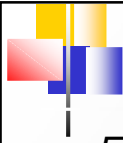


Analysis of Verification Tools for Security Protocols

Sergey Reznick, Igor Kotenko

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

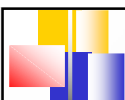
RE-TRUST Workshop, June 19, 2008



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

RE-TRUST Workshop, June 19, 2008



Needham-Schroeder Public Key Protocol

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

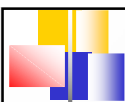
RE-TRUST Workshop, June 19, 2008



NSPK specification

1. A \rightarrow S : A,B
2. S \rightarrow A : {KPb, B}KSs
3. A \rightarrow B : {Na, A}KPb
4. B \rightarrow S : B,A
5. S \rightarrow B : {KPa, A}KSs
6. B \rightarrow A : {Na, Nb}KPa
7. A \rightarrow B : {Nb}KPb

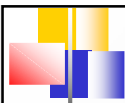
RE-TRUST Workshop, June 19, 2008



Attack for NSPK

- 3. $A \rightarrow B : \{Na, A\}_{KPb}$
 - i.3. $A \rightarrow I : \{Na, A\}_{KPi}$
 - ii.3. $I(A) \rightarrow B : \{Na, A\}_{KPb}$
- 6. $B \rightarrow A : \{Na, Nb\}_{KPa}$
 - ii.6. $B \rightarrow I(A) : \{Na, Nb\}_{KPa}$
 - i.6. $I \rightarrow A : \{Na, Nb\}_{KPa}$
- 7. $A \rightarrow B : \{Nb\}_{KPb}$
 - i.7. $A \rightarrow I : \{Nb\}_{KPi}$
 - ii.7. $I(A) \rightarrow B : \{Nb\}_{KPb}$

RE-TRUST Workshop, June 19, 2008



Formal analysis is necessary

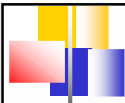
- NSPK announcement year: 1978
- NSPK flaw finding year: 1995
- Conclusion: any security protocol must be formally verified

RE-TRUST Workshop, June 19, 2008




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

SPIIRAS

RE-TRUST Workshop, June 19, 2008



No silver bullet

- Trade-off between different criteria
- Combination of tools

SPIIRAS

RE-TRUST Workshop, June 19, 2008



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



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.



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



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

```
-> 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;
```

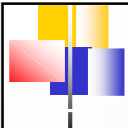
RE-TRUST Workshop, June 19, 2008



Murphi invariants for NSPK

```
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;
```

RE-TRUST Workshop, June 19, 2008

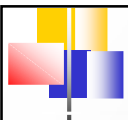


Murphi score

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

SPIIRAS

RE-TRUST Workshop, June 19, 2008

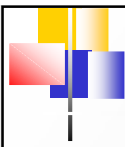


Other model checking tools

- SPIN: bigger community
- PRISM: probability support
- Casper: CSP based, better notation

SPIIRAS

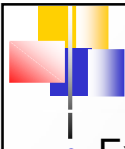
RE-TRUST Workshop, June 19, 2008



Main drawback of model checking approach

- States enumeration causing combinatorial explosion

RE-TRUST Workshop, June 19, 2008



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

RE-TRUST Workshop, June 19, 2008



AVISPA

- HPSL: High-Level Protocol Specification Language
- IF: Intermediate Format
- HPSL2IF: converter
- 4 backends:
 - OFMC
 - CL-AtSE
 - SATMC
 - TA4SP
- SPAN: visual tool

RE-TRUST Workshop, June 19, 2008



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

RE-TRUST Workshop, June 19, 2008



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

RE-TRUST Workshop, June 19, 2008



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

RE-TRUST Workshop, June 19, 2008



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 ($L(A_n) = L(A_{n+1})$)
 - “ $l \rightarrow r$ ” represents a protocol step.
- Benefit: approximation to infinite number of sessions
- Drawback: hang-ups

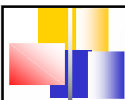
RE-TRUST Workshop, June 19, 2008



SPAN

- GUI tool to model protocol
- Intended behaviour mode:
- Behaviour with intruder mode

RE-TRUST Workshop, June 19, 2008



NSPK modelled via HPSL (1/7)

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

RE-TRUST Workshop, June 19, 2008



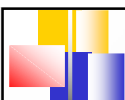
NSPK modelled via HPSL (2/7)

transition

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

end role

RE-TRUST Workshop, June 19, 2008



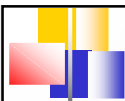
NSPK modelled via HPSL (3/7)

```
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
```

RE-TRUST Workshop, June 19, 2008



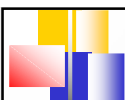
NSPK modelled via HPSL (4/7)

```
transition
```

1. State = 1 \wedge RCV($\{Na'.A\}_{Kb}$) =|>
 State' := 3 \wedge Nb' := new() \wedge
 SND($\{Na'.Nb'\}_{Ka}$)
 \wedge secret(Nb',nb,{A,B})
 \wedge witness(B,A,alice_bob_nb,Nb')
3. State = 3 \wedge RCV($\{Nb\}_{Kb}$) =|>
 State' := 5 \wedge request(B,A,bob_alice_na,Na)

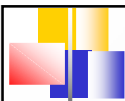
```
end role
```

RE-TRUST Workshop, June 19, 2008



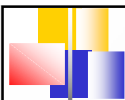
NSPK modelled via HPSL (5/7)

```
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)  $\wedge$  bob
        (A,B,Ka,Kb,SB,RB)
    end role
```



NSPK modelled via HPSL (6/7)

```
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)  $\wedge$  session(a,i,ka,ki)  $\wedge$ 
    session(i,b,ki,kb)
  end role
```

NSPK modelled via HPSL (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 proving tools
- Comparison, conclusion

RE-TRUST Workshop, June 19, 2008



Isabelle

- General purpose theorem prover
- 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

RE-TRUST Workshop, June 19, 2008



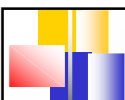
Isabelle example of verification (1/2)

```
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"
```



Isabelle example of verification (1/2)

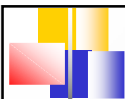
```
(*The spy MAY say anything he CAN say. We  
do not expect him to invent new nonces here,  
but he can also use NS1. Common to all  
similar protocols.*)  
  
| Fake: "[|evsf ∈ ns_public; X ∈ synth (analz  
  (spies evsf))|]"  
      ==> Says Spy B X # evsf ∈ ns_public"
```



Isabelle example of verification (1/2)

(*Alice initiates a protocol run, sending a nonce to Bob*)

```
| NS1: "[|evs1 ∈ ns_public; Nonce NA ∉ used
evs1|] ==> Says A B (Crypt (pubEK B)
\<lbrace>Nonce NA, Agent A\<rbrace>)
# evs1 ∈ 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.*)


lemma A_trusts_NS2_lemma [rule_format]:

```
"[|A ∉ bad; B ∉ bad; evs ∈ ns_public|]
==> Crypt (pubEK A) \<lbrace>Nonce NA, Nonce
NB\<rbrace> ∈ parts (spies evs) -->
  Says A B (Crypt(pubEK B) \<lbrace>Nonce NA,
Agent A\<rbrace>) ∈ set evs -->
  Says B A (Crypt(pubEK A) \<lbrace>Nonce NA,
Nonce NB\<rbrace>) ∈ 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

SPIIRAS

RE-TRUST Workshop, June 19, 2008



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

SPIIRAS

RE-TRUST Workshop, June 19, 2008




Other theorem proving tools

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



Other theorem proving 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*

SPIIRAS


RE-TRUST Workshop, June 19, 2008



Comparison table

	Model checking based tools	AVISPA	Isabelle	Other theorem proving based tools
Automation	Yes	Yes	Semi	Yes
Model simplicity	Usually no	Yes	No	Yes
Flexibility	Not much	Not perfect	Perfect	Not much
Attack trace extraction	Yes	Yes	Inapplicable	Not always
Community	Just for SPIN	Yes	Yes	Little

RE-TRUST Workshop, June 19, 2008




Conclusion

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

SPIIRAS

RE-TRUST Workshop, June 19, 2008



Finish

Any questions ?

SPIIRAS

RE-TRUST Workshop, June 19, 2008