                    Scalable Multicast Key Distribution

Status of this Memo

   This memo defines an Experimental Protocol for the Internet
   community. This memo does not specify an Internet standard of any
   kind. Discussion and suggestions for improvement are requested.
   Distribution of this memo is unlimited.

Abstract

   The benefits of multicasting are becoming ever-more apparent, and its
   use much more widespread. This is evident from the growth of the
   MBONE [1]. Providing security services for multicast, such as traffic
   integrity, authentication, and confidentiality, is particularly
   problematic since it requires securely distributing a group (session)
   key to each of a group's receivers. Traditionally, the key
   distribution function has been assigned to a central network entity,
   or Key Distribution Centre (KDC), but this method does not scale for
   wide-area multicasting, where group members may be widely-distributed
   across the internetwork, and a wide-area group may be densely
   populated.

   Even more problematic is the scalable distribution of sender-specific
   keys. Sender-specific keys are required if data traffic is to be
   authenticated on a per-sender basis.

   This memo provides a scalable solution to the multicast key
   distribution problem.

   NOTE: this proposal requires some simple support mechanisms, which,
   it is recommended here, be integrated into version 3 of IGMP. This
   support is described in Appendix B.

1. Introduction

   Growing concern about the integrity of Internet communication [13]
   (routing information and data traffic) has led to the development of
   an Internet Security Architecture, proposed by the IPSEC working
   group of the IETF [2]. The proposed security mechanisms are
   implemented at the network layer - the layer of the protocol stack at
   which networking resources are best protected [3].

Unlike many network layer protocols, the Core Based Tree (CBT)
multicast protocol [4] makes explicit provision for security; it has
its own protocol header, unlike existing IP multicast schemes
[10,11], and other recently proposed schemes [12].

In this document we describe how the CBT multicast protocol can
provide for the secure joining of a CBT group tree, and how this same
process can provide a scalable solution to the multicast key
distribution problem. These security services are an integral part
of the CBT protocol [4]. Their use is optional, and is dependent on
each individual group's requirements for security. Furthermore, the
use of the CBT multicast protocol for multicast key distribution does
not preclude the use of other multicast protocols for the actual
multicast communication itself, that is, CBT need only be the vehicle
with which to distribute keys.

Secure joining implies the provision for authentication, integrity,
and optionally, confidentiality, of CBT join messages. The scheme we
describe provides for the authentication of tree nodes (routers) and
 receivers (end-systems) as part of the tree joining process. Key
distribution (optional) is an integral part of secure joining.

Network layer multicast protocols, such as DVMRP [7] and M-OSPF [9],
do not have their own protocol header(s), and so cannot provision for
security in themselves; they must rely on whatever security is
provided by IP itself. Multicast key distribution is not addressed to
any significant degree by the new IP security architecture [2].

The CBT security architecture is independent of any particular
cryptotechniques, although many security services, such as
authentication, are easier if public-key cryptotechniques are
employed.

What follows is an overview of the CBT multicasting. The description
of our proposal in section 6.1 assumes the reader is reasonably
familiar with the CBT protocol. Details of the CBT architecture and
protocol can be found in [7] and [4], respectively.

2. Overview of BCT Multicasting

CBT is a new architecture for local and wide-area IP multicasting,
being unique in its utilization of just one shared delivery tree per
group, as opposed to the source-based delivery tree approach of
existing IP multicast schemes, such as DVMRP and MOSPF.

A shared multicast delivery tree is built around several so-called
core routers. A group receiver's local multicast router is required
to explicitly join the corresponding delivery tree after receiving an

IGMP [8] group membership report over a directly connected interface.
A CBT join message is targeted at one of the group's core routers.
The resulting acknowledgement traverses the reverse-path of the join,
resulting in the creation of a tree branch. Routers along these
branches are called non-core routers for the group, and there exists
a parent-child relationship between adjacent routers along a branch
of the same tree (group).

3. How the CBT Architecture Complements Security

   The CBT architecture requires "leaf" routers to explicitly join a CBT
   tree. Hence, CBT is not data driven; the ack associated with a join
   "fixes" tree state in the routers that make up the tree. This so-
   called "hard state" remains until the tree re-configures, for
   example, due to receivers leaving the group, or because an upstream
   failure has occurred. The CBT protocol incorporates mechanisms
   enabling a CBT tree to repair itself in the event of the latter.

   As far as the establishment of an authenticated multicast
   distribution tree is concerned, DVMRP, M-OSPF, and PIM, are at a
   disadvan- tage; the nature of their "soft state" means a delivery
   tree only exists as long as there is data flow. Also, routers
   implementing a multicast protocol that builds its delivery tree based
   on a reverse-path check (like DVMRP and PIM dense mode) cannot be
   sure of the previous-hop router, but only the interface a multicast
   packet arrived on.

   These problems do not occur in the CBT architecture. CBT's hard state
   approach means that all routers that make up a delivery tree know who
   their on-tree neighbours are; these neighbours can be authenticated
   as part of delivery tree set-up. As part of secure tree set-up,
   neighbours could exchange a secret packet handle for inclusion in the
   CBT header of data packets exchanged between those neighbours,
   allowing for the simple and efficient hop-by-hop authentication of
   data packets (on-tree).

   The presence of tree focal points (i.e. cores) provides CBT trees
   with natural authorization points (from a security viewpoint) -- the
   formation of a CBT tree requires a core to acknowledge at least one
   join in order for a tree branch to be formed. Thereafter,
   authorization and key distribution capability can be passed on to
   joining nodes that are authenticated.

   In terms of security, CBT's hard state approach offers several
   additional advantages: once a multicast tree is established, tree
   state maintained in the routers that make up the tree does not time
   out or change necessarily to reflect underlying unicast topology.
   The security implications of this are that nodes need not be subject

to repeated authentication subsequent to a period of inactivity, and
tree nodes do not need to re-authenticate themselves as a result of
an underlying unicast topology change, unless of course, an network
(node) failure has occurred.

Hard-state protocol mechanisms are often thought of as being less
fault tolerant than soft-state schemes, but there are pros and cons
to both approaches; we see here that security is one of the pros.

4. The Multicast Key Distribution Problem

We believe that multicast key distribution needs to be combined with
group access control. Without group access control, there is no point
in employing multicast key distribution, since, if there are no group
restrictions, then it should not matter to whom multicast information
is divulged.

There are different ways of addressing group access control. The
group access control we describe requires identifying one group
member (we suggest in [14] that this should be the group initiator)
who has the ability to create, modify and delete all or part of a
group access control list. The enforcement of group access control
may be done by a network entity external to the group, or by a group
member.

The essential problem of distributing a session (or group) key to a
group of multicast receivers lies in the fact that some central key
management entity, such as a key distribution centre (KDC) (A Key
Distribution Centre (KDC) is a network entity, usually residing at a
well-known address. It is a third party entity whose responsibility
it to generate and distribute symmetric key(s) to peers, or group
receivers in the case of multicast, wishing to engage in a "secure"
communication. It must therefore be able to identify and reliably
authenticate requestors of symmetric keys.), must authenticate each
of a group's receivers, as well as securely distribute a session key
to each of them. This involves encrypting the relevant message n
times, once with each secret key shared between the KDC and
corresponding receiver (or alternatively, with the public key of the
receiver), before multicasting it to the group. (Alternatively, the
KDC could send an encrypted message to each of the receivers
individually, but this does not scale either.) Potentially, n may be
very large. Encrypting the group key with the secret key (of a
secret-public key pair) of the KDC is not an option, since the group
key would be accessible to anyone holding the KDC's public key, and
public keys are either well-known or readily available. In short,
existing multicast key distribution methods do not scale.

The scaling problem of secure multicast key distribution is
compounded for the case where sender-specific keys need to be
distributed to a group. This is required for sender-specific
authentication of data traffic. It is not possible to achieve per-
sender authentication, given only a group session key.

Recently a proposal has emerged, called the Group Key Management
Protocol (GKMP) [15]. This was designed for military networks, but
the authors have demonstrated how the architecture could be applied
to a network like the Internet, running receiver-oriented multicast
applications.

GKMP goes a considerable way to addressing the problems of multicast
key distribution: it does not rely on a centralised KDC, but rather
places the burden of key management on a group member(s). This is the
approach adopted by the CBT solution, but our solution can take this
distributed approach further, which makes our scheme that much more
scalable. Furthermore, our scheme is relatively simple.

The CBT model for multicast key distribution is unique in that it is
integrated into the CBT multicast protocol itself. It offers a
simple, low-cost, scalable solution to multicast key distribution. We
describe the CBT multicast key distribution approach below.

5. Multicast Security Associations

   The IP security architecture [2] introduces the concept of "Security
   Associations" (SAs), which must be negotiated in advance during the
   key management phase, using a protocol such as Photuris [20], or
   ISAKMP [21]. A Security Association is normally one-way, so if two-
   way communication is to take place (e.g. a typical TCP connection),
   then two Security Associations need to be negotiated. During the
   negotiation phase, the destination system normally assigns a Security
   Parameter Index to the association, which is used, together with the
   destination address (or, for the sender, the sender's user-id) to
   index into a Security Association table, maintained by the
   communicating parties. This table enables those parties to index the
   correct security parameters pertinent to an association. The
   security association parameters include authentication algorithm,
   algorithm mode, cryptographic keys, key lifetime, sensitivity level,
   etc.

   The establishment of Security Associations (SA) for multicast
   communication does not scale using protocols like Photuris, or
   ISAKMP. This is why it is often assumed that a multicast group will
   be part of a single Security Association, and hence share a single
   SPI. It is assumed that one entity (or a pair of entities) creates
   the SPI "by some means" (which may be an SA negotiation protocol,

like [20] and [21]), which is then simply multicast, together with
the SA parameters, to the group for subsequent use. However, this
precludes multicast receivers from performing sender-specific origin
authentication; all a receiver can be sure of is that the sender is
part of the multicast Security Association.

We advocate that the primary core, either alone, or in conjunction
with the group initiator, establish the security parameters to be
used in the group communication. These are distributed as part of the
secure join process. Thereafter, individual senders can distribute
their own key and security parameters to the group. In the case of
the latter, there are two cases to consider:

+   the sender is already a group member. In this case, the sender
     can decide upon/generate its own security parameters, and multi-
     cast them to the group using the current group session key.

+   the sender is not a group member. In this case, before the
     sender begins sending, it must first negotiate the security
     parameters with the primary core, using a protocol such as Pho-
     turis [20] or ISAKMP [21]. Once completed, the primary core
     multicasts (securely) the new sender's session key and security
     parameters to the group.

Given that we assume the use of asymmetric cryptotechniques
throughout, this scheme provides a scalable solution to multicast
origin authentication.

Sender-specific keys are also discussed in section 8.

6. The CBT Multicast Key Distribution Model

The security architecture we propose allows not only for the secure
joining of a CBT multicast tree, but also provides a solution to the
multicast key distribution problem [16]. Multicast key distribution
is an optional, but integral, part of the secure tree joining
process; if a group session key is not required, its distribution may
be omitted.

The use of CBT for scalable multicast key distribution does not
preclude the use of other multicast protocols for the actual
multicast communication. CBT could be used solely for multicast key
distribution -- any multicast protocol could be used for the actual
multicast communication itself.

The model that we propose does not rely on the presence of a
centralised KDC -- indeed, the KDC we propose need not be dedicated
to key distribution. We are proposing that each group have its own

group key distribution centre (GKDC), and that the functions it
provides should be able to be "passed on" to other nodes as they join
the tree. Hence, our scheme involves truly distributed key
distribution capability, and is therefore scalable. It does not
require dedicated KDCs. We are proposing that a CBT primary core
initially take on the role of a GKDC.

6.1 Operational Overview

When a CBT group is created, it is the group initiator's
responsibility to create a multicast group access control list (ACL)
[14]. It is recommended that this list is a digitally signed
"document", the same as (or along the lines of) an X.509 certificate
[9], such that it can be authenticated. The group initiator
subsequently unicasts the ACL to the primary core for the group. This
communication is not part of the CBT protocol. The ACL's digital
signature ensures that it cannot be modified in transit without
detection. If the group membership itself is sensitive information,
the ACL can be additionally encrypted with the public key of the
primary core before being sent. The ACL can be an "inclusion" list
or an "exclusion" list, depending on whether group membership
includes relatively few, or excludes relatively few.

The ACL described above consists of group membership (inclusion or
exclusion) information, which can be at the granularity of hosts or
users. How these granularities are specified is outside the scope of
this document. Additionally, it may be desirable to restrict key
distribution capability to certain "trusted" nodes (routers) in the
network, such that only those trusted nodes will be given key
distribution capability should they become part of a CBT delivery
tree. For this case, an additional ACL is required comprising
"trusted" network nodes.

The primary core creates a session key subsequent to receiving and
authenticating the message containing the access control list. The
primary core also creates a key encrypting key (KEK) which is used
for re-keying the group just prior to an old key exceeding its life-
time. This re-keying strategy means that an active key is less
likely to become compromised during its lifetime.

The ACL(s), group key, and KEK are distributed to secondary cores as
they become part of the distribution tree.

Any tree node with this information can authenticate a joining
member, and hence, secure tree joining and multicast session key
distribution are truly distributed across already authenticated tree
nodes.

6.2 Integrated Join Authentication and Multicast Key Distribution

   For simplicity, in our example we assume the presence of an
   internetwork-wide asymmetric key management scheme, such as that
   proposed in [17]. However, we are not precluding the use of
   symmetric cryptographic techniques -- all of the security services we
   are proposing, i.e. integrity, authentication, and confidentiality,
   can all be achieved using symmetric cryptography, albeit a greater
   expense, e.g. negotiation with a third party to establish pairwise
   secret keys. For these reasons, we assume that a public (asymmetric)
   key management scheme is globally available, for example, through the
   Domain Name System (DNS) [17] or World Wide Web [18].

   NOTE: given the presence of asymmetric keys, we can assume digital
   signatures provide integrity and origin authentication services
   combined.

   The terminology we use here is described in Appendix A. We formally
   define some additional terms here:

   +   grpKey: group key used for encrypting group data traffic.

   +   ACL: group access control list.

   +   KEK: key encrypting key, used for re-keying a group with a new
        group key.

   +   SAparams: Security Association parameters, including SPI.

   +   group access package (grpAP): sent from an already verified tree
        node to a joining node.

        [token_sender, [ACL]^SK_core, {[grpKey, KEK,
        SAparams]^SK_core}^PK_origin-host,
        {[grpKey, KEK, SAparams]^SK_core}^PK_next-hop]^SK_sender

        NOTE: SK_core is the secret key of the PRIMARY core.

   As we have already stated, the elected primary core of a CBT tree
   takes on the initial role of GKDC. In our example, we assume that a
   group access control list has already been securely communicated to
   the primary core. Also, it is assumed the primary core has already
   participated in a Security Association estabishment protocol [20,21],
   and thus, holds a group key, a key-encrypting key, and an SPI.

      NOTE, there is a minor modification required to the CBT protocol
      [4], which is as follows: when a secondary core receives a join,
      instead of sending an ack followed by a re-join to the primary,

the secondary forwards the join to the primary; the ack travels
from the primary (or intermediate on-tree router) back to the join
origin. All routers (or only specific routers) become GKDCs after
they receive the ack.

We now demonstrate, by means of an example, how CBT routers join a
tree securely, and become GKDCs. For clarity, in the example, it is
assumed all routers are authorised to become GKDCs, i.e. there is no
trusted-router ACL.

In the diagram below, only one core (the primary) is shown. The
process of a secondary joining the primary follows exactly what we
describe here.

In the diagram, host h wishes to join multicast group G. Its local
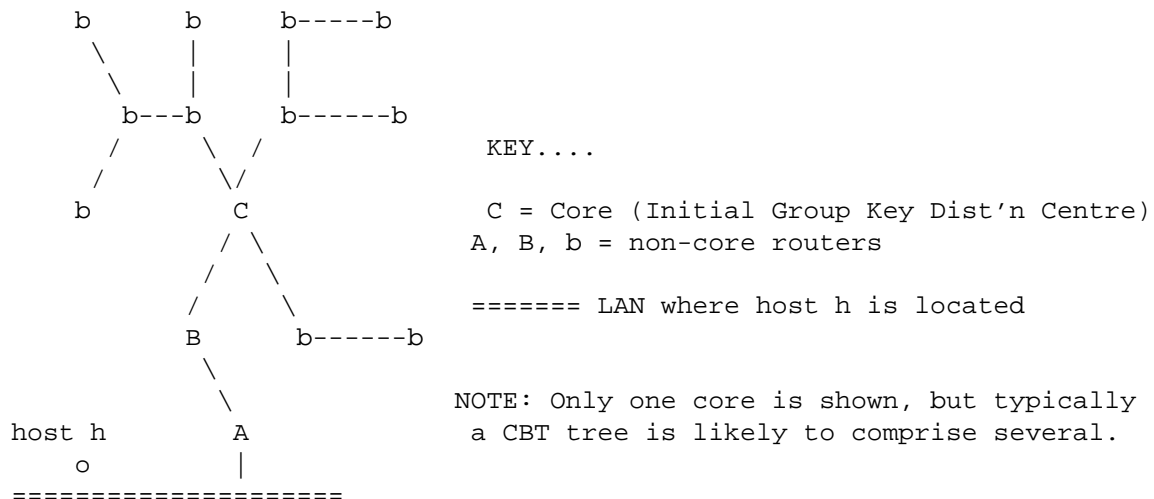multicast router (router A) has not yet joined the CBT tree for the
group G.

```
  b       b      b-----b
   \      |      |
    \     |      |
    b---b     b------b
   /     \  /               KEY....
  /       \/
 b         C                C = Core (Initial Group Key Dist'n Centre)
          / \               A, B, b = non-core routers
         /   \
        /     \             ======= LAN where host h is located
       B      b------b
        \
         \                  NOTE: Only one core is shown, but typically
host h        A             a CBT tree is likely to comprise several.
   o          |
=====================
```

Figure 1: Example of Multicast Key Distribution using CBT

A branch is created as part of the CBT secure tree joining process,
as follows:

+   Immediately subsequent to a multicast application starting up on
     host h, host h immediately sends an IGMP group membership
     report, addressed to the group. This report is not suppressible
     (see Appendix B), like other IGMP report types, and it also
     includes the reporting host's token, which is digitally signed

     h --> DR (A): [[token_h]^SK_h, IGMP group membership report]

(A host's token differs in two respects compared with tokens
defined in [9]. To refresh, a token assists a recipient in the
verification process, and typically contains: recipient's
unique identity, a timestamp, and a pseudo-random number. A
token is also usually digitally signed by its originator.
Firstly, A host's token does not contain the intended
recipient's identity, since this token may need to traverse
several CBT routers before reaching a GKDC. A host does not
actually know which router, i.e. GKDC, will actually
acknowledge the join that it invoked. Secondly, the host's
token is digitally signed -- this is usual for a token.
However, tokens generated by routers need not be explicitly
digitally signed because the JOIN-REQUESTs and JOIN-ACKs that
carry them are themselves digitally signed.)

+   In response to receiving the IGMP report, the local designated
     router (router A) authenticates the host's enclosed token. If
     successful, router A formulates a CBT join-request, whose target
     is core C (the primary core). Router A includes its own token in
     the join, as well as the signed token received from host h. The
     join is digitally signed by router A.

     NOTE 1: router A, like all CBT routers, is configured with the
     unicast addresses of a prioritized list of cores, for different
     group sets, so that joins can be targeted accordingly.

     NOTE 2: the host token is authenticated at most twice, once by
     the host's local CBT router, and once by a GKDC. If the local
     router is already a GKDC, then authentication only happens once.
     If the local router is not already a GKDC, a failed authentica-
     tion check removes the overhead of generating and sending a CBT
     join-request.

     Router A unicasts the join to the best next-hop router on the
     path to core C (router B).

         A --> B: [[token_A], [token_h]^SK_h, JOIN-REQUEST]^SK_A

+   B authenticates A's join-request. If successful, B repeats the
     previous step, but now the join is sent from B to C (the pri-
     mary, and target), and the join includes B's token. Host h's
     token is copied to this new join.

         B --> C: [[token_B], [token_h]^SK_h, JOIN-REQUEST]^SK_B

+   C authenticates B's join. As the tree's primary authorization
     point (and GKDC), C also authenticates host h, which triggered
     the join process. For this to be successful, host h must be

included in the GKDC's access control list for the group. If h
is not in the corresponding access control list, authentication
is redundant, and a join-nack is returned from C to B, which
eventually reaches host h's local DR, A.

Assuming successful authentication of B and h, C forms a group
access package (grpAP), encapsulates it in a join-ack, and digi-
tally signs the complete message. C's token, host h's signed
token, a signed ACL, and two (group key, KEK) pairs are included
in the group access package; one for the originating host, and
one for the next-hop CBT router to which the join-ack is des-
tined. Each key pair is digitally signed by the issuer, i.e. the
primary core for the group. The host key pair is encrypted using
the public key of the originating host, so as to be only deci-
pherable by the originating host, and the other key pair is
encrypted using the public key of the next-hop router to which
the ack is destined -- in this case, B. Host h's token is used
by the router connected to the subnet where h resides so as to
be able to identify the new member.

        C --> B: [[token^h]^SK_h, grpAP, JOIN-ACK]^SK_C

+   B authenticates the join-ack from C. B extracts its encrypted
      key pair from the group access package, decrypts it, authenti-
      cates the primary core, and stores the key pair in encrypted
      form, using a local key. B also verifies the digital signature
      included with the access control list. It subsequently stores
      the ACL in an appropriate table. The originating host key pair
      remains enciphered.

      The other copy of router B's key pair is taken and deciphered
      using its secret key, and immediately enciphered with the public
      key of next-hop to which a join-ack must be passed, i.e. router
      A. A group access package is formulated by B for A. It contains
      B's token, the group ACL (which is digitally signed by the pri-
      mary core), a (group key, KEK) pair encrypted using the public
      key of A, and the originating host's key pair, already
      encrypted. The group access package is encapsulated in a join-
      ack, the complete message is digitally signed by B, then for-
      warded to A.

        B --> A: [[token^h]^SK_h, grpAP, JOIN-ACK]^SK_B

+   A authenticates the join-ack received from B. A copy of the
      encrypted key pair that is for itself is extracted from the
      group access package and deciphered, and the key issuer (primary
      core) is authenticated. If successful, the enciphered key pair
      is stored by A. The digital signature of the included access

control list is also verified, and stored in an appropriate
table. The key pair encrypted for host h is extracted from the
group access package, and is forwarded directly to host h, which
is identified from the presence of its signed token. On
receipt, host h decrypts the key pair for subsequent use, and
stores the SA parameters in its SA table.

    A --> h: [[token^h]^SK_h, {grpKey, KEK, SAparams}^PK_h]

Going back to the initial step of the tree-joining procedure, if the
DR for the group being joined by host h were already established as
part of the corresponding tree, it would already be a GKDC. It would
therefore be able to directly pass the group key and KEK to host h
after receiving an IGMP group membership report from h:

    A --> h: [[token^h]^SK_h, {grpKey, KEK, SAparams}^PK_h]

If paths, or nodes fail, a new route to a core is gleaned as normal
from the underlying unicast routing table, and the re-joining process
(see [4]) occurs in the same secure fashion.

7. A Question of Trust

The security architecture we have described, involving multicast key
distribution, assumes that all routers on a delivery tree are trusted
and do not misbehave. A pertinent question is: is it reasonable to
assume that network routers do not misbehave and are adequately
protected from malicious attacks?

Many would argue that this is not a reasonable assumption, and
therefore the level of security should be increased to discount the
threat of misbehaving routers. As we described above, routers
periodically decrypt key pairs in order to verify them, and/or re-
encrypt them to pass them on to joining neighbour routers.

In view of the above, we suggest that if more stringent security is
required, the model we presented earlier should be slightly amended
to accommodate this requirement. However, depending on the security
requirement and perceived threat, the model we presented may be
acceptable.

We recommend the following change to the model already presented
above, to provide a higher level of security:

All join-requests must be authenticated by a core router, i.e. a join
arriving at an on-tree router must be forwarded upstream to a core if
the join is identified as being a "secure" join (as indicated by the
presence of a signed host token).

   The implication of this is that key distribution capability remains
   with the core routers and is not distributed to non-core routers
   whose joins have been authenticated. Whilst this makes our model
   somewhat less distributed than it was before, the concept of key
   distribution being delegated to the responsibility of individual
   groups remains. Our scheme therefore retains its attractiveness over
   centralized schemes.

8. The Multicast Distribution of Sender-Specific Keys

   Section 5, in part, discussed the scalable distribution of sender-
   specific keys and sender-specific security parameters to a multicast
   group, for both member-senders, and non-member senders. If asymmetric
   cryptotechniques are employed, this allows for sender-specific origin
   authentication.

   For member-senders, the following message is multicast to the group,
   encrypted using the current group session key, prior to the new
   sender transmitting data:

           {[sender_key, senderSAparams]^SK_sender}^group_key

   Non-member senders must first negotiate (e.g. using Photuris or
   ISAKMP) with the primary core, to establish the security association
   parameters, and the session key, for the sender. The sender, of
   course, is subject to access control at the primary. Thereafter, the
   primary multicasts the sender-specific session key, together with
   sender's security parameters to the group, using the group's current
   session key. Receivers are thus able to perform origin
   authentication.

                        Photuris or ISAKMP
          1. sender <---------------------> primary core

        2. {[sender_key, senderSAparams]^SK_primary}^group_key

   For numerous reasons, it may be desirable to exclude certain group
   members from all or part of a group's communication. We cannot offer
   any solution to providing this capability, other than requiring new
   keys to be distributed via the establishment of a newly-formed group
   (CBT tree).

9. Summary

   This memo has offered a scalable solution to the multicast key
   distribution problem. Our solution is based on the CBT architecture
   and protocol, but this should not preclude the use of other multicast
   protocols for secure multicast communication subsequent to key
   distribution. Furthermore, virtually all of the functionality present
   in our solution is in-built in the secure version of the CBT
   protocol, making multicast key distribution an optional, but integral
   part, of the CBT protocol.

Appendix A

   The following terminology is used throughout this document:

   +   PK_A indicates the public key of entity A.

   +   SK_A indicates the secret key of entity A. The secret key can be
        used by a sender to digitally sign a digest of the message,
        which is computed using a strong, one-way hash function, such as
        MD5 [19].

   +   Unencrypted messages will appear enclosed within square brack-
        ets, e.g. [X, Y, Z]. If a message is digitally signed, a super-
        script will appear outside the right hand bracket, indicating
        the message signer. Encrypted messages appear enclosed within
        curly braces, with a superscript on the top right hand side out-
        side the closing curly brace indicating the encryption key, e.g.
        $\{X, Y, Z\}^{PK\_A}$.

   +   a token is information sent as part of a strong authentication
        exchange, which aids a receiver in the message verification pro-
        cess. It consists of a timestamp, t (to demonstrate message
        freshness), a random, non-repeating number, r (to demonstrate
        message originality), and the unique name of the message
        recipient (to demonstrate that the message is indeed intended
        for the recipient). A digital signature is appended to the
        token by the sender (which allows the recipient to authenticate
        the sender). The token is as follows:

             $[t\_A, r\_A, B]^{SK\_A}$ -- token sent from A to B.

   +   A --> B: -- denotes a message sent from A to B.

Appendix B

   The group access controls described in this document require a few
   simple support mechanisms, which, we recommend, be integrated into
   version 3 of IGMP. This would be a logical inclusion to IGMP, given
   that version 3 is expected to accommodate a variety of multicast
   requirements, including security. Furthermore, this would remove the
   need for the integration of a separate support protocol in hosts.

   To refresh, IGMP [8] is a query/response multicast support protocol
   that operates between a multicast router and attached hosts.

   Whenever an multicast application starts on a host, that host
   generates a small number of IGMP group membership reports in quick
   succession (to overcome potential loss). Thereafter, a host only

issues a report in response to an IGMP query (issued by the local
multicast router), but only if the host has not received a report for
the same group (issued by some other host on the same subnet) before
the host's IGMP random response timer expires. Hence, IGMP,
incorporates a report "suppression" mechanism to help avoid "IGMP
storms" on a subnet, and generally conserve bandwidth.

We propose that IGMP accommodate "secure joins" - IGMP reports that
indicate the presence of a digitally signed host (or user) token.
These report types must not be suppressible, as is typically the case
with IGMP reports; it must be possible for each host to independently
report its group presence to the local router, since a GKDC bases its
group access control decision on this information.

This functionality should not adversely affect backwards
compatibility with earlier versions of IGMP that may be present on
the same subnet; the new reports will simply be ignored by older IGMP
versions, which thus continue to operate normally.

Security Considerations

Security issues are discussed throughout this memo.

Author's Address

Tony Ballardie,
Department of Computer Science,
University College London,
Gower Street,
London, WC1E 6BT,
ENGLAND, U.K.

Phone: ++44 (0)71 419 3462
EMail: A.Ballardie@cs.ucl.ac.uk

References

[1] MBONE, The Multicast BackbONE; M. Macedonia and D. Brutzman;
available from http://www.cs.ucl.ac.uk/mice/mbone_review.html.

[2] R. Atkinson. Security Architecture for the Internet Protocol; RFC
1825, SRI Network Information Center, August 1995.

[3] D. Estrin and G. Tsudik. An End-to-End Argument for Network Layer,
Inter-Domain Access Controls; Journal of Internetworking & Experience,
Vol 2, 71-85, 1991.

[4] A. Ballardie, S. Reeve, N. Jain. Core Based Tree (CBT) Multicast -
Protocol Specification; Work in Progress, 1996. Available from:
ftp://cs.ucl.ac.uk/darpa/IDMR/draft-ietf-idmr-cbt-spec-XX.txt.

[5] R. Atkinson. IP Authentication Header; RFC 1826, SRI Network
Information Center, August 1995.

[6] R. Atkinson. IP Encapsulating Security Payload; RFC 1827, SRI Net-
work Information Center, August 1995.

[7] A. Ballardie. Core Based Tree (CBT) Multicast Architecture; Work
in progress, 1996. Available from:
ftp://cs.ucl.ac.uk/darpa/IDMR/draft-ietf-idmr-cbt-arch-XX.txt

[8] W. Fenner. Internet Group Management Protocol, version 2 (IGMPv2),
Work in progress, 1996.

[9] CCITT Data Communication Networks Directory (Blue Book). Recommen-
dation X.509, Authentication Framework.

[10] T. Pusateri. Distance-Vector Multicast Routing Protocol (DVMRP)
version 3. Working draft, February 1996.

[11] J. Moy. Multicast Extensions to OSPF; RFC 1584, SRI Network
Information Center, March 1994.

[12] D. Estrin et al. Protocol Independent Multicast, protocol specif-
ication; Work in progress, January 1996.

[13] R. Braden, D. Clark, S. Crocker and C. Huitema. Security in the
Internet Architecture. RFC 1636, June 1994.

[14] A. Ballardie and J. Crowcroft. Multicast-Specific Security
Threats and Counter-Measures. In ISOC Symposium on Network and Distri-
buted System Security, February 1995.

[15] H. Harney, C. Muckenhirn, and T. Rivers. Group Key Management
Protocol (GKMP) Architecture. Working draft, 1994.

[16] N. Haller and R. Atkinson. RFC 1704, On Internet Authentication.
SRI Network Information Center, October 1994.

[17] C. Kaufman and D. Eastlake. DNS Security Protocol Extensions.
Working draft, January 1996.

[18] T. Berners-Lee, R. Cailliau, A. Luotonen, H. Frystyk Nielsen, A.
Secret. The World Wide Web. Communications of the ACM, 37(8):76-82,
August 1994.

   [19] R. Rivest. RFC 1321, The MD-5 Message Digest Algorithm, SRI Net-
   work Information Center, 1992.

   [20] P. Karn, W. Simpson. The Photuris Session Key Management Proto-
   col; Working draft, January 1996.

   [21] D. Maughan, M. Schertler. Internet Security Association and Key
   Management Protocol; Working draft, November 1995.