KHIP — A Scalable Protocol for Secure Multicast Routing*

Clay Shields J.J. Garcia-Luna-Aceves

{clay,jj}@cse.ucsc.edu
Computer Engineering Department
Baskin School of Engineering
University of California, Santa Cruz, CA 95064
http://www.soe.ucsc.edu/research/ccrg/

Abstract

We present Keyed HIP (KHIP), a secure, hierarchical multicast routing protocol. We show that other shared-tree multicast routing protocols are subject to attacks against the multicast routing infrastructure that can isolate receivers or domains or introduce loops into the structure of the multicast routing tree. KHIP changes the multicast routing model so that only trusted members are able to join the multicast tree. This protects the multicast routing against attacks that could form branches to unauthorized receivers, prevents replay attacks and limits the effects of flooding attacks. Untrusted routers that are present on the path between trusted routers cannot change the routing and can mount no denial-of-service attack stronger than simply dropping control messages. KHIP also provides a simple mechanism for distributing data encryption keys while adding little overhead to the protocol.

1 Introduction

A multicast routing protocol provides efficient many-to-many delivery across a network by constructing a tree over all sources and receivers in the network. Multicasting preserves bandwidth by sending data packets only once over any link of the multicast routing tree. There are two common ways to form the multicast routing tree. *Sender-initiated* protocols, such as DVMRP [8] and PIM-DM [9], build a separate tree from each source to all possible receivers using a *flood-and-prune* mechanism. Data from any source is initially flooded to all possible receivers, and receivers that do not want to receive the multicast send explicit remove messages, called *prunes*, that travel back towards the source and remove unneeded branches from the tree. These protocols construct multiple *source-routed* trees, each rooted at and with minimum delivery latency from a particular source. However, they are more expensive in terms of router memory requirements and control traffic overhead than the second type of multicast protocol. *Receiver-initiated* protocols, such as CBT [4], OCBT [20], and PIM-SM [9], require that each receiver that desires to participate in the multicast send an explicit join message towards a known point, frequently called a *core*. The core returns a message that forms a branch of the multicast tree back to the receiver. Instead of using a different tree for each source, these protocols build a single *shared tree* spanning all sources and receivers. While shared trees may increase the latency of packet delivery, they greatly reduce the memory requirements at routers and lower the amount of control traffic required. This reduction in routing overhead make shared trees ideal for routing between heterogeneous routing domains, and more recent hierarchical, intra-domain multicast routing protocols use shared trees [21, 16].

Multicast routing protocols today lack effective security mechanisms that allow for privacy or for safe transmission of proprietary information. This is due to the fact that multicast routing is fundamentally different than unicast routing, because multicasting involves a large number of participants whose identities are not known across the entire group; accordingly, the mechanisms needed to multicast securely are correspondingly different [3] than those needed for secure unicasting. There are three problems that need to be solved to provide a solution for secure multicasting. The first is the problem of *authentication*, which requires that participants prove their identity before they are allowed to join the group and receive encryption keys for the session. The second problem is that of *authorization*, which implies that only those entities with specific permission may use or alter the multicast routing tree of a given group, preferably after they have been suitably authenticated. The third problem is *integrity*, which requires that data and control packets originated at an authorized source not be intercepted or altered while traversing the multicast tree, and that the possibility of a denial-of-service attack preventing the transmission of such packets be minimized or eliminated.

Many existing protocols focus on providing multicast-security services for data packets at the application layer [26, 7, 24]. These protocols typically assume that all group members know which other entities are allowed to send and receive data, and that each member has a known public key that can be used to exchange a symmetric key or to check a signature on data packets to verify the authenticity of the sources. While these protocols are very effective for their purpose, they use existing insecure multicast routing for data transmission, which permits many possible attacks against the protocol. First, the fact that multicast routing provides a very efficient way to distribute data also means that there

*This work was supported in part by the Defense Advanced Research Projects Agency (DARPA) under Grant Numbers F19628-96-C-0038 and F30602-97-1-0291

is a very efficient method of launching a denial-of-service attack against all members. If some malicious sender were to send data at a high rate to a multicast group using this type of security, that data would be copied and forwarded over all branches of the tree to all receivers. An attacker can disrupt service to all receivers by purposely saturating the multicast routing tree. This may occur because either the saturated network lacks capacity to carry the legitimate traffic, or the receivers become saturated because the verification of messages must take place at the receivers.

Additionally, any attacker can listen to the traffic generated by the group members, because there is no control over the multicast group membership even though there is control over the secure group membership. While encryption is used to protect the data contents from exposure, there are situations in which this may not meet a desired level of security. Because the use of encryption is often a political issue and because secure multicast group members might be located in different countries throughout the world, the strength of the encryption being used may be limited in some cases by the laws of one or multiple countries where the participants reside. It might also be true that the long-term secrecy of the data is desirable, or simply that it would be to an attacker's advantage to know which parties were communicating. There is also the small risk of exposure through attacks against the encryption [5, 25]. Because of these attacks, it can sometimes be necessary to limit the ability of an attacker to access the secure traffic. To do this, the multicast model must be changed so that authentication, authorization and integrity checking occur in the routing protocol. In this way, the construction of tree branches occurs only between authorized senders and receivers. Current protocols do not do this; they only protect the data messages and not the control messages of the routing protocol.

A number of attacks are possible when authentication, authorization and integrity checking of control messages are not used, and any router in the multicast tree can send control messages affecting the entire multicast routing tree. This is particularly true of receiver-initiated protocols that rely on a shared tree for data delivery. The problem arises because an individual router never receives any assurance that the routers that are forwarding its join message towards any core, rendezvous point or root domain[1] being used to construct the multicast group. A corrupt router may choose instead to acknowledge the join message itself, without ever joining the tree. In this case, the corrupt router would have sole access to the data being transmitted from the new branch or could feed a particular receiver or subset of receivers on the branch below a false stream of data. A more active form of this attack, shown in Figure 1, could consist of several corrupted routers working together to isolate several multicast participants from the rest of the multicast tree by forwarding the join requests they receive towards one of the corrupted routers instead of the core. These types of attacks do not necessarily isolate only a few routers; with the introduction of hierarchical multicast routing protocols that use shared trees to form a backbone between domains, entire domains may be targeted [21, 16]. It is possible to imagine many other similar scenarios, all of which spring from the lack of trusted multicast routing.

Other attacks against the multicast routing may not be

[1]Each of which is just a specific name given by a particular protocol to label the known point that all members desiring multicast service must contact to join the group.

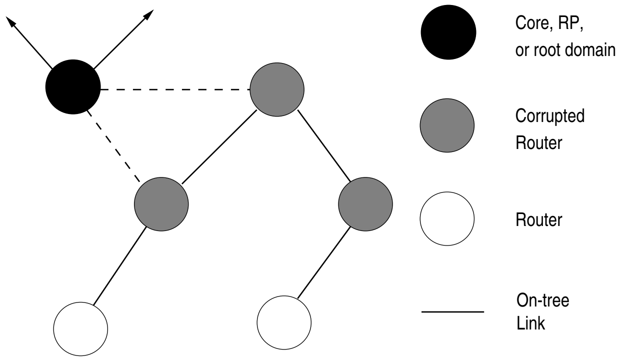

Figure 1: Establishment of a corrupted multicast tree

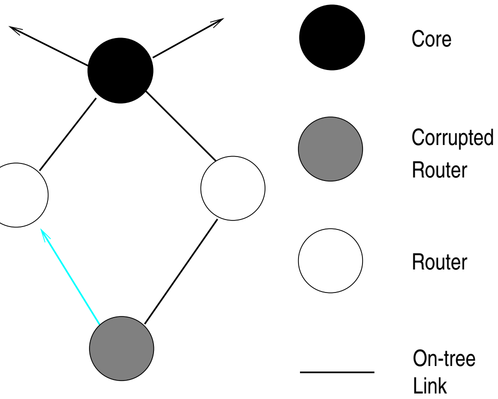

Figure 2: Corrupt router building a loop in the multicast tree

designed to replace or limit access to the data packets sent along the tree. Instead, an attacker may want to deny multicast service over a wide area. In this case, a corrupted router could be used to introduce loops into the multicast tree, as shown in Figure 2. In this simplest case, the corrupt router joins the tree on it shortest path to the core, then sends another join request to another router which has a path to the core that does not pass through the corrupt router. This would cause each data packets to traverse the loop as long as their TTL allowed, with each active source adding to the traffic, and receivers receiving multiple copies of each packet. While this would congest the corrupted router as much as any other, an attacker who gained illicit access might not care. There are less malicious reasons for wanting to alter the structure of the tree, as well. An attacker may want to reduce the costs of their routing, and by changing the shape of the tree by forging control messages, could reroute traffic so that it traveled a path for which the attacker was not financially responsible, placing the burden on elsewhere.

To maintain security in the face of attacks, a secure multicast routing protocol must limit the construction of tree branches to those links required to connect authorized senders and receivers. This requires placing trust in some of the routers of the network, and ensuring that the trusted routers are able to communicate securely with each other, even across a series of untrusted routers. Though compromise of a trusted router could result in the same type of attacks described above, it should be easier to harden and monitor a relatively few trusted routers, limiting the points of possible attack. The protocol should also limit the effects of flooding attacks, and should provide a scalable method for distributing keys. The protocol must do this while maintaining an efficient multicast service that does not duplicate data packets over any link and minimizes the overhead required in terms of control traffic and router memory.

This paper presents a solution that meets all the above criteria. Using techniques similar to those proposed by Gong and Shacham [13], we present simple extensions to HIP [21], called Keyed HIP (KHIP), that creates a secure multicast routing service. KHIP provides security of multicast routing by adding several elements to the multicast model. First, KHIP creates an authentication service that is trusted to issue certificates to authenticated hosts and routers which meet the criteria for joining the group. These certificates are used in constructing the multicast tree. A router connecting to a parent higher in the tree (possibly through several untrusted routers) signs the control message and includes its certificate to prove to its parent that it is authorized to modify the tree routing; the parent includes its certificate in a signed reply so that the child may trust it has joined a secure tree. The tree is divided into a number of sub-branches, each with its own key shared between the trusted members within that branch. Using this key, the trusted members can protect against flooding or replay attacks and efficiently distribute data encryption keys if desired.

Section 2 provides an overview of HIP, the hierarchical multicast routing protocol to be made secure. Section 3 describes how the HIP tree is divided into sub-branches, describes the messages used to create the KHIP tree, and shows how HIP messages are made secure. Section 4 provides an informal analysis of the effectiveness of different attacks against the routing. Section 5 presents a review of previous work.

2 Overview of HIP

HIP [21] is a hierarchical multicast routing protocol that uses the Ordered Core Based Tree protocol (OCBT) [20] to route multicast data between heterogeneous multicast domains. HIP aligns with existing multicast domains and makes each domain on the tree appear as an OCBT *virtual router* by organizing the border routers of the domain to simulate the output of a single OCBT router. The tree can thus be constructed of routers and domains, and any domain may contain other domains within, recursion therefore providing as many levels of hierarchy as necessary. Figure 3 shows the structure of a hierarchical multicast routing tree constructed using HIP. In this figure there are five different multicast domains, each outlined with a dashed line and numbered one through five. Notice that domain number five is completely enclosed within domain four. Figure 4 shows the shape of the tree at the highest level, where each domain appears as a single router on the tree.

While HIP does not solve the problem of multicast address allocation, it does provide a *location service* for distributing the mapping of center point[2] location to the multicast address. In the figure, the center point is located at router A. The creator of the multicast group sends a message advertising the existence of the group and the center point location of the group to the highest level of the hierarchy, where it is stored by a directory service. Receiver initiated domains with individual receivers wishing to join the group can later request and receive the location of the center point. For sender-initiated domains, the advertisement can be sent on a known multicast address which is subscribed to by all virtual routers containing sender initiated domains. The advertisement, which can contain the first packet of multicast data, is sent over this *all-virtual-router* multicast address to all sender-initiated domains, where it is flooded throughout the domain. If the internal source-routed tree is not pruned back completely, the border routers making up the virtual router for the domain then issue a join request for the group.

The multicast tree is formed between routers and domains using OCBT, a hard-state, receiver-initiated protocol that supports the use of multiple cores. Each core is labeled with a number representing that core's logical level. Lower numbered cores join to higher numbered cores, and routers on the tree are labeled with the level of the core answering their join requests. Control messages carry the level of core they are attempting to reach, so that lower-level branches break to allow formation of higher-level branches. The labeling guarantees loop freedom at all times.

Receivers wishing to join the tree send a message called a *join request* towards the core they wish to join, labeled with the level zero to indicate that any on-tree member can answer. The join request travels hop-by-hop towards the specified core, setting up transient state along the way. When the join request hits the core or some branch of the tree already in place, the receiving router returns a message called a *join acknowledgment* that carries the level of core or branch the join request encountered. The join acknowledgment returns along the reverse path of the join request and forms a branch of the tree back to the original requester. Because OCBT is a hard-state protocol, once branches are formed they remain in place until explicitly removed from the tree, or a router or link failure requires that the tree be rebuilt. A node sensing that its parent link or router has failed sends a message called a *flush* that travels from parent to all children tearing down the tree. This process continues down to leaf routers or down-tree cores, which then rejoin as needed. If a node no longer has any children or receivers on a subnet that wish to get the multicast tree, it sends a message called a *quit notice* to its parent. Any node receiving a quit notice that has no other children also leaves the tree by sending a quit notice to its parent. Once the tree is constructed, data flows over the tree in a very simple fashion. Any node receiving a data packet one one interface simply forwards it on over every other on-tree interface. In the event of a network partition, OCBT cores initiate a diffusing computation that returns the identities of the other cores within its partition. Using this information, a multicast tree can be constructed within each partition. Once the partition has merged, the trees within each partition merge.

With any receiver initiated multicast protocol the problem arises as to where to put the core. HIP solves this by using existing multicast domain border routers as cores. In any virtual router, the *exit router* that has the shortest path to the group center point is a level-two router, and all other routers are level-one routers. In Figure 3, routers F, N, Q and I would be exit routers for the center point at router A, as they have the shortest paths to the center point out of all border routers in their domains. A router wishing to join the multicast group simply directs its join request to any border router — it does not need to know the center point of the

[2] The center point is a term for what serves as the core of the inter-domain multicast tree. It is not called a core to differentiate it from cores that are local to some particular multicast domain.

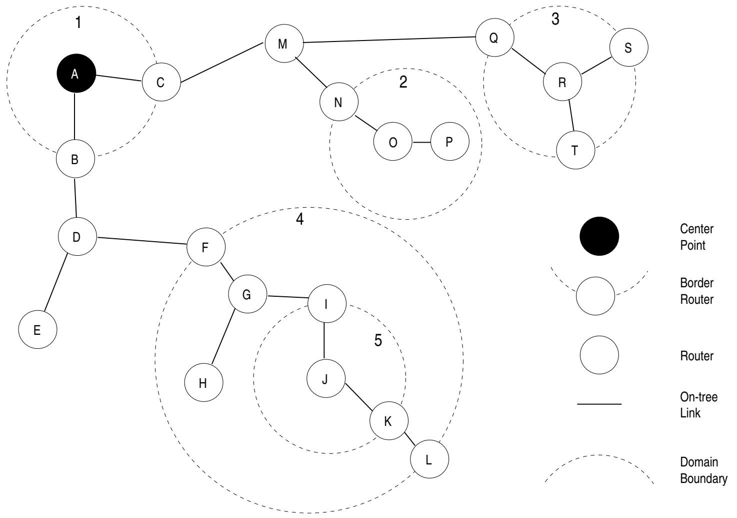

Figure 3: The structure of a hierarchical multicast routing tree created by HIP

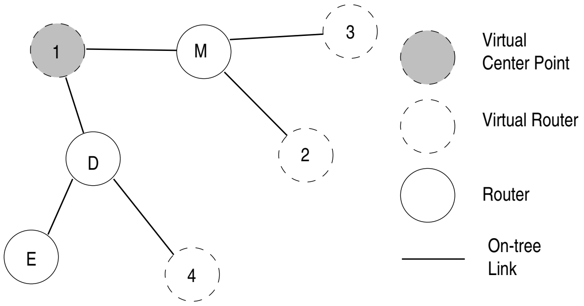

Figure 4: The HIP tree as it appears at the highest level

group. When the join request arrives at a border router that is not the exit router, it sends an acknowledgment and then joins the exit router itself. In the case in which a domain contains or is itself a center point, it is possible to use some internal router as the core. It will be the level-two core, and all border routers will be level-one cores. In Figure 3, router A will therefore be labeled with the level two, while routers B and C will be labeled level one. As border routers are generally fixed and change infrequently, using them as cores allows other routers in the domain to be given a fixed list of border routers to use as cores.

3 Keyed HIP

In extending HIP, the design goals are to provide scalable mechanisms for authentication and authorization, so that only privileged members are able to obtain the proper cryptographic credentials for retrieving keys and certificates that allow access to the multicast trees, and only those holding certificates can cause new branches to form in the tree for transmission or reception. We also wish to prevent or limit replay and flooding attacks. We assume that an attacker can be positioned along any link or control any non-trusted router so that they may drop, replay, delay, alter or issue any data or control packet. Additionally, we assume that an attacker may not be on a path used by the multicast tree and may send control messages to group members in an attempt to create branch of the tree to the attacker, either to receive data or to launch a denial-of-service attack via flooding. We do not require that clocks be synchronized tightly across the network. We do assume that some method of public key distribution and verification is possible. Finally, since HIP and OCBT use the existing unicast routing, we assume that some secure form of unicast routing is available, both for inter-domain and intra-domain routing. A number of proposals have been made for each protocol type[14, 23, 18, 22]. We assume that secure unicast routing provides a reliable path between routers in the network.

Keyed HIP (KHIP) creates a hierarchy of sub-branches over all the branches of the multicast tree, which removes the need for a single shared key for the entire group. While a protocol that can compute a single key across all receivers [24, 26, 7] may be used, and in some circumstances may be desirable, the natural organization of KHIP provides a simple, efficient mechanism for distributing encryption keys. Each sub-branch shares a common key among its members for use in encrypting sequence information within the sub-branch. Data packets can easily be reprocessed at the root of the sub-branch for transmission on the next sub-branch towards the center point; similarly, children of a sub-branch that are roots for another sub-branch re-process that data before re-transmission to their children. Re-processing is made less expensive in terms of the amount of computation needed by encrypting the data in a random key and

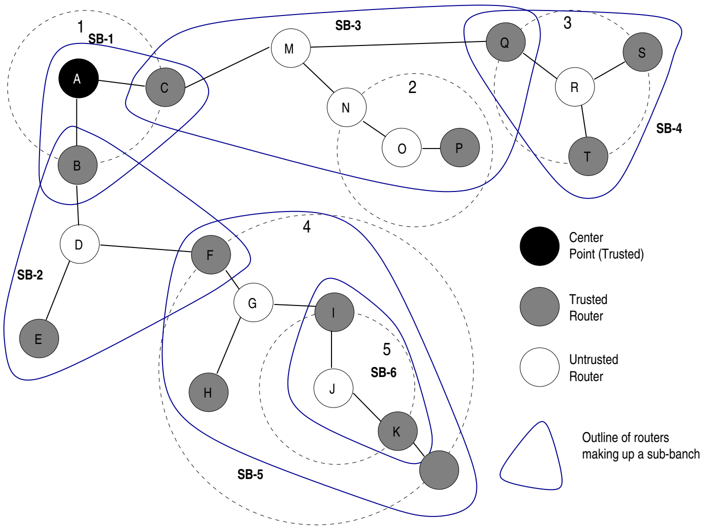

Figure 5: Secure sub-branches on the HIP tree

encrypting that key in the shared branch key. Re-processing consists of decrypting and re-encrypting only the random key and adding appropriate nonces for the new sub-branch. This encryption and decryption is necessary anyway to ensure the integrity of the sequence numbering and nonces; processing the random key only changes the size of the header that needs to be processed.

When new members join the tree they need only authenticate themselves to the root of the sub-branch they are joining, distributing the load of processing new members across many participants already in the tree. The nature of HIP, which relies on the existing border routers of multicast domains to serve as OCBT cores, provides a natural placement of trusted routers that act as the roots of sub-branches in the same location. A domain wishing to participate in a secure multicast session can therefore ensure its security and configure their border routers so that they posses a public/private key pair and are able to request a certificate from the authentication service. The authentication service is a hierarchy of trusted servers that maintain an access list for each multicast group and issues certificates, signed with a well known authentication server key, for trusted members to present as proof they are eligible to join a particular group.

Figure 5 shows how a HIP tree might be broken down into sub-branches. Router A is the center point of the tree and as such must possess a valid certificate to prove to receivers joining that it is a valid center point. Figure 6 shows the structure of the secure tree. The border routers of domain one are also trusted, and a sub-branch, labeled *SB-1*, is formed with router A as the root and routers B and C as its secure children. Router C is the root of its own sub-branch, with child routers P and Q. Notice that a secure branch can traverse untrusted routers, and that the border routers of a domain do not need to be trusted in order to support internal trusted routers, as seen in domain two. However, it is recommended that a domain that is to contain a large number of members of the secure multicast group ensure that its border routers are secure, as this increases the scalability by lowering the number of children any one secure parent must service. Router Q supports its own sub-branch that consists of the other border routers of domain three. Router B is also root of a sub-branch, with children E and F. Domain four contains a sub-domain, numbered five, and the sub-branch *SB-5* rooted at F traverses the virtual trusted router, which contains its own sub-branch, labeled *SB-6*. Secure data traversing the virtual trusted router from F to L will be re-processed at router I for transmission across *SB-6* to router K, where they will be re-processed to appear as if it came directly from F. While this may seem strange, it is congruent with the operation of HIP, which requires a domain to act as a single router on the higher-level tree. In this case, domain five appears as a single router to both G and L, so the data that L receives must be the same as if the virtual router were a real one.

3.1 Building a Secure Multicast Tree

To describe the methods through which individual members request and receive certificates and then communicate with each other to build the multicast tree, we introduce the following notation, similar to that used by Gong, Needham and Yahalom in their work commonly referred to as *GNY logic* [11]. The communicating entities consist of the authentication service AS, the HIP center point location service LP, the group initiator I, receivers of the multicast group (who on a shared tree are also senders) R, roots of the sub-branches within the structure of the tree C, one of which will be the center point for the tree CP. Roots preside over sub-branches, B made up of some number of receivers. Any member of the group, M, has a public key, K_{+M} and the corresponding private key, K_{-M}, and uses these to prove its identity to the authentication service and other group members. Two entities, say A and B, can share a common key K_{AB}. The transmission of messages between entities is indicated with an arrow, $A \rightarrow B$, with a message that is encrypted being enclosed with braces with the key as a subscript, $\{message\}_{K_{AB}}$. We also make use of digital signatures, which consist of a cryptographic hash of the message encrypted with the private key of the signer. Digitally-signed messages are indicated in brackets, with the key being used to sign it as a subscript, $[message]_{K_{-M}}$.

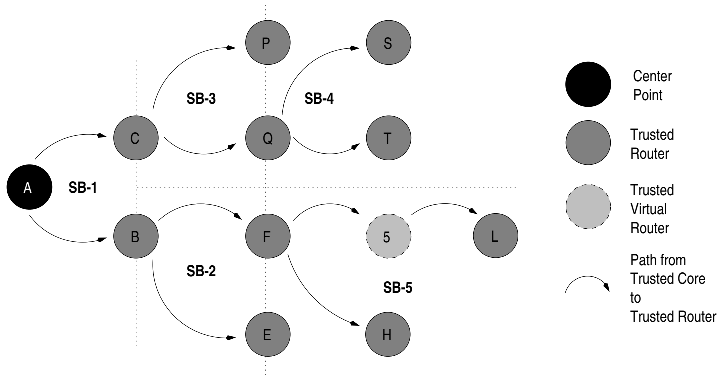

Figure 6: The shape of the secure KHIP tree

3.1.1 Authentication service and certificates

The first necessary addition to the KHIP multicast model is an *authentication service*. This service maintains the list of who is allowed what access to specific multicast groups. The policies that it might enforce are beyond the scope of this paper; however, it is easy to envision that the access lists might consist only of those entities who have paid for service or who are part of some governmental or commercial group that wish to share data securely. The authentication service issues *certificates*, credentials that verify the holders identity and specify what type of access the holder is allowed to the multicast routing. The authentication service owns a well-known public key used to sign certificates and is assumed to be secure against compromise. This service might be a single server within a domain that regulates its internal multicast service, or it could be a robust hierarchy of world-wide servers, with lower-level trusted servers possessing keys signed by the highest-level authority. Such a model has already been proposed [6].

When a member wishes to become part of the secure multicast group, it must first secure a certificate for the group from the authentication service. It requests a certificate from the authentication service using its own public key to authenticate its identity. If authenticated and approves, it receives a certificate consisting of the member's IP address, its public key, the multicast group or range of groups that the member is authorized to use, what permissions the member has for that group, and a time stamp and lifetime for the certificate. In the GNY notation, a certificate for some member M will appear as:

$$CERT_M = \\ [IP_M, K_{+M}, MC, Perm, TS, Life]_{K_{-AS}}$$

The possible permissions are $Create$, $Join$ and $Destroy$. As the names imply, these allow a member to initiate a group with a particular address, subscribe to the group as a sender or receiver, and terminate a secure group. With modification to the routing protocol, it would also be possible to specify $Listen$ and $Send$ for finer granularity of access control. The time stamp and lifetime are used to expire the certificate, and should be reasonably short. Given that it is difficult to revoke issued certificates in a way that is scalable and can reliably reach all entities who could honor them, this will force a receiver to occasionally re-request a new certificate. If the access list has changed in the interim, it will be denied. A bad certificate will therefore never last more than the lifetime specified. This solution is deemed preferable as it is easier to expand the hierarchy of authentication services than to attempt to create a method of issuing certificate revocations.

3.1.2 Group creation

To start a secure multicast session, some initiator must request and receive a certificate with the appropriate permissions, then communicate the center point of the group to the location service. The initiator send its IP address, its public key and the multicast address and permissions desired, which in this case would include *create*, to the authentication service. The transmission from the initiator to the authentication service appears as:

$$1 : I \rightarrow AS : \\ [IP_I, K_{+I}, MC, Perm]_{K_{-I}}$$

Upon receipt of this message, the authentication service retrieves the initiator's public key from the appropriate server and uses it to verify the IP address and public key contained in the message. It then checks the access list to see if the initiator is allowed to create the requested multicast groups. If these checks succeed, then the authorization service adds a time stamp and lifetime to the certificate, signs the certificate with the authentication service private key and returns it to the initiator. The reply is then:

$$2 : AS \rightarrow I : \\ CERT_I = [IP_I, K_{+I}, MC, Perm, TS, Life]_{K_{-AS}}$$

Now the initiator can send the group creation message to the location service so that the start of the group can be announced. This message includes the signed certificate, the center point for the group, the scope of the group (which specifies which domains the group should will cover) and a group lifetime, which should not exceed the lifetime of the certificate. Once the location service receives the creation request, it verifies the signature on the request and on the certificate it contains. Notice that the location service need not retrieve the initiator's public key as it is contained in the signed certificate. If the request passes these checks, the group center point is distributed as appropriate given the scope. The message sent from the initiator to the location service is:

$$3: I \rightarrow LS: \\ [Create, CERT_I, CP, scope, lifetime]_{K_{-I}}$$

As the center point will serve as the root of the multicast tree, it needs to obtain a proper certificate as well. This is necessary as even though it need not join the tree, it will need authenticate with those joining the tree. This occurs in the same manner as the initiator's certificate request, though the permissions are for *Join* rather than *Create*. Though the center point can be a distributed entity consisting of a number of border routers of a domain operating together, they do operate as subordinates to one master router, and only that router need retrieve a key as it will be processing and replying to all messages. This exchange appears as:

$$4: CP \rightarrow AS: \\ [IP_{CP}, K_{+CP}, MC, Perm]_{K_{-CP}}$$

$$5: AS \rightarrow CP: \\ CERT_{CP} = \\ [IP_{CP}, K_{+CP}, MC, Perm, TS, Life]_{K_{-AS}}$$

3.1.3 Host to router communication

The protocol described above details only the process of routers constructing the multicast tree; it does not describe the communication between hosts and routers. Today, hosts communicate with their designated multicast router using IGMP. For secure multicast, a secure version of IGMP needs to be developed [3]. A host could then authenticate with its router using the same four way authentication mechanism described below, first with an exchange to find the designated router and receive a nonce, then the message to the router with a host nonce and the rely bearing the key and sequence information. With this mechanism in place, the host to router connection will appear simply as the connection to leaves of the secure multicast tree.

3.1.4 Building the tree - securing OCBT

Once the group has been created and the center point has received its certificate, the process of building the tree can start. HIP uses OCBT to build the multicast tree. OCBT uses only four types of control messages in normal operation: a *join request (JR), join acknowledgment (JA), quit notice (QN)* and *flush (FL)*. The join request travels from receiver to core and sets up temporary state so that the join acknowledgment can traverse the same path back from core to receiver and instantiate a branch of the multicast tree. The other two messages are used to tear down branches of a tree, either following after a link or router failure, or when it is necessary to break a lower-level tree branch to allow a higher-level branch to form.

Since one design goal is to prevent branches from being built to unauthorized receivers, we add digital signatures to the join request and the join acknowledgment to ensure that the endpoints of a particular path along a branch are trusted receivers. Neither the flush message nor the quit notice need to be signed, as they do not result in construction of a branch and also may be generated by untrusted routers on the path between to trusted routers in response to link or router failures. In order to prevent attackers from replaying join requests or acknowledgments, we add two more messages to the original protocol. Because a member of the group does not know the identity of the identity or the location of the trusted core that it will reach when sending a join request towards the center point, it instead finds and exchanges messages with the trusted core higher on the tree. These two messages, called a *core request (CR)* and *core acknowledgment (CA)*, are signed by each party and carry nonces that ensure that the messages are fresh. Using time stamps instead of nonces would lead to a simple replay attack in which an attacker could quickly replay the join request to a different core router, which would accept it as valid and send the join acknowledgment back, building a branch to the attacker.

The exchange of messages that leads to the formation of a branch between some receiver and some core is then as follows. First, the receiver router sends a core request message. This message travels hop-by-hop towards the center point. When it reaches some branch of the tree it is forwarded from child to parent until it reaches some trusted core. If it does not reach a branch of the tree it will eventually arrive at the center point. The message contains the receivers certificate, a random nonce and is signed by the receiver, and looks like:

$$6: R \rightarrow C: \\ [CR, CERT_R, N_R]_{K_{-R}}$$

One the trusted core or the center point receives the core request, the signature and certificate are checked and if passed, a core acknowledgment is sent directly to the initiating receiver. This acknowledgment contains the core's certificate, the nonce created by the receiver, a new nonce created by the core and is signed by the core. This transmission of the acknowledgment is:

$$7: C \rightarrow R: \\ [CA, CERT_C, N_R, N_C]_{K_{-C}}$$

Once the receiver obtains the core acknowledgment, it forms and sends a join request. This join request carries the receiver's certificate and the nonce supplied by the core, and it is signed by the receiver. The join request is sent, hop-by-hop, towards the trusted core, and appears as:

$$8: R \rightarrow C: \\ [JR, CERT_R, N_C]_{K_{-R}}$$

Unlike a normal OCBT join request, the join request is not acknowledged by the first on-tree router it reaches. Instead, the join request is forwarded up the tree to the trusted core, which verifies the signature, nonce and certificate, then sends the join acknowledgment back down the tree, initiating the branch back to the receiver. When the receiver gets the acknowledgment, it verifies the nonce, certificate and signature to prevent replays of other join acknowledgments. The acknowledgment also contains some information encrypted in the receiver's public key; the branch key, K_B, which is used to send data to the multicast group as described below, a branch identification number, ID, which is unique to each trusted receiver serviced by that particular core, and a starting sequence number for data, SEQ. The ID and sequence number together make up nonces for each data packet sent by any member. This final acknowledgment is:

$$9 : C \rightarrow R : \\ [JA, CERT_C, N_R, \{K_B, ID, SEQ\}_{K_{+R}}]_{K_{-C}}$$

3.2 Operation and Maintenance of the Secure Multicast Tree

3.2.1 Data flow

Each individual receiver that is part of the sub-branch under some trusted core is given a shared branch key, K_B, a unique identifier on that sub-branch, ID, and a starting sequence number. While it is possible to use other protocols for group key exchange [24, 26, 7], the unique nature of a KHIP tree provides an efficient method of doing so, which we consider here. Using KHIP's key exchange protocol, when a receiver wants to send data, it creates a random encryption key, K_{Rand} and encrypts the data with that key. It then creates a packet that consists of the encrypted data and a package of information encrypted in the branch key. This information consists of the random key used to encrypt the data, the sender's branch ID number, the next sequence number, and a checksum of the encrypted data. The packet is then sent on to the branch. When other members of the branch receive the data, they decrypt the random key, sequence information and checksum. The ID and sequence information are used as nonces; each receiver keeps track of the sequence number from each different ID. Sequence numbers are used rather than random nonces to facilitate storage. If packets arrive in sequence and none are lost, then only one nonce need be kept for each sender. In the worst case, however, packet loss over the network or an attacker reordering packets can cause the required storage space to be increased. Each receiver need keep track only of sequence numbers in his sub-branch, limiting the number of receivers for which nonces need be kept. However, a router may have many individual hosts on a sub-net to service. In this case, one host on the sub-net can serve as a core for the other members on the same sub-net. This will result in duplicate data packets being sent over the sub-net, once from the router to appointed host and once from the host to other hosts, but will maintain the scalability of the protocol.

A branch member who is serving as a core and has parents or children on another sub-branch will re-process the packet before sending it on. It will decrypt the random key, verify the sequence number and checksum, then replace the sequence information with it's own sequence information for the new sub-branch, as if it were the originator. It will then re-encrypt the key, ID and sequence information and the checksum and send it out over the new sub-branch. This method consumes less processing time than would re-encrypting the entire packet, as the random key and sequence information can be much smaller than the data enclosed. A data transmission from a sender to its sub-branch would then be:

$$10a : R \rightarrow B : \\ \{Data\}_{K_{Rand}}, \\ \{K_{Rand}, ID, SEQ, CS(\{Data\}_{K_{Rand}})\}_{K_B}$$

The checksum is intended to help prevent *cut-and-paste attacks*, in which the attacker replaces the encrypted data but leaves the other information intact. The attacker can not replace the data at will without it being detected, assuming it can tell where the date ends and the key and sequence number starts, though it can replace it with something, including old meaningful data, that produces the same checksum. Though computationally expensive, the function chosen for the checksum should be a cryptographic hash, or else the attacker will be able to replace the encrypted data with possibly random data that has the same checksum.

In some cases, it may be necessary for an end receiver to verify the identity of the original sender. In this case, the originator of the data can include his IP address with and digitally sign the data. Using the included IP address, a receiver can lookup the originator and retrieve his public key, then use that to verify the origin of the data. This looks like:

$$10b : R \rightarrow B : \\ \{[Data]_{K_{-R}}, IP_R\}_{K_{Rand}}, \\ \{K_{Rand}, ID, SEQ, CS(\{Data, IP_R\}_{K_{Rand}})\}_{K_B}$$

If lookups of public keys are expensive and a sender is not sending a large amount of data, it may be worthwhile to include the certificate issued to access the multicast group with the data. By verifying the authentication server signature on the certificate, each receiver can then use the enclosed public key to verify the original sender's identity. This method of verifying the sender's identity appears as:

$$10c : R \rightarrow B : \\ \{[Data]_{K_{-R}}, CERT_R\}_{K_{Rand}}, \\ \{K_{Rand}, ID, SEQ, CS(\{Data, CERT_R\}_{K_{Rand}})\}_{K_B}$$

3.2.2 Re-keying

Though there is no single encryption key for the entire multicast group, each sub-branch key will need to change branch keys K_B as receivers leave, either following their legitimate departure or following a link or router failure that removes a receiver from the tree. It will also be necessary to occasionally to restart the sequence numbers used as nonces. In these cases the core of the sub-branch creates a new branch key and starting sequence number, add those to the old branch key for use as a nonce, encrypts it with the public key of each authenticated member of the sub-branch, signs it and sends it multicast to all members. The structure of the re-keying message is:

$$11: C \rightarrow B:$$
$$[IP_{R_1}, \{K_{B_{New}}, K_{B_{Old}}, SEQ\}_{K_{+R_1}}]_{K_{-C}},$$
$$[IP_{R_2}, \{K_{B_{New}}, K_{B_{Old}}, SEQ\}_{K_{+R_2}}]_{K_{-C}}, \ldots$$

Following a key change, the old encryption key and nonce sequence remains valid for a short period to allow messages in transit that are encrypted in the old key to be accepted when they arrive. Note that the scalability of this method of re-keying is limited, and while it is expected to work on the number of routers found in a sub-branch, it likely would not scale well to a sub-net with hundreds of receivers.

3.2.3 Tree Maintenance

In OCBT, when a router determines that a link to a child on the tree or the child itself has failed, it only needs remove information about the child from its routing state. When a child detects that a parent router or the link to the parent has failed, it needs to take action to assure that its children and itself can rejoin the tree as needed. Therefore, a router detecting a failure will send a flush message to all its children and removes state concerning them. Each non-core child receiving a flush message forwards it to all its children. This process results in the tree being removed from the point of failure down to the individual receivers or cores, who then have the responsibility of rejoining the tree. Other tree maintenance occurs when a router receives a higher-level join request, at which point the router sends a quit notice to its parent and becomes part of the higher-level branch that is forming. In this case the router maintains its children.

With KHIP, untrusted routers need to be able to respond properly to link and router failures but should not be able to cause branches of the tree to be formed to untrusted routers, including routers that were once part of the tree out of the necessity of being on the path to a trusted router, but no longer should be. This requires some small changes to the maintenance mechanism of OCBT. First, quit notices need to be forwarded up to the next trusted core in the tree, so that if a trusted receiver quits the tree, the core router can remove state about it and start the re-keying process. Second, an untrusted router that is forced to quit from its parent to join a higher-level branch that is forming must also send a flush message to destroy the tree below it. This is necessary because the trusted cores or receivers below that router have to authenticate themselves with the new trusted core and receive the branch key for their sub-branch.

Finally, to limit the effects of attack based on forging, replaying or failing to deliver control messages and to detect expires certificates, we require that each sub-branch periodically be destroyed and re-constructed by the trusted core sending a flush message to all its children. Each receiver will then re-authenticate with the higher core. Each receiver should also keep a timer to ensure that they receive these flush messages periodically; if they do not, then they should quit from the tree and attempt to rejoin to make sure the core's certificate has not expired.

4 Denial-of-service attacks and attacks by untrusted routers

There are a number of denial-of-service attacks that are possible against members of a multicast group. The most potent type is a flooding attack, one that uses the natural efficiency of multicast to attempt to drown all receivers in a barrage of worthless data that is spread to all members of the group. Less potent types that use forged or replayed control messages or that simply do not process or forward control messages are also effective at denying service to individual receivers or branches of the multicast tree. KHIP limits the effect of flooding attacks, first by limiting the spread of branches so that an attacker cannot easily access the tree to send to it, and second by verifying the encrypted sequence number and ID enclosed in each data packet. An attacker who happens to be on the path between a trusted core and receiver can attempt to flood the entire multicast tree, but since it is not in possession of the proper sequence information or branch key it cannot construct packets properly and the flooded data will be detected and dropped at each trusted receiver and at the root for that sub-branch. If an attacker were to try replaying old data, this too would be detected when the sequence number was seen as already having been received. Even if the attacker preserved old data from previous sessions it would be improbable that the root of the sub-branch would have chosen the same random branch key to make the packets decrypt properly, as the branch keys changes regularly.

KHIP does not defend against other types of denial-of-service attacks, however. Untrusted routers that lie on the path between a trusted core and one or more trusted receivers can deny service to the receivers, and hence to any sub-branch that they may be the root for. In addition, some of these attacks may cause branches to be formed between the attacker and either the trusted core or trusted receiver. We consider these branches *malformed* as they do not span a path from trusted core to trusted receiver. In all cases these branches are transient as they will be removed when the core periodically flushes the tree to force receivers to re-authenticate. If the trusted receivers do not receive this periodic flush, they will quit the tree themselves. It is important to note that the malformed branches would have been part of the tree branch crossing the corrupt router anyway, so the corrupt router does not gain any additional information than it otherwise would have, and that this type of denial of service is no stronger than simply not forwarding control messages to build the tree in the first place. Malformed branches last only as long as the period allowed between core and receiver re-authentication, because the tree is destroyed and re-built at those times. The attacker also does not gain access the branch key with these attacks. Table 1 shows the effects of replaying, forging or dropping control messages.

In most cases, forgery or replay of control messages or data are detectable, because these messages are signed or encrypted in the branch shared key, cryptographic operations which the attacker cannot duplicate. These instances are marked "Detect" in the figure, as the receiver of a message can tell it has been altered when the signature does not match the message. There are also cases in which a router on the path can cause a denial-of-service attack by dropping control message or data packets. These cases are marked "DOS". A solution to corrupt routers purposely dropping packets would be to use a multi-path unicast routing protocol to determine alternate paths towards the center point,

	CR	CA	JR	JA	FL	QN	Data
Replay	No effect/ DOS - core	Detect	Detect	Detect	DOS/ Transient Malformed Branch C ← A	Transient Malformed Branch A ← R	Detect
Forge	Detect	Detect	Detect	Detect	DOS/ Transient Malformed Branch C ← A	Transient Malformed Branch A ← R	Detect
Drop	DOS-receiver	DOS-receiver	DOS-receiver	DOS/ Transient Malformed Branch C ← A	Transient Malformed Branch A ← R	Transient Malformed Branch C ← A	DOS

Table 1: Effects of on-tree untrusted router attacks.

bypassing the misbehaving router.

In some instances the attacker can maintain a branch of the tree to themselves from either the core or receiver while denying service to one of the trusted participants. In those cases in which transient malformed branches exist, the table entry is labeled as such, with an arrow pointing from the child to the parent of the malformed branch. These cases occur because the untrusted routers on the path between trusted routers need to be able to issue flush and quit notices in response to link and router failures. Since these routers are not trusted, they cannot be issued certificates to use to sign the quit and flush control messages. A simple but costly solution to this is to require every router in the network to be trusted, which is nice for scalability of data transmission but not for certificate distribution. Instead, we tolerate the possibility of these attacks as they are, in effect, just a combination of otherwise possible attacks. An untrusted router can listen to traffic that flows across it, and it can prevent formation of branches by not passing control messages. Attacks that allow the temporary formation of malformed branches are simply a combination of these two other attacks, though less potent as the attacker will only receive data from one direction.

As KHIP supports heterogeneous multicast routing protocols, those protocols are possible points of compromise, especially if they are not secure protocols. A possible solution is to secure and trust the entire domain and have the KHIP-speaking border routers distribute the un-encrypted data to the trusted domain. This is clearly not always practical, however, and domains should use some secure multicast routing, either KHIP or some future secure protocol.

5 Related Work

The first efforts on providing secure multicast service focused on establishing a method of distributing a common shared key to all members of a multicast group [19, 15]. While these protocols were effective for that purpose, they were unscalable, because either they required a single server compute the key for a group, or they required extensive knowledge about the group membership. More recently, distributed and scalable methods of keying a multicast group have been proposed [24, 26, 7]. These protocols, while effective for distributing keys across a multicast group, do not solve the problem of authorization, which is needed to prevent unauthorized receivers from listening to the group and unauthorized senders from mounting a flooding attack. Gong and Shacham [12] were the first to point out the need for some type of authentication and authorization mechanism for multicast. They also clearly stated the goals that a secure multicast protocol design should meet: *compatibility* with existing protocols, *scalability* to the scope of the global Internet, *transparency* to higher-level protocols, *localizablity* for gradual introduction of the technology and *flexibility* to support a variety of policies. However, the authors did not create a protocol that met these criteria.

The first attempt to provide for authentication and authorization in an existing multicast routing protocol came in some simple extensions to CBT [4] that attempted to regulate access to the multicast tree at the first hop router [3]. Ballardie and Crowcroft pointed out the need for *Secure IGMP*[3], which could present cryptographic credentials from the host to the router. In other ways their protocols did not meet any reasonable design requirements, however. There was no mechanism for key distribution, and since all authorization was done at the leaf router on the tree, a corrupt router compromised the entire scheme. Additionally, rather than preventing flooding attacks, the protocol attempted to detect and squelch such attacks by randomly sampling packets and, upon detection of unauthorized traffic, sending messages towards the putative source that prevented it from forwarding traffic onto the tree. The problem with this scheme is that it leads to a simple and effective denial-of-service attack. An attacker, in conjunction with one corrupt router, could send unauthorized traffic that was forged with the source address of the target of the attack. When these packets were detected, the innocent target would be removed from the tree, victim of the forgers.

Gong and Shacham examined the problems inherent in maintaining the efficiency of multicast routing while providing a secure service in [13]. This work introduced four methods of reducing the size and number of control messages needed to authenticate group members and to distribute en-

[3]Internet Group Management Protocol (IGMP) [10] is the protocol that a hosts uses to communicate with an attached router to initiate their connection with the multicast group.

cryptions keys to the group. First, they pointed out that a multicast tree consisted of branches, each of which could utilize different control information than other branches. A node on the tree could then tailor messages for each branch separately, rather than send information needed by only one branch to all branches. Second, they showed that an intermediate node could do some message re-processing, including re-arranging or re-encrypting the message so long as the message's integrity and origin are maintained. This is significant, because it means that a sender does not need to know the topography or group membership to trim control information from a message. Instead, simply knowing a few nodes down different branches, a node can combine this with the first point to tailor messages for several small branches, at the bottom of which the messages are reprocessed for sub-branches. The authors also point out that shared tree protocols are ideal for this type of re-processing, because protocols that have distributed cores can use them as natural spots for message re-processing, in effect breaking one flat tree down into a hierarchy of smaller trees, each of which has its own control traffic. Third, they point out that group re-keying need not take place only at the time a member joins or leaves the group, but can be pre-computed; they call this *hot start authentication*. Finally, they extend the idea of hot start authentication to *continuous authentication*, under which each member needs to periodically re-authenticate to receive the current key. These four ideas re-occur in later works in the area [17, 24].

Some of the above ideas appeared in a RFC that again attempted to produce a secure version of CBT [1]. Under this scheme, called the Scalable Multicast Key Distribution (SMKD), the central CBT core is given an authorization list that it uses to verify signed join requests from receivers, each of whom has some public/private key pair. As the tree grows, the access list and a shared group key are distributed along each branch of the tree. The major problems with this scheme are that no provision is made for re-keying the group should some member leave, and no mechanism is supplied for updating the access list should it change while the group is in existence. The CBT protocol has changed since the time this RFC was released [2]; CBT is no longer a hard-state protocol nor does it support multiple cores. Both the use of multiple cores and the hard state were needed for the scalability of the original key distribution mechanism.

Recently, an application-level implementation of multicast security called Iolus [17] has been proposed. Iolus uses multiple multicast groups, each group with a different multicast key, connected by "group security controllers (GSC)" that re-key and forward traffic between groups. The use of different multicast groups reduces the problem of changing the key, as when some member leaves the group only that group needs to receive new keys, instead of all the groups in the session. Iolus clearly follows the first two ideas presented by Gong and Shacham [13], with control messages being destined for a specific multicast group instead of a particular branch of the multicast tree and with the GSC doing re-processing of the messages that need to travel between groups. While Iolus provides for secure key distribution and re-keying when necessary, it is still necessary to implement multicast security at the routers. Implementing network security at the application level does provide for authentication, in that it gives the appropriate encryption keys to qualified receivers. However, it does not provide for authorization at the network level; therefore it does not protect the routing infrastructure against unauthorized senders mounting flooding attacks and it does not prevent unauthorized receivers from joining the tree and receiving encrypted data.

Iolus is also inefficient in utilizing network resources. An Iolus session uses multiple multicast addresses, where as single multicast group, by definition, uses only one. The mechanisms that claim multiple multicast addresses and make sure that the correct ones are distributed to their local area are certain to be more costly than those to distribute a single address globally. Iolus can also lead to multicast packets being duplicated repeatedly over the same link. This is antithetical to multicast routing protocol design, and can occur when the GSC is placed improperly. As the GSC communicates with different multicast groups, traffic from one multicast group may arrive and be destined to go out to one or more other multicast groups. If the path to any receiver in another group lies along the same path as an incoming packet, that packet will cross the link again on its way out from the GSC. This duplication can occur a number of times equal to the one less than the number of multicast groups the particular GSC is servicing for the session. The problem arises as the placement of the GSC is not necessarily related to the network topography. Even if care is taken to place the GSC, any receiver obtaining the incorrect address for the local multicast group for the session creates a situation where packet duplication can occur.

6 Conclusions

We have described Keyed HIP (KHIP), a new protocol for secure, hierarchical multicast routing. KHIP maintains the efficiency of multicast routing of HIP [21] while providing authentication services and secure routing so that only authorized receivers may use the multicast tree and obtain keys for sending or receiving data. KHIP adds an authentication service that issues certificates to entities who are allowed access and who authenticate themselves with a known public key. These certificates are included in signed control messages to prove that the sender has the authority to alter the tree. The tree itself is divided into sub-branches, and messages within each sub-branch also carry nonces to prevent forgery or replay attacks that could build a branch of the tree to an unauthorized router. Each sub-branch can also use a shared key for data transmission, thus obviating the need for a single key shared across the entire tree. The headers of data packets are re-processed for transmission between sub-branches. The amount of work needed for re-processing is minimized by encrypting the data in a random key and encrypting that random key with the shared sub-branch key. Re-processing is thus limited to re-encrypting the random key in the new branch key and adding new nonces for transmission in the new sub-branch. This increases only the size of the headers that need to be processed, and does not increase the number of encryptions or decryptions. Untrusted routers can only eavesdrop on the encrypted data flow if they happen to lie on the path between two authorized entities. While some denial-of-service attacks by these untrusted routers are possible using unsigned control messages, they are no more effective than if the untrusted router was simply dropping control packets. Keyed HIP is the first secure, hierarchical multicast routing protocol. It meets the needs of security while providing delivery of data across many receivers.

References

[1] A. Ballardie. Scalable Multicast Key Distribution. RFC 1949, May 1996.

[2] A. Ballardie, B. Cain, and Z. Zhang. Core Based Trees (CBT version 3) Multicast Routing. Internet-Draft, March 1998.

[3] A. Ballardie and J. Crowcroft. Multicast-Specific Security Threats and Counter-measures. In *Proceedings of the Symposium on Network and Distributed System Security*, pages 2–16, San Diego, CA, February 1995. IEEE Computer Society Press.

[4] A. Ballardie, P. Francis, and J. Crowcroft. Core Based Trees (CBT): An Architecture for Scalable Inter-Domain Multicast Routing. In *Proc. ACM SIGCOMM'93*, pages 85–95, October 1993.

[5] E. Biham. How to Forge DES-Encrypted Messages in 2^{28} Steps. Technical report, Technion, 1996.

[6] S. Boeyen, T. Howes, and P. Richard. Internet X.509 Public Key Infrastructure Operational Protocols - LDAPv2. RFC 2559, April 1999.

[7] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas. Multicast Security: A Taxonomy and Efficient Construction. In *Proc. IEEE Infocom*, March 1999.

[8] S. Deering. *Multicast routing in a datagram internetwork*. PhD thesis, Stanford University, Palo Alto, California, December 1991.

[9] S.E. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei. An Arcitecture for Wide-Area Multicast Routing. In *Proc.of the ACM SIGCOMM94*, pages 126–135, London, UK, September 1994.

[10] W. Fenner. Internet Group Management Protocol, Version 2. RFC 2236, November 1997.

[11] L. Gong, R. Needham, and R. Yahalom. Reasoning about Belief in Cryptographic Protocols. In *Proceedings of IEEE Computer Society Symposium on Research in Security and Privacy*, pages 234–48, May 1990.

[12] L. Gong and N. Shacham. Elements of Trusted Multicasting. In *Proceedings: 1994 International Conference on Network Protocols*, Boston, MA, October 1994. IEEE Computer Society Press.

[13] L. Gong and N. Shacham. Trade-offs in Routing Private Multicast Traffic. In *Proceedings of GLOBECOM '95*, pages p. 2124–8, Singapore, November 1995. IEEE Computer Society Press.

[14] R. Hauser, T. Przygienda, and G. Tsudik. Reducing the Cost of Security in Link-State Routing. In *Proceedings of the Symposium on Network and Distributed System Security*, pages 93–99, San Diego, CA, Feb 1997.

[15] F. Jordan and M. Medina. Secure Multicast Communications using a Key Distribution Center. In P. Viega and D. Khakar, editors, *Proceeding of IFIP TC6 International Conference on Information Networks and Data Communication.*, pages 367–380, Funchal, Portugal, April 1994. Elsevier Science B.V.

[16] S. Kumar, P. Radoslavov, D. Thaler, C. Alaettinoglu, D. Estrin, and M. Handley. The MASC/BGMP Architecture for Inter-domain Multicast Routing. In *Proceedings of ACM SIGCOMM98*, pages 93–104, Vancouver, Canada, August 1998.

[17] S. Mittra. Iolus: A Framework for Scalable Secure Multicasting. In *In Proceedings of ACM SIGCOMM97*, Cannes, France, September 1997.

[18] S. Murphy and M. Badger. Digital Signature Protection of the OSPF Routing Protocol. In *Proceedings of the Symposium on Network and Distributed System Security*, pages 93–102, San Diego, CA, Feb 1996.

[19] S.H. Ong and S.H. Goh. A Generic Multicast-key Determination Protocol. In *Proceedings of IEEE Singapore International Conference on Networks/International Conference on Information Engineering '93*, pages p. 518–22, Singapore, Sept 1993.

[20] C. Shields and J.J. Garcia-Luna-Aceves. The Ordered Core Based Tree Protocol. In *Proceedings of the IEEE INFOCOM97*, Kobe, Japan, April 1997. IEEE.

[21] C. Shields and J.J. Garcia-Luna-Aceves. Hierarchical Multicast Routing. In *Proc. Seventeenth Annual ACM SIGACT-SIGOPS Symposium on principles of distributed computing (PODC 98)*, Puerto Vallarta, Mexico, June 28–July 2 1998.

[22] B. Smith and J.J. Garcia-Luna-Aceves. Efficient Security Mechanisms for The Border Gateway Routing Protocol. *Computer Communications (Elsevier)*, 21(3):203–210, 1998.

[23] B. Smith, S. Murthy, and J.J. Garcia-Luna-Aceves. Securing Distance Vector Routing Protocols. In *Proc. Internet Society Symposium on Network and Distributed System Security*, San Diego, California, February 1997.

[24] D. Wallner, E. Harder, and Ryan C. Agee. Key Management for Multicast: Issues and Architectures. Informational RFC, September 1998.

[25] M. Wiener. Efficient DES Key Search. Technical Report TR-244, School of Computer Science, Carleton University, Ottowa, Canada, May 1994.

[26] C. Wong, M. Gouda, and S. Lam. Secure Group Communications Using Key Graphs. In *Proceedings of the ACM SIGCOMM98*, pages 68–79, 1998.