Automated Validation of Internet Security Protocols

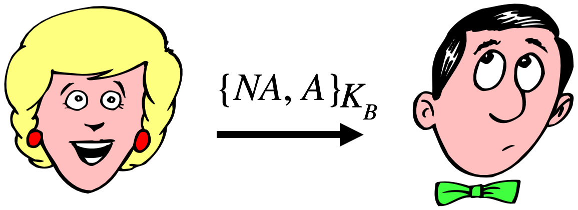
Luca Viganò
The AVISPA Project

Motivation

- 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 the next generation of security protocols and to improve their security, it is of utmost importance 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.

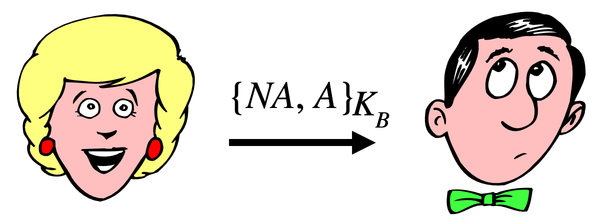
The state of the art... "yesterday"

- Several semi-automated tools have been developed to analyze protocols under the perfect cryptography assumption, but (in most cases) they are limited to small and medium-scale protocols.
  - For example, Clark/Jacob protocol library: NSPK, NSSK, Otway-Rees, Yahalom, Woo-Lam, Denning-Sacco, ...

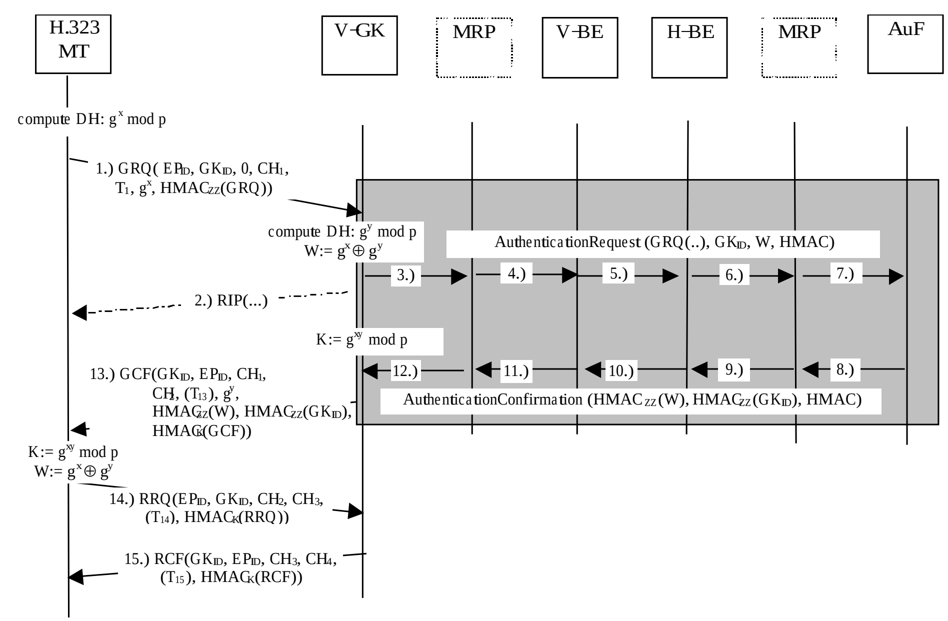


The state of the art... "yesterday"

- Several semi-automated tools have been developed to analyze protocols under the perfect cryptography assumption, but (in most cases) they are limited to small and medium-scale protocols.
  - For example, Clark/Jacob protocol library: NSPK, NSSK, Otway-Rees, Yahalom, Woo-Lam, Denning-Sacco, ...



  - Scaling up to large-scale Internet security protocols is a considerable scientific and technological challenge.



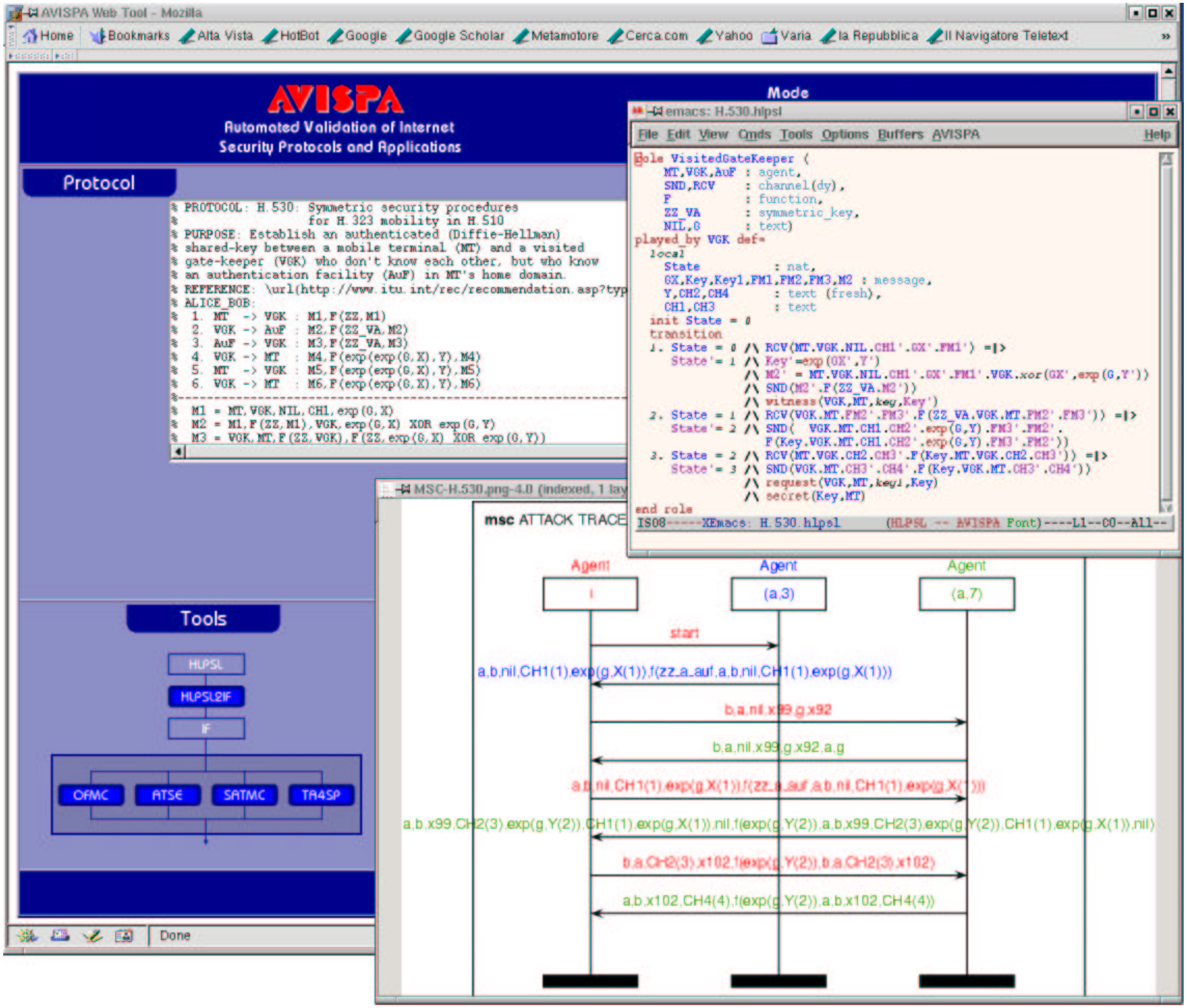
The state of the art... today and tomorrow

- Some tools (AVISPA, ProVerif, Casper/FDR, Scyther, NRL, ...) are taking up this challenge and
  - developing languages for specifying industrial-scale security protocols and their properties,
  - advancing analysis techniques to scale up to this complexity.
- These technologies are migrating to companies and standardization organizations.
- Also: extensions to
  - even more complex protocols and properties (group protocols, broadcast, ad-hoc networks, emerging properties, etc.)
  - Web Services,
  - and so on.

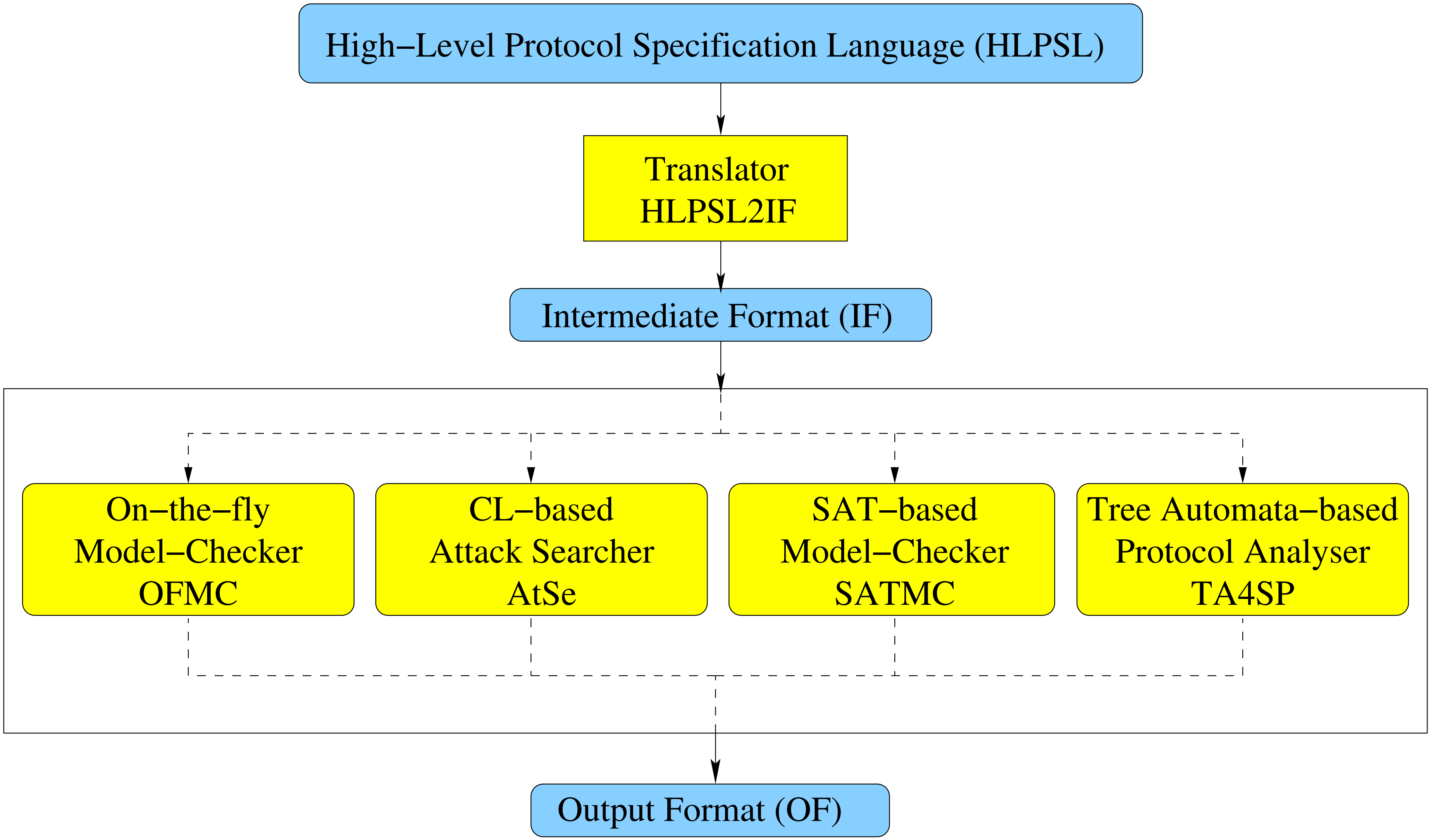
The AVISPA Tool

- A push-button integrated tool supporting the protocol designer in the debugging and validation of protocols.
  - Provides a role-based (& TLA-based) specification language for security protocols, properties, channels and intruder models.
  - Integrates different back-ends implementing a variety of state-of-the-art automatic analysis techniques.
- Assessed on a large collection of practically relevant, industrial protocols (the AVISPA Library).
- Large user base (the AVISPA users mailing list).

The Web Interface www.avispa-project.org



The AVISPA Tool: architecture



The AVISPA Tool: the back-ends

From protocol falsification to abstraction-based verification.

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

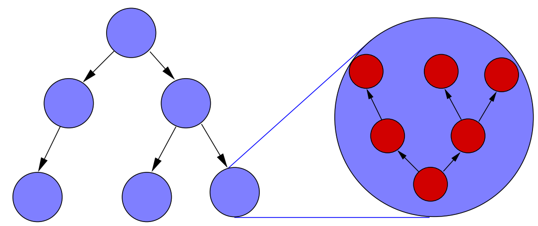
CL-AtSe (Constraint-Logic-based Attack Searcher) applies constraint solving with simplification heuristics and redundancy elimination techniques.

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

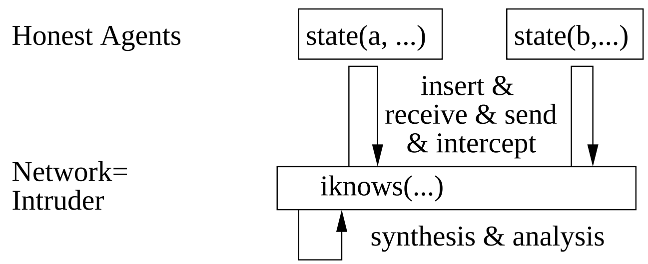
TA4SP (Tree Automata based on Automatic Approximations for the Analysis of Security Protocols) approximates the intruder knowledge by using regular tree languages and rewriting to produce under and over approximations.

Graphical overview of some symbolic reductions

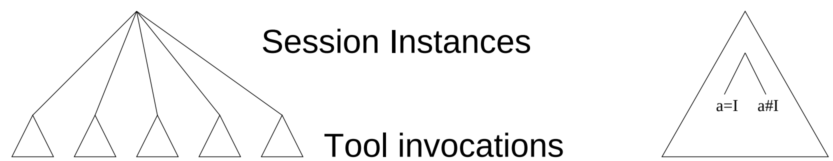
- The Lazy Intruder



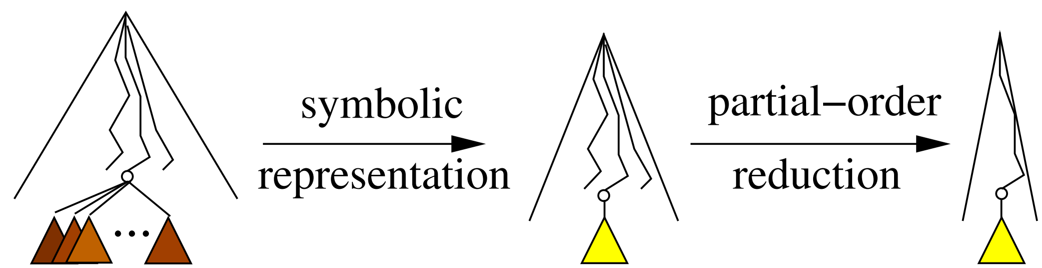
- Compressions



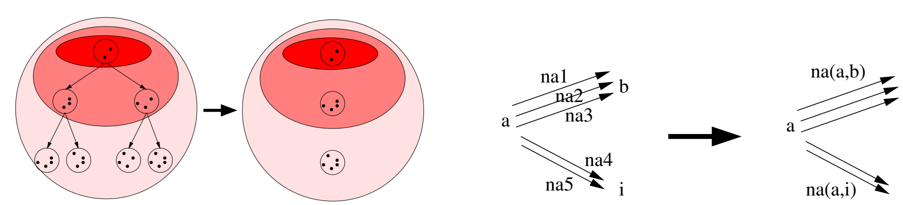
- Symbolic Sessions



- Constraint Differentiation



- Abstractions (data and control)



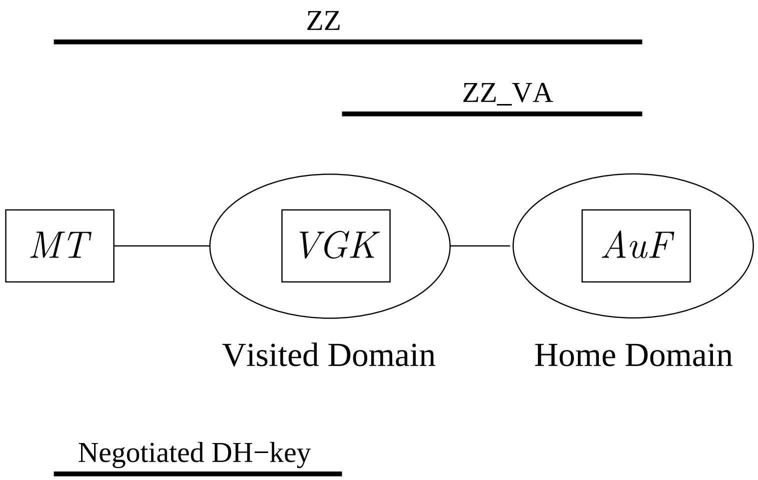
The AVISPA Tool and the AVISPA Library: Results

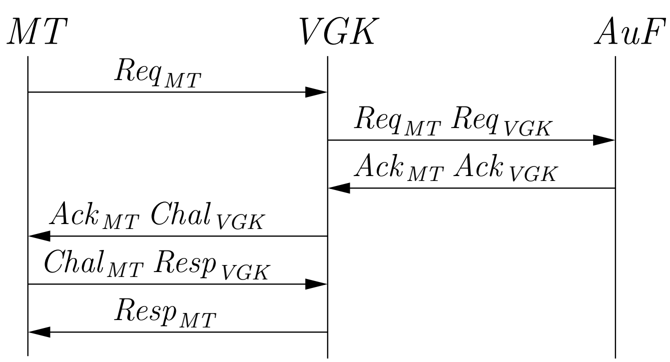
- Beyond Clark/Jacob (few seconds for entire library, with new attacks).
- A library of 384 problems from 79 protocols that have recently been or are currently being standardized by the IETF (problem = protocol + property).
- Analysis:
  - 215 problems in 87 min.
  - Several new attacks (e.g. H.530 protocol).

Problems		OFMC			CL-atse			SATMC			
Protocol	#P	P	A	T	P	A	T	P	A	TE	TS
UMTS_AKA	3	3	0	0,02	3	0	0,01	3	0	0,11	0,00
AAAMobileIP	7	7	0	0,75	7	0	0,20	7	0	1,32	0,01
ISO-PK1	1	1	1	0,02	1	1	0,00	1	1	0,05	0,00
ISO-PK2	1	1	0	0,05	1	0	0,00	1	0	1,62	0,00
ISO-PK3	2	2	2	0,04	2	2	0,01	2	2	0,27	0,00
ISO-PK4	2	2	0	0,54	2	0	0,03	2	0	1.153	1,16
LPD-MSR	2	2	2	0,02	2	2	0,02	2	2	0,17	0,02
LPD-IMSR	2	2	0	0,08	2	0	0,01	2	0	0,43	0,01
CHAPv2	3	3	0	0,32	3	0	0,01	3	0	0,55	0,00
EKE	3	3	2	0,19	3	2	0,04	3	2	0,22	0,00
TLS	3	3	0	2,20	3	0	0,32	3	0	-	0,00
DHCP-delayed	2	2	0	0,07	2	0	0,00	2	0	0,19	0,00
Kerb-Cross-Realm	8	8	0	11,86	8	0	4,14	8	0	113,60	1,69
Kerb-Ticket-Cache	6	6	0	2,43	6	0	0,38	6	0	495,66	7,75
Kerb-V	8	8	0	3,08	8	0	0,42	8	0	139,56	2,95
Kerb-Forwardable	6	6	0	30,34	6	0	10,89	0	0	-	-
Kerb-PKINIT	7	7	0	4,41	7	0	0,64	7	0	640,33	11,65
Kerb-preauth	7	7	0	1,86	7	0	0,62	7	0	373,72	2,57
CRAM-MD5	2	2	0	0,71	2	0	0,74	2	0	0,40	0,00
PKB	1	1	1	0,25	1	1	0,01	1	1	0,34	0,02
PKB-fix	2	2	0	4,06	2	0	44,25	2	0	0,86	0,02
SRP_siemens	3	3	0	2,86	0	0	-	0	0	-	-
EKE2	3	3	0	0,16	0	0	-	0	0	-	-
SPEKE	3	3	0	3,11	0	0	-	0	0	-	-
IKEv2-CHILD	3	3	0	1,19	0	0	-	0	0	-	-
IKEv2-DS	3	3	1	5,22	0	0	-	0	0	-	-
IKEv2-DSx	3	3	0	42,56	0	0	-	0	0	-	-
IKEv2-MAC	3	3	0	8,03	0	0	-	0	0	-	-
IKEv2-MACx	3	3	0	40,54	0	0	-	0	0	-	-
h.530	3	1	1	0,64	0	0	-	0	0	-	-
h.530-fix	3	3	0	4.278	0	0	-	0	0	-	-
lipkey-spkm-known	2	2	0	0,23	0	0	-	0	0	-	-
lipkey-spkm-unknown	2	2	0	7,33	0	0	-	0	0	-	-

Also: TA4SP establishes in a few minutes that a number of protocols (EKE, EKE2, IKEv2-CHILD, IKEv2-MAC, TLS, UMTS_AKA, MS-ChapV2) guarantee secrecy.

An example: the H.530 Protocol






```
1. MT   -> VGK : MT,VGK,NIL,CH1,G^DHX,
                 F(ZZ,MT,VGK,NIL,CH1,G^DHX)
2. VGK -> AuF  : MT,VGK,NIL,CH1,G^DHX,
                 F(ZZ,MT,VGK,NIL,CH1,G^DHX),
                 VGK,G^DHX XOR G^DHY,
                 F(ZZ_VA,MT,VGK,NIL,CH1,G^DHX,
                   F(ZZ,MT,VGK,NIL,CH1,G^DHX),
                   VGK,G^DHX XOR G^DHY)
3. AuF -> VGK  : VGK,MT,F(ZZ,VGK),
                 F(ZZ,G^DHX XOR G^DHY),
                 F(ZZ_VA,VGK,MT,F(ZZ,VGK),
                   F(ZZ,G^DHX XOR G^DHY))
4. VGK -> MT   : VGK,MT,CH1,CH2,G^DHY,
                 F(ZZ,G^DHX XOR G^DHY),
                 F(ZZ,VGK),
                 F((G^DHX)^DHY,VGK,MT,CH1,CH2,G^DHY,
                   F(ZZ,G^DHX XOR G^DHY),F(ZZ,VGK))
5. MT -> VGK   : MT,VGK,CH2,CH3,
                 F((G^DHX)^DHY,MT,VGK,CH2,CH3)
6. VGK -> MT   : VGK,MT,CH3,CH4,
                 F((G^DHX)^DHY,VGK,MT,CH3,CH4)
```

Protocol proposed (and patented) by Siemens. Modeling time, ca. 1 day. Analysis time, ca. 1 second. New patent filed, ca. 1 year.

Summary: the present and the future

- AVISPA package (& web-interface): www.avispa-project.org
- Current work:
  - Extending the AVISPA library with further protocols and properties.
  - Unbounded verification using abstractions.
  - Algebraic properties.
  - Guessing intruder and other intruder models (and channels).
  - Web-services.
  - Combining cryptographic and formal proof techniques.
- Integration of other tools via HLPSL/IF (e.g. translator from HLPSL to Applied Pi Calculus to then apply ProVerif).
- A Security Protocol Animator Tool.

Road map

- Motivation.
- The AVISPA Tool.
- ☞ **OFMC in more detail.**
- Algebraic properties.
- Conclusions and outlook.

Formal analysis of security protocols

- Challenging as general problem is undecidable.
- Several sources of infinity in protocol analysis:
  - Unbounded number of possible intruder messages (unbounded message depth).
  - Unbounded number of sessions or protocol steps (and agents).
- Possible approaches:
  - Falsification identifies attack traces but does not guarantee correctness.
  - Verification proves correctness but is difficult to automate (requires induction and often restrictions).
- Symbolic techniques to reduce the search space without excluding or introducing attacks.

Two key challenges and their solutions

Two key challenges of model-checking security protocols:

1. The prolific Dolev-Yao intruder model.

2. Concurrency: number of parallel sessions executed by honest agents.

Two key challenges and their solutions

Two key challenges of model-checking security protocols:

1. The prolific Dolev-Yao intruder model.
    - No bound on the messages the intruder can compose.
    - Lazy Intruder: symbolic representation of intruder.
      "Often just as if there were no intruder!"

2. Concurrency: number of parallel sessions executed by honest agents.

Road map

- Motivation.
- The AVISPA Tool.

☞ **OFMC in more detail.**

  - Lazy Intruder.
  - Constraint Differentiation.

- Algebraic properties.
- Conclusions and outlook.

Protocol model

- Protocol modeled as an infinite-state transition system.
  - States: local states of honest agents and current knowledge of the intruder.
  - Transitions: actions of the honest agents and the intruder.
- The Dolev-Yao intruder:
  - Controls the entire network.
  - Perfect cryptography.
  - Unbounded composition of messages.
- Security properties: attack predicates on states.
- Also: protocol-independent declarations (operator symbols, algebraic properties, intruder model,...)

Lazy Intruder: overview

- Many different approaches based on different formalisms, e.g.:
  - Process calculi (e.g. [Amadio & Lugiez], [Boreale & Buscemi])
  - Strand spaces (e.g. [Millen & Shmatikov], [Corin & Etalle])
  - Rewriting (e.g. [Chevalier & Vigneron], [BMV])
- But they all share the same basic ideas:
  - Avoid the naïve enumeration of possible messages the intruder can send.
  - Use variables and constraints for messages sent by the intruder.

The Lazy Intruder: idea

$$1. \quad A \rightarrow B : \; M, A, B, \{\!| NA, M, A, B |\!\}_{K_{AS}}$$

The Lazy Intruder: idea

$$1. \quad \mathrm{i}(A) \rightarrow B : \; M, A, B, \{\!| NA, M, A, B |\!\}_{K_{AS}}$$

The Lazy Intruder: idea

$$1. \quad i(A) \rightarrow B : \; M, A, B, \{\!| NA, M, A, B |\!\}_{K_{AS}}$$

Which concrete value is chosen for these parts makes a difference only later.

The Lazy Intruder: idea

$$1. \quad \mathrm{i}(A) \to B : \; M, A, B, \{|NA, M, A, B|\}_{K_{AS}}$$

Which concrete value is chosen for these parts makes a difference only later.

Idea: postpone this decision.

$$1. \quad \mathrm{i}(A) \to B : \; x_1, x_2, B, x_3 \qquad \mathit{from}(\{x_1, x_2, x_3\}, IK)$$

IK: current Intruder Knowledge

The Lazy Intruder: idea

$$1.\quad \mathrm{i}(A) \rightarrow B:\ M, A, B, \{\!|NA, M, A, B|\!\}_{K_{AS}}$$

Which concrete value is chosen for these parts makes a difference only later.

Idea: postpone this decision.

$$1.\quad \mathrm{i}(A) \rightarrow B:\ x_1, x_2, B, x_3 \qquad \textit{from}(\{x_1, x_2, x_3\}, IK)$$

IK: current Intruder Knowledge

from-constraints are evaluated in a demand-driven way,
hence lazy intruder.

The Lazy Intruder: formally

- Constraints of the lazy intruder:

$$\mathit{from}(T, IK)$$

- $[\![\mathit{from}(T, IK)]\!] = \{\sigma \mid \mathrm{ground}(T\sigma \cup IK\sigma) \wedge (T\sigma \subseteq \mathcal{DY}(IK\sigma))\}$

  where $\mathcal{DY}(IK)$ is the closure of IK under Dolev-Yao rules.

- Semantics hence relates *from*-constraints to the Dolev-Yao model.

The Lazy Intruder: formally

- Constraints of the lazy intruder:

$$\mathit{from}(T, IK)$$

- $[\![\mathit{from}(T, IK)]\!] = \{\sigma \mid \mathrm{ground}(T\sigma \cup IK\sigma) \wedge (T\sigma \subseteq \mathcal{DY}(IK\sigma))\}$

  where $\mathcal{DY}(IK)$ is the closure of IK under Dolev-Yao rules.

- Semantics hence relates *from*-constraints to the Dolev-Yao model.

- **Theorem.** Satisfiability of (well-formed) *from*-constraints is decidable.

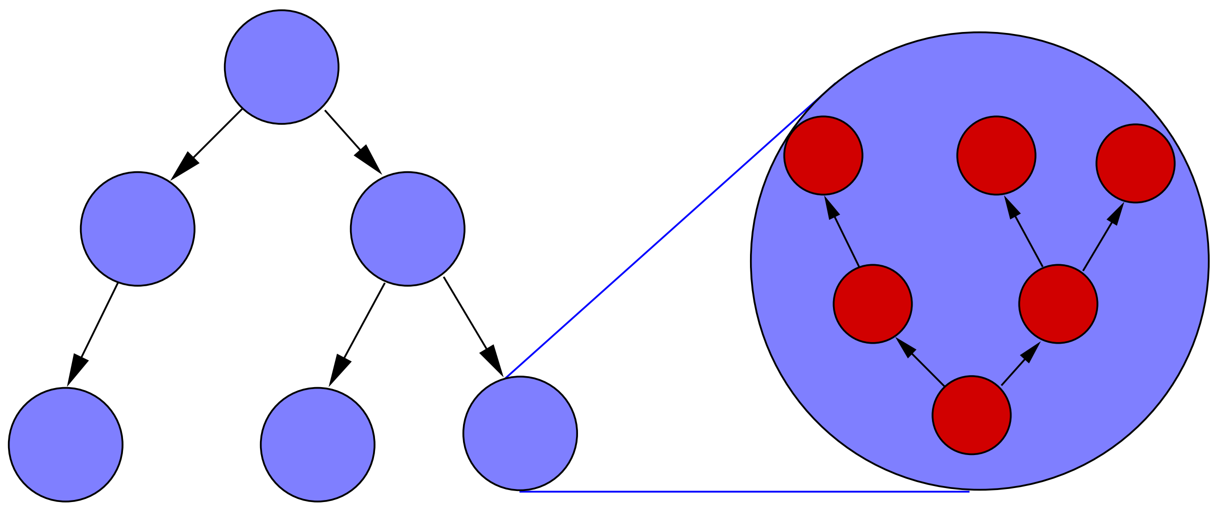
- A restriction on the depth of messages is not necessary.

- Non-atomic keys can easily be handled.

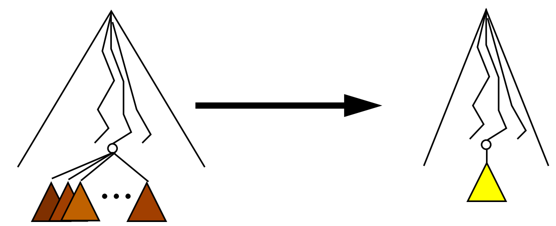
Integration: symbolic transition system

- Symbolic state = term with variables + constraint set
- $[\![(t, C)]\!] = \{t\sigma \mid \sigma \in [\![C]\!]\}$ (a set of ground states).
- Two layers of search:

  Layer 1: search in the symbolic state space
  Layer 2: constraint reduction



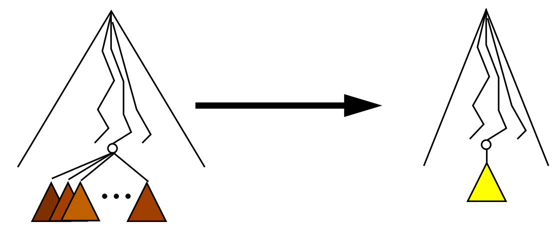
NSPK and the Lazy Intruder



We allow messages to contain variables and employ unification.

$$A \rightarrow B : \ \{NA, A\}_{K_B}$$
$$B \rightarrow A : \ \{NA, NB\}_{K_A}$$
$$A \rightarrow B : \ \{NB\}_{K_B}$$

NSPK and the Lazy Intruder

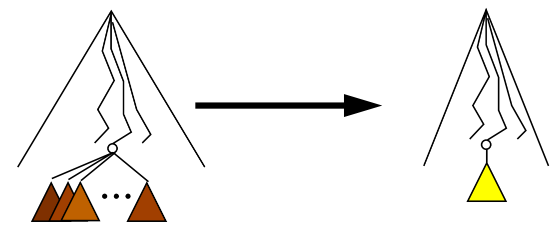


We allow messages to contain variables and employ unification.

$a \to \mathrm{I} : \{na, a\}_{K_\mathrm{I}}$

$$\begin{aligned} A \to B &: \{NA, A\}_{K_B} \\ B \to A &: \{NA, NB\}_{K_A} \\ A \to B &: \{NB\}_{K_B} \end{aligned}$$

NSPK and the Lazy Intruder

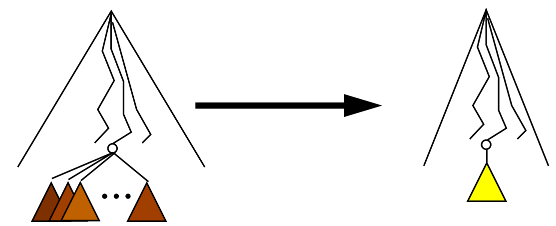


We allow messages to contain variables and employ unification.

$$a \to \mathrm{I} : \{na, a\}_{K_\mathrm{I}}$$
$$\mathrm{I} \to b : X_1$$

$$A \to B : \quad \{NA, A\}_{K_B}$$
$$B \to A : \quad \{NA, NB\}_{K_A}$$
$$A \to B : \quad \{NB\}_{K_B}$$

NSPK and the Lazy Intruder



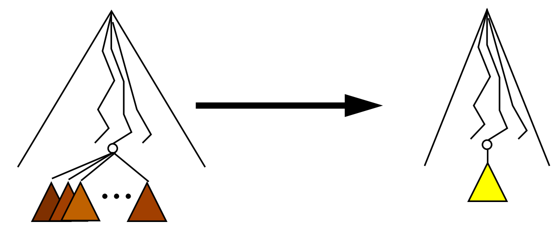
We allow messages to contain variables and employ unification.

$a \longrightarrow \mathrm{I} : \{na, a\}_{K_{\mathrm{I}}}$
$\mathrm{I} \longrightarrow b : X_1$
$b \longrightarrow \mathrm{I} : \{X_2, nb\}_{K_{X_3}}$

$X_1 = \{X_2, X_3\}_{K_b}$

$A \rightarrow B : \ \{NA, A\}_{K_B}$
$B \rightarrow A : \ \{NA, NB\}_{K_A}$
$A \rightarrow B : \ \{NB\}_{K_B}$

NSPK and the Lazy Intruder



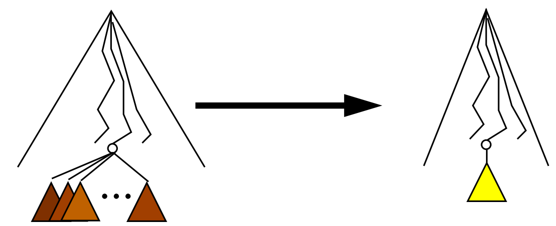
We allow messages to contain variables and employ unification.

$$a \rightarrow \mathrm{I} : \{na, a\}_{K_{\mathrm{I}}}$$
$$\mathrm{I} \rightarrow b : \{X_2, X_3\}_{K_b}$$
$$b \rightarrow \mathrm{I} : \{X_2, nb\}_{K_{X_3}}$$
$$\mathrm{I} \rightarrow a : \{X_2, nb\}_{K_{X_3}}$$

$$X_1 = \{X_2, X_3\}_{K_b}$$

$$A \rightarrow B : \{NA, A\}_{K_B}$$
$$B \rightarrow A : \{NA, NB\}_{K_A}$$
$$A \rightarrow B : \{NB\}_{K_B}$$

NSPK and the Lazy Intruder

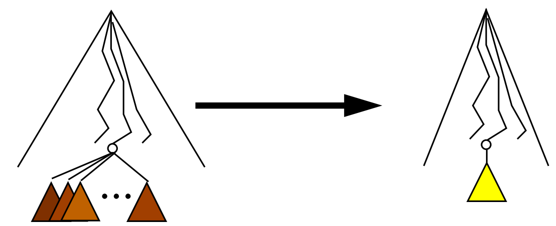


We allow messages to contain variables and employ unification.

$$a \longrightarrow \mathrm{I} : \{na, a\}_{K_{\mathrm{I}}}$$
$$\mathrm{I} \longrightarrow b : \{X_2, X_3\}_{K_b} \qquad X_1 = \{na, a\}_{K_b}$$
$$b \longrightarrow \mathrm{I} : \{X_2, nb\}_{K_{X_3}}$$
$$\mathrm{I} \longrightarrow a : \{X_2, nb\}_{K_{X_3}}$$
$$a \longrightarrow \mathrm{I} : \{nb\}_{K_{\mathrm{I}}} \qquad X_2 = na,\ X_3 = a$$

$$A \rightarrow B : \quad \{NA, A\}_{K_B}$$
$$B \rightarrow A : \quad \{NA, NB\}_{K_A}$$
$$A \rightarrow B : \quad \{NB\}_{K_B}$$

NSPK and the Lazy Intruder



We allow messages to contain variables and employ unification.

$a \rightarrow \mathrm{I} : \{na, a\}_{K_\mathrm{I}}$

$\mathrm{I} \rightarrow b : \{na, a\}_{K_b}$ $\quad X_1 = \{na, a\}_{K_b}$

$b \rightarrow \mathrm{I} : \{na, nb\}_{K_a}$

$\mathrm{I} \rightarrow a : \{na, nb\}_{K_a}$

$a \rightarrow \mathrm{I} : \{nb\}_{K_\mathrm{I}}$ $\quad X_2 = na, \; X_3 = a$

$\mathrm{I} \rightarrow b : \{nb\}_{K_\mathrm{b}}$

$A \rightarrow B : \; \{NA, A\}_{K_B}$

$B \rightarrow A : \; \{NA, NB\}_{K_A}$

$A \rightarrow B : \; \{NB\}_{K_B}$

Road map

- Motivation.
- The AVISPA Tool.

☞ **OFMC in more detail.**

  - Lazy Intruder.
  - Constraint Differentiation.

- Algebraic properties.
- Conclusions and outlook.

Two key challenges and their solutions

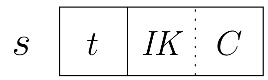
Two key challenges of model-checking security protocols:

1. The prolific Dolev-Yao intruder model.
   - No bound on the messages the intruder can compose.
   - Lazy Intruder: symbolic representation of intruder.
     "Often just as if there were no intruder!"

2. Concurrency: number of parallel sessions executed by honest agents.
   - Often addressed using Partial-Order Reduction (POR).
   - POR is limited when using the lazy intruder technique.
   - Constraint Differentiation: general, POR-inspired reduction technique extending the lazy intruder — correct and complete.

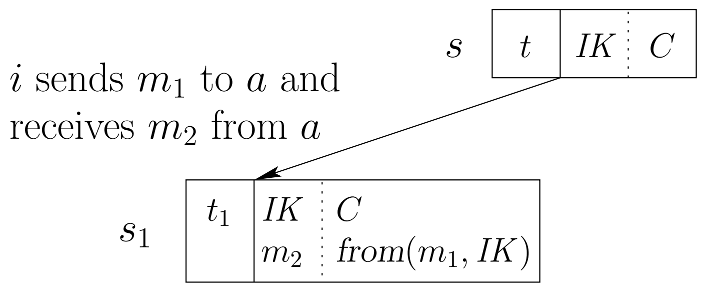
Constraint Differentiation: idea

Typical situation: 2 independent actions executable in either order:



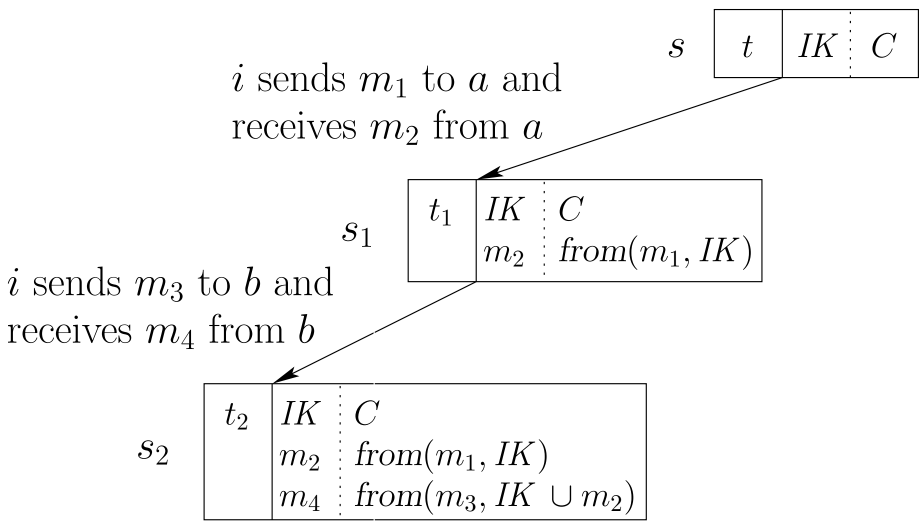
Constraint Differentiation: idea

Typical situation: 2 independent actions executable in either order:



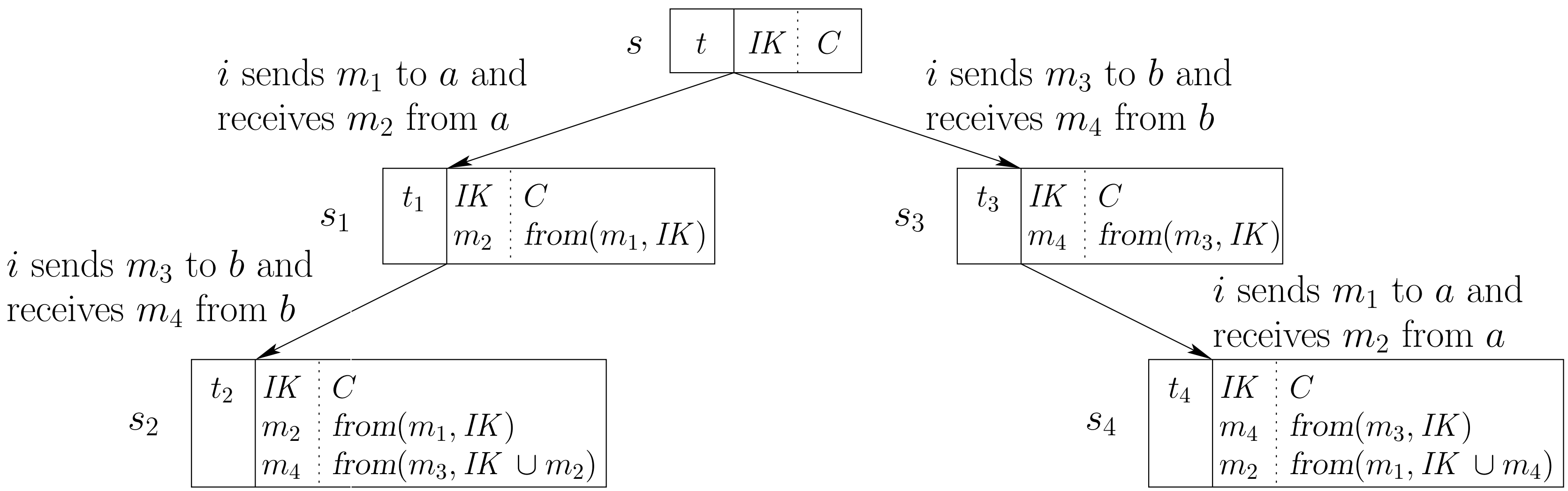
Constraint Differentiation: idea

Typical situation: 2 independent actions executable in either order:



Constraint Differentiation: idea

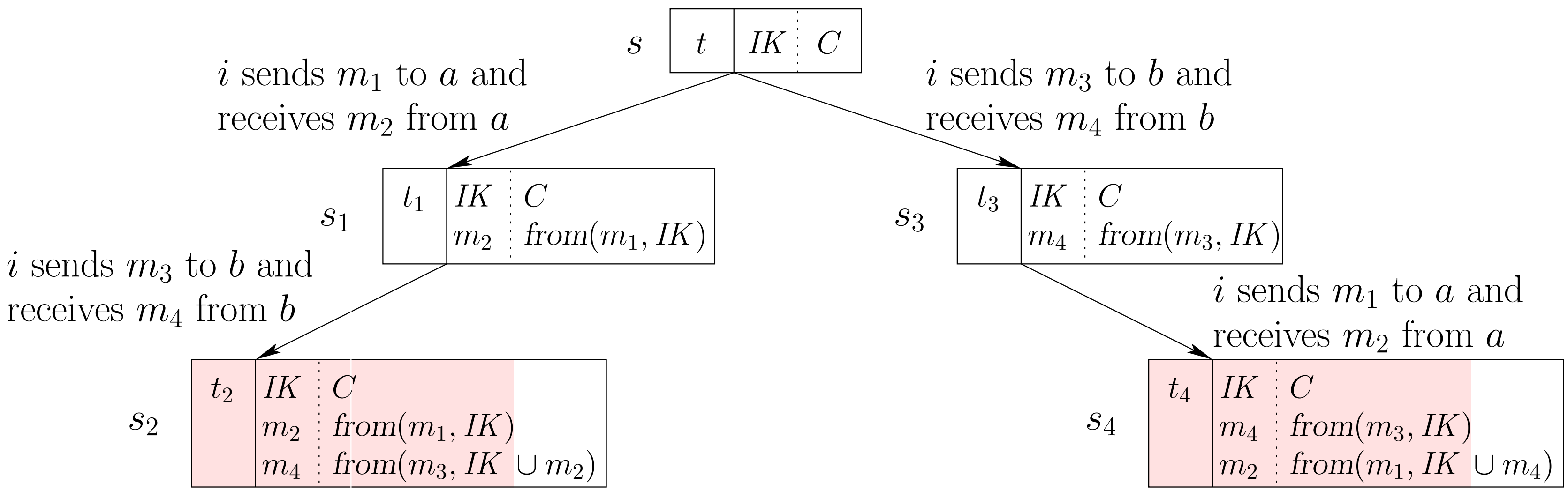
Typical situation: 2 independent actions executable in either order:



(where $t_2 = t_4$)

Constraint Differentiation: idea

Typical situation: 2 independent actions executable in either order:

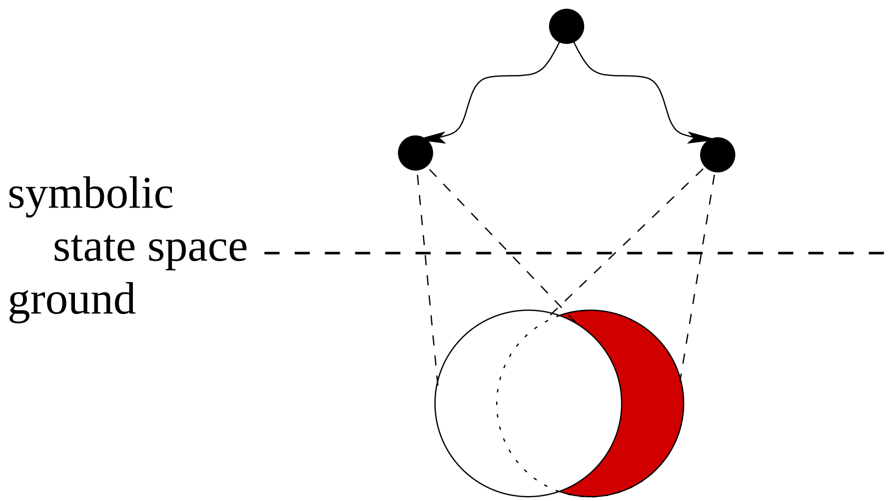


(where $t_2 = t_4$)

Idea: exploit redundancies in the symbolic states, i.e. reduction exploits overlapping of the sets of ground states.

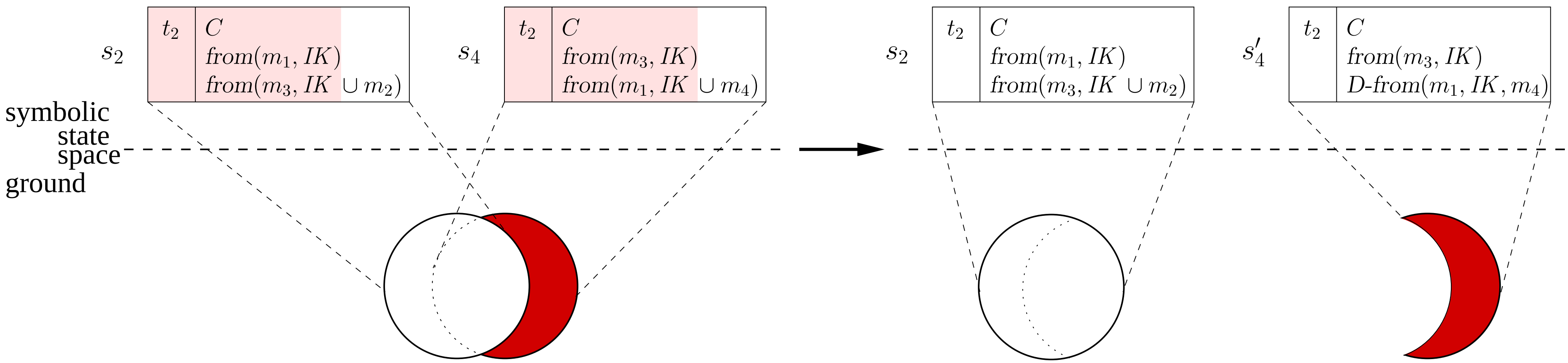
Constraint Differentiation: idea

Typical situation: 2 independent actions executable in either order:



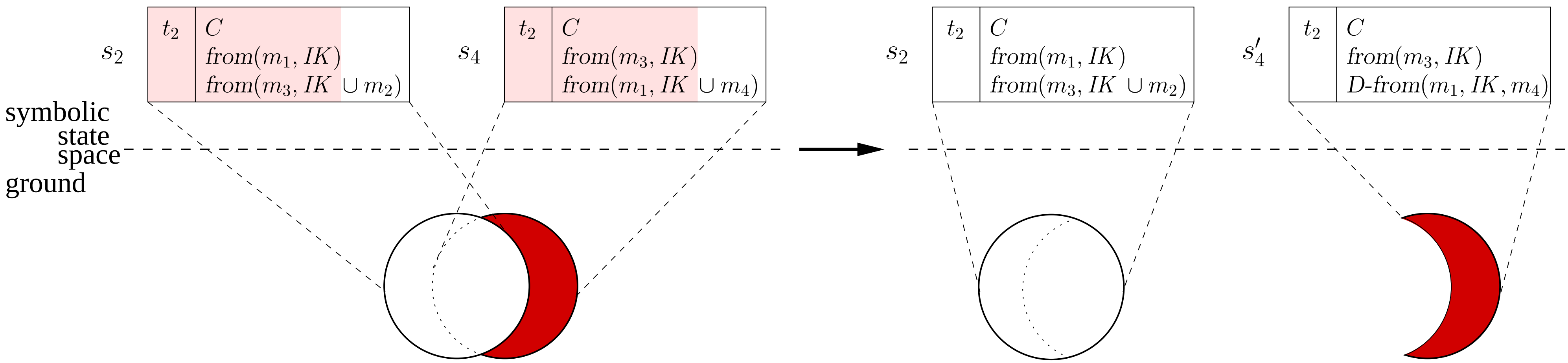
Idea: exploit redundancies in the symbolic states, i.e. reduction exploits overlapping of the sets of ground states.

Constraint Differentiation (1)



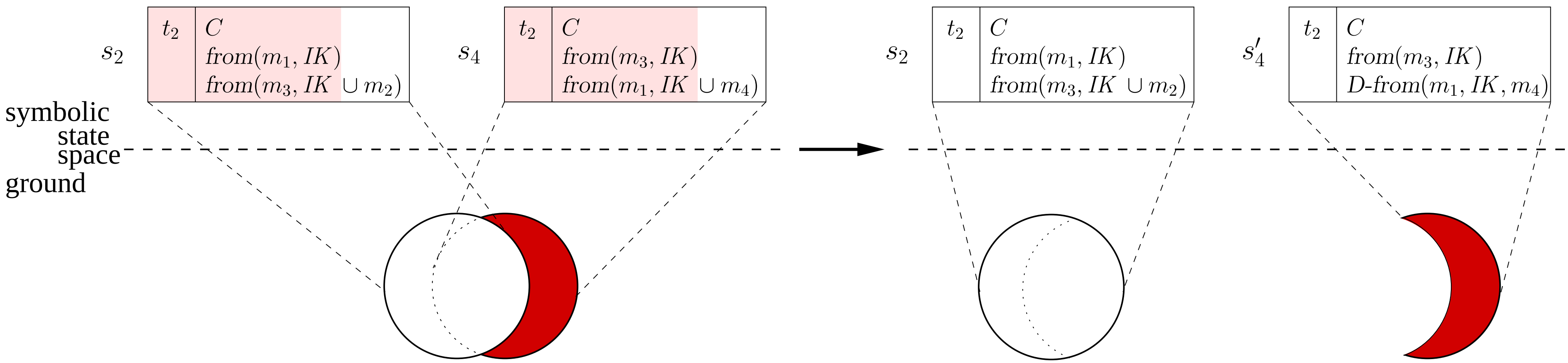
- New kind of constraints: $D\text{-}from(T, IK, NIK)$.
- Intuition:
  - Intruder has just learned some new intruder knowledge NIK.

Constraint Differentiation (1)



- New kind of constraints: $D\text{-}from(T, IK, NIK)$.
- Intuition:
  - Intruder has just learned some new intruder knowledge NIK.
  - All solutions $[\![from(T, IK \cup NIK)]\!]$ are "correct"

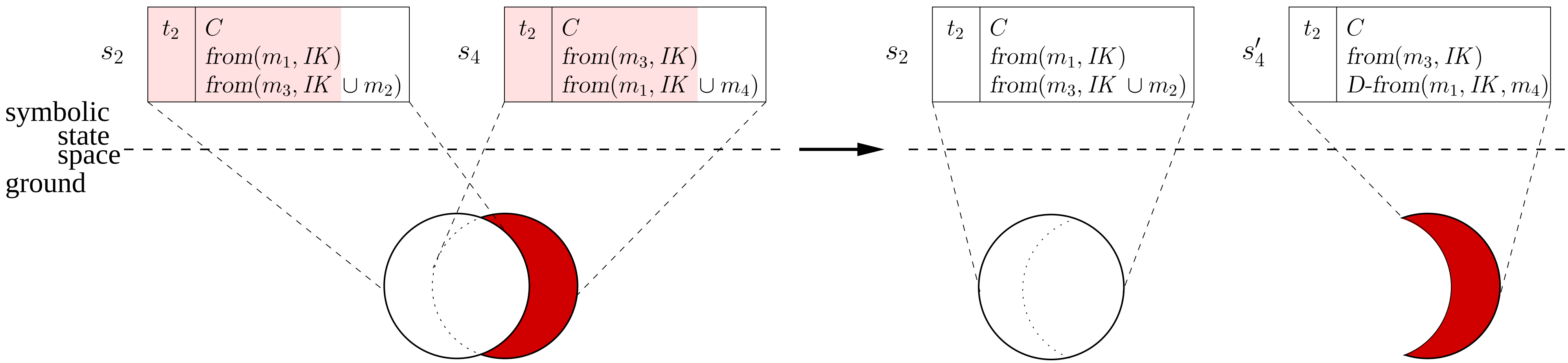
Constraint Differentiation (1)



- New kind of constraints: $\textit{D-from}(T, IK, NIK)$.
- Intuition:
  - ▶ Intruder has just learned some new intruder knowledge NIK.
  - ▶ All solutions $[\![\textit{from}(T, IK \cup NIK)]\!]$ are "correct" but a solution is interesting only if it requires NIK.

$$[\![\textit{D-from}(T, IK, NIK)]\!] = [\![\textit{from}(T, IK \cup NIK)]\!] \setminus [\![\textit{from}(T, IK)]\!].$$

Constraint Differentiation (2)



- $[\![\textit{D-from}(T, IK, NIK)]\!] = [\![\textit{from}(T, IK \cup NIK)]\!] \setminus [\![\textit{from}(T, IK)]\!]$

- **Theorem.** *Satisfiability of (well-formed) D-from constraints is decidable.*

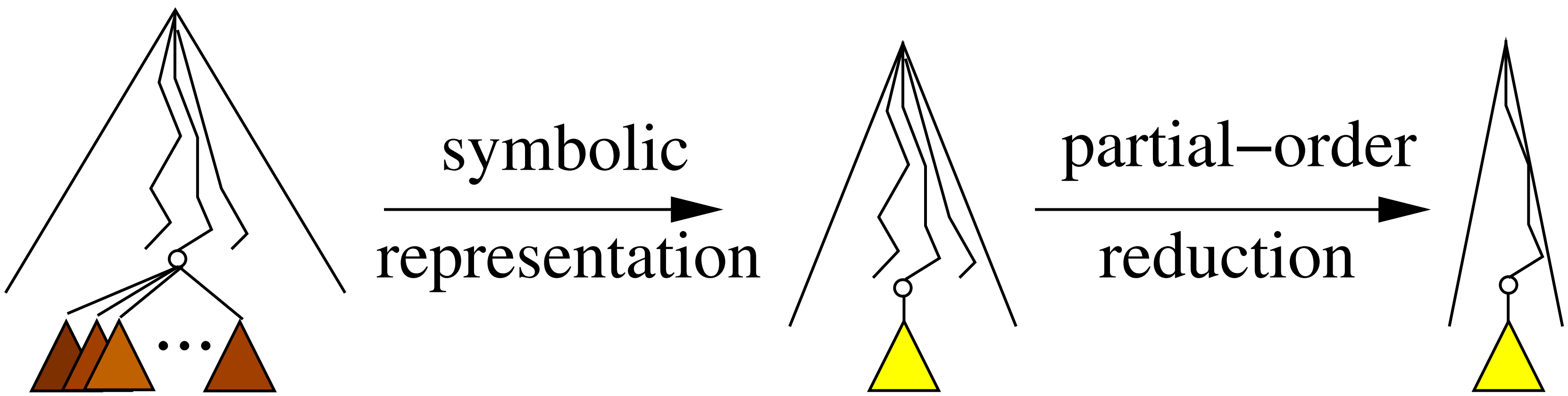
- **Theorem.** $[\![s_2]\!] \cup [\![s_4]\!] = [\![s_2]\!] \cup [\![s'_4]\!]$

Constraint Differentiation: experimental results

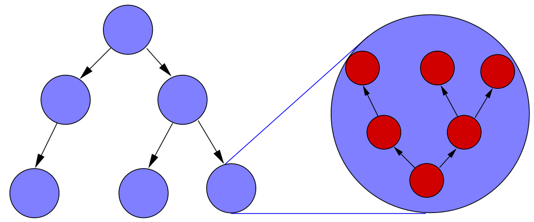
IKE Aggressive Mode Pre-Shared Key without and with CD: the nodes for each ply of the search tree and search time

IKE Aggressive Mode Pre-Shared Key												
Mode:	without CD						with CD					
Scenario:	$[a,b],[a,i]$		$[a,b],[a,i],[i,a]$		$[a,b],[a,i],[i,a],[b,i]$		$[a,b],[a,i]$		$[a,b],[a,i],[i,a]$		$[a,b],[a,i],[i,a],[b,i]$	
Ply	s1	s2	s1	s2	s1	s2	s1	s2	s1	s2	s1	s2
1	3	3	4	4	5	5	3	3	4	4	5	5
2	7	7	14	14	23	23	5	5	10	10	16	16
3	13	14	43	45	97	100	7	8	19	21	40	43
4	17	27	112	139	368	420	6	12	30	44	86	111
5	15	53	238	422	1228	1727	5	17	35	81	150	261
6	15	101	393	1262	3501	6989	3	18	31	139	218	578
7		191	483	3699	8232	27835		20	22	215	241	1174
8		410	420	10637	15288	108927		23	8	319	203	2290
9		720		29783	21168	417862		22		436	136	4112
10		960		79939	18900	1565354		12		527	48	7025
11		990		201861		5695140		9		602		11062
12		990		467533		TO		5		576		16390
13				929500		TO				428		22544
14				1583582		TO				233		27443
15				2132130		TO				177		31024
16				1801800		TO				53		29595
17						TO						10531
18						TO						10531
19						TO						7857
20						TO						2371
Nodes	71	4467	1708	7242353	68811	TO	30	155	160	3866	1144	197426
Time	0.16s	13.66s	4.64s	40655.50s	3m41s	TO	0.08s	0.49s	0.49s	21.60s	4.17s	26m30s

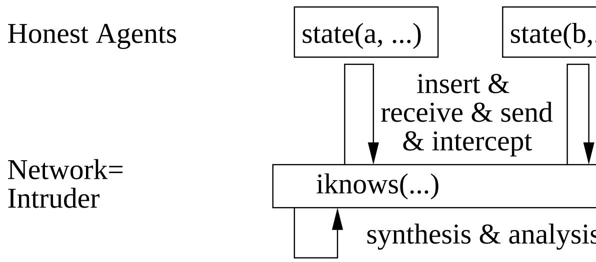
Lazy Intruder and Constraint Differentiation



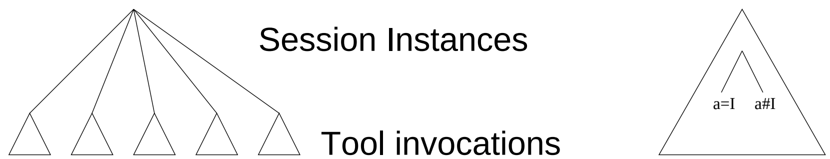
Graphical overview of some symbolic reductions



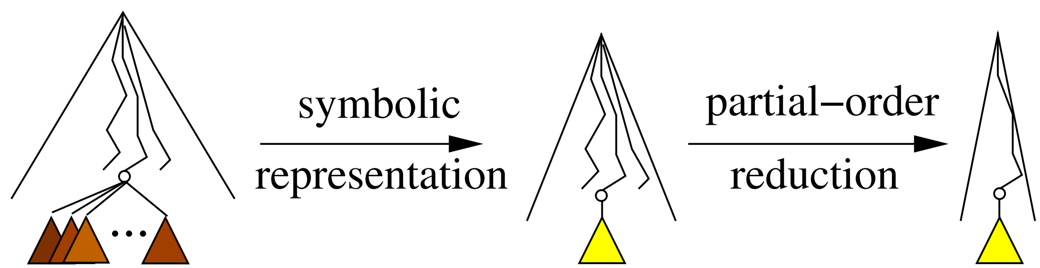
- The Lazy Intruder



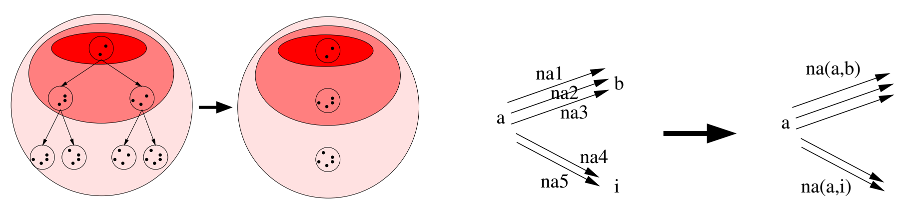
- Compressions



- Symbolic Sessions



- Constraint Differentiation



- Abstractions (data and control)

Road map

- Motivation.
- The AVISPA Tool.
- OFMC in more detail.
- ☞ **Algebraic properties.**
- Conclusions and outlook.

Context: messages in the free term algebra

Common Dolev-Yao-style model:

1. A –> B: enc(K,NA)
2. B –> A: enc(K,(NA,NB))

- Messages are represented by terms:
  - constant symbols: agent names, keys, . . .
  - function symbols: cryptographic operations
- The terms are interpreted in the free term algebra:

$$f(t_1, \ldots, t_n) \approx g(s_1, \ldots, s_m)$$
$$\text{iff}$$
$$(f = g) \ \wedge \ (t_1 \approx s_1) \ \wedge \ \ldots \ \wedge \ (t_n \approx s_n)$$

Context: messages in the free term algebra

Common Dolev-Yao-style model:

- Intruder deduction: given a set of ground terms IK, $\mathcal{DY}(IK)$ is the least closure of IK under a set of deduction rules like

$$\frac{m \in \mathcal{DY}(IK) \quad k \in \mathcal{DY}(IK)}{\{\!|m|\!\}_k \in \mathcal{DY}(IK)} \qquad \frac{\{\!|m|\!\}_k \in \mathcal{DY}(IK) \quad k \in \mathcal{DY}(IK)}{m \in \mathcal{DY}(IK)}$$

- Reflects the perfect cryptography assumption.

- Core of all protocol analysis problems.

- Well-understood for the free algebra.

Why algebraic properties are necessary

- Example Diffie-Hellman key-exchange:

$$
\begin{array}{ll}
1. & A \rightarrow B: \quad g^x \bmod p \\
2. & B \rightarrow A: \quad g^y \bmod p
\end{array}
$$

Why algebraic properties are necessary

- Example Diffie-Hellman key-exchange:

$$
\begin{array}{ll}
1. \quad A \rightarrow B: & g^x \bmod p \\
2. \quad B \rightarrow A: & g^y \bmod p \\
\hline
A: & B: \\
key = (g^y)^x & \approx \quad (g^x)^y \\
\hline
\end{array}
$$

Why algebraic properties are necessary

- Example Diffie-Hellman key-exchange:

$$
\begin{array}{c}
1. \quad A \rightarrow B : \quad g^x \bmod p \\
2. \quad B \rightarrow A : \quad g^y \bmod p \\
\hline
\qquad\qquad A : \qquad\qquad B : \\
key = \quad (g^y)^x \quad \approx \quad (g^x)^y \\
\hline
3. \quad A \leftrightarrow B : \quad \{\!|\ldots|\!\}_{g^{xy}}
\end{array}
$$

Why algebraic properties are necessary

- Example Diffie-Hellman key-exchange:

$$\begin{array}{l} 1.\quad A \rightarrow B:\quad g^x \bmod p \\ 2.\quad B \rightarrow A:\quad g^y \bmod p \\ \hline \qquad\qquad A: \qquad\qquad B: \\ key = \quad (g^y)^x \quad \approx \quad (g^x)^y \\ \hline 3.\quad A \leftrightarrow B:\quad \{\!|\ldots|\!\}_{g^{xy}} \end{array}$$

- Need commutativity of exponentiation to represent this protocol.
- Minimum: the algebraic properties necessary for legal protocol execution.
- Affects also authentication/agreement goals.
- Degree of abstraction and aspects to model:

$$\mathrm{dec}(k, \{\!|m|\!\}_k) \approx m \qquad X\|(Y\|Z) \approx (X\|Y)\|Z \qquad X \oplus Y \oplus X \approx Y$$

Examples: explicit encryption and decryption

- Most formal models lack explicit decryption operator.
- If a principal A knows an encrypted message and the corresponding key, assume A can decrypt message.
  - Implicit assumption that A never decrypts a message that wasn't encrypted in the first place.
  - Usually justified by assumption that A can check format of decrypted message.
- What if format checking isn't implemented? Or what if it is, but you are trying to verify that it works properly?
- In that case, need to model both encryption and decryption explicitly, plus their cancellation, e.g. $\mathrm{dec}(k, \{|m|\}_k) \approx m$.

Examples: explicit pairing and associativity

- Most formal systems assume boundaries between unambiguous terms.
- If a principal gets "$A \| NA$" won't that be confused with NB (or part of NB)?
- Even when type confusion addresses types of single terms.
- To get more realistic model, need explicit pairing and associativity, e.g.

$$fst(X\|Y) \approx X \qquad fst(X)\|snd(X) \approx X$$

$$snd(X\|Y) \approx Y \qquad X\|(Y\|Z) \approx (X\|Y)\|Z$$

Examples: exclusive-or

- Cheap and has provable security properties.
  - If A sends $X \oplus R$, where R is a random secret, then an observer learns no more about X than before it saw the message.
- On the other hand, commutativity and cancellation properties make it tricky to reason about:

$$X \oplus Y \approx Y \oplus X \qquad X \oplus X \approx e$$

$$(X \oplus Y) \oplus Z \approx X \oplus (Y \oplus Z) \qquad X \oplus e \approx X$$

Hard problems to solve

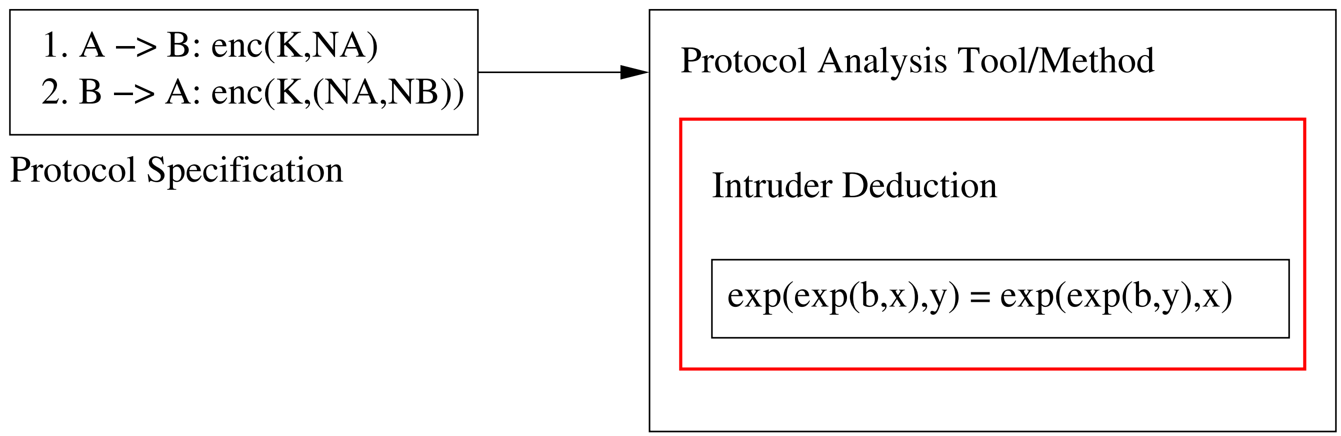
Consider the quotient algebra $\mathcal{T}(\Sigma, V)_{/\approx_E}$ for a set of equations E

- E-Unification problem: $\exists\sigma.\ s\sigma \approx_E t\sigma$?
- Intruder deduction problem: $t \in \mathcal{DY}_E(IK)$?

$$\frac{s \in \mathcal{DY}_E(IK)}{t \in \mathcal{DY}_E(IK)}\ s \approx_E t$$

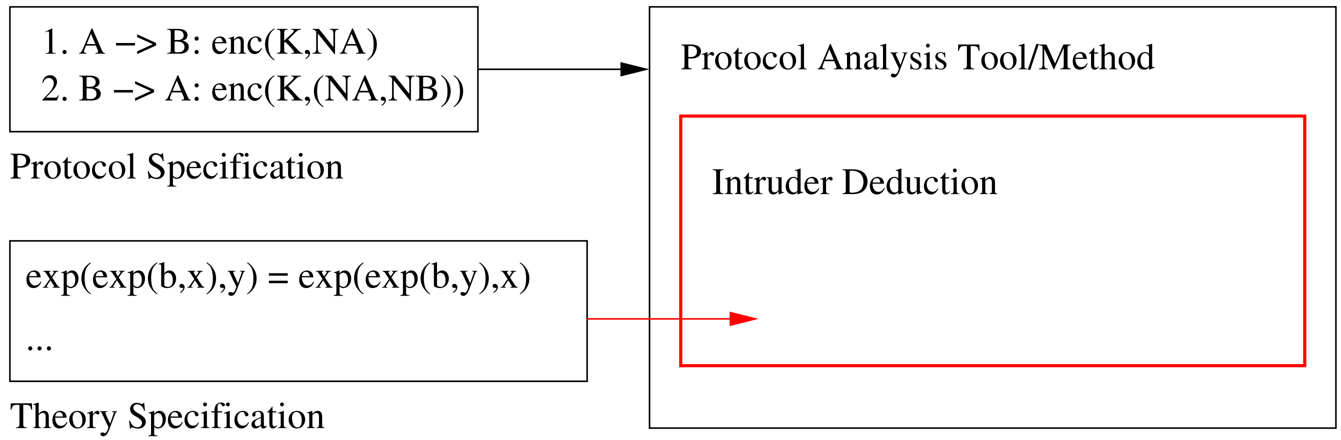
- Symbolic intruder deduction problem: $\exists\sigma.\ t\sigma \in \mathcal{DY}_E(IK\sigma)$?
- In general, these core problems are undecidable.

Existing work on algebraic intruder deduction



- More and more protocol analysis tools consider algebraic properties.
- Extensions for theories like exponentiation and bitwise xor.
  - Specialized algorithms for hard-wired theories.
- (Modular) rewriting approaches.
  - Parametrized over set of rewrite rules.
  - Built-in modular theory.

A framework for algebraic intruder deduction



- General methods for intruder deduction parametrized over algebraic theory E.
  - The theory E is read from a theory specification file.
  - Supports a large class of theories.
  - Independent of protocol analysis method.

Framework: supported theories $E = F \cup C$

$$\mathrm{dec}(x_1, \{|x_2|\}_{x_1}) \approx x_2$$

$$(x_1^{x_2})^{x_3} \approx (x_1^{x_3})^{x_2}$$

$$x_1 \oplus x_2 \approx x_2 \oplus x_1 \qquad x_1 \oplus x_1 \approx e$$

$$x_1 \oplus (x_2 \oplus x_3) \approx (x_1 \oplus x_2) \oplus x_3 \qquad x_1 \oplus e \approx x_1$$

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.

$$a \oplus b \oplus a \;\rightarrow_{C/F}\; e \oplus b \;\rightarrow_{C/F}\; b\,.$$

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

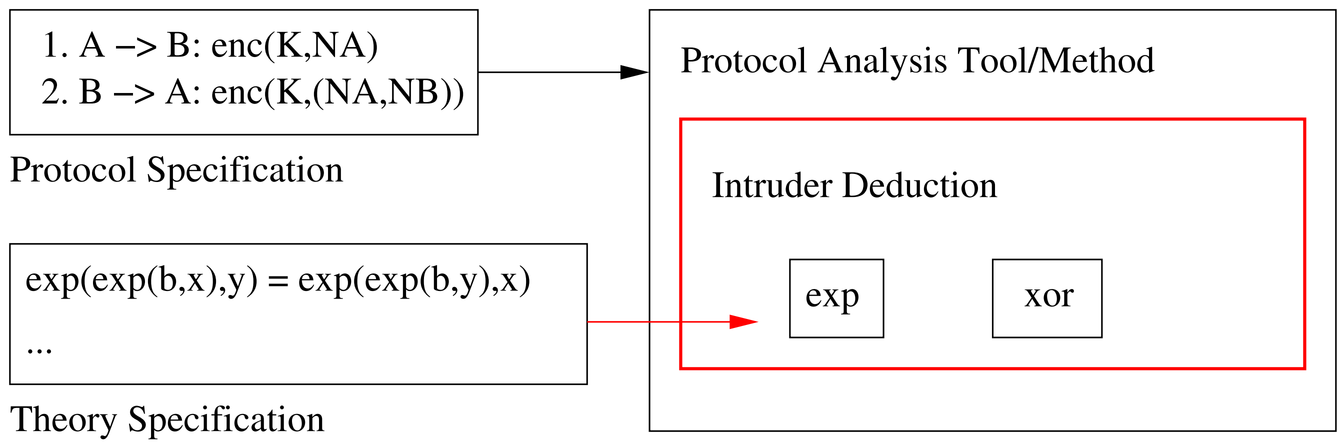
Framework: restrictions

- E-unification and E-deduction in general undecidable for the supported theories.

- We therefore introduce restrictions, trading them for generality and flexibility:
  - We bound the terms that can be substituted for variables.
  - We limit the number of deduction steps of the intruder.

- Many protocol analysis methods already require such restrictions.
  - Typed models in security protocol analysis are special cases of these restrictions.

Framework: modular design



Open for integration of specialized unification algorithms.

- Uses the more efficient specialized algorithms when available.
  - Usually without any bounds on variables and deduction steps.
- Uses the general methods otherwise.

Summary

A framework for algebraic intruder deduction, implemented in OFMC.

- General methods for intruder deduction parametrized over algebraic theory E.
- Modular design:
  - Large class of theories.
  - Independent from protocol analysis method.
  - Open for integration of existing specialized unification algorithms.
- Trading restrictions on variables and deduction steps for generality and flexibility.
- Provides a basis for a formalization of off-line guessing.
  - Making explicit intermediate steps of a guessing attack.
  - General and uniform definition, independent of the underlying intruder model and behavior of cryptography.

Road map

- Motivation.
- The AVISPA Tool.
- OFMC in more detail.
- Algebraic properties.

☞ **Conclusions and outlook.**

Conclusions and outlook

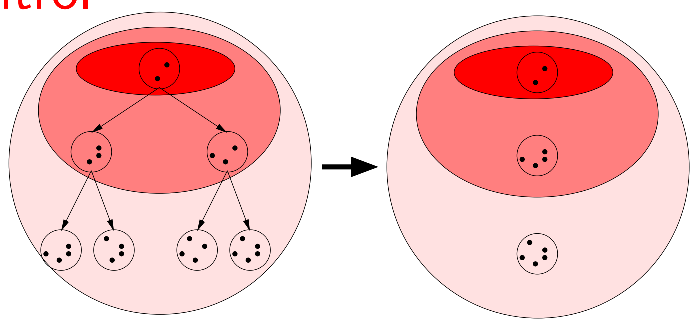
- The AVISPA Tool is a state-of-the-art, integrated environment for the automatic validation of Internet security protocols.

  AVISPA package (& web-interface): www.avispa-project.org

- Current work:
  - Extending the AVISPA library with further protocols and properties.
  - Unbounded verification using abstractions.
  - Algebraic properties.
  - Guessing intruder and other intruder models (and channels).
  - Web-services.
  - Combining cryptographic and formal proof techniques.

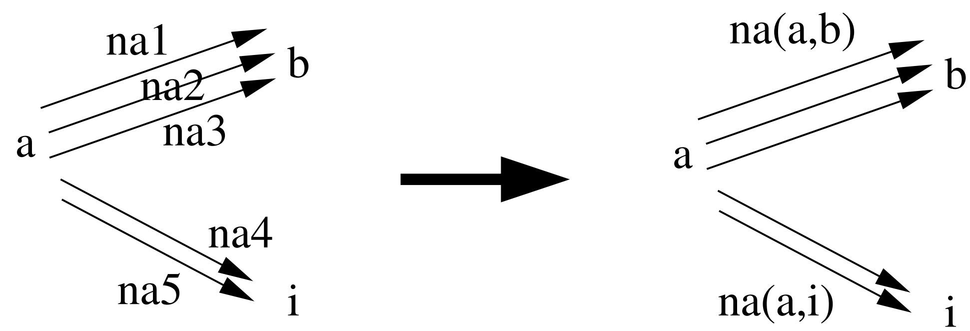
Abstractions

Control



- It abstracts the interleavings away completely.
- One computes the fixed-point of reachable facts rather than of reachable states.
- There is an unbounded number of sessions.

Data



- Idea: Partition fresh data into (finitely many) equivalence classes.
- Example: use as an equivalence relation on fresh data whether they were created by the same agent for the same purpose.

Web Services

- Web Services (WS): a series of standards that add higher-layer semantics and quality of service to web-based and XML-based communication, in particular among enterprises.
- Structure is far more complex than standard security protocols.
  - Requires model simplifications, approximations, and abstractions (and showing that these do not exclude attacks).
- Case study: Secure WS-ReliableMessaging Scenario [Fossacs'06]
  1. an automated analysis based on symbolic protocol analysis techniques under the assumption of perfect cryptography,
  2. an analysis closer to real cryptography based on explicit cryptographic assumptions on the underlying crypto-algorithms.

  Both analyses have positive results: they demonstrate that at the abstraction level of each analysis, the protocol is error-free.
- Future work: link the 2 kinds of analysis for WS in the style of previous proofs of soundness of Dolev-Yao models.