Security Models
Lecture 5
Tools

Pascal Lafourcade

2020-2021

Outline of Today:

1. Summerize

Outline of Today:

1. Summerize
2. AVISPA

Outline of Today:

1. Summerize
2. AVISPA
3. Scyther

Outline of Today:

1. Summerize
2. AVISPA
3. Scyther
4. Proverif

Outline of Today:

1. Summerize
2. AVISPA
3. Scyther
4. Proverif
5. Comparaison

Outline of Today:

1. Summerize
2. AVISPA
3. Scyther
4. Proverif
5. Comparaison
6. Algebraic Properties

Outline of Today:

1. Summerize
2. AVISPA
3. Scyther
4. Proverif
5. Comparaison
6. Algebraic Properties
7. FFGG

Outline of Today:

1. Summerize
2. AVISPA
3. Scyther
4. Proverif
5. Comparaison
6. Algebraic Properties
7. FFGG
8. Conclusion

Outline

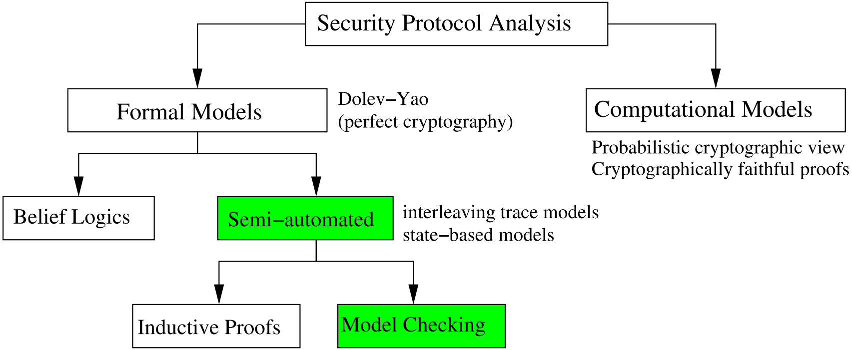
1. Summerize
2. AVISPA
3. Scyther
4. Proverif
5. Comparaison
6. Algebraic Properties
7. FFGG
8. Conclusion

Complexity

Complexity depends of intruder capabilities. In classical Dolev-Yao intruder model we (pair + encryption) we have the following results:

- Passive Intruder
  Problem is polynomial
- Bounded Number of sessions
  Problem is NP-complete
  Tools can verify 3-4 sessions: useful to finds flaws ! OFMC, Cl-Atse, SATMC, FDR, Athena...
- Unbounded Number of sessions
  Problem is in general undecidable
  Tools are corrects, but uncomplete (can find false attacks, can not terminate) Proverif, TA4SP, Scyther.

Formal Landscape and Our Focus



N.B. Challenging as general problem is undecidable due e.g. to the possibility of unbounded number of protocol sessions.

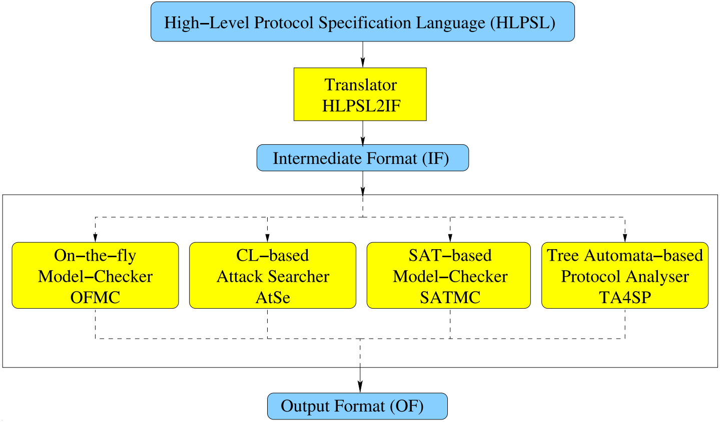
Tools studied Today

- **Avispa** : Plateform with 4 tools: OFMC, CL-AtSe, SATMC, and TA4SP.
- **Proverif**: Analyses unbounded number of session using over-approximation with Horn Clauses.
- **Scyther**: Verifies bounded and unbounded number of session with backwards search based on partially ordered patterns.

Outline

1. Summerize
2. AVISPA
3. Scyther
4. Proverif
5. Comparaison
6. Algebraic Properties
7. FFGG
8. Conclusion

The AVISPA Tool: Architecture



The AVISPA Tool: the Back-Ends

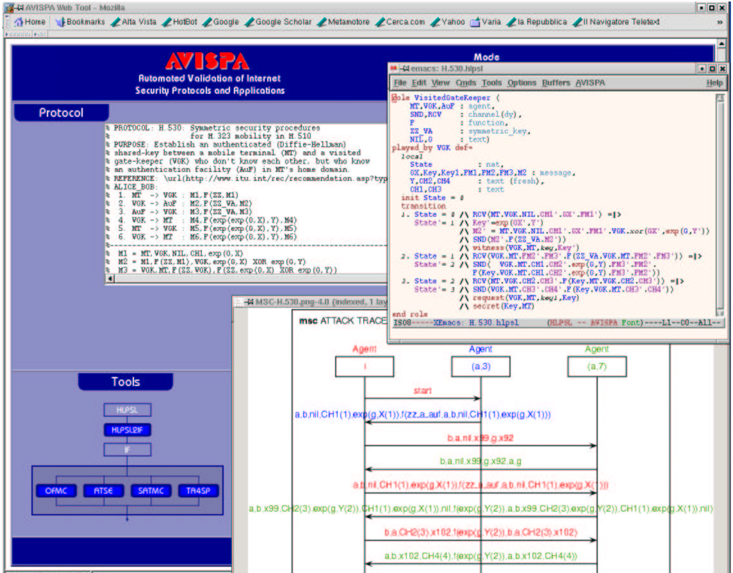
On-the-fly Model-Checker (OFMC) employs several symbolic techniques to explore the state space in a demand-driven way.

CL-AtSe (Constraint-Logic-based Attack Searcher) applies constraint solving with simplification heuristics and redundancy elimination techniques.

The SAT-based Model-Checker (SATMC) builds a propositional formula encoding all the possible traces (of bounded length) on the protocol and uses a SAT solver.

TA4SP (Tree Automata based on Automatic Approximations for the Analysis of Security Protocols) approximates the intruder knowledge by using regular tree languages and rewriting to produce under and over approximations.

The Web Interface www.avispa-project.org



```
% PROTOCOL: H.530: Symmetric security procedures
%                  for H.323 mobility in H.510
% PURPOSE: Establish an authenticated (Diffie-Hellman)
% shared-key between a mobile terminal (MT) and a visited
% gate-keeper (VGK) who don't know each other, but who know
% an authentication facility (AuF) in MT's home domain.
% REFERENCE: \url{http://www.itu.int/rec/recommendation.asp?typ
% ALICE_BOB:
%  1. MT  -> VGK : M1,F(ZZ,M1)
%  2. VGK -> AuF : M2,F(ZZ_VA,M2)
%  3. AuF -> VGK : M3,F(ZZ_VA,M3)
%  4. VGK -> MT  : M4,F(exp(exp(G,X),Y),M4)
%  5. MT  -> VGK : M5,F(exp(exp(G,X),Y),M5)
%  6. VGK -> MT  : M6,F(exp(exp(G,X),Y),M6)
%-------------------------------------------------------------
%  M1 = MT,VGK,NIL,CH1,exp(G,X)
%  M2 = M1,F(ZZ,M1),VGK,exp(G,X) XOR exp(G,Y)
%  M3 = VGK,MT,F(ZZ,VGK),F(ZZ,exp(G,X) XOR exp(G,Y))
```

```
role VisitedGateKeeper (
    MT,VGK,AuF : agent,
    SND,RCV    : channel(dy),
    F          : function,
    ZZ_VA      : symmetric_key,
    NIL,G      : text)
played_by VGK def=
  local
    State                  : nat,
    GX,Key,Key1,FM1,FM2,FM3,M2 : message,
    Y,CH2,CH4              : text (fresh),
    CH1,CH3                : text
  init State = 0
  transition
  1. State = 0 /\ RCV(MT.VGK.NIL.CH1'.GX'.FM1') =|>
     State'= 1 /\ Key'=exp(GX',Y')
               /\ M2' = MT.VGK.NIL.CH1'.GX'.FM1'.VGK.xor(GX',exp(G,Y'))
               /\ SND(M2'.F(ZZ_VA.M2'))
               /\ witness(VGK,MT,key,Key')
  2. State = 1 /\ RCV(VGK.MT.FM2'.FM3'.F(ZZ_VA.VGK.MT.FM2'.FM3')) =|>
     State'= 2 /\ SND(  VGK.MT.CH1.CH2'.exp(G,Y).FM3'.FM2'.
                  F(Key.VGK.MT.CH1.CH2'.exp(G,Y).FM3'.FM2'))
  3. State = 2 /\ RCV(MT.VGK.CH2.CH3'.F(Key.MT.VGK.CH2.CH3')) =|>
     State'= 3 /\ SND(VGK.MT.CH3'.CH4'.F(Key.VGK.MT.CH3'.CH4'))
               /\ request(VGK,MT,key1,Key)
               /\ secret(Key,MT)
end role
ISO8-----XEmacs: H.530.hlpsl      (HLPSL -- AVISPA Font)----L1--C0--All--
```

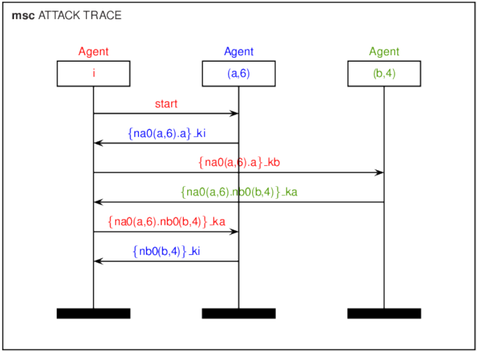
Results in AVISPA

```
AVISPA Tool Summary

OFMC    : UNSAFE
CL-AtSe : UNSAFE
SATMC   : UNSAFE
TA4SP   : INCONCLUSIVE

Refer to individual tools output for details
```

MSC Description of the Attack in AVISPA



Needham-Schroeder : Alice

```
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 /\ RCV(start) =|>
        State':= 2 /\ Na' := new() /\ SND({Na'.A}_Kb)
                   /\ secret(Na',na,{A,B})

    2.  State  = 2 /\ RCV({Na.Nb'}_Ka) =|>
        State':= 4 /\ SND({Nb'}_Kb)
end role
```

Needham-Schroeder: Bob

```
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

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

    3.  State  = 3 /\ RCV({Nb}_Kb) =|>
end role
```

Needham-Schroeder: Session, Environement & Goal

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

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

- Agent: names of principles
- public key: asymmetric keys
- symmetric key: symmetric keys
- nat: natural numbers
- function: to model hash functions etc
- bool: Boolean values for modeling flags

Kinds of variables:

- State variables: Those that are within the scope of a role.
- Declared at the top of a role
- Unprimed versions indicate current state
- Primed versions indicate next state

Role Definition

1. Role declaration: its name and the list of formal arguments, along with (in the case of basic roles) a player declaration;
2. Declaration of local variables and ownership rules, if any;
3. Initialization of variables, if required;
4. Declaration of accepting states, if any;
5. Knowledge declarations, if applicable;
6. Either (optionally) : a transition section (for basic roles) or a composition section (for composed roles).

Outline

Scyther

- Alternative: backwards search based on patterns
    - Security properties represented by claim events in the protocol.
    - Supports symmetric and asymmetric keys, cryptographic hash functions, key-tables, multiple protocols in parallel, composed keys, etc (but no user-definable algebraic functions)
    - Can perform unbounded verification of protocols
    - Provides *complete characterization* of protocol roles: Answer to: "after execution of a protocol role, what events must also have occurred?"
- Also state-of-art. Freely available for download for Windows, Linux and Mac OS X.
- Will be used in the exercise sessions.

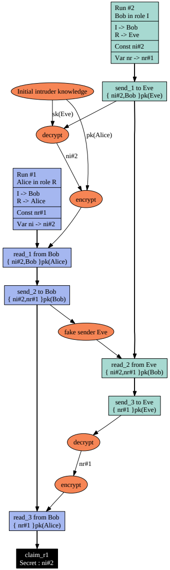
Input

```
protocol ns3(I,R) {
  role I {
    const ni: Nonce;
    var nr: Nonce;
    send_1(I,R, {ni,I}pk(R) );
    read_2(R,I, {ni,nr}pk(I) );
    send_3(I,R, {nr}pk(R) );

    claim_i1(I,Secret,ni);
    claim_i2(I,Nisynch);
  }
  role R {
    var ni: Nonce;
    const nr: Nonce;
    read_1(I,R, {ni,I}pk(R) );
    send_2(R,I, {ni,nr}pk(I) );
    read_3(I,R, {nr}pk(R) );

    claim_r1(R,Secret,ni);
    claim_r2(R,Nisynch);
} }
```

Output (<0.02 seconds)



[Id 1] Protocol ns3, role R, claim type Secret

Outline

1. Summerize
2. AVISPA
3. Scyther
4. **Proverif**
5. Comparaison
6. Algebraic Properties
7. FFGG
8. Conclusion

Proverif

Proverif uses spi-calculus or Horn Clauses

`analyser -in horn toto.pv` OR `analyser -in pi toto.pv`

Proverif: Horn Clauses

```
(* Needham Shroeder Lowe  *)
pred c/1 elimVar,decompData.
nounif c:x.
fun pk/1.
fun encrypt/2.

query c:secret[].
reduc

(* Initialization *)
c:c[];
c:pk(sA[]);
c:pk(sB[]);

c:x & c:encrypt(m,pk(x)) -> c:m;
c:x -> c:pk(x);
c:x & c:y -> c:encrypt(x,y);
```

Proverif: Horn Clauses

```
(* The protocol *)
(* A *)
c:pk(x) -> c:encrypt((Na[pk(x)], pk(sA[])), pk(x));

c:pk(x) & c:encrypt((Na[pk(x)], y), pk(sA[]))
-> c:encrypt((y,k[pk(x)]), pk(x));

(* B *)
c:encrypt((x,y), pk(sB[]))
-> c:encrypt((x, Nb[x,y], pk(sB[])), y);

c:encrypt((x,pk(sA[])))
& c:encrypt((Nb[x, pk(sA[])], z), pk(sB[]))
-> c:encrypt(secret[], pk(z)).
```

Proverif

```
goal reachable: c:secret[]

rule 7 c:secret[]
  any c:x_182
  rule 1 c:encrypt(secret[],pk(x_182))
    rule 5 c:encrypt((Na[pk(x_168)],pk(sA[])),pk(sB[]))
      2-tuple c:(Na[pk(x_168)],pk(sA[]))
        0-th c:Na[pk(x_168)]
          rule 7 c:(Na[pk(x_168)],pk(sA[]))
            any c:x_168
            rule 4 c:encrypt((Na[pk(x_168)],pk(sA[])),pk(x_168))
              rule 6 c:pk(x_168)
                any c:x_168
        rule 9 c:pk(sA[])
      rule 8 c:pk(sB[])
    rule 5 c:encrypt((Nb[Na[pk(x_168)],pk(sA[])],x_182),pk(sB[]))
      2-tuple c:(Nb[Na[pk(x_168)],pk(sA[])],x_182)
        0-th c:Nb[Na[pk(x_168)],pk(sA[])]
          rule 7 c:(Nb[Na[pk(x_168)],pk(sA[])],k[pk(x_168)])
            any c:x_168
            rule 3 c:encrypt((Nb[Na[pk(x_168)],pk(sA[])],k[pk(x_168)]),pk(x_168))
              rule 6 c:pk(x_168)
                any c:x_168
              rule 2 c:encrypt((Na[pk(x_168)],Nb[Na[pk(x_168)],pk(sA[])]),pk(sA[]))
                rule 5 c:encrypt((Na[pk(x_168)],pk(sA[])),pk(sB[]))
                  2-tuple c:(Na[pk(x_168)],pk(sA[]))
                    0-th c:Na[pk(x_168)]
                      rule 7 c:(Na[pk(x_168)],pk(sA[]))
                        any c:x_168
                        rule 4 c:encrypt((Na[pk(x_168)],pk(sA[])),pk(x_168))
                          rule 6 c:pk(x_168)
                            any c:x_168
                    rule 9 c:pk(sA[])
                  rule 8 c:pk(sB[])
        any c:x_182
      rule 8 c:pk(sB[])

RESULT goal reachable: c:secret[]
```

What is the spi-calculus?

The spi-calculus is an extension of the pi-calculus designed to represent cryptographic protocols.

The pi-calculus is a process calculus:

- processes communicate: they can send and receive messages on channels several processes can execute in parallel.
- In the pi-calculus, messages and channels are names, that is, atomic values a, b, c, ...

What is the spi-calculus ? (continued)

Example:

$$\overline{c} < a > | c(x).\overline{d} < x >$$

The first process sends a on channel c, the second one inputs this message, puts it in variable x and sends x on channel d.

The link with cryptographic protocols is clear:

- Each participant of the protocol is represented by a process
- The messages exchanged by processes are the messages of the protocol.

However, in protocols, messages are not necessarily atomic values. The names of the pi calculus are replaced by terms in the spi

calculus.

Proverif

Pi calculus + cryptographic primitives

$M, N ::=$	terms
x, y, z	variable
$a, b, c, k,$	name
$f(M_1, ..., M_n)$	constructor application

$P, Q ::=$	processes
$\overline{M} < N > .P$	output
$M(x).P$	input
let $x = g(M_1, ..., M_n)$ in P else Q	destructor application
if $M = N$ then P else Q	conditional
0	nil process
$P \vert Q$	parallel composition
$!P$	replication
$(\nu a)P$	restriction

Example: Denning Sacco

Message 1. $A \rightarrow B : \{\{k\}_{sk_A}\}_{pk_B}$
Message 2. $B \rightarrow A\{s\}_k, k$ fresh

$(\nu sk_A)(\nu sk_B)$ let $pk_A = pk(sk_A)$ in let $pk_B = pk(sk_B)$ in
$\overline{c} < pk_A > \overline{c} < pk_B >$.
(A) $!c(x\_pk_B).(\nu k)\overline{c} < pencrypt(sign(k, sk_A), x\_pk_B) >$
$.c(x)$.let$s = sdecrypt(x, k)$ in 0
(B) $|!c(y)$.let$y' = pdecrypt(y, sk_B)$ in let $k = checksign(y', pk_A)$ in
$\overline{c} < sencrypt(s, k) >$

Proverif: Pi Calculus

```
free c.
(* Public key cryptography *)
fun pk/1.
private fun sk/1.
(* just encryption, no signing *)
fun encrypt/2.
reduc decrypt(encrypt(x,pk(y)),sk(y)) = x.

(* Symmetric key cryptography *)
fun symcrypt/2.
reduc symdecrypt(symcrypt(z,j),j) = z.

(* Effectively the claim signals *)
private free secretANa, secretANb, secretBNa, secretBNb, secretAtoB, secretBtoA

(* Security claims to verify *)
query attacker:secretANa;
      attacker:secretANb;
      attacker:secretAtoB;
      attacker:secretBNa;
      attacker:secretBNb;
      attacker:secretBtoA.
```

Proverif: Pi Calculus

```
let processA =
        (* Choose the other host *)
         in(c, X);
        new Na;
        out(c, encrypt((Na,X),pk(X)));
        in(c,m2);
        let (=Na, nb) = decrypt(m2, sk(A)) in
        out(c, encrypt(nb,pk(X)));
        if X = A then
                out(c, symcrypt(secretANa, Na));
                out(c, symcrypt(secretANb, nb))
                else        if X = B then
                        out(c, symcrypt(secretAtoB, Na));
                        out(c, symcrypt(secretAtoB, nb));
                        out(c, symcrypt(secretANa, Na));
                        out(c, symcrypt(secretANb, nb)).
```

Proverif: Pi Calculus

```
let processB =
       in(c,m1);
       let (na,Y) = decrypt(m1, sk(B)) in
       new Nb;
       out(c, encrypt((na, Nb), pk(Y)));
       in(c,m3);
       let (=Nb) = decrypt(m3, sk(B)) in
       if Y = A then
              out(c, symcrypt(secretBtoA, na));
              out(c, symcrypt(secretBtoA, Nb));
              out(c, symcrypt(secretBNa, na));
              out(c, symcrypt(secretBNb, Nb))
              else       if Y = B then
                     out(c, symcrypt(secretBNa, na));
                     out(c, symcrypt(secretBNb, Nb)).
```

Proverif: Pi Calculus

```
let processBbyA =
       in(c,m1);
       let (na,Y) = decrypt(m1, sk(A)) in
       new Nb;
       out(c, encrypt((na, Nb), pk(Y)));
       in(c,m3);
       let (=Nb) = decrypt(m3, sk(A)) in
       if Y = A then
              out(c, symcrypt(secretBtoA, na));
              out(c, symcrypt(secretBtoA, Nb));
              out(c, symcrypt(secretBNa, na));
              out(c, symcrypt(secretBNb, Nb))
              else       if Y = B then
                     out(c, symcrypt(secretBNa, na));
                     out(c, symcrypt(secretBNb, Nb)).
```

Proverif: Pi Calculus

```
process
        new A;
        new B;
        new I;

        out(c,A);
        out(c,B);
        out(c,I);
        out(c,sk(I));

        ((!processA) | (!processB) | (!processBbyA))
```

Proverif: Pi Calculus

```
RESULT not attacker:secretANa[] is true.
RESULT not attacker:secretANb[] is false.
RESULT not attacker:secretAtoB[] is true.
RESULT not attacker:secretBNa[] is false.
RESULT not attacker:secretBNb[] is false.
RESULT not attacker:secretBtoA[] is false.
```

Outline

1. Summerize
2. AVISPA
3. Scyther
4. Proverif
5. Comparaison
6. Algebraic Properties
7. FFGG
8. Conclusion

Necessity of Tools

- Protocols are small recipes.
- Non trivial to design and understand.
- The number and size of new protocols.
- Out-pacing human ability to rigourously analyze them.

GOAL : A tool is finding flaws or establishing their correctness.

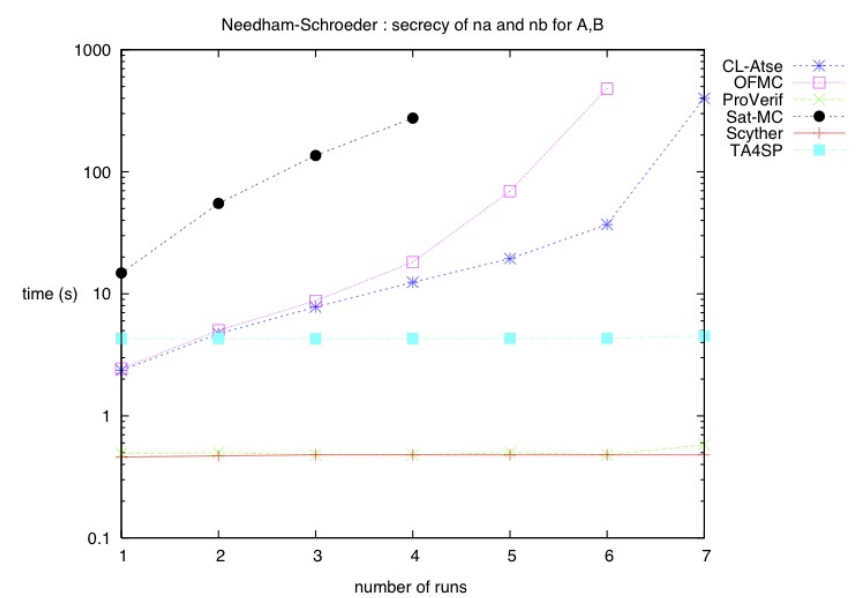
- completely automated,
- robust,
- expressive,
- and easily usable.

Existing Tools: AVISPA, Scyther, Proverif, Hermes, Casper/FDR, Murphi, NRL ...

Comparison of Tools Dealing with Algebraic Properties ?

Bibliography

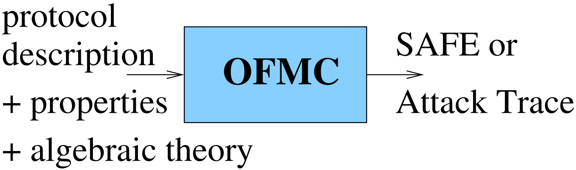
- **Time performence comparison of AVISPA Tools**
  L. Vigano “Automated Security Protocol Analysis With the AVISPA Tool” ENTCS 2006.
- **Usability comparison between AVISPA and HERMES**
  M. Hussain and D. Seret “A Comparative study of Security Protocols Validation Tools: HERMES vs. AVISPA”.In the 8th International Conference Advanced Communication Technology, ICACT’06.
- Cas Cremers and Pascal Lafourcade. **Comparing State Spaces in Automatic Security Protocol Verification**. In Michael Goldsmith and Bill Roscoe (eds.), Proceedings of the 7th International Workshop on Automated Verification of Critical Systems (AVoCS’07), Oxford, UK, September 2007, Electronic Notes in Theoretical Computer Science, pages 49-63. Elsevier Science Publishers.


Needham-Schroeder : secrecy of na and nb for A,B
1000
100
time (s)
10
1
0.1
1
2
3
4
5
6
7
number of runs
CL-Atse
OFMC
ProVerif
Sat-MC
Scyther
TA4SP

Outline

1. Summerize
2. AVISPA
3. Scyther
4. Proverif
5. Comparaison
6. **Algebraic Properties**
7. FFGG
8. Conclusion

Model Checking with OFMC



Input:

- Transition system (initial state + transition relation).
- Goal : insecure states (i.e., attacks).
- Algebraic properties.

Output:

- SAFE indicating security for bounded sessions or
- An attack trace.

Supported Theories by OFMC

$$\begin{aligned} (x_1^{x_2})^{x_3} &\approx (x_1^{x_3})^{x_2} \\ x_1 \oplus x_2 &\approx x_2 \oplus x_1 \\ x_1 \oplus (x_2 \oplus x_3) &\approx (x_1 \oplus x_2) \oplus x_3 \end{aligned}$$

Finite Theories F:
The F-equivalence class of every term is finite.

$$\begin{aligned} dec(x_2, \{x_1\}_{x_2}) &\approx x_1 \\ x_1 \oplus x_1 &\approx 0 \\ x_1 \oplus 0 &\approx x_1 \end{aligned}$$

Cancellation theories C:
One side of each equation is a variable of the other side, or a constant.

Supported Theories by OFMC

$$\begin{aligned} (x_1^{x_2})^{x_3} &\approx (x_1^{x_3})^{x_2} \\ x_1 \oplus x_2 &\approx x_2 \oplus x_1 \\ x_1 \oplus (x_2 \oplus x_3) &\approx (x_1 \oplus x_2) \oplus x_3 \end{aligned}$$

$$\begin{aligned} dec(x_2, \{x_1\}_{x_2}) &\approx x_1 \\ x_1 \oplus x_1 &\approx 0 \\ x_1 \oplus 0 &\approx x_1 \end{aligned}$$

Finite Theories F:
The F-equivalence class of every term is finite.

Cancellation theories C:
One side of each equation is a variable of the other side, or a constant.

Rewriting with C modulo F, e.g.

$$a \oplus b \oplus a \rightarrow_{C/F} 0 \oplus b \rightarrow_{C/F} b\ .$$

We require: $\rightarrow_{C/F}$ is convergent.

On-the-Fly Model-Checker (OFMC)

- Common language for specifying protocols and security properties.
- Supports symmetric and asymmetric keys, cryptographic hash functions, key-tables, user-definable algebraic functions, etc.

Input

```
PROTOCOL Needham-Schroeder;
Identifiers
  A, B: user;
  Na, Nb: nonce;
  Ka, Kb: public_key;
Messages
  1.  A -> B:  {A,Na}Kb
  2.  B -> A:  {Na,Nb}Ka
  3.  A -> B:  {Nb}Kb
Intruder_knowledge Spy, b, ka, kb, kspy;
Goal correspondence_between A B;
```

Output (<1 second)

```
A -> Spy: {A,Na}Kspy
Spy -> B: {A,Na}Kb
B -> A: {Na,Nb}Ka
A -> Spy: {Nb}Kspy
Spy -> B {Nb}Kb
```

Secure or not ?

One protocol: K secret key between A and B ?

$$\begin{array}{lcll} A & \rightarrow & S: & A, B, \{A \oplus N_A\}K_S, \{N_A \oplus c\}K_S \\ S & \rightarrow & B: & A, B, S \\ B & \rightarrow & S: & B, A, \{B \oplus N_B\}K_S, \{N_B \oplus c\}K_S \\ S & \rightarrow & A: & K \oplus \{N_A\}K_S \\ S & \rightarrow & B: & K \oplus \{N_B\}K_S \end{array}$$

OFMC answers : SAFE with exlusive-or $\oplus$

Secure or not ?

One protocol: K secret key between A and B ?

$$\begin{array}{lcll} A & \rightarrow & S: & A, B, \{A \oplus N_A\}K_S, \{N_A \oplus c\}K_S \\ S & \rightarrow & B: & A, B, S \\ B & \rightarrow & S: & B, A, \{B \oplus N_B\}K_S, \{N_B \oplus c\}K_S \\ S & \rightarrow & A: & K \oplus \{N_A\}K_S \\ S & \rightarrow & B: & K \oplus \{N_B\}K_S \end{array}$$

OFMC answers : SAFE with exlusive-or $\oplus$

But with $\{x \oplus y\}_{K_S} = \{x\}_{K_S} \oplus \{y\}_{K_S}$

There is an attack !

Tools Dealing with Exclusive-Or and Diffie-Hellman

- **Avispa**:
  - OFMC: On-the-fly Model-Checker employs several symbolic techniques to explore the state space in a demand-driven way.
  - CL-Atse: Constraint-Logic-based Attack Searcher applies constraint solving with simplification heuristics and redundancy elimination techniques.
- **Proverif**: Analyses unbounded number of session using over-approximation with Horn Clauses.
  - XOR-ProVerif and DH-ProVerif: are two tools developed by Kuesters et al for analyzing cryptographic protocols with Exclusive-Or and Diffie-Hellman properties, using ProVerif

PC DELL E4500 Intel dual Core 2.2 Ghz with 2 GB of RAM.
Work done with Sylvain Vigier qnd Vanessa Terrade, presneted to FAST 09.

Exclusive-Or Summary

Tools Protocols	Avispa		ProVerif
	OFMC	CL-Atse	XOR-ProVerif
Bull	UNSAFE Survey secrecy attack 0.08 s	UNSAFE Survey secrecy attack 0.08 s	No result XOR-ProVerif Does not end
Bull v2	The analysis Does not end time search: 20 h	SAFE 1 h 10 min	No result XOR-ProVerif Does not end
WEP	UNSAFE Survey secrecy attack 0.01 s	UNSAFE Survey secrecy attack less than 0.01 s	UNSAFE Survey secrecy attack less than 1 s
WEP v2	SAFE 0.01 s	SAFE less than 0.01 s	SAFE less than 1 s
Gong	SAFE 19 s	SAFE 1 min 34 s	No result Does not end
Salary Sum	UNSAFE New secrecy attack 0.45 s	UNSAFE New secrecy attack 11 min 16 s	UNSAFE Survey secrecy attack Does not end
TMN	UNSAFE New secrecy attack 0.04 s	UNSAFE New secrecy attack less than 0.01 s	UNSAFE New secrecy attack less than 1 s
EAuction	SAFE less than 1s	SAFE 0.59 s	SAFE less than 1 s

Diffie-Hellman Summary

Tools Protocols	Avispa		ProVerif
	OFMC	CL-Atse	DH-ProVerif
D.H	UNSAFE Survey authentication attack 0.01 s	UNSAFE Survey authentication attack less than 0.01 s	UNSAFE Survey authentication attack less than 1 s
IKA	UNSAFE Survey authentication and secrecy attack less than 0.01 s	UNSAFE Survey authentication and secrecy attack less than 0.01 s	UNSAFE 1s+2min 33s SAFE 3s + 1s

Conclusion

- Usually same attacks with OFMC, CL-Atse, and XOR-ProVerif or DH-ProVerif.
- Attack most of the time identical to those of the survey (except for Salary Sum and TMN)

Conclusion for Exclusive-Or

- OFMC terminates it is globally faster that CL-Atse.
- But for protocols using a large number of Exclusive-Or operations, *e.g.* for instance in the Bull's protocol, OFMC does not terminates whereas CL-Atse does.
- the number of Exclusive-Or used in a protocol is the parameter which increases verification time.
- If the number of variables and constants is not too large ProVerif is very efficient and faster that Avispa tools.

Conclusion for Diffie-Hellman

All protocols were analyzed quickly by all the tools.
This confirms the polynomial complexity of DH-ProVerif and the fact that this equational theory is less complex than Exclusive-Or.

Next Time

Playing with Tools

- Scyther
- Avispa: OFMC, Cl-Atse, SATMC, TA4SP
- Proverif
- Non Interference...

Outline

1. Summerize
2. AVISPA
3. Scyther
4. Proverif
5. Comparaison
6. Algebraic Properties
7. **FFGG**
8. Conclusion

FFGG by J.Millen

1 $A \rightarrow B : A$
2 $B \rightarrow A : B, N, M$
3 $A \rightarrow B : A, \{N, M, S\}_{PkB}$ for B view $A, \{N, X, S\}_{PkB}$
4 $B \rightarrow A : N, X, \{X, S, N\}_{PkB}$

FFGG by J.Millen

1 $A \rightarrow B : A$

2 $B \rightarrow A : B, N, M$

3 $A \rightarrow B : A, \{N, M, S\}_{PkB}$ for B view $A, \{N, X, S\}_{PkB}$

4 $B \rightarrow A : N, X, \{X, S, N\}_{PkB}$

Is S secret ?

FFGG by J.Millen

1 $A \rightarrow B : A$

2 $B \rightarrow A : B, N, M$

3 $A \rightarrow B : A, \{N, M, S\}_{PkB}$ for B view $A, \{N, X, S\}_{PkB}$

4 $B \rightarrow A : N, X, \{X, S, N\}_{PkB}$

Is S secret ?

Parallel Attack

1.1 $A \rightarrow B$: A

2.1 $A \rightarrow B$: A

1.2 $B \rightarrow I(A)$: B, N_1, M_1

2.2 $B \rightarrow I(A)$: B, N_2, M_2

1.2 $I(B) \rightarrow A$: B, N_1, N_2 a

1.3 $A \rightarrow B$: $A, \{N_1, N_2, S\}_{PkB}$ b

1.4 $B \rightarrow A$: $N_1, N_2, \{N_2, S, N_1\}_{PkB}$ c

2.3 $I(A) \rightarrow B$: $A, \{N_2, S, N_1\}_{PkB}$ d

2.4 $B \rightarrow A$: $N_2, S, \{S, N_1, N_2\}_{PkB}$

Outline

1. Summerize
2. AVISPA
3. Scyther
4. Proverif
5. Comparaison
6. Algebraic Properties
7. FFGG
8. **Conclusion**

Summary

Today

- Hermes
- Scyther
- Avispa
- Proverif

Conclusion

- Automatic verification is necessary.
- Tool are very helpful for design and verification.
- Use your favorite tool.
- Modeling of a protocol is quite tricky.
- Know the limitations of the tool and what you are checking.

Next

- Others Protocols
- Others properties
- Others Tools: Maude NPA, TA4SP, new OFMC

Thank you for your attention.

Questions ?