Automatic Detection of Attacks on Cryptographic Protocols: a Case Study

Ivan Cibrario Bertolotti[1], Luca Durante[1], Riccardo Sisto[2], Adriano Valenzano[1]

[1]IEIIT - CNR

[2]Dipartimento di Automatica e Informatica
Politecnico di Torino

Outline

- Introduction and motivation
- Spi calculus and S^3A
- The case study:
  - The Yahalom protocol and its variants
  - Analysis of the Yahalom protocols with S^3A
- Conclusions

Formal verification of cryptographic protocols

- Research in this area has recently made much progress:
  - Verification of more complex protocols
  - Verification under less restrictive assumptions
- Different techniques are now available.
  - They generally feature *complementary* strengths and weaknesses.

Aim of this paper

- Show the strengths of a new approach
  - Based on spi-calculus and testing equivalence
  - Theory presented in

    L. Durante, R. Sisto, A. Valenzano: "Automatic testing equivalence verification of spi calculus specifications", ACM Trans. Softw. Eng. Method. 12(2): 222-284 (2003)
  - Implemented by the prototype tool S^3A

- By a case study
  - Verification of several versions of the Yahalom protocol

Spi calculus

- Formal specification language for cryptographic protocols (Abadi, Gordon, 1998)
- W.r.t. other formalisms enables more precise and detailed descriptions
  - e.g. explicit description of decryptions and checks
- Being completely *untyped*, enables detection of all kinds of type flaw attacks.

Testing Equivalence

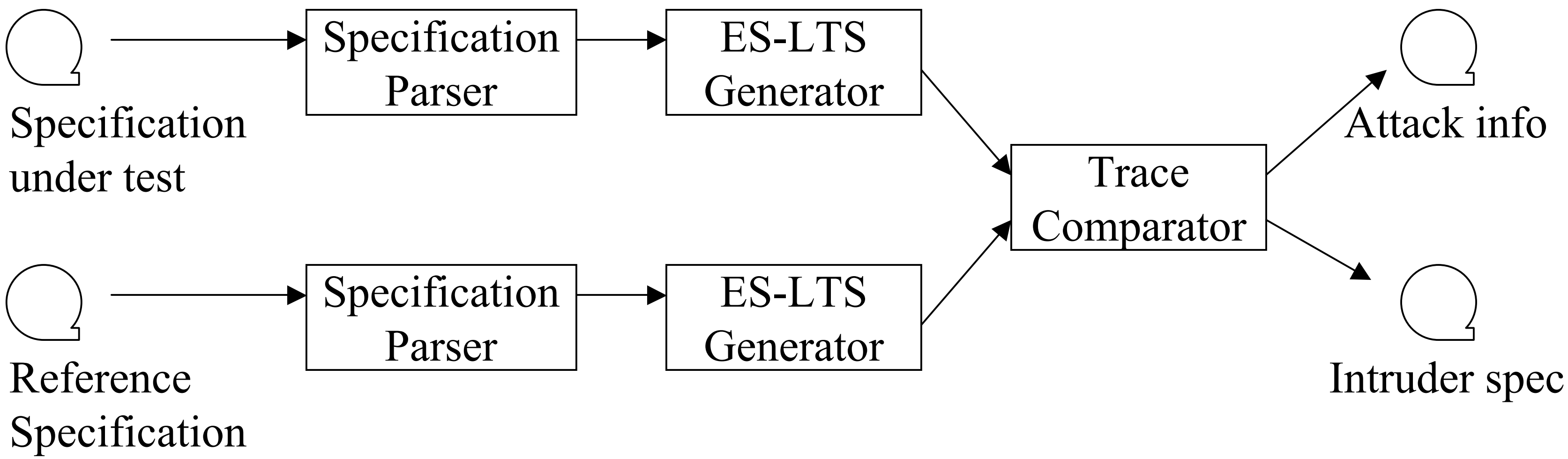
- Intuitive definition: two processes A and B are testing equivalent ($A \cong B$) if an external observer cannot distinguish them by testing
- Secrecy:

  $Inst(M) \cong Inst(M') \ \forall M, M'$
- Authenticity:

  $Inst(M) \cong Inst_{spec}(M) \ \forall M$

S^{3}A

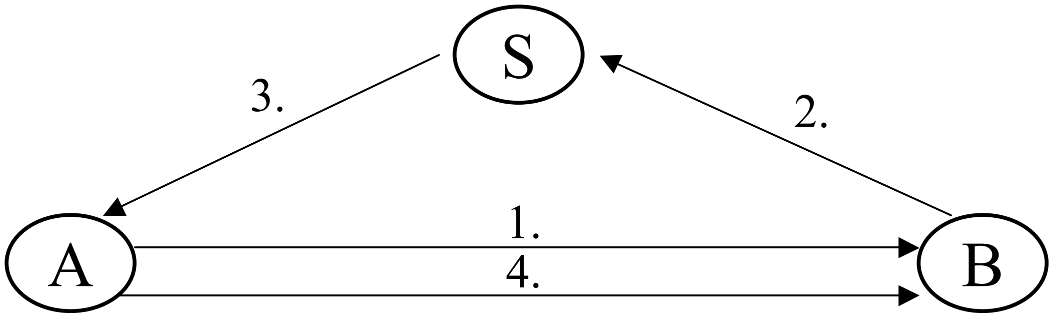
- Implements testing equivalence verification of spi calculus specifications by state space exploration



Main Features of S^3A

- Completely automatic check (push button)
- Symbolic representation of messages
  - No artificial restriction on message length and structure
  - No restriction on the possibility of finding out type-flaws
- Enhanced performance by reductions based on partial orders and symmetries

The Yahalom Protocol



1. $A \rightarrow B : A, n_A$
2. $B \rightarrow S : B, \{A, n_A, n_B\}_{KBS}$
3. $S \rightarrow A : \{B, KAB, n_A, n_B\}_{KAS}, \{A, KAB\}_{KBS}$
4. $A \rightarrow B : \{A, KAB\}_{KBS}, \{n_B\}_{KAB}$

The Server specification in spi-calculus

server(I, R, KIS, KRS) =

c(xR, x).

[xR is R]

case x of $\{xI, xnI, xnR\}_{KRS}$ in

[xI is I]

$(\nu\ KIR)\ (\bar{c}\ \langle \{xR, KIR, xnI, xnR\}_{KIS}, \{xI, KIR\}_{KRS} \rangle.\ 0)$

Analysis of a weakened version of the protocol: missing a check

server_weak(I, R, KIS, KRS) =

c(xR, x).

[xR is R]

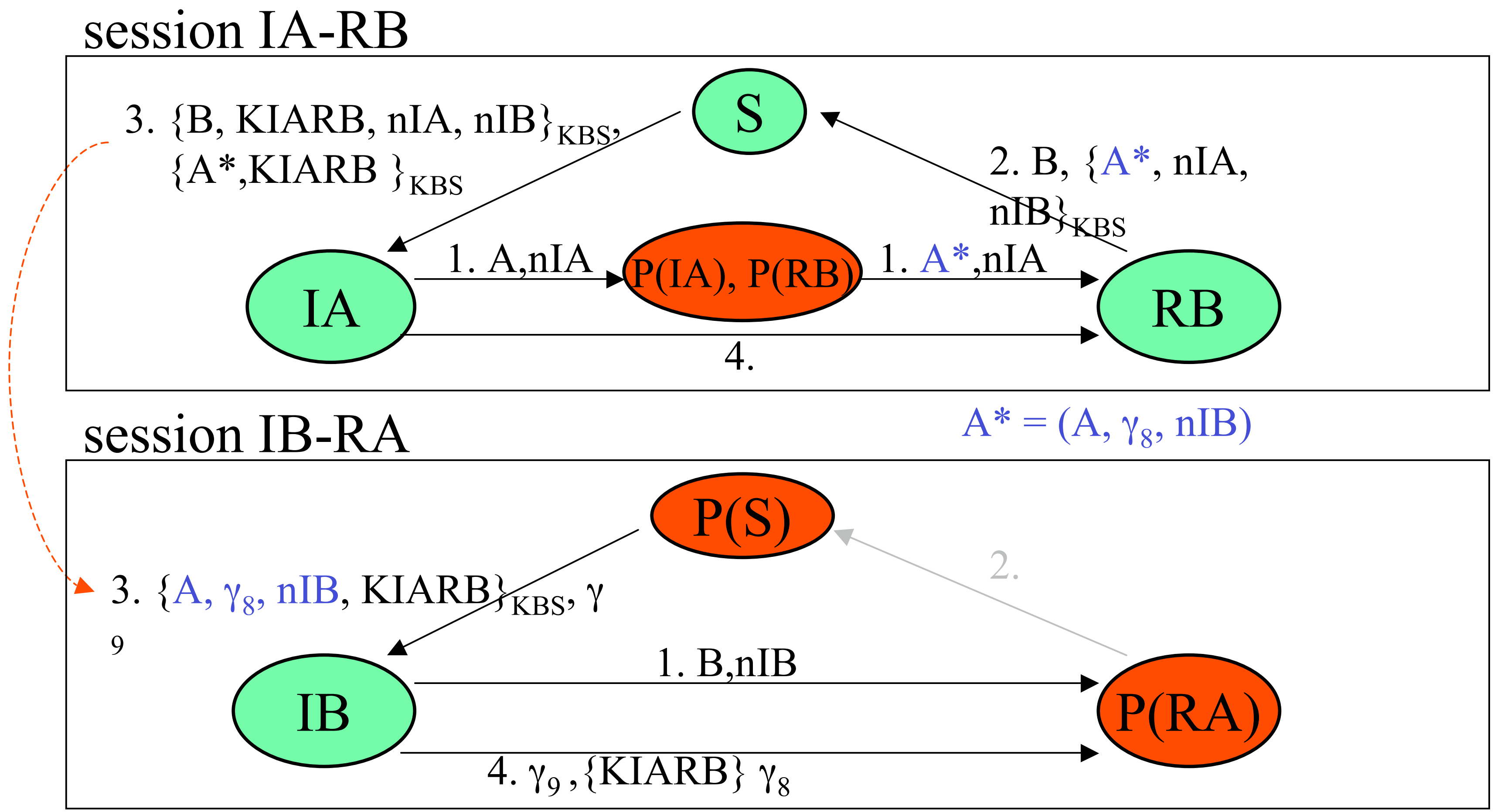
case x of $\{xI, xnI, xnR\}_{KRS}$ in

~~[xI is I]~~

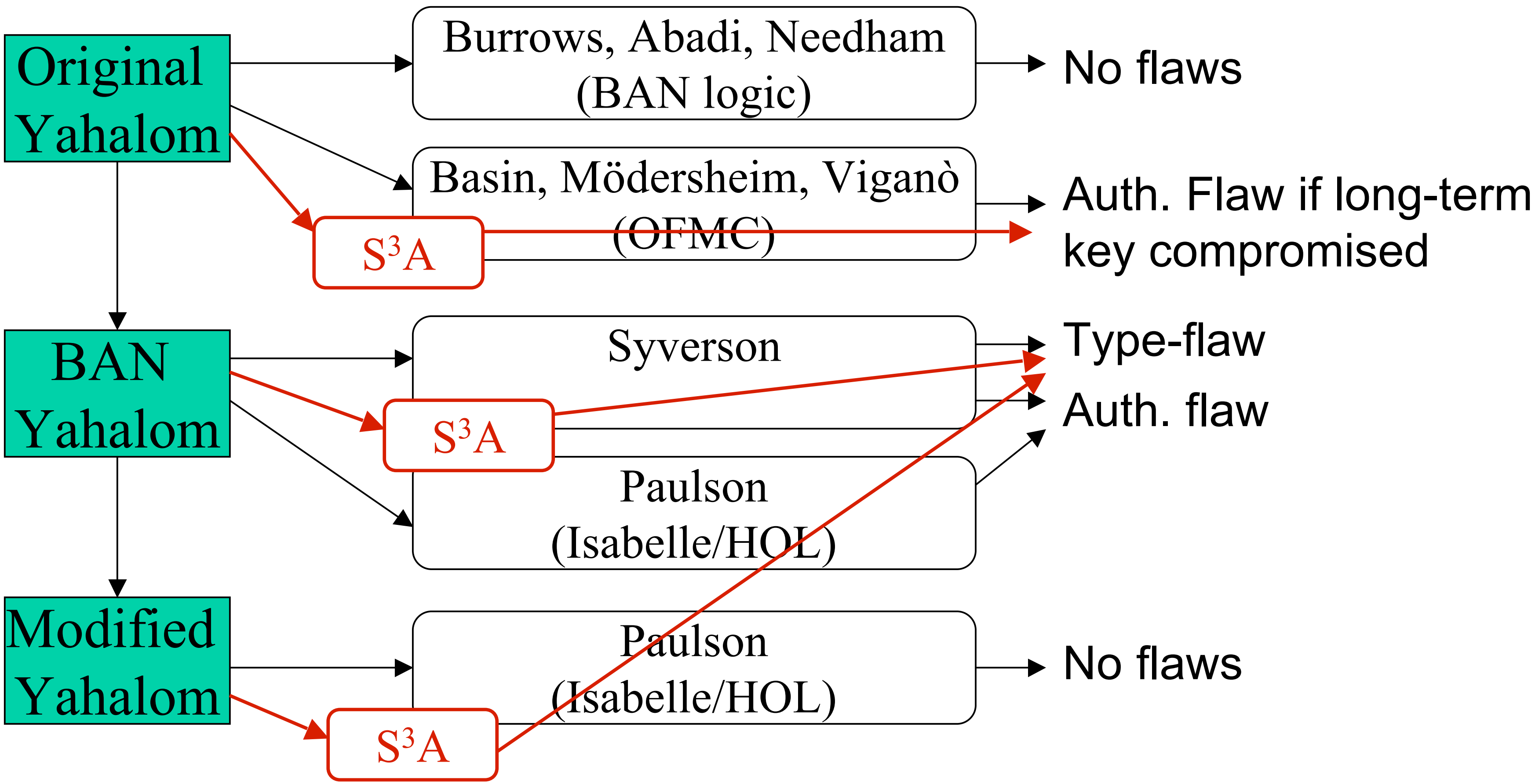
$(\nu\ KIR)\ (\bar{c}\ \langle\{xR, KIR, xnI, xnR\}_{KIS}, \{xI, KIR\}\ \}_{KRS}\rangle.$

0)

The Attack found by S^3A



Other results



Conclusions

- The verification method implemented by S^3A
  - lets automatically discover type flaw attacks, even if they are previously unknown and too complex to be found by hand
  - lets verify protocol versions with partial decode/check operations
- The performance of S^3A is comparable to the one of other state-of-the-art tools even if it performs more sophisticated checks

Conclusions (contd)

- Studying the Yahalom protocol with S^3A we found that
  - Modified Yahalom is affected by the same type-flaw attack that affects BAN-Yahalom

The Initiator Specification in spi-calculus

initiator(I, R, KIS) =

$(\nu\ n_I)\ (\overline{c}\ \langle I, n_I \rangle.$

$c\ (x, y).$

case x of $\{xR, xKIR, xnI, xnR\}_{KIS}$ in

$[xR \text{ is } R]\ [xnI \text{ is } n_I]$

$c\ \langle y, \{xnR\}_{xKIR} \rangle.$

0)