SMUG MEETING
GLENWOOD, MD
September 23-24, 1999

Notes by David Balenson

PARTICIPANTS

1. Thomas Hardjono, Nortel, 978-288-4538, thardjono@baynetworks.com
2. Ran Canetti, IBM Research, canetti@watson.ibm.com
3. Senthil Sengodan, Nokia, 781-359-5112, senthil.sengodan@nokia.com
4. Joe Wesley, Sun, 781-442-0165, joe.wesley@sun.com
5. Eric Harder, US DoD, 301-688-0850, ejh@tycho.ncsc.mil
6. David Balenson, NAI Labs, 443-259-2358, balenson@tislabs.com
7. Hugh Harney, Sparta, 410-381-9400, x203, hh@sparta.com
8. Carl Muckenhirn, Sparta, 410-381-9400, cfm@columbia.sparta.com
9. Allison Mankin, USC/ISI, 703-812-3706, mankin@isi.edu
10. Brad Weiss, Cisco, 408-526-4796, bew@cisco.com
11. Pankaj Rohatgi, IBM Research, 914-784-7251, rohatgi@watson.ibm.com
12. Pete Dinsmore, NAI Labs, 443-259-2346, pdinsmore@nai.com
13. Patrick McDaniel, UMich, 734-665-4126, pdmcdan@eecs.umich.edu
14. Amit Kleinman, NDS, 972-2-589-4343, akleinmann@ndsisrael.com
15. Atul Prakash, Umich, 734-763-1585, aprakash@umich.edu

D A Y O N E

AGENDA, RAN CANETTI

Thursday, 9/23

9:30-10:10 Presentations:
- 9:30-9:50 Patrick McDaniel, U of Michigan, Antigone, a framework for secure group communication policy management
- 9:50-10:10 Pankaj Rohatgi, IBM Research, The Hybrid Signature Scheme

10:15 - 12:30 Framework and Building Blocks discussion
- 10:15-10:30 Ran Canetti (IBM Research): Presentation of the suggested Framework and Building Blocks draft, following the Oslo discussions.
- 10:30-12:30 Open discussion

12:30-13:30 Lunch

13:30-14:30 Framework and Building Blocks discussion - conclusion
- Open discussion, with the hope to reach a rough consensus.

14:30-15:00 Break

15:00-17:00 Data Transform BBs
- 15:00-15:15 Pankaj Rohatgi: Overview of desired functionality and requirements from the data transforms (with emphasis on authentication).
- 15:15-17:00 Open discussion

Friday, 9/24

9:30-12:00 Group-Controller-to-Group-Members BBs discussion
- 9:30-9:45 Thomas Hardjono (Nortel): Overview of desired functionality and requirements from the Group-Member-to-Group-Control protocols.
- 9:45-12:00 Open discussion

12:00-13:00 Lunch
13:00-16:00 Group-Controller-to-Policy-Server BBs discussion
13:00-13:15 Pete Dinsmore (NAI): Overview of desired functionality and requirements from the Group-Controller-to-Policy-Server protocols.
13:15-13:30 Hugh Harney (Sparta): Overview of desired functionality and requirements from the Group-Controller-to-Policy-Server protocols.
13:30-16:00 Open discussion
16:00: Adjourn meeting.

ANTIGONE
Patrick McDaniel

Antigione: A flexible framework for secure group communications
- Patrick McDaniel, Atul Prakash, EECS Department, University of Michigan
- Peter Honeyman, CITI, University of Michigan

Secure group policies
- what is the design space of secure group communications policies?
- can the range of policies be addressed with a single toolkit?

(Do we really need another?)

Problem statement
- in many of these approaches, both the policy dimensions and policies from which secure group is defined is limited
  - application developers must accept the avail policies
    - features
    - security
    - performance

(Antigone) Secure Group Policy
- session rekeying policy
- application message policy
- membership policy
- process failure policy

Session relying policy
- when is the group security context changed?
- membership *sensitivity*
  - JOIN
  - LEAVE
  - PROCESS_FAILURE
  - MEMBER_EJECT
- time sensitivity

Application message policy
- what are the guarantees provided for application messages?
  - threat model
- guarantees
  - integrity

    - confidentiality
    - group authenticity
    - sender authenticity

Membership policy
- what information about the membership, if any, is delivered to the group members?
    - delivery guarantees
- sensitive to group membership events

Process failure policies
- under what circumstances is a failure detected?
    - crash model
- what can we assume about malicious parties?
    - fault masking
    - denial of service

What is Antigone?
- middleware layer
- flexible interface for definition and implementation of wide range of secure group policies
- infrastructure independent
- transport layer independence
- policy implementing abstractions

Antigone architecture
- application
- 3 layers
    - predefined policies
        - rekeying policy
        - application message policy
        - membership policy
        - process failure policy
    - mechanisms
        - authenticate
        - join
        - rekey/group membership
        - application message
        - failure detection
        - leave
    - broadcast transport layer at bottom, single abstraction for multicasts group
        - point-to-point
        - asymmetric multicast
        - symmetric multicast
- multicast/TCP
- IP

Mechanism layer
- mechanisms
    - authentication
    - join
    - rekeying/group membership

    - application messaging
    - failure detection (and *recovery*)
    - leave
- *micro-protocols*

Example: Authentication
a -> SL: A, I_0
SL -> TTP: SL, A, I_1
TTP -> SL: $\{\{A\}_{KSL}$ XOR $\{SL\}_{KA}, I_1\}_{KSL}$
SL -> A: SL, A, {GID, A, I_0, I_2, [Policy Block], $PU_G\}_{\{SL\}KA}$
- authenticate user to the group (Leighton-Micali)
- establish a long term *pair key* between SL and A
- distribute a *policy block*

Predefine policy layers
- intended as demonstration of policy implementation
- event based communication between layers
- policy independent from mechanism
- the policies available in many existing systems can be constructed using Antigone

Example: session rekeying policy

Policy	JOIN	LEAVE	FAILURE	EJECT
Time sensitive	N	N	N	N
Leave sensitive	N	Y	Y	Y
Join sensitive	Y	N	N	Y
Membership sensitive	Y	Y	Y	Y

- membership events are detected by mechanism layer
- events signaled to policy layers
- session rekeyed according to application policy

Transport layer
- single abstraction providing an unreliable group broadcast channel
    - symmetric multicast
    - point-to-point
    - asymmetric multicast

sender --> session leader --> receiver1, 2, … n

Conclusions
- using Antigone, applications can define and implement a wide range of policies
- policy independent from mechanisms
- mechanism independence
- predefine policies represent those provided by existing systems
- transport layer independence

Future work
- explore design space of secure group policies
    - explore adaptive / role based policies
- implement additional mechanisms

    - authentication, session keying, application messaging, failure detection and recovery
- benchmark implementation
- integrate with real world applications

Contact information
- source and documentation is available at http://antigone.citi.umich.edu/
- please send all questions and comments to pdmcdan@eecs.umich.edu

A HYBRID SIGNATURE SCHEME FOR MULTICAST SOURCE AUTHENTICATION
Pankaj Rohatgi, IBM TJ Watson Research Center

I-D written some time back. Upcoming paper in ACSAC conference. Multicast source auth problem.

Secure multicast secure authentication problem
- how to authenticate the source of a multicast data packet?
    - unicast approach does not work:
        - shared secret based MAC on each packet does not identify source
        - useful only for group authentication: data originates for some unknown group member
    - fresh approach needed for identifying source
    - signatures on each packet is too expensive
    - 1024 bits RSA speeds:
        - 400 MHz Pentium: 50/s
        - specialized HW: 200/s

More efficient alternatives?
- elliptic curve cryptography?
- stream signatures [GR]
    - need reliability
    - large space overhead
- asymmetric MACs [CGIMNP]
    - secure only when collusion is bounded
- flow signatures [WL] -- also subject of recent draft
- hybrid signatures ("what I am proposing")

Flow signatures [WL]
- basic philosophy
    - very few signatures are possible every second
    - how to accommodate high bitrate applications?
        - application sends very few, very large packets
            - each packet can be individually signed
        - or, if applications sends arbitrary sized packets the aggregate a single public key signature over several packets

Wong-Lam scheme
- split each second into few intervals
- during each time interval
    - intercept and buffer all outgoing packets

- at end of time interval
  - create an authentication (hash) tree of all collected packets (hash of packets at leaves)
  - sign root of hash tree
  - append authentication information top each packet: root signature + hashes on path from leaf to root
  - transit modified packets

Analysis of Wong Lam scheme
- advantages:
  - good for smooth unidirectional flows, e.g., video, audio
- disadvantages:
  - latency unacceptable in interactive applications
    - distributed games/simulations
- cannot handle multiple flows (non-intuitive)
  - suppose signature rate = 10/s
    - can handle single 1000 packets with latency 0.2s
    - cannot handle 50 low-bitrate flows (1 packet/s) unless latency = 5s or information shared between flows
- authentication overhead not uniform
  - more overhead with higher packet rate
- does not handle irregular conditions well
  - bursty packet rate: higher overhead during higher traffic load
  - irregular processor loading: translates to large variation in latency

Hybrid signature scheme
- generalization of off-line/on-line signatures
  - faster signature rate
  - substantially smaller signature size
- off-line/on-line signatures [EGM]
  - off-line computations
  - repeat:
    - generate a 1-time public-private key pair
    - certify 1-time public key with long term signature key (such as RSA)
    - store key-pair, certificate in buffer
  - on-line computation: for singing message m
    - sign m using 1-time private key from buffer
    - append certificate for corresponding 1-time public key to signature
  - 1-time key operates 10-20 X faster than regular signatures

Generalizing off-line/on-line signatures for improved speed
- off-line: thread 1
  - generate k-time key-pair (from k, 1-time key pairs)
  - certify k-time public key using long term signature key
  - store k-time key-pair, certificate in buffer
- on-line: thread 2, to sign message m
  - if current k-time key exhausted, pick fresh key from buffer
  - sign m with some i'th use (I=1,…,k) of current k-time key
  - append i'th time signature with certificate and authentication information for i'th use of k-time key

- observation: with increasing k, sustainable hybrid signature approaches 1-time key generation rate ~ 10-20 x regular signature rate

Advantages of approach
- easy to obtain 10X improvement over regular signature schemes
- negligible latency
- easily handles multiple flows
- fixed signature size
- handle irregular conditions very well: if reasonable buffer of keys is maintained by off-line thread then
  - gracefully handles bursty traffic
  - smooths over regular processor loading

Disadvantage
- large size of overhead makes scheme impractical
  - size of i'th signature from k-time key (largest overhead)
  - size of certificate for k-time public key
  - size of authentication information to identify i'th use of k-time key
- size overhead > 1 Kbyte/packet for reasonable speedup
- all three components must be reduced to make scheme practical

Primer on 1-time signatures
- for signing 160 bit values: (SHA-1 hashes)
- let f: 1-way, 80 bit -> 80 bit function
- private key: 160 pairs of 80-bit random numbers $<r_{0,0}, r_{1,0}>$, $<r_{0,1}, r_{1,1}>$, …, $<r_{0,160}, r_{1,160}>$
- public key $<f(r_{0,0}), f(r_{1,0})>$, $<f(r_{0,1}), f(r_{1,1})>$, …, $<f(r_{0,160}), f(r_{1,160})>$
- signature on m = m_0 m_1 …. m_{160}: $r_{m0,0}$, $r_{m1,1}$, …., $r_{m160,160}$
- size = 160 * 2 * 10 bytes
- better schemes exist, but overhead which is proportional to number of bits to sign still substantial
- tradeoff between size and speed (favors speed)

Reducing signature size overhead
- why sign a 160 bit hash of message instead of a 80 bit hash?
  - birthday attack finds collisions in $2^{|Hash|/2}$ steps
  - adversary creates two messages m1 and m2 such that Hash(m1) = Hash(m2)
  - m1 is favorable to signer, m2 is favorable to adversary
  - adversary asks signer to sign m1
  - later adversary claims m2 was signed

TCRS and Commitments
- target collision resistant functions (TCRs aka Universal One Way Hash Functions) [BR, NY]
- T(t,m): $(01,)^n$ X (0,1)* -> $(0,1)^n$, a function family indexed by n-bit key t on message m, is a TCR family if all bpounded adversaries have low probability of winning the following game:
  - adversary chooses m1
  - signer randomly chooses key t and outputs t, T(t,m1)
  - adversary tries to find an m2 s.t. T(t,m1) = T(t,m2)

- TCRs good enough for signatures; to sign m, signer picks t at random and signs <t,T(t,m)>
- no birthdays!! n could be around 80 :-)
- but must sign an 80 bit key & and 80 bit hash :-(

TCRs and Commitments
- constraints
  - must sign 80 bit key t and 80 bit T(t,m)
- could sign t using long term key (off-line) and T(t,m) using k-time key (on-line)
- BUT cannot disclose t to adversary before m is chosen for 80 bit TCRs :-(
- trick: use commitments; sign commitment to t in off-line process and sign T(t,m) and open commitment of t during on-line process
- “practical” commitments are fairly cheap, overhead of commitment << savings from smaller one-time signatures

Modified on-line/off-line scheme
- off-line
  - generate k-time, 80-bit key pair and k commitments to indices (keys) t1,…,tk of 80-bit TCR hash functions to be used for hashing k messages
  - use long term signature key to certify k-time key pair and commitments; store in buffer
- on-line: for message m
  - compute signature-I(TCR-Hash(ti,m)), open commitment to ti and append certificate for k-time key and authentication information for i’th use
- advantages: 2X reduction ion size, 2X improvement in speed; trade speed improvement for additional size savings!

Additional savings
- identifying and authenticating I’th use of a k-time key when k-time key constructured from k, 1-time keys
  - straightforward approach: use a hash tree
  - space saving approach use a TCR hash tree [BR]
- certificate size
  - use Bellare-Rogaway’s technique to embed messages into RSA/Rabin signatures; certificate consists of a single RSA/Rabin signature on a substantial amount of useful data

A 36-time scheme w/ 80-bit security parameter
- off-line:
  - create 36-time keys from 36, 1-time keys with cmmts
  - public key: roots of 6 TCR trees consisting of 6 public keys each; fits in one 1024 bit RSA block, certificate size = 128 bytes
- on-line:
  - signature size: 290 bytes
  - opening commitment: 10 bytes (implicitly other bytes are derived from 1-time signature)
  - identifying/authenticating i’th use: 50 bytes
- signature rate: 360 sigs/s sustainable on 266 MHz Pentium II, 25000 sigs/s burst rate!

FRAMEWORK AND BUILDING BLOCKS FOR SMUG

Eds: Thomas Hardjono, Ran Canetti, Mark Baugher, Pete Dinsmore

Status of draft
- follows Oslo meeting
- an attempt to capture rough consensus of SMuG
- will be revised following this meeting

The Building Block Approach
- partitions the complex problem of multicast security to “digestible” portions
- focuses the research towards standardizable protocols
- allows SMuG to work in parallel on various BBs
- provides a common framework and terminology for SMuG

The BBS are Functional
- each building block is described by its functionality, not by the algorithm or protocol (e.g., encryption, key mgmt)
- there can be different ways to instantiate each building block

A reference framework: the building blocks at a glance

(insert framework diagram here)

Components of a secure multicast protocol
group members: sender(s) and receivers
group controller and key server (GC+KS)
…

Why put [GC+KS] in one box?
- simplify overall design
- do not mandate separation for a standard
- avoid standardizing GC-2-KS protocols, at least at first
- simplify life for group members
- can always refine ,,,

Four building blocks
- data encryption building block
  - member to member (1-2-M or M-2-M)
- data and source auth building block
  - member to member (1-2-M or M-2-M)
- group control and key mgmt BB
  - GC+KS to members (also GC+KS-2-GC+KS)
- group policy mgmt BB
  - GC+KS to Policy Server(s). Also PS-2-PS)

DISCUSSION

General agreement that reasonable framework to start with!

Rohatgi question if want to consider data encryption translation servers/GWs. Disney example. Model assumes translation server would be a member of the group, i.e., an end.

Should we include in framework?

Recommend against and not allow.
Recommend against but allow.
Allow.

Canetti: Note that this puts trust in translators. Also translators (for instance ISPs) will have liability problems. In previous meetings Brian Whetten suggested that this does not match their business model.

Also, Don't see any security advantage in key translation.

DB: should allow!! simply acknowledge/reference and explain how framework apply, rather than show as explicit element in framework.

Key hierarchies?

So, go into more in depth discussion of Problem Area 1 --

SECURE MULTICAST DATA FLOW ISSUES, Pankaj Rohatgi

SMuG framework and building blocks

Sender(s) -> receiver(s) data flows
- full standardization required for interoperability, acceptance …
- what should these flows carry?
  - transformed multicast data traffic: raw data needs to be transformed to provide:
    - secrecy
    - access control
    - group authentication
    - source authentication
  - should control information (e.g., key updates) be multiplexed in?
    - pros: good for synchronization, …
    - cons: complexity, lack of modularity…
      - some schemes require different communication primitives for control data

Data transforms
- current consensus: three types of transforms
- encryption transformation: for secrecy / access control
  - based on block/stream ciphers, e.g., as in IPsec/SSL
- group authentication transformation
  - based on MACs, e.g., CBC-MAC or HMAC as in AH
- source authentication transformation
  - based on regular digital signatures, flow signatures, stream signatures, hybrid signatures, asymmetric MACs, etc.

Issues with data transforms

- should transforms be independent or should composite transforms be defined for common usage scenarios?
  - e.g., an ESP like composition transform for encryption/group-authentication
- should all transforms be applied at the same level in the networking stack, either application or network layer?
  - pros: consistency is cleaner, no conflict on granularity of group membership, i.e., either host or application
  - cons: need to re-invent the wheel, e.g., cannot re-use IPsec components
- issues with application level transforms
  - best suited when group membership granularity is at application level
  - faster to implement and deploy
  - less efficient, especially in multiple group members/host scenarios
  - requires modification to current multicast applications
- issues with network level transform
  - most suitable when group membership granularity is at host level
  - slower to implement and deploy since OS kernel changes are required
  - more efficient, especially in multiple group members/host scenarios
  - if done right, secure multicast should be transparent to current multicast applications
- interaction between transforms and reliability mechanisms
  - most multicast reliability mechanisms based on local recovery from packet loss
  - reliability mechanism must reside above net layer
  - encryption transform may interfere with local recovery, if local recovery agent os not group member; problem when encryption is performed at network layer
- in multi-transform scenarios order of transform application could be important
  - source authentication transform needs to be applied before any end or group-authentication transform
    - good cryptographic practice if non-repudiation required
    - encrypted data may get decrypted and re-encrypted by a different key at gateways between different domains, e.g., in Iolus or because of legal restrictions on permissible encryption schemes
  - to minimize impact of denial-of-service attacks, better to have group authentication transformation performed last, i.e., checked first

Comment from DB: Other reasons for translation: scale, crypto exposure, dynamism

Issues with encryption transforms

- well understood / deployed
- based on block/stream ciphers
- guiding principle:
  - try to re-use/mimic existing transforms
- should be able to work with hardware based conditional access modules

Issues with group authentication transforms

- techniques for group authentication well known/deployed
  - MACs such as 3DES-CBC-MAC or HMAC-SHA1 etc.
- should one reuse IPsec authentication transforms AH, or part of ESP directly or replicate their functionality at some other layer?

Issues with source authentication transforms

- still an active area of research
- currently known efficient techniques are complex
  - may require grouping/buffering of packets
  - may involve introduction of latency
  - may require complex software arch such as use of multiple thread/processes with issues of synchronization, timers, etc.
  - without further investigations, it may be too premature to introduce these into the networking layer in the OS kernel
- long term source authentication keys likely to be used by principals across different multicast flows and also for other applications
  - certification of keys is expensive, principals are likely to have very few authentication keys
  - secret authentication keys may be stored internally in a hardware device such a smart-card
  - design needs to allow for source authentication transformation to be done outside secure multicast software module
  - design needs to prevent out-of-context substitution attacks of data authenticated with long term authentication key
- source authentication transform may be the logical place to implement replay protection
- will a single transform even be enough?
  - schemes which combine an efficient and low security source authentication mechanism at a packet level with a less efficient but more secure authentication mechanism at a packet aggregate level

Canetti: Also question of whether above or below reliability?

Summary of data transform issues to be resolved by SMuG

- types of data transforms and their placement
  - consensus on three types of data transforms
  - no consensus on placement yet. Either follow rigid approach and fix placement of transforms or allow a flexible approach permitting transforms at both application and network level
  - rigid approach needs to be studied with respect to effects on reliability mechanisms, granularity of membership, etc.
  - flexible approach needs to be studied with regard to problems created by inconsistency
- format of all transforms
  - guiding principle: where possible re-use or borrow heavily from existing transforms, e.g., SSL/IPSEC
- agree upon a minimal set of mandatory transforms
- fix order of application transforms or allow for flexible approach
  - fixed order of source authentication followed by data encryption for group authentication makes sense
  - need to investigate security issues raised by allowing a flexible approach
- further investigate interaction between reliability mechanism and data transforms

LUNCH BREAK

DISCUSSION

Amit: question about line between GC+KS and receiver representing two different protocols.

Amit: clarify where policy is stored. Currently mentioned in 7.2.3 Key Management, but probably belongs in 7.2.4, Policy Mgmt.

Switch directions slightly. First look at 3 data transforms and possible placement at various layers, then come back to the document.

Without Reliable Multicast	Application	Transport	Network
Group authentication	x	x	x
Group confidentiality	x	x	x
Source authentication	x	x	x

With Reliable Multicast	Application	Transport	Network
Group authentication	x		x
Group confidentiality	x		x
Source authentication	x?	x	

Question of who does retransmission
- sender/source
- repair node

Carl: but why do we care? Big buffer issue. Sender auth maintained, then don't have to worry.

Ran: We should have two different source authentication solutions: One that assumes reliable transmission and one that doesn't.

Ran: possible transforms
1) network layer enc: ESP, KEK?, others?
2) network layer group auth: ESP, AH, KEK?, others?
3) network layer source auth: [WL], [R], [C+], sig, others?
4) appl. layer enc:
5) appl. layer group auth:

KEK scheme
- The session key is used only to encrypt an ephemeral key; the data is encrypted using the ephemeral key. In addition, the encryption of the ephemeral key appears at every packet.
- This gives a slight efficiency advantage since group key can be changed less frequently. The ephemeral key is changed more frequently. The recipient does not need to decrypt the ephemeral key for each packet. Instead, it decrypts the ephemeral key once whenever it is changed, and then caches this key until the next refreshment occurs.

Ran: mandate / recommend an order? GA[GC[SA[M]]]

DB: Need separate transforms? Why not GA with GC, e.g., enc w/ MDC

GA first to preclude DoS attacks. IPsec sec arch suggests authentication after encryption in ESP.

AAA, NNA, NNN, but not ANN (as demonstrated by Hugh - / \ /)

Policy/Group Controller discussions (Hardjono moderating)

Step 1. Group Policy
- data sensitivity/confidentiality
- access control rules (e.g., who can join)
- source authentication (or not) (non-repudiation or not)
- member data policy
- failure policy
- rekeying policy
- group authentication (or not)
- infrastructure
- compromise policy

Roles

Step 2. Create Group - GC create group parameters/state
- initialization
  - group key(s)
  - get certificate (e.g., CAs)
- join request from member (user auth)
- get key (secure channel)

List of member events:
- re-sync
- evict
- quit (leave)
- freeze / thaw
- (switch group address)
- join
- periodic rekey
- policy update

- group shutdown
- invalidate key (e.g., key lost or compromised?)

D A Y T W O

RMT AND SMUG
Allison Mankin

Co-chair Reliable Multicast Transport (RMT) WG in IETF

RMT and Secure Multicast - SMuG Meeting 24 Sep 99
- RFC 2357
  - Transport policies about whether can even publish reliable multicast
  - Could break Internet given lots of retransmission
  - Congestion control requirements; security requirements too!
- Building blocks and protocol instantiations
- Appeal to SMuG

RFC 2357 - Security Considerations for RM Protocols
- The analysis must document which of the various parties -- senders, routers (more generally, data forwarders), receivers, retransmission sources -- must be trusted in order to ensure secure operation and privacy of the transmitted data, to what degree, and why. (One issue to address here are "man-in-the-middle" attacks.)
- To what degree can data be manipulated so that at least a subset of the receivers receive different copies? Does the protocol allow a group of receivers to determine whether they all received the same data?
- What limitations are placed on the retransmission mechanism to prevent it from being abused to flood network links with excessive traffic? Which parties must be trusted to ensure this, and to what degree, and why? The presumption will be that either a congestion control mechanism will inherently limit the volume of retransmission traffic, and that this limiting influence is robust under concerted attack; or that retransmission requests will be signed in a cryptographically strong manner so that abuses of the mechanism can be traced back to their source. Protocols that do not provide either of these forms of protection face a great burden of proof that they don't threaten network stability.

(Some discussion regarding authentication of retransmission requests, etc.)

Building Blocks (BB) Planned
- NAK-Based Reliability
- FEC Repair
- Congestion Control
- Generic Router Support
- Common headers (probably part of above)
- Tree Configuration
- Security

Protocol Instantiations (PI Planned
- NAK-only

- tree-based ACK
- open look
- RMT will ensure that PI(s) meet needs for reliable key distribution
- each PI will have to meet the security requirement, mostly with analysis, but also with mechanisms from the Security BB

(Control message auth proving to be one of the most important security requirements for RMT.)

(Take action to work with RMT folks to address specific security concerns that may require tradition, non-multicast security solutions. Thomas to follow-up.)

Appeal to SMuG Members
- some volunteers to draft the Security BB
- Volunteers to draft/review security analyses for Protocol Instantiations
- Commentary/critique on RFC 2357 requirements - on mailing list - is the state of the art up to meeting it? Is the state of the art further along?

Contacts
- RMT mailing list and archives
  - send message to rmt-request@lbl.gov w/ Subject of archive help
- Chairs
  - Roger Kermode, Motorola, ark008@mail.mot.com
  - Lorenzo Vicisano, Cisco, lorenzo@cisco.com
  - Allison Mankin, USCISI, mankin@isi.edu

Points made in ensuing discussion:
- Authenticating control messages is a denial of service problem. Should be dealt with in the broader context of protecting IP multicast from denial of service attacks.
- Different RM protocols may have different susceptibility to denial of service via false ACKs/NAKs. Seems that FEC-based protocols are less susceptible.

NEXT MEETING AT IETF

One day on Sunday? Yes. At IETF hotel. Nov 7th.

BACK TO BB DISCUSSION

Come up w/ set of requirements for each one of the BBs/protocols, then give out homework assignments.

“Driving a car” analogy.

Will return to this, but Framework I-D discussion first.

DISCUSSION OF FRAMEWORK I-D

Quick overview of draft.

Comments due by EOB Monday, October 4th. New version to be prepared and posted by I-D cutoff date of October 22nd. Copy to be sent to list and/or posted on Web page.

Ran: Define two BBs between GC+KS and sender/receiver.

Amit: Sender doesn't necessarily send to multicast addresses; may send unicast to a server that multicasts.

So, is the server the sender? No, because the original sender must be part of the group to do the encryption.

Pete: change policy server to a box, rather than database. Call something other than "server". Call is policy "Manager" for now. Double-check Policy WG terminology for possible consistency.

Chose one of m-M or N-N and use consistently throughout.

Eliminate line in Sender and Receiver subsection regarding "In addition …"

Hugh: Section 5, bullets. Should these be **may** include …

Joe: Use the term data confidentiality instead of multicast confidentiality.

Hugh: multicast - oriented toward multicast communications protocol vs. group, which is a set of entities that may be communicating via multicast. Agreed to distinguish the two. We are addressing secure multicast.

Carl: really secure IP multicast.

DMB: Make sure BB discussion consistent with framework.

Ran: Plan to add group authentication as BB. We include data integrity with authentication.

DMB: Section 7.1, do we know for sure that our notion of BB is same as RMT? Ran: No, should not say this, just say what we mean.

Ran: Split GC+KS into two different BBs?

Carl: Yes, but one for group establishment, other for keys. Describe it in this document now!

Suggestion agreed upon by group.

Joe: BBs provide flexible not useful security services.

Amit: group security policy database discussion in Multicast Key Management subsection really belongs in Policy Management subsection.

Ran: clean up redundancy in Section 7.3.

DMB: Add references for policy work, including DCCM, GKMP RFCs 2093 and 2094, and Antigone.

Patrick: remove implication that only two types of policies

Atal/Thomas: move risks of BBs from RMT from page 13 to similar discussion page 8.

BACK TO BUILDING BLOCK DISCUSSION

Actions for Next meeting:

I. Data handling. Ran/Pankaj to pursue.

II. Group key management. Review state-of-the-art of the research as potentially directly applicable to GC+KS. Thomas/Hugh/Eric/Ran to pursue. Presentation at DC SMuG.

III. Need review of PWG work and potential value for us. Review of research in general and potential value. Roadmap way forward. Pete/Hugh/Patrick to prepare. Presentation at DC SMuG.

Need list of GC+KS (Problems Area 2) activities

Advertisement from GC+KS to members
- group name
- (group address
- time
- secure/NOT
- description
- first contact
- (multicast probe)
- ALL SIGNED

Joining the group (UNICAST):
- exchange identities
- exchange permissions (ACL + policies)
- policies (pushed to member)
  - member enforce

* Establish some group long-term security parameters

** Establish keying material
  - Secret group key
  - (SA, GA, GC, compromise material)
  - auxiliary keys
  - public keys for source authentication

(*) and (**) are separate: the first is membership management and the second is group key management

We agreed that implementations would probably use a (keyid, key) pair to handle synchronization of when you receive key material and when it is valid.

Rekey

three types

- cryptoperiod expired - key refresh
- key compromised, with no membership change
- membership change
- delete key (might be delete group)

rekey must be authenticated (by group controller?)

lots of discussion about whether or not group controller can send to multicast group. Some implementations ONLY have one sender, the data source.

will rekey always be multicast? probably not, but will be necessary to scale

what happens between two GC+KS's?

- Use any secure point to point communication mechanism.
- user-auth by one GC+KS is accepted by all GC+KS
- GC+KS agree/share keying material
- GC+KS agree who is main controller

Thomas: should the 3 subgroups assigned actions for next SMuG capture current discussions in their presentations?

Decision: Meet again for a full day on Sunday before IETF, Nov 7.

Meeting adjourned.