Minutes of Secure Multicast Group (SmuG) 12th Meeting at IETF50, March 2001

1. Status of SMuG - Thomas Hardjono, Nortel

Thomas said that the SMuG drafts have nearly all been migrated to MSEC. We should now run SMuG in parallel with MSEC, where SMuG supports MSEC in the way that the RMRG group supports RMT.

Thomas showed the standard SMuG Framework slide and indicated that we now need to handle the distributed side of the framework. That will be a predominate topic, because people will be asking in MSEC about scalability. Furthermore, SMuG must pay attention what happens in MSEC.

There are future work items (discussed at the end of the meeting). Again, once things become solid enough we should push them over to MSEC.

Hugh asked if we should make some of the SMuG documents information RFCs? Ran replied that yes, and we can do this directly from SMuG.

Mark will be taking Thomas' position as a SMuG chair. This will put new blood into SMuG, and also allows Thomas to focus on MSEC.

2. Anycast Security presentation - Lakshiminath Dondeti, Nortel

Anycast is a cousin of multicast, so this is a very valid SMuG topic to consider. It delivers packets to the "topologically nearest" network element with the address. This provides fault-tolerance.

Anycast routing is defined in RFC 1546. Anycast servers use ARP and Link-level multicast for advertisements and listening. Forwarding is similar to unicast in the network, and last-hop routing is similar to that in multicasting. Any server responding to an anycast request puts its own address as the source of the reply.

The security issues with anycast are similar to those in multicast, along with a few more. Server advertisements are unauthorized, anycast servers are not proven to be authentic, and anycast communications are not checked for integrity or otherwise secure.

Hugh mentioned that he has developed some technology which is similar to anycast. You have to start with policy dissemination to solve the problems of clients knowing that a server is legitimate.

Tassos asked if the problem is authenticating the server or the information that it gives you? Thomas replied that a client sends the anycast message, and then wants to authenticate that the responding server. But it might trust the server after that initial exchange.

Tassos suggested that you can't have 2 unicast messages linked together (i.e. trusting the second one because you trusted the first one) because you don't know if there are going to the same server. But it's not a problem because you're going to switch to a different mode (unicast to the server) after the first packet. Lakshiminath said that with DNS there is no correlation of anycast messages, they're all independent. But with other applications you might need to keep state between anycast messages, especially if you want

fault tolerance.

The issues with Anycast server advertisements include clients having no way of identifying legitimate servers. Any host can advertise itself as an anycast server for purposes of DoS and/or spoofing a real server. The proposed solution is to use anycast server access control measures.

Pete asked if this issue is any different than the multicast case, and the answer was no. Pete asked if this wasn't similar to the problems with routing security? The answer was that you could, but using group access control solution are much simpler. A discussion on the applicability of routing security ensued, where Thomas pointed out that ISPs don't like signed LSAs because they make the routing messages much larger. Hugh pointed out that in the any case you need not only signed messages, but a locally defined policy.

Hugh observed that it sounds as if an administrator is be setting up an anycast service, so it's probably not a router but actually a server. Signed router updates don't apply. The server has policy it is passing. At the application layer it needs a policy so it can distinguish between valid servers and invalid servers. Lakshiminath agreed that certain valid servers need to be defined in the registration process. The IPv6 people thought of routers with ACLs to verify a client can use that service.

Ran asked if the solution was servers authenticating the clients, or vice versa? Lakshiminath said that in the DNS anycast case the client sends out the DNS request, and he just wants to get back a response without doing authentication. Ran reasoned that this puts computational overhead on the servers to do public key operations. Hugh suggested that you can use group keying but you have to give the key to all clients and servers. Brian suggested that if you have the servers and all first hop routers in a group, you could exclude the clients but solve the authentication problem down to the last hop. If you use group keys this may work -- first hop routers probably don't want to do public key verification either. Hugh agreed that if an ISP wants to provide its own services, it can protect it's own servers and its own 1st hop routers. This keeps you from having to trust all of the clients, which is the harder problem.

Pete pointed out that the solution is fine when using anycast for a service, but if you use it for an application than this method won't work. Everyone generally agreed that this is a good research problem.

Hugh asked if there the requirement that there should be no public key operations for verifying server authenticity? Lakshiminath replied no, but that they wouldn't be considered for the immediate solution.

Tassos asked if it would be well advised to have en entity in the local network which could be queried to find if a server is valid? In which case there would be not a client dealing with certificates but the local trusted entity. It was suggested that anycast could route all responses to this proxy, and that the client only contact the proxy. However the client would lose fault tolerance, which is one of the valuable properties of anycast.

Someone suggested that imagine you contact an anycast server without authenticating them, and then do the authentication check after receiving the unicast replay. It's a weaker approach but would be satisfactory to some people. Verify, then put on the authentication list. Pete mentioned that the

fundamental properly of anycast is finding the server. You want to verify the response, but also avoid the DoS attack of invalid servers.

The issues with securing anycast communication include making the communications private. The proposed solution is an IPSec counterpart for anycast. We may also draw from group security work. IPSec could be used for securing anycast, but the fault-tolerant nature of anycast service location would be lost. IKE would have to change slightly: the initiator (client) would send a "requests for service" to the anycast address whereas the responder (server) would reply with source address as its unicast address.

Ran asked about using an Akamai server approach. Lakshiminath agreed that they use an anycast sort of approach. Looking at an MIT paper describing this, a small group of servers keep all state amongst the servers, with one management station deciding when to switch sessions between servers. Anycast probably don't do that though. Hugh suggested that if the anycast group shared a private key (and the pubkey was published), wouldn't that work to solve some of the migration problems? Lakshiminath agreed, and pointed out that you could then also do anycast group authentication. Brian pointed out that revocation of the keypair without service loss then becomes an obvious difficulty. Someone suggested that the group could be torn down to start a new one in that case, but Hugh responded that it was not necessarily workable as the group could use an LKH tree or OFT for the group members.

Tassos said that the problem is not security, but that you want the session to stay up even if the server goes down. Lakshiminath replied that when we throw in security then we have to deal with it. He's not trying to solve the connection migration problem, just the security management worries.

There was a discussion considering whether different servers should share the same unicast IP address so that if one dies the other can take over. But Thomas said that if those servers are in different subnets that won't work. Lakshiminath pointed out that anycast provides more than fault tolerance, there's a primitive load-balancing going on as well.

Someone asked which is the main problem we're trying to solve? Lakshiminath said that there are important problems, and urgent problems! We're doing this to solve the important problems because others are working on the urgent problems already. E.g., IPv6 just solved the access-control problem because that's urgent. But they've asked us to find a better solution. But we could solve the easy important problem now first if we can.

3. NTRU Overview - Ari Singer, NTRU

Ari didn't go over the details of the NTRU algorithm this time. He can send previous presentations to those who need them. He was presenting because he feels that NTRU provides technology which would be particularly suitable for multicast. One problem in the secure multicast space is slow public key encryption, and NTRU provides faster public key technology.

NTRU provide public key encryption and signatures. It is based on hard problems which are different than the ones the RSA and ECC algorithms use. It provides very fast encryption, decryption, digital signature generation and verification, and key generation. This is done with a very small footprint and memory requirements. There are patent issues, of course.

The point to this group is that NTRU greatly diminishes the tradeoffs of speed and power consumption costs. Also, this group is looking at problems for the future, and he expects NTRU to be widely deployed in the future.

The algorithm has had big name people to review it, and the reviews have been public. It's all on their web site.

Thomas pointed out that the IETF doesn't standardize ciphers, although it references them For example Scott Fluhrer is currently doing the AES draft for IPSEC, and suggests that Ari could do the same for NTRU. Ran pointed out that he could also specify how IKE would use NTRU for authentication (signatures and encrypted nonce cases). PKIX also has an algorithm list.

Ran suggested two places NTRU could be used in SMuG and MSEC areas: one is key management for the SA1 (in GDOI or GSAKMP). The other is for source authentication to authenticate each packet individually. The main trouble here is the length of the signature -- it's on the order of 2000 bits. That might be OK for larger packets though. For unicast, the overhead is 12-20 bytes. Ari mentioned that as NTRU keys grow in size, they become about the same length as ECC. At 2048 bits they're close; at 1024 bits they are larger than RSA.

4. Formalizing Security Requirements for GDOI - Cathy Meadows, NRL

In her group at NRL, they are doing research in crypto protocols. She's been working with the GDOI group analyzing the GDOI protocol. But she wants to present this to SMuG now (even though GDOI has moved to MSEC) because this opened some interesting research issues.

The goals of her work are to provide set of formal requirements for GDOI that can provide formal analysis, and provide guidance and understanding for the writers/implementers/users of GDOI.

She writes the formal specification for GDOI requirements using the NPATRL requirements language developed at NRL. However, the language isn't a very readable notation which limits its use. This has prompted research into developing more readable notation for NPATRL.

GDOI has a maze of requirements which include: access control, authentication, secrecy, freshness, and PFS. These interact, which is why it is a maze. For example, secrecy interacts with freshness of keys, etc.

There are some challenges in developing requirements for group protocols. Pairwise protocols have notion of a session where secrecy means keys are not learned by parties not involved in the session, and freshness means that a key is unique to a session. But in a group protocol session, much more open ended: many keys may be distributed in one session, and principals may join and leave the group during a session.

The secrecy requirements for GDOI include forward access control, back access control and PFS. Other requirements may govern the affects of stealing keys.

Hugh asked if there is an assumption that everyone in the group knows where they should get there keys, who can give them updates, etc.? Cathy replied yes, we assume only one well-known GCKS. That would be a good separate requirement to express in this language.

There are two kinds of freshness issues with respect to each role. The first issue is from the GCKS point of view. The requirement is to assume that there is only one valid KEK at a time. There was a discussion about LKH trees which resulted in an understanding that an LKH tree can be considered a single KEK, with the support keys attached to it. The second issue is from the group members point of view. The freshest KEK is the one most recently known to a principal. I.e., the group member just knows about the last key he got, which may not be the latest KEK if he missed a key update. You don't want a group member replacing a KEK he has with an older KEK.

Her next plans are several:

a. To go over requirements with the GDOI designers for accuracy and completeness. This was done for earlier versions.
b. Develop NPATRL into a logic that can provide a formal justification for a "mix/match" approach using ideas from logic programming.
c. Develop "user-friendly" presentation for NPATRL language. She is investigating use of fault trees for this.

5. Large Group Multicast - Hugh Harney, SPARTA

One thing we talked about a long time ago was how to scope the problems of multicast security to get specifications out in a reasonable time. One thing was to split the framework into 3 areas (policy, key management, data transforms). Another was to split between a centralized concept and distributed concept.

We pushed off the distributed concept until we fixed the easier centralized problem, which we've now done for key management (GDOI) and data transforms (MESP). GSAKMP however tries to solve the whole puzzle. The rest of this talk describes how GSAKMP solves the key management distribution problem.

The goals for dealing with large groups are distributed processing, distributed bandwidth usage (for processing of keys), and having a dynamic architecture for a group.

If you have the right atmosphere to allow a group member to disseminate the group key, go ahead. Allow the security infrastructure to group. Security discovery protocols are needed in order to discover the closest key server since (like the anycast case) it might not be pre-configured.

There is be a "group owner" who owns the data. Someone asked if a single administrative entity defines all the policy for a group? Is the policy retained in that one entity forever, or does it extend? Hugh said that he did not add polling group members to find out the policy (e.g., use of mechanism).

There is a "group controller" which is authorized by the group owner to generate keys, do access control, etc. The authorization is kept in the policy token. The policy token describes:

- access control (who's allowed in the group)
- authorization (who can give out the key, who can cause re-keys, etc.)
- mechanisms (entire security for the group -- data crypto policy, crypto policy for messages to group controller, etc)

Thomas suggested that for simplicity we say that the group owner is the

highest entity on the authorization tree. Say there are two GCKSs. Either the policy is given out to the first GCKS which says it's delegatable to the second GCKS, or there are two policy tokens (one for each GCKS).

Someone asked if when you specify the mechanisms, do you also specify how to protect the data on the platform? Hugh replied that you could put that in the policy token if you wanted. That would have to come after we solve the policy problem for the group itself though. You could tie into secure OS mechanism.

Someone asked Hugh if he was assuming an out-of-band mechanism for policy transferal, and he agreed that "magic happens"!

A group owner can allow a group controller to share policy and key distribution with a delegated controller. The delegate isn't a GCKS -- they won't be generating keys, but will be distributing and enforcing access control.
You can either have delegates pull the policy, or you have the group controller push data. Either way, it's a mutually-suspicious exchange where they both validate the role of the other.

Someone asked if a delegated controller can recursively delegate? Hugh said that the policy token could certainly allow that.

A "group member" doesn't know the group infrastructure. He's only allowed to get the group keys.

GSAKMP needed a discovery protocol, and they use a concentric ring multicast approach. Cathy asked if more than 1 server can respond, and the answer was yes.

The discovery protocol works like this: A member sends a "request to join" message. A delegated controller discovers it, and sets up a unicast message with the sender, using IKE or SSL or whatever. (Or nothing if the exchange isn't sensitive. The keys will still be encrypted). The delegated controller sends down keys, etc. and shuts the connection down. Now the member can participate in the group.

So if you have one main GCKS with delegated controllers, they essentially have all categories of SAs (SA1/SA2/SA3) open with the GCKS.

There doesn't need to be one PKI. The policy token can have rules based on multiple sets for access control, etc. For example, anyone with an Entrust certificate from Nortel might be allowed to get to a certain IP address subnet, where anyone from Cisco with a Verisign certificate from Verisign might be allowed to get to a different IP address subnet. But it's all declared in the same policy token.

Someone asked if Hugh had implemented a policy language for GSAKMP? The answer was yes, see the source code.

Cathy asked if you can have more than one Group controller. Hugh clarified that at any one time you may have only one. The group owner can write a new token and/or revoke a group controller, however.

Ran asked what the application was in mind when developing GSAKMP. Hugh said two things impacted the work: 1) satellite system with highly interactive

groups between satellites and changing sets of group stations. 2) Simulation systems which needed multicast security. They wanted to configure 1000s of group members in 5 minutes, then go to another group.

6. Concast - Ken Calvert, University of Kentucky

Ken said that while we has going to talk about Concast in particular, the concepts here are not specific to Concast

The issue is users trusting the infrastructure. The farther out you go in the network, the more you trust the network. That's true for unicast and multicast forwarding, Nack suppression in routing protocols, etc. Services are becoming more sophisticated which makes for harder trust considerations. For example, more routers can munge the packets.

The problem is that many multicast applications involve feedback, and the feedback can be lost. A M-to-1 channel is needed.

Concast solves this with scalability through abstraction. In Concast (like anycast) a single identifier represents a group of senders. This trades extra processing in routes for reduced bandwidth requirements.

Tassos asked how the routers decide to forward the message. Do they decide, then one does it which requires latency? Ken said no. Someone asked how this differs from PGM? Thomas observed that it's a merger of PGM and PGMS.

An example of Comcast usage is the scenario where a multicast sender needs feedback from group members to adjust its transmission rate. All senders reply on a particular Comcast address to the sender. Someone asked what happens if some responders do not respond? You then get a sample out of the population which doesn't have a value. Ken said that you can make it reliable by making sure you get an answer from everyone.

Some final observations were that with Comcast you get an "overlay" for free. Also authentication and authorization are handled separately.

7. Future Work Items - Thomas Hardjono, Nortel

Distributed Designs in SMuG Framework

Solving the distributed problem is in the SMuG charter, and the MSEC people will be looking for solutions from SMuG. Ran mentioned that there are two kinds of distributed design: a) trust is centralized but there are replicated controllers for scalability, and b) delegated distributed policy generation where there is no single point of failure (there may not even be a central controlling entity).

Hugh noted that one mode for GSAKMP is that we can move the control around, but there is only one at a time. The other way is more like a Cliques approach, where we come up with a policy, etc. But the policy token could support both.

Lakshiminath asked if there is a single group owner in the 1-to-M case? Thomas replied that there is one group owner and one group controller. Lakshiminath pointed out that the group owner gets to delegate access control to others. With M-to-M you have to talk about the policy definition being distributed.

Hugh said that at any one time you need a single SA for the whole group. If you have divergent policies, you need to bring them together and decide what is the *group* policy. Once you have it you can distribute it appropriately. It doesn't have to come from a centralized server -- could be "group speak".

Pete asked about the working on the M-to-M multicast case?
Thomas agreed that M-to-M must be considered, but it still will be a ways off.

L3-Receiver Access Control (i.e., IGMP)

This is needed by multicast routing protocols
Brad said that a bunch of people think it's an important problem. There are two parts to the problem:

1) If you have a host behind a non-broadcast domain, and it trusts its upstream router (e.g., in a DSL access network) you need a way to describe multicast policy. How to define multicast policy so an object describing multicast (GRA stuff), 1-to-M, M-to-M, whatever.
2) Authentication to the 1st hop router.

Anycast Security

As described in Lakshiminath's presentation, this is needed by the IPv6 community.

Multicast routing protocols security

PIM-SM/SSM control packet authentication is an issue.
The only draft he knows of is the ad-hoc one out of PIM will use IPSEC AH, and leave the rest to SMuG. It needs key management for multicast routing.
Thomas worked on an ad-hoc draft for this.

RM-protocols security

NAK-based RM protocols security is needed. Thomas did presentation at last IETF on this. The RM security assumes we are going to solve some of these problems. Other RM protocols can be looked at too.

Other topics

Ran suggested looking at the Xcast work. This was the small group multicast which had a BOF 2 IETFs ago. They didn't get a group, but are working underground. Since this is research, it
might be worth looking at even if they aren't accepted from IETF. A general discussion followed after which it was noted that there wasn't any support for their protocols because their design was flawed. The consensus was that this probably shouldn't be a high priority.

Hugh suggested looking at security for overlay networks (e.g., CPN).

Brad suggested looking at APAX and BEEP. BEEP is peer-to-peer application which includes framing and other semantics for applications. The message passing framework is APAX -- multiple APAX
networks of multiple BEEP channels which is kind of an overlay network.
He's looking at MAPEX (Multicast APEX).

Hugh mentioned that one interesting part is securing the overlay protocols themselves.

Thomas noted that for all of these projects we need people to drive them, or they won't happen.

8. Next Meeting - Thomas Hardjono, Nortel

We used to have meetings between IETFs. The next meeting being in London, maybe it would be better to meeting in the USA mainland someplace in between, or instead of in London. Hugh pointed out that most of us have to go to MSEC and will be in London anyway. We might not have travel budgets to travel someplace else as well. So given travel budgets, it will probably be held at the London IETF.

It was decided that rather than hold the meeting on the Sunday previous to the IETF, that it would be better to hold it during the lunch break between IETF sessions for couple of days.

Attendees

Ran Canetti	canetti@watson.ibm.com
Carsten Bormann	cabo@tzi.org
Ari Singer	asinger@ntru.com
Paul Judge	judge@cc.gatech.com
Peter Dinsmore	peter_dinsmore@nai.com
Yann Guinamano	yann@guinamano.com
Radha Poovendran	radha@ee.washington.edu
Angela Schuett	amschue@tycho.ncsc.mil
Haixiang He	haixiang@nrotelnetworks.com
Jean-Michel Combes	jeanmichel.combes@rd.francetelecom.com
Ken Calvert	calvert@netlab.uky.edu
David Grawrock	david.grawrock@intel.com
Hugh Harney	hh@sparta.com
Cathy Meadows	meadows@itd.nrl.navy.mil
Thomas Hardjono	hardjono@nortelnetworks.com
Tassos Nakassis	anakassis@nist.gov
Brian Weis	bew@cisco.com
Lakshminath Doneti	ldondeti@nortelnetworks.com
Fred Kaudel	fred.kaudel@flukenetworks.com
Hitoshi Kaudel	asaeda@yamato.ibm.com
Ross Finlayson	finlayson@live.com
Brad Cain	bcain@cereva.com