Secure Reliable Group Key Management

Yang Richard Yang

11/14/2001

Review: DDoS

- Prevention
  - ingress filtering
  - router filtering [Park et al. 2001]

- Detection
  - end-to-end signaling
    - ICMP: out-band [Bellovin 2000]
    - Marking: in-band [Savage et al. 2000]
  - auditing
    - Source Path Isolation Engine (SPIE) [Snoeren et al. 2001]
**Example: Multiple Attackers**

![Diagram showing an example of multiple attackers and an attack path leading to a victim.]

**Advanced Marking Scheme (AMS) I**

[Song et al. 2001]

- **Main idea:** assume that the victim knows the map of its upstream routers
  - relatively easy to obtain in practice, although incurs a high overhead
- **Use two 11-bit hash functions** (to distinguish the order of the two routers) \( h \) and \( h' \) to hash the endpoints addresses and XOR them 
  \[ h(R_{start}) \oplus h'(R_{end}) \]
- **Use the other 5 bits** to encode the distance from edge to the victim
Example: Multiple Attackers

Advancing Marking Scheme II

- Use two sets of hash functions instead of two hash functions
- Use $w$ bits to encode the hash function and $11 - w$ to encode its value
- Intuition
  - probability for collision in AMS I is $1/2^{11}$
  - probability for collision in AMS II is $1/2^{(11-w)}$, where $s = 2^w$ because the victim will need to match the values of $s$ hash functions (the implementation uses a threshold approach)
- Choose $w = 3$
- Lesson: always question my statement
Group Key Management System

- A class of applications is based on a group communications model
- The function of a group key management system is to provide access control to the shared symmetric group key
- Two types of access control:
  - backward access control: change the group key after a new user joins the group
  - forward access control: change the group key after a current user leaves the group

Key Trees

[Wallner et al. Internet Draft, Wong et al. SIGCOMM '98]
New User u9 Joins the Group

User u9 Leaves the Group
A Lower Bound on Rekey Encoding

Summary of the Lower Bound

- \( \Omega(\log(n)) \) amortized per request lower bound if forward access control is required, regardless of whether backward access control is required.
- Snowink, et al. proved a similar bound independently at about the same time. This proof is much shorter 😊
System Model

- There is only one key server
- Key server implements forward access control
- After the i-th operation, all users in the group share a common group key $g_i$
- When updating keys, the key server uses one key $k$ to encrypt another key $k'$
- The adversary has access to all past communications. However, it can access one key $k$ only if it has received $k$ from the key server, or $k$ is encrypted by a key $k'$ that it has access, etc.

Rekey Encryption Graphs

- A model to capture all communications
- Rekey encryption graphs
  - A sequence of directed graphs $G_i$
  - $G_i$ captures the communication cost of rekeying the first $i$ requests
Nodes and Edges of \( G_i \)

- **Nodes**
  - individual key nodes
  - other key nodes

- **Edges**
  - A key node \( k \) to key node \( k' \) if the key server has sent \( k' \) by encrypting it using \( k \)

- **Properties**
  \[ G_i \subseteq G_{i+1} \]

Subgraph \( S_i \) of Graph \( G_i \)

- User node \( u \) when \( u \) is in the group (joined and not left)
- The group key \( g_i \)
- Any node or edge that is on a path from an individual key node in \( S_i \) to \( g_i \)
- Reduce \( S_i \) to a tree
- \( S_i \subseteq G_i \subseteq G_{i+1} \)
Properties of Subgraph $S_i$

- Denote $N$ as the number of individual key nodes in $S_i$
- Denote $i(x)$ as the in-degree of node $x$
- Define cost of user node $u$ as
  \[ c(u) = \sum_{x \text{ is on } u\text{'s path to group key}} i(x) \]
- Define $C = \max_{u \text{ is a user node}} c(u)$

Properties of $S_i$ (cont’)

- Define $n(x)$ as the number of individual key nodes reachable to $x$
  \[ s(N) = \sum_{\text{all user } u} c(u) = \sum_{\text{all internal node } y} i(y) n(y) \]
- We can prove (using induction, and because $x\ln(x)$ is convex) that
  \[ s(N) \geq N\ln N \]
- Therefore,
  \[ C \geq \ln N \]
Wasted Communication
After a User leaves

- Suppose user $u$ left and not re-join.
- Because of forward access control, $c(u)$ is the wasted communication cost: in $S_i$, not in $S_j$, for any $j > i$.

Construction of Lower Bound Requests

- First $n$ joins, followed by $n$ leaves.
- If the departed user has the highest cost, the $i$-th leave request gives $\ln(n+1-i)$ edges.
- Therefore, the number of edges in $G_{2n}$ is greater than or equal to:

$$\ln(n) + \ln(n-1) + \ldots + \ln(2) = \Theta(n \ln(n))$$

- Therefore, the per request communication cost is $\Omega(\ln(n))$.
System Components

Distributed Registrars Protocol
Rekey Encoding Component

- The encoding component can process user requests either individually or in a batch.
- Periodic batch encoding has the following benefits:
  - Batch rekeying improves efficiency.
  - Periodic rekeying can reduce out-of-sync problems.
  - Inter-dependency among rekey messages.
  - Inter-dependency between data messages and rekey messages.
  - For periodic rekeying, a user knows when the group key will be changed.
- Why do we emphasize periodic rekeying?

Periodic Batch Encoding Algorithms

- Objectives
  - Reduce the number of encrypted keys.
  - Maintain the balance of the updated key tree.
  - Make it easy for a user to identify its location and keys.
- Basic ideas of the batch algorithms
  - Always replace departed users with new users.
  - If J > L, how to add the remaining J-L new users?
**Key ID**

![Key ID diagram]

**Insertion**

- **Key server:**
  - maintain $n_k$, the ID of the largest valid key node
  - replace empty nodes from $n_k+1$ to $dn_k+d$ with new users
  - if not enough to insert new users, starting from $n_k+1$, split a u-node, put the split u-node as the left-most child, and $n_k++$
  - put $n_k$ in the rekey message
- **For a user with an ID $n$:**
  - while ($n$ is not in the range from $n_k+1$ to $dn_k+d$)
    - $n = dn+1$
Insertion Example:
Add 5 new users

Insertion Example:
First Use the Empty Nodes
Insertion Example:

Now split $n_k+1$

\[ n_k=4 \]

\[
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
5 & 6 & 7 & 8 \\
9 & 10 & 11 & 12 \\
13 & 14 & 15 & \\
\end{array}
\]

old user with 4

new user1 new user2 new user3 new user4

Batch Encoding Performance
Batch Encoding Performance Gains

Rekey Transport Component

- **Function:** encapsulate encrypted keys into packets, and provide reliable rekey transport

- **Issues:**
  - what is the workload?
  - what is the transport protocol? how to analyze its performance?
Key Assignment: Key Tree? Rekey Subtree

Key Assignment: Key Tree? Rekey Subtree

(changed from k1-9)

(changed from k123)

(changed from k789)

leave
Key Assignment Algorithms

- Three algorithms
  - BFA: large average, small variance
  - DFA: small average, large variance
  - R-BFA: small average, small variance

What is the assumption of the three algorithms?
**Average and Standard Deviation of the Number of Packets**

N=2048, L=1000, d=4, #encs/pkt=25

![Graph showing average and standard deviation](image)

**Total Number of Packets in a Rekey Message**

N=2048, L=1000, d=4, #encs/pkt=25

![3D graph](image)
Rekey Multicast Transport Protocol

- Two requirements:
  - soft real-time: each rekey message should be delivered with high probability before next rekey interval
  - eventual reliability
- Protocol summary:
  - proactive adaptive FEC reliable multicast
    - with \( n \) original packets, the key server generates \( k \) repair packets and sends \( n+k \) packets; with any \( n \) out of the \( n+k \) packets, a receiver can recover the original \( n \) packets
  - reliable unicast re-synchronization [proposed by Wong et al.]
- Performance analysis:
  - a rekey multicast protocol with sparse workload can be converted to a conventional reliable multicast protocol

Effects of the Proactivity Factor

![Graph showing the effects of the proactivity factor on bandwidth overhead and transport latency.](image)
How to Adapt the Proactivity Factor?

- Depends on target:
  - set the target number of NACKs $N^*$

- Receiver
  - if receiver $i$ needs $r_i$ more packets to have $n$ packets, reports $r_i$ to the key server

- Key server
  - send $k+n$ packets
  - Let $N$ denote the number of NACKs received
  - if $(N > N^*)$
    - sort the $r_i$ in increasing order \{$r_1, r_2, r_3, \ldots, r_{N-N^*}, \ldots, r_N$\}
    - $k += r_{N-N^*}$
  - How to reduce $k$?

- Is this adjustment policy stable? How to make it stable?

Overall System Performance

- Need user membership dynamics to generate workload
- Evaluations assume that the time each user stays in the group is exponentially distributed
Bandwidth Requirement vs. Rekey Interval

mean_time=180 sec, packet_size=25*20+57, J=L

rekey interval T (s)

bandwidth (kbps)

N=512
N=1024
N=2048
N=4096