Analysis of Verification Tools for Security Protocols

Sergey Reznick, Igor Kotenko

Computer Security Research Group
St. Petersburg Institute for Informatics and Automation of Russian Academy of Sciences

Agenda

• Example of the flaw in protocol design
• Requirements for protocol verification tools
• Approaches to protocol verification
• Model checking based protocol verification
• AVISPA
• Isabelle
• Other theorem proofing tools
• Conclusion
Needham-Schroeder Public Key Protocol

- Participants: Alice, Bob, Server
- Target of participants: mutually exchange nonces generated by Alice and Bob
- Security goals:
  - confidentiality
  - authenticity

NSPK specification

1. A -> S : A,B
2. S -> A : {KPb, B}KSs
3. A -> B : {Na, A}KPb
4. B -> S : B,A
5. S -> B : {KPa, A}KSs
6. B -> A : {Na, Nb}KPa
7. A -> B : {Nb}KPb
3. A -> B : \{Na, A\}KPa
   i.3. A -> I : \{Na,A\}KPi
      ii.3. I(A) -> B : \{Na,A\}KPa
6. B -> A : \{Na, Nb\}KPa
   ii.6. B -> I(A) : \{Na,Nb\}KPa
   i.6. I -> A : \{Na,Nb\}KPa
7. A -> B : \{Nb\}KPa
   i.7. A -> I : \{Nb\}KPi
      ii.7. I(A) -> B : \{Nb\}KPa

---

Formal analysis is necessary

- NSPK announcement year: 1978
- NSPK flaw finding year: 1995
- Conclusion: any security protocol must be formally verified
Agenda

- Example of the flaw in protocol design
- Requirements for protocol verification tools
- Approaches to protocol verification
- Model checking based protocol verification
- AVISPA
- Isabelle
- Other theorem proving tools
- Conclusion

Verification tool selection

- A lot of tools available
- «Best» one is needed
Criteria for verification tools

- Automation
- Model simplicity
- Flexibility
- Attack trace extraction
- Community

No silver bullet

- Trade-off between different criteria
- Combination of tools
Example of the flaw in protocol design
Requirements for protocol verification tools
Approaches to protocol verification
Model checking based protocol verification
AVISPA
Isabelle
Other theorem proving tools
Conclusion

Approaches to verification

Model checking
- Murphi
- Other tools: SPIN, PRISM, Casper etc.

Theorem proving
- AVISPA (Constraints logic, lazy calculation, SAT, term rewriting)
- Isabelle (manual theorem proofs in the first order logic)
- Other tools: CAPSL, ProVerif etc.
Example of the flaw in protocol design
- Requirements for protocol verification tools
- Approaches to protocol verification
- Model checking based protocol verification
- AVISPA
- Isabelle
- Other theorem proving tools
- Conclusion

Murphi
- General purpose model checking tool
- Finite state machine + temporal logic condition
- Methodology:
  - Formulate the protocol as behaviour of the participants
  - Create model of the intruder
  - Formulate correctness condition
  - Simulate/verify the protocol
Fragment of the Murphi specification for NSPK

```plaintext
-> var outM: Message;   -- outgoing message
begin
  undefine outM;
  outM.source := i;  outM.dest := j;
  outM.key := j;
  outM.mType := M_NonceAddress;
  outM.nonce1 := i;  outM.nonce2 := i;
  multisetadd (outM,net);
  ini[i].state := I_WAIT;  ini[i].responder := j;
end;
```

Murphi invariants for NSPK

```plaintext
invariant "responder correctly authenticated"
forall i: InitiatorId do
  ini[i].state = I_COMMIT &
  ismember(ini[i].responder, ResponderId)
  ->
  res[ini[i].responder].initiator = i &
  ( res[ini[i].responder].state = R_WAIT |
    res[ini[i].responder].state = R_COMMIT )
end;
```
Murphi score

- Automation: yes
- Model simplicity: no
- Flexibility: very low
- Attack trace extraction: yes
- Community: no

Other model checking tools

- SPIN: bigger community
- PRISM: probability support
- Casper: CSP based, better notation
Main drawback of model checking approach

- States enumeration causing combinatorial explosion

Agenda

- Example of the flaw in protocol design
- Requirements for protocol verification tools
- Approaches to protocol verification
- Model checking based protocol verification
- AVISPA
- Isabelle
- Other theorem proving tools
- Conclusion
AVISPA

- HLPSL: High-Level Protocol Specification Language
- IF: Intermediate Format
- HLPSL2IF: converter
- 4 backends:
  - OFMC
  - CL-AtSE
  - SATMC
  - TA4SP
- SPAN: visual tool

OFMC

- Tree representation:
  - Root is initial state
  - Node's children are states to which the system can transfer for 1 transition
  - Some nodes are attack states
- Tree is infinite in both width and depth
- Tree is formalized as datatype in the lazy programming language
- Benefit: fast answer
CL-AtSe

- Models each protocol step by constraints on the intruder’s list of knowledges
- A protocol step is executed by adding new constraints to the system and reduce/eliminate other constraints accordingly.
- At each step the system state is tested against the provided set of security properties.
- Benefit: fast answer

SATMC

- Fully automatic translation from security protocol specifications into propositional logic
- Combines a reduction of protocol insecurity problems to planning problems and well-known SAT-reduction techniques developed for planning
- Drawback: hang-ups
**TA4SP**

- Approach is based on rewriting on regular tree languages
  - A0 represents the initial configuration of the network
  - a term rewriting R is applied until reaching a possible stabilization of the process \( L(A_n) = L(A_{n+1}) \)
    - “I → r” represents a protocol step.
- Benefit: approximation to infinite number of sessions
- Drawback: hang-ups

---

**SPAN**

- GUI tool to model protocol
- Intended behaviour mode:
- Behaviour with intruder mode
role
alice (A, B: agent,
    Ka, Kb: public_key,
    SND, RCV: channel (dy))
played_by A def=

    local State : nat,
    Na, Nb: text

init State := 0

transition
0. State = 0 \land RCV(start) =|>
    State':= 2 \land Na' := new() \land SND({Na'.A}_Kb)
    \land secret(Na',na,{A,B})
    \land witness(A,B,bob_alice_na,Na')
2. State = 2 \land RCV({Na.Nb'}_Ka) =|>
    State':= 4 \land SND({Nb'}_Kb)
    \land request(A,B,alice_bob_nb,Nb')

end role
NSPK modelled via HLPSL (3/7)

role bob(A, B: agent,
    Ka, Kb: public_key,
    SND, RCV: channel (dy))
played_by B def=

local State : nat,
    Na, Nb: text

init State := 1

NSPK modelled via HLPSL (4/7)

transition
    1. State = 1 \land RCV({Na'.A}_Kb) =|>
       State':= 3 \land Nb' := new() \land
       SND({Na'.Nb'}_Ka)
       \land secret(Nb',nb,{A,B})
       \land witness(B,A,alice_bob_nb,Nb')

    3. State = 3 \land RCV({Nb}_Kb) =|>
       State':= 5 \land request(B,A,bob_alice_na,Na)

end role
role session(A, B: agent, Ka, Kb: public_key)
def=
local SA, RA, SB, RB: channel (dy)
composition
    alice(A,B,Ka,Kb,SA,RA) \land bob
                  (A,B,Ka,Kb,SB,RB)
end role

NSPK modelled via HLPSL (5/7)

role environment() def=
    const a, b : agent,
    ka, kb, ki : public_key,
    na, nb,
    alice_bob_nb,
    bob_alice_na : protocol_id
intruder_knowledge = \{a, b, ka, kb, ki, inv(ki)\}
composition
    session(a,b,ka,kb) \land session(a,i,ka,ki) \land
    session(i,b,ki,kb)
end role

NSPK modelled via HLPSL (6/7)
NSPK modelled via HLPSL (7/7)
goal
  secrecy_of na, nb
authentication_on alice_bob_nb
authentication_on bob_alice_na
end goal
environment()

AVISPA score

- Automation: yes
- Model simplicity: yes
- Flexibility: not perfect
- Attack trace extraction: yes
- Community: yes
Agenda

- Example of the flaw in protocol design
- Requirements for protocol verification tools
- Approaches to protocol verification
- Model checking based protocol verification
- AVISPA
- Isabelle
- Other theorem proofing tools
- Comparison, conclusion

Isabelle

- General purpose theorem proofer
- Very flexible
- Different paradigm:
  - Make suggestion (based on experience, intuition etc.)
  - Proceed with proof of suggested claims
  - If proof does not succeed then either
    - Use another hints
    - Change your mind and prove opposite claim
  - Special skills/intuition are necessary
Header{"Verifying the Needham-Schroeder Public-Key Protocol"}
theory NS_Public_Bad imports Public begin

inductive_set ns_public :: "event list set"
where
  (*Initial trace is empty*)
Nil: "[] ∈ ns_public"

(*The spy MAY say anything he CAN say. We do not expect him to invent new nonces here, but he can also use NS1. Common to all similar protocols.*)

Fake: "[evsf ∈ ns_public; X ∈ synth (analz (spies evsf))]"
  ==> Says Spy B X # evsf ∈ ns_public"
(\*Alice initiates a protocol run, sending a nonce to Bob\*)

| NS1: "[evs1 \in ns\_public; Nonce NA \notin used evs1] \implies\ Says A B (Crypt (pubEK B) \{\text{Nonce NA, Agent A}\})
# evs1 \in ns\_public"

(Isabelle example of verification (2/2))

(\*Authentication for A: if she receives message 2 and has used NA to start a run, then B has sent message 2.\*)

\text{lemma A\_trusts\_NS2\_lemma [rule_format]:}

\text{"[A \notin bad; B \notin bad; evs \in ns\_public]\} \\
\implies\ Crypt (pubEK A) \{\text{Nonce NA, Nonce NB}\} \in parts (spies evs) -->
Says A B (Crypt(pubEK B) \{\text{Nonce NA, Agent A}\}) \in set evs -->
Says B A (Crypt(pubEK A) \{\text{Nonce NA, Nonce NB}\} \in set evs"

apply (erule ns\_public.induct)
apply (auto dest: Spy\_not\_see\_NA unique\_NA)
done
Isabelle score

- Automation: semi
- Model simplicity: no
- Flexibility: perfect
- Attack trace extraction: inapplicable
- Community: yes

Agenda

- Example of the flaw in protocol design
- Requirements for protocol verification tools
- Approaches to protocol verification
- Model checking based protocol verification
- AVISPA
- Isabelle
- Other theorem proofing tools
- Comparison, conclusion
Other theorem proofing tools

- ProVerif: convenient notation, false negatives, attack trace is not extracted
- CAPSL: convenient notation, authentication oriented

Other theorem proofing tools score

- Automation: yes
- Model simplicity: yes
- Flexibility: little
- Attack trace extraction: not always
- Community: little
Agenda

- Example of the flaw in protocol design
- Requirements for protocol verification tools
- Approaches to protocol verification
- Model checking based protocol verification
- AVISPA
- Isabelle
- Other theorem proving tools
- Comparison, conclusion

Comparison table

<table>
<thead>
<tr>
<th></th>
<th>Model checking based tools</th>
<th>AVISPA</th>
<th>Isabelle</th>
<th>Other theorem proving based tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automation</td>
<td>Yes</td>
<td>Yes</td>
<td>Semi</td>
<td>Yes</td>
</tr>
<tr>
<td>Model simplicity</td>
<td>Usually no</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Flexibility</td>
<td>Not much</td>
<td>Not perfect</td>
<td>Perfect</td>
<td>Not much</td>
</tr>
<tr>
<td>Attack trace extraction</td>
<td>Yes</td>
<td>Yes</td>
<td>Inapplicable</td>
<td>Not always</td>
</tr>
<tr>
<td>Community</td>
<td>Just for SPIN</td>
<td>Yes</td>
<td>Yes</td>
<td>Little</td>
</tr>
</tbody>
</table>
Conclusion

- AVISPA is main tool
- Isabelle is needed where flexibility is critical

Any questions?