Attacking Multicast Group
Key Management Protocols

Graham Steel and Alan Bundy
Multicast Key Management Protocols

**Aim:** To maintain a secure key for multicast within a group as agents join and leave

- Analysis of these protocols is challenging:
  - Modelling the protocols, posing security conjectures, searching in the model created

- Aims of this talk:
  - Demonstrate efficacy of CORAL approach
  - Describe what modifications other tools would need to tackle these protocols
CORAL

Refutes incorrect inductive conjectures
Uses a method borrowing theory from ‘Proof by Consistency’
- a refutation complete method for proving inductive theorems
First-order version of Paulson model

By refuting a security property $\forall \text{trace}. P(\text{trace})$, we obtain the attack as the instantiation of $\text{trace}$

Tested on several known attacks (from Clark-Jacob corpus)
New attacks on Asokan–Ginzboorg
Example - Tagdhiri Jackson

Originally proposed by Tanaka + Sato. T+J found flaws using Alloy + SAT checker, proposed improved protocol.

Flaw due to retention of old keys

However, their model did not include an active attacker!

CORAL used to model + attack the improved version
### Tanaka-Sato/Taghdiri-Jackson

<table>
<thead>
<tr>
<th><strong>Join:</strong></th>
<th><strong>Send:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. $M_i \rightarrow S : { \text{join} }<em>{K</em>{M_i}}$</td>
<td>1. $M_i \rightarrow S : { \text{send}(n) }<em>{I</em>{K_{M_i}}}$</td>
</tr>
<tr>
<td>2. $S \rightarrow M_i : { I_{K_{M_i}}, G_k(n) }<em>{K</em>{M_i}}$</td>
<td>2. $S \rightarrow M_i : { n', G_k(n') }<em>{I</em>{K_{M_i}}}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Leave:</strong></th>
<th><strong>Receive:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. $M_i \rightarrow S : { \text{leave} }<em>{I</em>{K_{M_i}}}$</td>
<td>1. $M_j \rightarrow S : { \text{read}(n) }<em>{I</em>{K_{M_j}}}$</td>
</tr>
<tr>
<td>2. $S \rightarrow M_i : { \text{ack. leave} }<em>{I</em>{K_{M_i}}}$</td>
<td>2. $S \rightarrow M_j : { G_k(n') }<em>{I</em>{K_{M_j}}}$</td>
</tr>
</tbody>
</table>

(and generate new key)
Modelling the Protocol

- Want to keep model general wrt no. of agents, scenario...
  - CORAL’s inductive model ideal for this
- Importance of knowing who is in the group at all times
  - Stored in trace
- Lots of fresh material needed
  - Use of counter, heuristic
Security Properties

Pereira–Quisquater properties unsuitable

Need *multicast group authenticity*

Throughout the evolution of the group, non-members should not be accepted as group members – whether sending or receiving.

Must make concrete conjectures in terms of trace.

Difficult without allowing ‘transient security breach’ to count as an attack.
Example

\[
m(\text{cons}(\text{sent}(Mj, \text{all}, \text{encr}(\text{hello}(Y), Gk), X\text{group}), \text{cons}(\text{sent}(X, Mj, \text{encr}(\text{pair}(Gk, \text{send}(Sq2)), Ikey), X\text{group}), \text{cons}(\text{sent}(Mj, \text{server}, \text{encr}(\text{send}(Sq2), Ikey), X\text{group}), Trace))), \text{Group}, \text{Keyseq}, \text{Tick}) = true \land \text{eqagent}(Mj, spy) = false \land \text{in}(Gk, \text{analz}(Trace)) = true \land \text{ingroup}(\text{triple}(\text{principal}(spy), X3, X2), X\text{group}, \text{Newgp}) = false
\]
Attack on Taghdiri Jackson

5. spy → server : \{ send(1) \}_{ik(spy)}

6. server → spy : \{ Gk(2), send(1) \}_{ik(spy)}

7. a → server : \{ send(2) \}_{ik(a)}

8. server → a : \{ Gk(2), send(2) \}_{ik(a)}

9. a → all : \{ hello(9) \}_{Gk(2)}

10. spy → server : \{ leave \}_{ik(spy)}

11. server → spy : \{ ackleave \}_{ik(spy)}

12. a → server : \{ send(2) \}_{ik(a)}

13. spy → a : \{ Gk(2), send(2) \}_{ik(a)}

14. a → all : \{ hello(14) \}_{Gk(2)}
Iolus

Join:

1. $M_i \rightarrow S : \{ \text{join} \}_{K_{Mi}}$
2. $S \rightarrow M_i : \{ \text{Ik}_{Mi}, Gk(n) \}_{K_{Mi}}$

Send:

1. $M_i \rightarrow \text{ALL} : \{ \text{message} \}_{Gk(n)}$

Leave:

1. $M_i \rightarrow S : \{ \text{leave} \}_{Ik_{Mi}}$
2. $S \rightarrow \text{ALL} : \{ \{ Gk_{n'} \}_{Ik_{M_j}} \ldots \} \forall j \neq i, M_j \in \text{group}$
Modelling Iolus

- For a general model, need lists for key update
  
  Needed this before for Asokan–Ginzboorg
  
  Straightforward in CORAL

- Control conditions become non-trivial
  
  Must work out what the key update message is
  
  Use recursive auxiliary function (as for A-G)

- No separate send/receive protocols
  
  Makes posing conjectures easier
Attack on Iolus

9. server → s(a) : \{ ik(11), Gk(11) \}_{longtermK(s(a))}
10. a → server : \{ leave \}_{ik(2)}
11. server → all : [\{ Gk(14) \}_{ik(11)}, \{ Gk(14) \}_{ik(5)}]
12. spy → server : \{ leave \}_{ik(5)}
13. server → all : [\{ Gk(26) \}_{ik(11)}]
14. spy → all : [\{ Gk(14) \}_{ik(11)}, \{ Gk(14) \}_{ik(5)}]
Summary

Strengths

- Natural, general model in inductive formalism
- Could pose novel security properties
- Found 3 new attacks on 2 protocols

Weaknesses

- Slow - up to 3 hours
- Posing conjectures tricky
  though easier second time, and not just CORAL
What Was Required

- Arbitrary number of agents
- Lists
- Auxiliary functions
- Conjectures involving temporal properties
Further Work

- More group protocols, with Diffie-Hellman operations
- API attacks - Bond–Clulow

More details

http://homepages.inf.ed.ac.uk/s9808756/coral