The High-Level Protocol Specification Language HLPSL developed in the EU project AVISPA

David von Oheimb
Siemens Corporate Technology, Munich

Automated Validation of Internet Security Protocols and Applications

Shared cost RTD (FET open) project IST-2001-39252

Context

- The world is distributed:
  - Our basic infrastructures are increasingly based on networked information systems.
- Essential to developing networked services and applications are protocols.
- In protocol design, a major problem is security errors.

Money: development and security updates are costing many millions of €.
Time: protocols are delayed by years.
Acceptance: confidence in network and application security is eroding.

Motivation

- Key: the number and scale of new security protocols under development is out-pacing the human ability to rigorously analyze and validate them.
- To speed up the development of new security protocols and to improve their security, it is important to have
  - tools that support the rigorous analysis of security protocols
  - by either finding flaws or establishing their correctness.
- Optimally, these tools should be completely automated, robust, expressive, and easily usable, so that they can be integrated into the protocol development and standardization processes.

AVISPA Project Objectives

1. Develop a rich specification language, HLPSL,
   for formalising industrial strength security protocols and their properties.

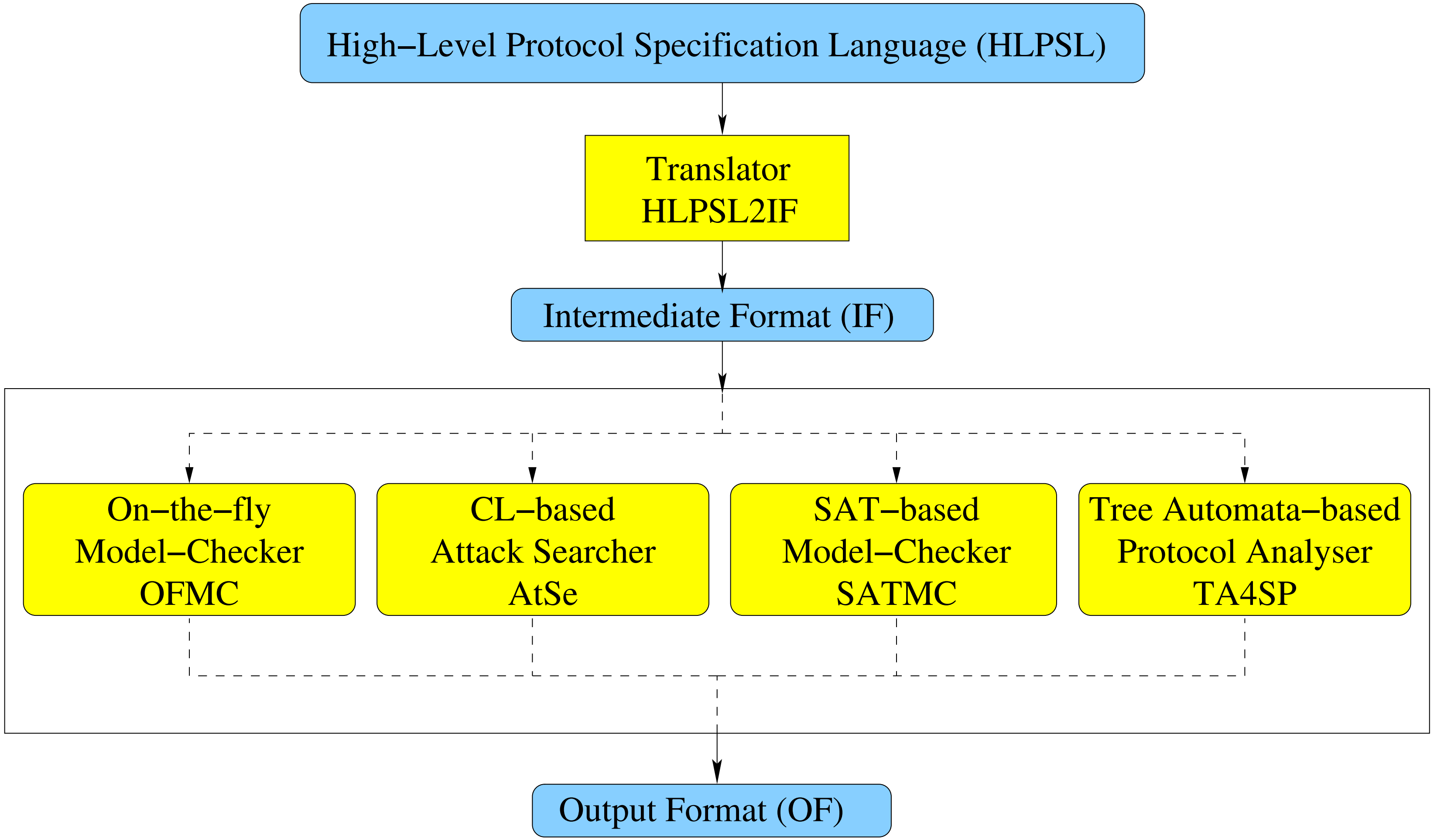
2. Advance state-of-the-art analysis techniques to scale up to this complexity.

3. Develop a toolset, the AVISPA Tool, based on these techniques.

4. Tune and assess the tool on a large library of practically relevant, industrial protocols.

5. Initiate migration of this technology to companies and standardisation organisations.

The AVISPA Tool



Running Example

NSPK Key Server Protocol:

if A *does not know* K_B,

$$A \rightarrow S : \quad A, B$$

$$S \rightarrow A : \quad \{B, K_B\}_{K_S^{-1}}$$

$$A \rightarrow B : \quad \{N_A, A\}_{K_B}$$

if B *does not know* K_A,

$$B \rightarrow S : \quad B, A$$

$$S \rightarrow B : \quad \{A, K_A\}_{K_S^{-1}}$$

$$B \rightarrow A : \quad \{N_A, N_B\}_{K_A}$$

$$A \rightarrow B : \quad \{N_B\}_{K_B}$$

non-trivial data structures (e.g., key rings) and
control flow not covered by (most) other tools!

Modular Specification Using Roles

- Basic roles:
  - Alice (initiator)
  - Bob (responder)
  - a central server
- Composed roles:
  - definition of a session: one Alice and one Bob,
  - instantiations: one server, several sessions.
- Each role has a local environment.

Header of Basic Role Bob

```
role bob(A,B: agent,
         Kb,Ks: public_key,
         KeyRing: (agent.public_key) set,
         SND,RCV: channel(dy))
   played_by B def=
   local
      State: nat,
      Na,Nb: text,
      Ka: public_key
   init
      State:=1
   transition
      ...
end role
```

Transitions of Bob

```
1a.   State  =1 /\ RCV({Na'.A}_Kb) /\ in(A.Ka',KeyRing)
 =|> State':=3 /\ Nb':=new() /\ SND({Na'.Nb'}_Ka')

1b.   State  =1 /\ RCV({Na'.A}_Kb) /\ not(in(A.Ka',KeyRing))
 =|> State':=2 /\ SND(B.A)

2.    State  =2 /\ RCV({A.Ka'}_inv(Ks))
 =|> State':=3 /\ Nb':=new() /\ SND({Na.Nb'}_Ka')
               /\ KeyRing':=cons(A.Ka',KeyRing)

3.    State  =3 /\ RCV({Nb}_inv(Kb))
 =|> State':=4
```

Composed Roles

```
role session(A,B: agent,
             Ka,Kb,Ks: public_key,
             KeyRings: agent -> (agent.public_key) set) def=

   local SND,RCV: channel(dy)

   composition
         alice(A,B,Ka,Ks,KeyRings(A),SND,RCV)
      /\   bob(A,B,Kb,Ks,KeyRings(B),SND,RCV)

end role
```

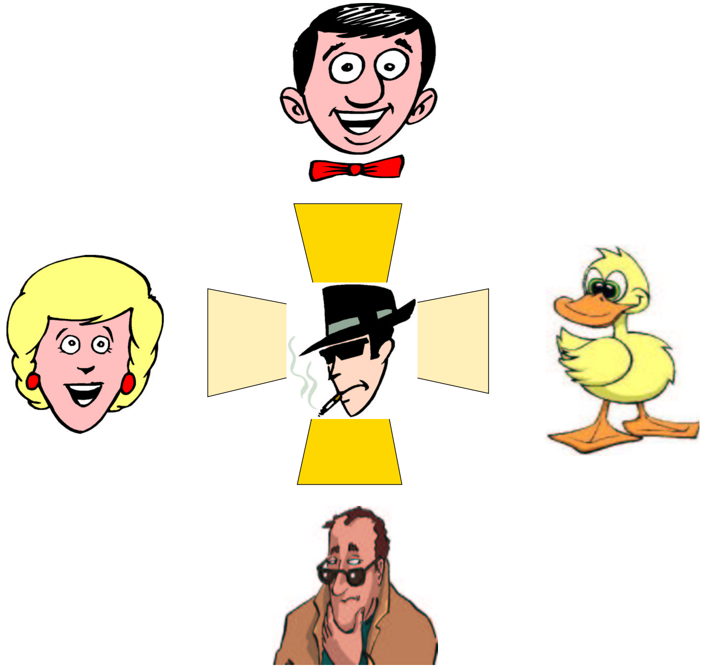
Global Environment

```
role environment() def=
   local KeyRings: agent -> (agent.public_key) set,
         KeyRing: (agent.public_key) set,
         SND,RCV: channel(dy)
   const a,b,s,i: agent,
         ka,kb,ks,ki: public_key
   init  KeyRings:={(a.{}), (b.{a.ka}), (i.{a.ka,b.kb})}
      /\ KeyRing :={a.ka,b.kb,s.ks,i.ki}
   intruder_knowledge={a,b,s,i,ka,kb,ks,ki,inv(ki)}
   composition
         server(s,ks,KeyRing,SND,RCV)
      /\ session(a,b,ka,kb,ks,KeyRings)
      /\ session(i,b,ki,kb,ks,KeyRings)
      /\ session(a,i,ka,ki,ks,KeyRings)
end role
```

Dolev-Yao Intruder Model

Intruder has full control over the network — he *is* the network:

- all messages sent by principals go to the intruder
- operations the intruder can do on messages:
  - forward, replay, suppress
  - decompose and analyze (if keys known)
  - modify, synthesize
  - send anywhere
- intruder cannot break cryptography
- intruder may play role(s) of (normal) principals
- intruder gains knowlege of compromised principals



HLPSL Type System

- Basic types available for specifying protocols:
  - agent, channel, boolean, integer, text, message, public key, symmetric key

  Variables can be assigned with “fresh” values (using `new()`).

- Type constructors:
  - functions, tuples, sets.

- Compound types like `{text.bool}_public_key`
  - describe how terms are constructed
  - allow search space optimizations

Declaring Goals

Three basic properties can be considered:

- secrecy
- weak authentication
- strong authentication (with replay protection)

```
goal
   secrecy_of na, nb
   authentication_on alice_bob_nb
   authentication_on bob_alice_na
end goal
```

Specifying Goal Facts

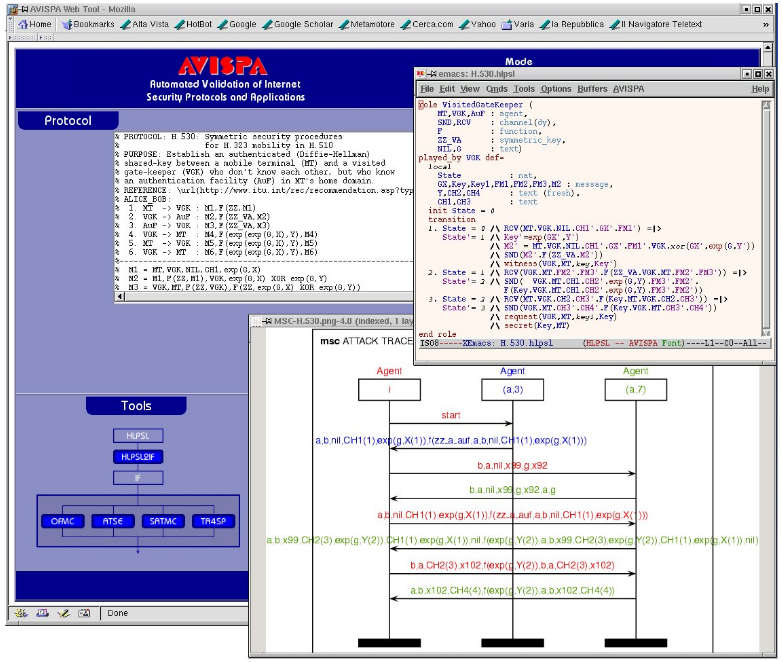
```
role bob...
   1a. State  =1 /\ RCV({Na'.A}_Kb) /\ in(A.Ka',KeyRing)
   =|> State':=3 /\ Nb':=new() /\ SND({Na'.Nb'}_Ka')
       /\ secret(Nb',nb,{A,B})
       /\ witness(B,A,alice_bob_nb,Nb')
       ...
end role

role alice...
   3.  State  =3 /\ RCV({Na.Nb'}_Ka)
   =|> State':=4 /\ SND({Nb'}_Kb)
       /\ request(A,B,alice_bob_nb,Nb')
end role
```

Properties of HLPSL

- easy to learn and read
- non-ambiguous semantics (in terms of TLA)
- strongly typed
- expressive, supporting...
  - modularity: composition, hiding
  - control flow
  - explicit intruder knowledge
  - cryptographic primitives: nonces, hashes, signatures
  - algebraic properties, e.g. of `xor` and `exp`

Industry Impact: Present and Future



- HLPSL and the AVISPA Tool:
  industrial strength, engineered for usability.
  - E.g., GUI, comprehensive documentation, user mailing list.
- Dissemination in industry forums:
  - talks and demos (e.g. at IETF)
  - patents/RFCs for improved protocols
  - AVISPA Library as a template and benchmark suite
- Technology Migration:
  - over 70 downloads of the AVISPA Tool from http://www.avispa-project.org/
  - over 50 subscribers to the users' mailing list.
- Active interest from industry and standardisation bodies (e.g. SIEMENS, SAP, IETF) in continued research and follow-up projects.