKMP	GSA Key Management Protocol
HI	Host Identity
HIP	Host Identity Protocol
HIT	Host Identity Tag
IKE	Internet Key Exchange
ISAKMP	Internet Security Association Key
KEK	Key Encryption Key
KS	Key Server
	Management Protocol
LKH	Logical Key Hierarchy
MIKEY	Multimedia Internet Keying
PIM-SM	Protocol Independent Multicast – Sparse Mode
PS	Policy Server
PT	Policy Token
SA	Security Association
SPI	Security Parameter Index
TEK	Traffic Encryption Key
TESLA	Time Efficient Stream Loss-Tolerant Authentication

9. REFERENCES

[1] Deering, S., “Multicast Routing in a Datagram Internetwork”, Ph.D. Thesis, Stanford University, 1991

[2] Deering, S., “Host Extensions for IP multicasting”, RFC1112, IETF, 1989

[3] Wallner, D., Harder, E., Agee, R., “Key Management for Multicast: Issues and Architectures”, RFC2627, IETF, June 1999.

[4] Hardjono, T., Weis, B. “The Multicast Group Security Architecture”, RFC3740, IETF, March 2004

[5] Baugher, M., Canetti, R., Dondeti, L., Lindholm, F., “Multicast Security (MSEC) Group Key Management Architecture”, RFC4046, IETF, April 2005

[6] Holbrook, H., Cain, B., “Source-Specific Multicast for IP”, RFC4607, IETF, August 2006

[7] Maughan, D., et. al., “Internet Security Association and Key Management Protocol (ISAKMP)”, RFC2408, IETF, November 1998

[8] Baugher, M., Weis, B., Hardjono, T., Harney, H., “The Group Domain of Interpretation”, RFC3547, IETF, July 2003

[9] Arkko, J., et. al., “MIKEY: Multimedia Internet KEYing”, RFC3830, IETF, August 2003

[10] Harney, H., Meth, U., Colegrove, A., “GSAKMP: Group Secure Association Key Management Protocol”, RFC4535, IETF, June 2006

[11] Fenner, B. et. al., “Protocol Independent Multicast – Sparse Mode (PIM-SM): Protocol Specification (revised)”, RFC4601, IETF, August 2006

[12] Perrig, A., et. al., “Timed Efficient Stream Loss-Tolerant Authentication (TESLA): Multicast Source Authentication Transform Introduction”, RFC4082, IETF, June 2005

[13] Kaufman, C. (editor), “Internet Key Exchange (IKEv2) Protocol”, RFC4306, IETF, December 2006

[14] Moskowitz, R., Nikander, P., Jokela, P., Henderson, T., “Host Identity Protocol”, Internet-Draft, <draft-ietf-hip-base-06>, work in progress, IETF, June 2006

[15] Laganier, J., Koponen, T., Eggert, L., “Host Identity Protocol (HIP) Registration Extension”, Internet-Draft, <draft-ietf-hip-registration-02>, work in progress, IETF, June 2006

[16] Jokela, P., Melén, J., Ylitalo, J., “HIP Service Discovery”, Internet-Draft, <draft-jokela-hip-service-discovery-00>, work in progress, IETF, June 2006

[17] “Security of Multimedia Broadcast / Multicast Service (MBMS)”, technical specification TS 33.246, 3GPP, September 2006

[18] Kent, S., Seo, K., “Security Architecture for the Internet Protocol”, RFC4301, IETF, December 2005

[19] Ballardie, A., “Scalable Multicast Key Distribution”, RFC1949, IETF, May 1996

[20] Harkins, D., Carrel, D. “The Internet Key Exchange (IKE)”, RFC2409, IETF, November 1998