
Attacking Multicast Group Key Management Protocols

Graham Steel and Alan Bundy



 School of
informatics

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 $\forall trace.P(trace)$, we obtain the
attack as the instantiation of *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

Join:

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

Send:

1. $M_i \rightarrow S : \{ \text{send}(n) \}_{K_{M_i}}$
2. $S \rightarrow M_i : \{ n', Gk(n') \}_{K_{M_i}}$

Leave:

1. $M_i \rightarrow S : \{ \text{leave} \}_{K_{M_i}}$
2. $S \rightarrow M_i : \{ \text{ack.leave} \}_{K_{M_i}}$
(and generate new key)

Receive:

1. $M_j \rightarrow S : \{ \text{read}(n) \}_{K_{M_j}}$
2. $S \rightarrow M_j : \{ Gk(n') \}_{K_{M_j}}$

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

$m(\text{cons}(\text{sent}(M_j, \text{all}, \text{encr}(\text{hello}(Y), G_k), X_{\text{group}}),$
 $\text{cons}(\text{sent}(X, M_j, \text{encr}(\text{pair}(G_k, \text{send}(S_{q2})), I_{\text{key}}), X_{\text{group}}),$
 $\text{cons}(\text{sent}(M_j, \text{server}, \text{encr}(\text{send}(S_{q2}), I_{\text{key}}), X_{\text{group}}),$
 $\text{Trace}))), \text{Group}, \text{Keyseq}, \text{Tick}) = \text{true} \wedge$
 $\text{eqagent}(M_j, \text{spy}) = \text{false} \wedge$
 $\text{in}(G_k, \text{anzl}(\text{Trace}) = \text{true} \wedge$
 $\text{ingroup}(\text{triple}(\text{principal}(\text{spy}), X_3, X_2), X_{\text{group}}, \text{Newgp}) = \text{false}$
 \rightarrow

Attack on Taghdiri Jackson

5. spy \rightarrow server : $\{ send(1) \}_{ik(spy)}$
6. server \rightarrow spy : $\{ Gk(2), send(1) \}_{ik(spy)}$
7. a \rightarrow server : $\{ send(2) \}_{ik(a)}$
8. server \rightarrow a : $\{ Gk(2), send(2) \}_{ik(a)}$
9. a \rightarrow all : $\{ hello(9) \}_{Gk(2)}$
10. spy \rightarrow server : $\{ leave \}_{ik(spy)}$
11. server \rightarrow spy : $\{ ackleave \}_{ik(spy)}$
12. a \rightarrow server : $\{ send(2) \}_{ik(a)}$
13. spy \rightarrow a : $\{ Gk(2), send(2) \}_{ik(a)}$
14. a \rightarrow all : $\{ hello(14) \}_{Gk(2)}$

Iolus

Join:

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

Send:

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

Leave:

1. $M_i \rightarrow S : \{ \text{leave} \}_{K_{M_i}}$
2. $S \rightarrow \text{ALL} : [\{ Gk_{n'} \}_{K_{M_j}} \dots] \forall j \neq i, M_j \in \text{group}$

Modelling lolus

- 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 lolus

- 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>
