Secure Group Communication

Professor Patrick McDaniel
CSE545 - Advanced Network Security
Spring 2011
Group communication

• An implementation of *l-to-n* or *n-to-n* communication
  ‣ via multicast
  ‣ via point-to-point communication
  ‣ via hybrid solutions

• Typical implementation building blocks
  ‣ Full mesh
  ‣ Ad hoc networks
  ‣ Broadcast
Multicast

• Multicast is a unreliable \textit{n-to-n} transmission channel
  ‣ Native IP protocol, supported as an unreliable, best-effort service
  ‣ All class D addresses 224.0.0.0-239.255.255.255 are reserved for multicast
  ‣ Routing and address management is complex
    • Largely implemented by grafting to source root transmission trees
    • Not supported by current networks
    • Enormous efforts in the 1990s, found to be too hard in practice

• \textit{Anycast}: finds the “nearest” instance of an address
Multicast routing (simplified)
Multicast routing (simplified)
Multicast routing (simplified)
Applications of Group/MC

• It was thought that future network systems would require more and more multipoint communication:
  ‣ Conferencing (Skype, WebEx ...)
  ‣ Information streaming (audio, stock feeds, ...)
  ‣ Bulk downloads (patch distribution ...)
  ‣ Services (distributed filesystems, federated databases ...)

• *Debatable*: if available, could be enabling technology.
The technical challenge ...

- Creating and sustaining a *group session* is much more complicated than in a point to point session.
  - Membership
  - Authentication (data)
  - Key management
Security Properties

• Confidentiality
  ‣ Group or recipient

• Authenticity
  ‣ Group or sender (source)
  ‣ Non-repudiation

• Other concerns
  ‣ Contributory keying
  ‣ TTP availability
  ‣ Membership dynamics
  ‣ Anonymity
Security Properties

- Confidentiality
  - Group or recipient

Antigone: 6 group concerns

1. How do you authenticate users?
2. How do you introduce users to the group?
3. How do you rekey/distribute membership?
4. How do you pass messages?
5. How do you detect failed members?
6. How do you remove members from the group?

- Other concerns
  - Contributory keying
  - Anonymity

- Membership dynamics
Group Membership

• Who is in the group?
  ‣ Transient membership (who is in the group now?)
    • Joining, leaving
    • Each moment of the group is called a view
  ‣ Admission control (who do I let in?)
    • Who decides?
  ‣ Anonymity (how do I the membership from leaking?)
  ‣ Notification (do I and how do I communicate to the members the current membership?)
Group Operations

• The group will proceed via:
  ‣ INIT: start the group (often by group owner)
  ‣ JOIN: a new member joins the group
  ‣ LEAVE: a member leaves the group
  ‣ EJECT: a member is forcibly removed from the group
  ‣ REKEY: change the session keys
  ‣ SHUTDOWN: shutdown the group, dispose session state

• Note: each operation may require changes in session state
A universe of potential group members $U = \{u_1, u_2, \ldots u_n\}$
A group owner $U_g$
A view $v$ is a (successive) state of the group at some time, $v = (1 \ldots \infty)$

Group $G_v = (M_v, S_v)$, where:

- $M_v$ is a view of membership, e.g., $M_v \subset U$,
- $S_v$ is the session state under view $v$, e.g., the session keys

The group is defined as a series of membership operations:

- $INIT() : G_1 = (M_v = \{U_g\}, S_1 = init\_key())$
- $JOIN(u_i) : G_{v+1} = (M_{v+1} = M_v + u_i, S_{v+1} = join\_key(S_v))$
- $LEAVE(u_i) : G_{v+1} = (M_{v+1} = M_v - u_i, S_{v+1} = leave\_rekey(S_v))$
- $EJECT(u_i) : G_{v+1} = (M_{v+1} = M_v - u_i, S_{v+1} = eject\_rekey(S_v))$
- $REKEY() : G_{v+1} = (M_{v+1} = M_v, S_{v+1} = rekey(S_v))$
- $SHUTDOWN() : G_{v+1} = (\emptyset, \emptyset)$

Note that the security for the group is largely defined by the $init\_key$, $join\_key$, $leave\_rekey$, $eject\_rekey$, and $rekey$ functions.
• The Group Key Management Protocol was the first standardized secure group communication protocol
  ‣ Intended for multicast, applies generally to groups
  ‣ First to handle access control, key generation, and key distribution to the members of the group (old style: KDC)
  ‣ Intended to be lightweight, simple and fast

• Parties:
  ‣ Originator/group controller : group owner (source)
  ‣ Group Key Management (GKM): key distribution application
  ‣ Receivers: group members
GKMP (The specifics)

Keys:

- Group Traffic Encrypting Key (GTEK)
- Group Key Encrypting Key (GKEK)
- Session Traffic Encrypting Key (STEK), per member
- Session Key Encrypting Key (SKEK), per member

Packages:

- Group Key Packet (GKP) for view $v$: $GKP_v = E(GKEK_v, \{GTEK_v, GKEK_{v+1}\})$
- Session Key Package (SKP): $SKP = \{STEK, SKEK\}$
- Group Rekey Package (GRP): $GRP_v = \{E(SKEK, GKP_v), Sig(orig., E(SKEK, GKP))\}$

Keying the group:

1. The GKM creates the session keys via the GKP, $v = 1$.
2. For each user in the session, distribute SKP and then GRP.
3. Rekeying for view $v$ uses the GTEK obtained in the previous step using a single multicast message.
4. Member eject can alternately disband the group and start over.
Keys:
- Group Traffic Encryption Key (GT EK)
- Group Key Encrypting Key (GKEK)
- Session Traffic Encryption Key (ST EK)
- Session Key Encrypting Key (SKEK)

Packages:
- Group Key Packet (GKP): \( GKP_v = GKEK_v \cdot \{ GTEK_v, GKEK_{v+1} \} \)
- Session Key Package (SKP): \( SKP = \{ ST EK, SKEK \} \)
- Group Rekey Package (GRP): \( GRP_v = GKEK_v \cdot \{ GKEK_{v+1}, GKP_v, Sig_{orig.}, GKEK_{v+1} \} \)

Keying the group:
1. The GKM creates the session keys via the GKP,
   \( GKP_v = GKEK_v \cdot \{ GTEK_v, GKEK_{v+1} \} \)
2. For each user in the session, distribute SKP and then GRP.
3. Rekeying for view uses the GTEK obtained in the previous step using a single multicast message.
4. Member eject can alternately disband the group and start over.
Sender Authenticity

• How do you authenticate precisely which sender was the origin of a message?
  ‣ Solution 1: digitally sign the data
    • Problem: too expensive
  ‣ Solution 2: Use stream signatures
    • Problem: delays reception
Cannetti et. al solution

1. Each user $u$ is assigned a subset of keys $R_u = \{R_u^1, R_u^2, \ldots, R_u^\gamma\}$ from a keyset $R$, $|R_u| = \gamma$.

2. A sender $s$ attempts to send message $M$ to the group, he computes $MAC(M, R_s^1), MAC(M, R_s^2), \ldots, MAC(M, R_s^\gamma)$ and sends it with the message.

3. Each recipient receives the message and validates each $MAC$ for each $R_s^i \in R_u$. If all validates, then the sender is authenticated.

- **Note 1**: this works only where each user is provided a unique set of keys (that’s the insight)
- **Note 2**: It is sufficient to send only a subset of the bits per MAC
- **Note 3**: gamma places the bounds on the collusion group before the system collapses
Logical Key Hierarchy (LKH)

- **Problem**: there is a lot of overhead in distributing keys to $n$ members individually
- **Idea**: could we somehow assign multiple keys in such a way as to allow us to efficiently distribute keys while still preserving session security requirements
- **Keys are organized in a (logical) hierarchical tree**
  - Group key is located at the root
  - Key encryption keys are the non-root, non-leaf nodes
  - Each member corresponds to one leaf node (per member KEK)
- **Updates the group key and the key encryption key by means of the encryption of internal key-nodes**
- **Rekey with only $O(\log_2(n))$ messages**
LHK Approach

\[ K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_8 \]
LHK Approach
LHK Approach

Each member is given the keys from itself to the root of the tree
LHK Approach

Each member is given the keys from itself to the root of the tree.

Group key used to secure communication.
LHK Leave/Eject

Replace all the keys held by the leaving/ejected member by encrypting them with their child node keys

rekey message:

$E(K_4, K_{34}')$, $E(K_4, K_{14}')$, $E(K_4, K_{18}')$

$E(K_{12}, K_{14}')$, $E(K_{12}, K_{18}')$

$E(K_{58}, K_{18}')$
Contributory keying

• It is often desirable to contribute the keying process to ensure safe operation, e.g., not using old keys, not using member as oracle
  ‣ Same is true of point to point communication

• N-way Diffie-Hellman is a way to achieve this with
N-way Diffie-Hellman

Diffie-Hellman can be extended to 3 parties to share a session key. Assume you have a group whose membership $M = \{m_1, m_2, m_3\}$ a large group prime $n$ and $g$ which is primitive mod $n$. The parties perform the following algorithm:

1. Each member $i$ of the group selects a large integer $\phi_i$ and computes:
   \[ X_i = g^{\phi_i} \mod n \]

2. $m_1$ sends $X_1$ to $m_2$, $m_2$ sends $X_2$ to $m_3$ and $m_3$ sends $X_3$ to $m_1$.

3. $m_1$ computes:
   \[ Z_1 = X_3^{X_1} \mod n \]
   and sends it to $m_2$.

4. $m_2$ computes:
   \[ Z_2 = X_1^{X_2} \mod n \]
   and sends it to $m_3$.

5. $m_3$ computes:
   \begin{align*}
   1. & \quad m_1 \text{ computes } k = Z_3^{X_1} \mod n = g^{\phi_1\phi_2\phi_3} \mod n \\
   2. & \quad m_2 \text{ computes } k = Z_1^{X_2} \mod n = g^{\phi_1\phi_2\phi_3} \mod n \\
   3. & \quad m_3 \text{ computes } k = Z_2^{X_3} \mod n = g^{\phi_1\phi_2\phi_3} \mod n
   \end{align*}
   and sends it to $m_1$. 

• Problem: \textit{l-effects-n} failure - a single group change (leave) can effect every member of the group
  ‣ For large groups, could be thousands of such changes per minute
  ‣ For example, thing about a PPV multicast TV channel

• Rather than changing the keying (as in LKH), could you change the structure of the group?
  ‣ Subgrouping - break down the groups into smaller units that can localize change
  ‣ Group is defined by a hierarchy of subgroups with a “security agent” leader
  ‣ These leaders mediate traffic between nodes
Setup

- Group security agents (GSA)
  - Group security controllers (GSC) -- group owners
  - Group security intermediaries (GSI) -- subgroup proxies, are members of their “parent” groups
• Each member $m_i$ of a subgroup $G_i$ is has a long term secret that is used to deliver a member specific group secret $K_{GSA-m_i}$.

• Each subgroup $G_i$ uses a subgroup session key $K_{GRP_i}$ to secure communication.

• On joins, a new $K'_{GRP_i}$ is created, encrypted, and multicast to the subgroup as $E(K_{GRP_i}, K'_{GRP_i})$.

• On leaves, a new $K'_{GRP_i}$ is created, encrypted, and multicast to the subgroup as $E(K_{GSA-m_1}, K'_{GRP_i}), E(K_{GSA-m_2}, K'_{GRP_i}), \ldots, E(K_{GSA-m_k}, K'_{GRP_i})$.

• For transmission, the sender of a message $d$ creates a one-time key $K_{ot}$, and encrypts $d$ with it, then encrypts $K_{ot}$ with the local subgroup key before sending it out: $E(K_{ot}, d), E(K_{GRP_i}, K_{ot})$

• A message passing between two subgroups is “translated” by the GSI by simply un-encrypting the one-time key with the source subgroup key and re-encrypting it with the key of the destination subgroup, e.g.,

$$E(K_{ot}, d), E(K_{GRP_i}, K_{ot}) \rightarrow E(K_{ot}, d), E(K_{GRP_j}, K_{ot})$$
Secure Group Communication

• In the end, the killer applications for group communication never emerged directly
  ‣ Routing broadcasts (multicast) was too complicated, lacked scalability, and not really needed (see Netflix)

• Results
  ‣ Much of the keying, translation, and other tricks were applicable to a broad range of later challenges
    • Sensor networks
    • Ad Hoc networks
    • P2P systems
    • Agent based systems
    • Web caching

• Prediction: multipoint communication not here yet
Mid-term

• In class: 4:15-5:30
  • 14 - 3 point short answer (42 pts)
  • 4 - 7 point long answer (28 pts)
  • 3 - 10 point word problems (30 pts)
• Bug finder winner: Damien (+10 pts)
Mid-term review

• Introduction

• Communication Security
  ‣ SSH
  ‣ DH
  ‣ SSL

• Routing Security
  ‣ Types of routing, how they work
  ‣ Attacks against routing
  ‣ BGP security solutions
    • sBGP, soBGP, IRV, etc.
• Secure group communication
  ‣ Models, operation, guarantees
  ‣ GKMP, LKH, ...

• Other things
  ‣ Papers we read, things passed out in class
  ‣ Discussions, opinions, etc.
  ‣ Projects