

Automatic SAT-Compilation of Protocol Insecurity Problems via Reduction to Planning*

Luca Compagna

joint work with Alessandro Armando



MRG-Lab – DIST, University of Genova

FORTE 2002 – Houston (Texas), November 13, 2002

*This work has been supported by the FET Open Assessment Project AVISS IST-2000-26410.

Introduction

- **Context:** Dramatic speed-up of SAT solvers in the last decade: problems with thousands of variables are now solved routinely in milliseconds.

This has led to breakthroughs in planning and hardware verification.

- **Approach:** In this work we have investigated if similar results can be obtained by applying SAT-based model-checking to security protocols.

Roadmap

- Example.
- The Model and Protocol Insecurity Problems.
- Encoding Protocol Insecurity Problems into SAT.
- Implementation and Experimental Results.
- Conclusions and Perspectives.
- Preliminary results (if there will be enough time).

An example

Consider the following simple protocol:

1. $A \rightarrow B : \{N_A\}_{K_{AB}}$
2. $B \rightarrow A : \{f(N_A)\}_{K_{AB}}$

An example

Consider the following simple protocol:

1. $A \rightarrow B : \{N_A\}_{K_{AB}}$
2. $B \rightarrow A : \{f(N_A)\}_{K_{AB}}$

This protocol **is flawed**. In fact, by executing

- 1.1. $A \rightarrow I(B) : \{N_A\}_{K_{AB}}$
- 2.1 $I(B) \rightarrow A : \{N_A\}_{K_{AB}}$
- 2.2 $A \rightarrow I(B) : \{f(N_A)\}_{K_{AB}}$
- 1.2 $I(B) \rightarrow A : \{f(N_A)\}_{K_{AB}}$

A believes to speak with B in the first session, but **A is really speaking with I** .

Modeling

- **Perfect cryptography:** an encrypted message can be neither altered nor read without the appropriate key.
- **Strong Typing:** all the messages considered in the analysis are assumed to be well-typed (type confusion is not allowed).
- **The Dolev-Yao attacker:**
 - controls all the traffic in the network.
 - can compose and send fraudulent messages from the knowledge he can glean from the observed traffic and his own initial knowledge.

Protocol Specification Language

We use an expressive formalism based on **first-order multiset rewriting**, called **IF**, (see Jacquemard, Rusinowitch, and Vigneron, "Compiling and Verifying Security Protocols" in LPAR 2000) that

- supports the specification of different protocol models and properties,
- has a clear, well-defined semantics, and
- results from automatic compilation of high-level protocol specifications in the "Alice&Bob"-style notation.

The Intermediate Format

States represented by multisets of terms of the form:

- $m(j, s, r, t)$ means that sender s has (supposedly) sent message t to principal r at protocol step j .

The Intermediate Format

States represented by multisets of terms of the form:

- $m(j, s, r, t)$ means that sender s has (supposedly) sent message t to principal r at protocol step j .
- $w(j, s, r, ak, ik, c)$ represents the **state of principal r** at step j of session c ; it means that r
 - knows the terms stored in the lists ak (*acquired knowledge*) and ik (*initial knowledge*), and
 - is waiting for a message from s (if $j \neq 0$).

The Intermediate Format

States represented by multisets of terms of the form:

- $m(j, s, r, t)$ means that sender s has (supposedly) sent message t to principal r at protocol step j .
- $w(j, s, r, ak, ik, c)$ represents the state of principal r at step j of session c ; it means that r
 - knows the terms stored in the lists ak (*acquired knowledge*) and ik (*initial knowledge*), and
 - is waiting for a message from s (if $j \neq 0$).
- $i(t)$ means that the intruder knows t .

The Intermediate Format

States represented by multisets of terms of the form:

- $m(j, s, r, t)$ means that sender s has (supposedly) sent message t to principal r at protocol step j .
- $w(j, s, r, ak, ik, c)$ represents the state of principal r at step j of session c ; it means that r
 - knows the terms stored in the lists ak (*acquired knowledge*) and ik (*initial knowledge*), and
 - is waiting for a message from s (if $j \neq 0$).
- $i(t)$ means that the intruder knows t .
- $fresh(n)$ means that n has not been used yet.

The IF: an example

- Initial State:

$w(0, \text{alice}, \text{alice}, [], [\text{alice}, \text{bob}, \text{kab}], 1) \cdot w(1, \text{alice}, \text{bob}, [], [\text{bob}, \text{alice}, \text{kab}], 1) \cdot \dots \cdot$
 $\text{fresh}(\text{nc}(\text{na}, 1)) \cdot i(\text{alice}) \cdot i(\text{bob})$

The IF: an example

- Initial State:

$w(0, \text{alice}, \text{alice}, [], [\text{alice}, \text{bob}, \text{kab}], 1) \cdot w(1, \text{alice}, \text{bob}, [], [\text{bob}, \text{alice}, \text{kab}], 1) \cdot \dots \cdot$
 $\text{fresh}(\text{nc}(\text{na}, 1)) \cdot i(\text{alice}) \cdot i(\text{bob})$

- Protocol Rules: $w(0, xA, xA, [], [xA, xB, xK], xC) \cdot \text{fresh}(\text{nc}(\text{na}, xC)) \Rightarrow$
 $m(1, xA, xB, \{\text{nc}(\text{na}, xC)\}_{xK}) \cdot$
 $w(2, xB, xA, [\text{nc}(\text{na}, xC)], [xA, xB, xK], xC)$

The IF: an example

- Initial State:

$w(0, \text{alice}, \text{alice}, [], [\text{alice}, \text{bob}, \text{kab}], 1) \cdot w(1, \text{alice}, \text{bob}, [], [\text{bob}, \text{alice}, \text{kab}], 1) \cdot \dots \cdot$
 $\text{fresh}(\text{nc}(\text{na}, 1)) \cdot i(\text{alice}) \cdot i(\text{bob})$

- Protocol Rules: $w(0, xA, xA, [], [xA, xB, xK], xC) \cdot \text{fresh}(\text{nc}(\text{na}, xC)) \Rightarrow$
 $m(1, xA, xB, \{\text{nc}(\text{na}, xC)\}_{xK}) \cdot$
 $w(2, xB, xA, [\text{nc}(\text{na}, xC)], [xA, xB, xK], xC)$

- Intruder Rules:

$m(xj, xs, xr, xmsg) \Rightarrow i(xs) \cdot i(xr) \cdot i(xmsg)$
 $i(\{xmsg\}_{xK}) \cdot i(xK) \Rightarrow i(xmsg) \cdot i(\{xmsg\}_{xK}) \cdot i(xK)$

The IF: an example

- Initial State:

$$w(0, \text{alice}, \text{alice}, [], [\text{alice}, \text{bob}, \text{kab}], 1) \cdot w(1, \text{alice}, \text{bob}, [], [\text{bob}, \text{alice}, \text{kab}], 1) \cdot \dots \cdot \text{fresh}(\text{nc}(\text{na}, 1)) \cdot i(\text{alice}) \cdot i(\text{bob})$$

- Protocol Rules: $w(0, xA, xA, [], [xA, xB, xK], xC) \cdot \text{fresh}(\text{nc}(\text{na}, xC)) \Rightarrow m(1, xA, xB, \{\text{nc}(\text{na}, xC)\}_{xK}) \cdot w(2, xB, xA, [\text{nc}(\text{na}, xC)], [xA, xB, xK], xC)$

- Intruder Rules:

$$m(xj, xs, xr, xmsg) \Rightarrow i(xs) \cdot i(xr) \cdot i(xmsg) \\ i(\{xmsg\}_{xK}) \cdot i(xK) \Rightarrow i(xmsg) \cdot i(\{xmsg\}_{xK}) \cdot i(xK)$$

- Bad States: $w(0, \text{alice}, \text{alice}, [], [\text{alice}, \text{bob}, \text{kab}], s(1)) \cdot w(1, \text{alice}, \text{bob}, [], [\text{bob}, \text{alice}, \text{kab}], 1)$

Protocol Insecurity Problems

A **Protocol Insecurity Problem** is a tuple $\langle \mathcal{S}, \mathcal{L}, \mathcal{R}, \mathcal{I}, \mathcal{B} \rangle$ where:

- \mathcal{S} is a set of atomic formulae of a sorted first-order language called *facts*;
- \mathcal{L} is a set of function symbols called *rule labels*;
- \mathcal{R} is a set of (deterministic) labelled *rewrite rules* of the form $L \xrightarrow{l} R$, where $L, R \subseteq \mathcal{S}$, and $l \in \mathcal{L}$;
- \mathcal{I} and \mathcal{B} are the *initial state* and a set of *bad states*.

Protocol Insecurity Problems

A **Protocol Insecurity Problem** is a tuple $\langle \mathcal{S}, \mathcal{L}, \mathcal{R}, \mathcal{I}, \mathcal{B} \rangle$ where:

- \mathcal{S} is a set of atomic formulae of a sorted first-order language called *facts*;
- \mathcal{L} is a set of function symbols called *rule labels*;
- \mathcal{R} is a set of (deterministic) labelled *rewrite rules* of the form $L \xrightarrow{l} R$, where $L, R \subseteq \mathcal{S}$, and $l \in \mathcal{L}$;
- \mathcal{I} and \mathcal{B} are the *initial state* and a set of *bad states*.

A **solution to a Protocol Insecurity Problem** is a sequence

$S_1 \xrightarrow{l_1} S_2 \xrightarrow{l_2} \dots \xrightarrow{l_{n-1}} S_n$, where $l_i \in \mathcal{L}$, $\mathcal{I} \equiv S_1$, and there exists $S_{\mathcal{B}} \in \mathcal{B}$ such that $S_{\mathcal{B}} \subseteq S_n$.

Encoding Protocol Insecurity Problems into SAT

The reduction of **Protocol Insecurity Problems** to **SAT** is carried out in two phases:

1. the **Protocol Insecurity Problem** is translated into **Planning Problem**;
2. the **Planning Problem** is then encoded into **SAT**, using standard encoding techniques, with iterative deepening on the number of steps.

Planning Problems

A **planning problem** is a tuple $\Pi = \langle \mathcal{F}, \mathcal{A}, Ops, I, G \rangle$, where:

- \mathcal{F} and \mathcal{A} are sets of ground atomic formulas called **fluents** and **actions** respectively.
- Ops is a set of expressions of the form

$$Pre \xrightarrow{Act} Add; Del,$$

where $Act \in \mathcal{A}$, and Pre , Add , and Del are finite sets of fluents such that $Add \cap Del = \emptyset$.

- I and G are boolean combinations of fluents representing the **initial state** and the **final states** respectively.

A **solution to a planning problem** is a sequence of actions whose execution leads from the initial state to a final state and the preconditions of each action hold in the state to which it applies.

Protocol Insecurity Problems as Planning Problems

Map IF facts into “fluents”:

IF facts	Planning fluents
$i(a)$	$i(a)$
$m(1, a, b, \{nc(na, 1)\}_{kab}, 1)$	$m(1, a, b, \{nc(na, 1)\}_{kab}, 1)$

Map IF rewrite rules into “actions”:

IF rules	Planning operators
$B \xRightarrow{P} A$	$B \xrightarrow{P} A; \neg B$
$A, B \xRightarrow{Q}$	$A, B \xrightarrow{Q} ; \neg A, \neg B$
$A, B \xRightarrow{R} A$	$A, B \xrightarrow{R} ; \neg B$

Encoding Planning Problems into SAT (1)

Fact: Given,

- a planning problem $\Pi = \langle \mathcal{F}, \mathcal{A}, Ops, I, G \rangle$, and
- a positive integer n ,

then it is possible to build a propositional formula Φ_{Π}^n such that any **model of Φ_{Π}^n** corresponds to a **partial-order plan of Π** .

Intuition: add an additional **time-index** parameter to each **action** or **fluent**, to indicate the state at which the action begins or the fluent holds.

Encoding Planning Problems into SAT (2)

The formula Φ_{Π}^n is given by the conjunction of the following axioms:

Universal Axioms: for each action $\alpha \in \mathcal{A}$ s.t.

$(Pre \xrightarrow{\alpha} Add; Del) \in Ops$ and for each $i = 0, \dots, n - 1$

$$\alpha_i \supset \bigwedge \{p_i \mid p \in Pre\}$$

$$\alpha_i \supset \bigwedge \{p_{i+1} \mid p \in Add\}$$

$$\alpha_i \supset \bigwedge \{\neg p_{i+1} \mid p \in Del\}$$

Cardinality: $O(n|\mathcal{A}||\mathcal{F}|)$, but usually $O(n|\mathcal{A}|r)$, where r is the maximal number of fluents mentioned in an operator (usually a small number).

Encoding Planning Problems into SAT (3)

Explanatory Frame Axioms: for all fluents $p \in F$ and for each $i = 1, \dots, n - 1$

$$\begin{aligned}
 (p_i \wedge \neg p_{i+1}) &\supset \\
 &\quad \bigvee \left\{ \alpha_i \mid \alpha \in \mathcal{A}, Pre \xrightarrow{\alpha} Add; Del \in Ops, p \in Del \right\} \\
 (\neg p_i \wedge p_{i+1}) &\supset \\
 &\quad \bigvee \left\{ \alpha_i \mid \alpha \in \mathcal{A}, Pre \xrightarrow{\alpha} Add; Del \in Ops, p \in Add \right\}
 \end{aligned}$$

Cardinality: $O(n|\mathcal{F}||\mathcal{A}|)$, but usually $O(n|\mathcal{F}|k)$, where k is a small number.

Encoding Planning Problems into SAT (4)

Conflict Exclusion Axioms: for each $i = 0, \dots, n - 1$

$$\neg(\alpha_i \wedge \alpha'_i)$$

for all $\alpha, \alpha' \in \mathcal{A}$ s.t. $\alpha \neq \alpha'$, $Pre \xrightarrow{\alpha} Add; Del \in Ops$,
 $Pre' \xrightarrow{\alpha'} Add'; Del' \in Ops$, and $Pre \cap Del' \neq \emptyset$ or $Pre' \cap Del \neq \emptyset$.

Cardinality: $O(n|\mathcal{A}|^2)$

Initial State Axioms: I_0 , i.e. the formula I in which each occurrence of a fluent is replaced by a fluent time-indexed with 0.

Cardinality: $O(|\mathcal{F}|)$.

Goal State Axioms: G_n , i.e. the formula G in which each occurrence of a fluent is replaced by a fluent time-indexed with n .

Cardinality: depending from the structure of G , typically small.

Is a direct application of this encoding feasible
for our task?

Is a direct application of this encoding feasible
for our task?

No!

Is a direct application of this encoding feasible
for our task?

No!

A number of optimizing transformations need
to be done in order to get encodings of
manageable size.

Optimizations

- Language Specialization.
- Fluent Splitting.
- Exploiting Static Fluents.
- Invariant-based Simplification.
- Reducing the Number of Conflict Exclusion Axioms.
- Step-Compression (`step_compression`).
- Building Encryption Properties into the Encoding.
- Exploiting Mutual Exclusion of w -terms.
- Graphplan Encoding and/or Unit Propagation.

Optimizations: Language Specialization

- **Specialize** $m(\dots)$, $w(\dots)$, and $i(\dots)$ by restricting the message terms to those allowed by the protocol.
- **Splitting messages** of the form $m(j, s, r, c(t_1, t_2))$ into two messages $m(j, s, r, t_1, 1)$ and $m(j, s, r, t_2, 2)$.

When used in combination, these optimizations often **reduce** the size of the language of **several orders of magnitude**.

Optimizations: Step-Compression

Simple form of partial-order reduction: significant savings by compressing these rule triples.

Impersonate	w(...)	i(...)
	=>	
	w(...)	i(...)
	m(...)	

step	w(...)
	m(...)
	=>
	w(...)
	m(...)

divert	m(...)
	=>
	i(...)

Optimizations: Step-Compression

Simple form of partial-order reduction: significant savings by compressing these rule triples.



Optimizations: Step-Compression

Simple form of partial-order reduction: significant savings by compressing these rule triples.

Impersonate	w(...)
	i(...)
step	=>
divert	w(...)
	i(...)

Introduction of Bounds and Constraints

- ☑ Bounding the Number of Session Repetitions
(`session_repetitions`).
- ☑ Constraining Variable Instantiation
(`constrain_rule_variables`).

Note: These may introduce incompleteness, which can be overcome by searching (using iterative deepening) the parameter space.

Constraining Variable Instantiation

Let us consider part of the Kao-Chow protocol:

2. $S \rightarrow B : \{A, B, Na, Kab\}Kas, \{A, B, Na, Kab\}Kbs$
3. $B \rightarrow A : \{A, B, Na, Kab\}Kas, \{Na\}Kab, Nb$

B cannot decrypt $\{A, B, Na, Kab\}Kas$. Therefore, the occurrences of A in the **first** component of the received message might be different to that inside the **second**.

This constraint imposes that the occurrences of A (as well as of B , Na , and Kab) in the **first** and in the **second** part of the message must coincide.

For instance, messages of the form

$$m(2, a, b, c(\{a, b, nc(na, s(1))\}kas, \{a, b, nc(nb, s(1))\}kbs)))$$

would be ruled out by the constraint.

Implementation

- **IF2SATE**: a translator from the IF to SATE (a STRIPS-like language).

6,200 lines of Prolog code.

- **SATE**: a compiler from SATE to SAT (DIMACS format).

2,800 lines of Prolog code.

NOTE: state-of-the-art tools carrying out the SAT encoding of planning problems (e.g. Medic, BlackBox) are unable to handle STRIPS languages with complex term structure (only individual constants allowed).

Experiments: Results

Protocol	Atoms	Clauses	EncTime	SolvTime
<i>ISO symmetric key 1-pass unilateral authentication</i>	679	2,073	0.18	0.00
<i>ISO symmetric key 2-pass mutual authentication</i>	1,970	7,382	0.43	0.01
<i>Andrew Secure RPC Protocol</i>	161,615	2,506,889	80.57	2.65
<i>ISO CCF 1-pass unilateral authentication</i>	649	2,033	0.17	0.00
<i>ISO CCF 2-pass mutual authentication</i>	2,211	10,595	0.46	0.00
<i>Needham-Schroeder Conventional Key</i>	126,505	370,449	29.25	0.39
<i>Woo-Lam II</i>	7,988	56,744	3.31	0.04
<i>Woo-Lam Mutual Authentication</i>	771,934	4,133,390	1,024.00	7.95
<i>Needham-Schroeder Signature protocol</i>	17,867	59,911	3.77	0.05
<i>Neuman Stubblebine repeated part</i>	39,579	312,107	15.17	0.21
<i>Kao Chow Repeated Authentication, 1</i>	50,703	185,317	16.34	0.17
<i>Kao Chow Repeated Authentication, 2</i>	586,033	1,999,959	339.70	2.11
<i>Kao Chow Repeated Authentication, 3</i>	1,100,428	6,367,574	1,288.00	MO
<i>ISO public key 1-pass unilateral authentication</i>	1,161	3,835	0.32	0.00
<i>ISO public key 2-pass mutual authentication</i>	4,165	23,883	1.18	0.01
<i>Needham-Schroeder Public Key</i>	9,318	47,474	1.77	0.05
<i>Needham-Schroeder Public Key with key server</i>	11,339	67,056	4.29	0.04
<i>SPLICE/AS Authentication Protocol</i>	15,622	69,226	5.48	0.05
<i>Encrypted Key Exchange</i>	121,868	1,500,317	75.39	1.78
<i>Davis Swick Private Key Certificates, protocol 1</i>	8,036	25,372	1.37	0.02
<i>Davis Swick Private Key Certificates, protocol 2</i>	12,123	47,149	2.68	0.03
<i>Davis Swick Private Key Certificates, protocol 3</i>	10,606	27,680	1.50	0.02
<i>Davis Swick Private Key Certificates, protocol 4</i>	27,757	96,482	8.18	0.13

Experiments: Results

Protocol	Atoms	Clauses	EncTime	SolvTime
<i>ISO symmetric key 1-pass unilateral authentication</i>	679	2,073	0.18	0.00
<i>ISO symmetric key 2-pass mutual authentication</i>	1,970	7,382	0.43	0.01
<i>Andrew Secure RPC Protocol</i>	161,615	2,506,889	80.57	2.65
<i>ISO CCF 1-pass unilateral authentication</i>	649	2,033	0.17	0.00
<i>ISO CCF 2-pass mutual authentication</i>	2,211	10,595	0.46	0.00
<i>Needham-Schroeder Conventional Key</i>	126,505	370,449	29.25	0.39
<i>Woo-Lam II</i>	7,988	56,744	3.31	0.04
<i>Woo-Lam Mutual Authentication</i>	771,934	4,133,390	1,024.00	7.95
<i>Needham-Schroeder Signature protocol</i>	17,867	59,911	3.77	0.05
<i>Neuman Stubblebine repeated part</i>	39,579	312,107	15.17	0.21
<i>Kao Chow Repeated Authentication, 1</i>	50,703	185,317	16.34	0.17
<i>Kao Chow Repeated Authentication, 2</i>	586,033	1,999,959	339.70	2.11
<i>Kao Chow Repeated Authentication, 3</i>	1,100,428	6,367,574	1,288.00	MO
<i>ISO public key 1-pass unilateral authentication</i>	1,161	3,835	0.32	0.00
<i>ISO public key 2-pass mutual authentication</i>	4,165	23,883	1.18	0.01
<i>Needham-Schroeder Public Key</i>	9,318	47,474	1.77	0.05
<i>Needham-Schroeder Public Key with key server</i>	11,339	67,056	4.29	0.04
<i>SPLICE/AS Authentication Protocol</i>	15,622	69,226	5.48	0.05
<i>Encrypted Key Exchange</i>	121,868	1,500,317	75.39	1.78
<i>Davis Swick Private Key Certificates, protocol 1</i>	8,036	25,372	1.37	0.02
<i>Davis Swick Private Key Certificates, protocol 2</i>	12,123	47,149	2.68	0.03
<i>Davis Swick Private Key Certificates, protocol 3</i>	10,606	27,680	1.50	0.02
<i>Davis Swick Private Key Certificates, protocol 4</i>	27,757	96,482	8.18	0.13

Experiments: Results

Protocol	Atoms	Clauses	EncTime	SolvTime
<i>ISO symmetric key 1-pass unilateral authentication</i>	679	2,073	0.18	0.00
<i>ISO symmetric key 2-pass mutual authentication</i>	1,970	7,382	0.43	0.01
<i>Andrew Secure RPC Protocol</i>	161,615	2,506,889	80.57	2.65
<i>ISO CCF 1-pass unilateral authentication</i>	649	2,033	0.17	0.00
<i>ISO CCF 2-pass mutual authentication</i>	2,211	10,595	0.46	0.00
<i>Needham-Schroeder Conventional Key</i>	126,505	370,449	29.25	0.39
<i>Woo-Lam II</i>	7,988	56,744	3.31	0.04
<i>Woo-Lam Mutual Authentication</i>	771,934	4,133,390	1,024.00	7.95
<i>Needham-Schroeder Signature protocol</i>	17,867	59,911	3.77	0.05
<i>Neuman Stubblebine repeated part</i>	39,579	312,107	15.17	0.21
<i>Kao Chow Repeated Authentication, 1</i>	50,703	185,317	16.34	0.17
<i>Kao Chow Repeated Authentication, 2</i>	586,033	1,999,959	339.70	2.11
<i>Kao Chow Repeated Authentication, 3</i>	1,100,428	6,367,574	1,288.00	MO
<i>ISO public key 1-pass unilateral authentication</i>	1,161	3,835	0.32	0.00
<i>ISO public key 2-pass mutual authentication</i>	4,165	23,883	1.18	0.01
<i>Needham-Schroeder Public Key</i>	9,318	47,474	1.77	0.05
<i>Needham-Schroeder Public Key with key server</i>	11,339	67,056	4.29	0.04
<i>SPLICE/AS Authentication Protocol</i>	15,622	69,226	5.48	0.05
<i>Encrypted Key Exchange</i>	121,868	1,500,317	75.39	1.78
<i>Davis Swick Private Key Certificates, protocol 1</i>	8,036	25,372	1.37	0.02
<i>Davis Swick Private Key Certificates, protocol 2</i>	12,123	47,149	2.68	0.03
<i>Davis Swick Private Key Certificates, protocol 3</i>	10,606	27,680	1.50	0.02
<i>Davis Swick Private Key Certificates, protocol 4</i>	27,757	96,482	8.18	0.13

Experiments: Results

Protocol	Atoms	Clauses	EncTime	SolvTime
<i>ISO symmetric key 1-pass unilateral authentication</i>	679	2,073	0.18	0.00
<i>ISO symmetric key 2-pass mutual authentication</i>	1,970	7,382	0.43	0.01
<i>Andrew Secure RPC Protocol</i>	161,615	2,506,889	80.57	2.65
<i>ISO CCF 1-pass unilateral authentication</i>	649	2,033	0.17	0.00
<i>ISO CCF 2-pass mutual authentication</i>	2,211	10,595	0.46	0.00
<i>Needham-Schroeder Conventional Key</i>	126,505	370,449	29.25	0.39
<i>Woo-Lam II</i>	7,988	56,744	3.31	0.04
<i>Woo-Lam Mutual Authentication</i>	771,934	4,133,390	1,024.00	7.95
<i>Needham-Schroeder Signature protocol</i>	17,867	59,911	3.77	0.05
<i>Neuman Stubblebine repeated part</i>	39,579	312,107	15.17	0.21
<i>Kao Chow Repeated Authentication, 1</i>	50,703	185,317	16.34	0.17
<i>Kao Chow Repeated Authentication, 2</i>	586,033	1,999,959	339.70	2.11
<i>Kao Chow Repeated Authentication, 3</i>	1,100,428	6,367,574	1,288.00	MO
<i>ISO public key 1-pass unilateral authentication</i>	1,161	3,835	0.32	0.00
<i>ISO public key 2-pass mutual authentication</i>	4,165	23,883	1.18	0.01
<i>Needham-Schroeder Public Key</i>	9,318	47,474	1.77	0.05
<i>Needham-Schroeder Public Key with key server</i>	11,339	67,056	4.29	0.04
<i>SPLICE/AS Authentication Protocol</i>	15,622	69,226	5.48	0.05
<i>Encrypted Key Exchange</i>	121,868	1,500,317	75.39	1.78
<i>Davis Swick Private Key Certificates, protocol 1</i>	8,036	25,372	1.37	0.02
<i>Davis Swick Private Key Certificates, protocol 2</i>	12,123	47,149	2.68	0.03
<i>Davis Swick Private Key Certificates, protocol 3</i>	10,606	27,680	1.50	0.02
<i>Davis Swick Private Key Certificates, protocol 4</i>	27,757	96,482	8.18	0.13

Experiments: Analysis

Over the Clark/Jacob library:

- **Coverage:** unsuited to detect type-flaws, but effective on the others.
- **Effectiveness:** it finds most of the known attacks.
- **Performance:** Encoding Time largely dominates Solving Time.

Conclusions & Perspectives

- SATMC **performs well** on the Clark/Jacob library.
- Optimizations bring **a lot!** Some remain to be implemented.

We expect orders of magnitude reduction in compilation time by:

- building **encryption properties** into the encoding,
- exploiting more **sophisticated encoding** (e.g. graphplan encoding), and
- applying **unit propagation** during the encoding phase.

Experiments: Unit Propagation

Protocol	SATMC		Compact	
	Atoms	Clauses	Atoms	Clauses
<i>ISO symmetric key 1-pass unilateral authentication</i>	679	2,073	188	562
<i>ISO symmetric key 2-pass mutual authentication</i>	1,970	7,382	293	960
<i>Andrew Secure RPC Protocol</i>	161,615	2,506,889	372	1,228
<i>ISO CCF 1-pass unilateral authentication</i>	649	2,033	207	642
<i>ISO CCF 2-pass mutual authentication</i>	2,211	10,595	274	894
<i>Needham-Schroeder Conventional Key</i>	126,505	370,449	370	1,220
<i>Woo-Lam II</i>	7,988	56,744	925	3,613
<i>Woo-Lam Mutual Authentication</i>	771,934	4,133,390	1,073	4,796
<i>Needham-Schroeder Signature protocol</i>	17,867	59,911	379	1,332
<i>Neuman Stubblebine repeated part</i>	39,579	312,107	802	4,342
<i>Kao Chow Repeated Authentication, 1</i>	50,703	185,317	654	2,855
<i>Kao Chow Repeated Authentication, 2</i>	586,033	1,999,959	696	2,938
<i>Kao Chow Repeated Authentication, 3</i>	1,100,428	6,367,574	978	5,093
<i>ISO public key 1-pass unilateral authentication</i>	1,161	3,835	302	855
<i>ISO public key 2-pass mutual authentication</i>	4,165	23,883	303	1,053
<i>Needham-Schroeder Public Key</i>	9,318	47,474	497	1,580
<i>Needham-Schroeder Public Key with key server</i>	11,339	67,056	758	2,161
<i>SPLICE/AS Authentication Protocol</i>	15,622	69,226	612	2,082
<i>Encrypted Key Exchange</i>	121,868	1,500,317	1,088	70,865

Experiments: Unit Propagation

Protocol	SATMC		Compact	
	Atoms	Clauses	Atoms	Clauses
<i>ISO symmetric key 1-pass unilateral authentication</i>	679	2,073	188	562
<i>ISO symmetric key 2-pass mutual authentication</i>	1,970	7,382	293	960
<i>Andrew Secure RPC Protocol</i>	161,615	2,506,889	372	1,228
<i>ISO CCF 1-pass unilateral authentication</i>	649	2,033	207	642
<i>ISO CCF 2-pass mutual authentication</i>	2,211	10,595	274	894
<i>Needham-Schroeder Conventional Key</i>	126,505	370,449	370	1,220
<i>Woo-Lam II</i>	7,988	56,744	925	3,613
<i>Woo-Lam Mutual Authentication</i>	771,934	4,133,390	1,073	4,796
<i>Needham-Schroeder Signature protocol</i>	17,867	59,911	379	1,332
<i>Neuman Stubblebine repeated part</i>	39,579	312,107	802	4,342
<i>Kao Chow Repeated Authentication, 1</i>	50,703	185,317	654	2,855
<i>Kao Chow Repeated Authentication, 2</i>	586,033	1,999,959	696	2,938
<i>Kao Chow Repeated Authentication, 3</i>	1,100,428	6,367,574	978	5,093
<i>ISO public key 1-pass unilateral authentication</i>	1,161	3,835	302	855
<i>ISO public key 2-pass mutual authentication</i>	4,165	23,883	303	1,053
<i>Needham-Schroeder Public Key</i>	9,318	47,474	497	1,580
<i>Needham-Schroeder Public Key with key server</i>	11,339	67,056	758	2,161
<i>SPLICE/AS Authentication Protocol</i>	15,622	69,226	612	2,082
<i>Encrypted Key Exchange</i>	121,868	1,500,317	1,088	70,865

Experiments: Graphplan Encoding

Protocol	SATMC		Blackbox	
	Atoms	Clauses	Atoms	Clauses
<i>ISO symmetric key 1-pass unilateral authentication</i>	679	2,073	396	1,001
<i>ISO symmetric key 2-pass mutual authentication</i>	1,970	7,382	494	1,349
<i>Andrew Secure RPC Protocol</i>	161,615	2,506,889	633	2,381
<i>ISO CCF 1-pass unilateral authentication</i>	649	2,033	413	1,059
<i>ISO CCF 2-pass mutual authentication</i>	2,211	10,595	522	1,455
<i>Needham-Schroeder Conventional Key</i>	126,505	370,449	767	2,418
<i>Woo-Lam II</i>	7,988	56,744	1,129	6,822
<i>Woo-Lam Mutual Authentication</i>	771,934	4,133,390	1,058	5,271
<i>Needham-Schroeder Signature protocol</i>	17,867	59,911	566	1,482
<i>Neuman Stubblebine repeated part</i>	39,579	312,107	1,086	8,857
<i>Kao Chow Repeated Authentication, 1</i>	50,703	185,317	891	5,416
<i>Kao Chow Repeated Authentication, 2</i>	586,033	1,999,959	904	5,382
<i>Kao Chow Repeated Authentication, 3</i>	1,100,428	6,367,574	1,015	7,209
<i>ISO public key 1-pass unilateral authentication</i>	1,161	3,835	576	1,614
<i>ISO public key 2-pass mutual authentication</i>	4,165	23,883	667	1,924
<i>Needham-Schroeder Public Key</i>	9,318	47,474	815	2,682
<i>Needham-Schroeder Public Key with key server</i>	11,339	67,056	1,486	5,298
<i>SPLICE/AS Authentication Protocol</i>	15,622	69,226	1,023	3,841
<i>Encrypted Key Exchange</i>	121,868	1,500,317	1,056	5,855

Experiments: Graphplan Encoding

Protocol	SATMC		Blackbox	
	Atoms	Clauses	Atoms	Clauses
<i>ISO symmetric key 1-pass unilateral authentication</i>	679	2,073	396	1,001
<i>ISO symmetric key 2-pass mutual authentication</i>	1,970	7,382	494	1,349
<i>Andrew Secure RPC Protocol</i>	161,615	2,506,889	633	2,381
<i>ISO CCF 1-pass unilateral authentication</i>	649	2,033	413	1,059
<i>ISO CCF 2-pass mutual authentication</i>	2,211	10,595	522	1,455
<i>Needham-Schroeder Conventional Key</i>	126,505	370,449	767	2,418
<i>Woo-Lam II</i>	7,988	56,744	1,129	6,822
<i>Woo-Lam Mutual Authentication</i>	771,934	4,133,390	1,058	5,271
<i>Needham-Schroeder Signature protocol</i>	17,867	59,911	566	1,482
<i>Neuman Stubblebine repeated part</i>	39,579	312,107	1,086	8,857
<i>Kao Chow Repeated Authentication, 1</i>	50,703	185,317	891	5,416
<i>Kao Chow Repeated Authentication, 2</i>	586,033	1,999,959	904	5,382
<i>Kao Chow Repeated Authentication, 3</i>	1,100,428	6,367,574	1,015	7,209
<i>ISO public key 1-pass unilateral authentication</i>	1,161	3,835	576	1,614
<i>ISO public key 2-pass mutual authentication</i>	4,165	23,883	667	1,924
<i>Needham-Schroeder Public Key</i>	9,318	47,474	815	2,682
<i>Needham-Schroeder Public Key with key server</i>	11,339	67,056	1,486	5,298
<i>SPLICE/AS Authentication Protocol</i>	15,622	69,226	1,023	3,841
<i>Encrypted Key Exchange</i>	121,868	1,500,317	1,056	5,855

The SATE Specification Language (1)

The SATE specification language allows to specify planning problems by means of the following main constructs:

- **Sort declaration**, e.g. `sort(msg), sort(pk), sort(nonce)`.
- **Super-Sort declaration**, e.g. `super_sort(msg, nonce)`.
- **Constant declaration**, e.g. `constant(encrypt(pk, msg), msg)`.
- **Initial State**, e.g.

```
facts([wk(0, mr(intruder), mr(a), etc, 1, true, 1),
      wk(1, mr(a), mr(b), etc, 1, true, 1),
      wk(0, mr(intruder), mr(a), etc, 1, true, 2),
      inknw(mr(a), mr(a), 1, 1), ... ,
      inknw(mr(b), primed(pk(kb)), 4, 1),
      i(mr(b)), ..., i(primed(pk(ki)))]).
```

The SATE Specification Language (2)

- **Goal**, e.g. `goal([i(nonce(na,X))])`.
- **Operator**, e.g.

```

action(step_0_1(XTime,Xc),
  true,
  [wk(0, mr(intruder), mr(a), etc, 1, true, Xc)],
  [m(1, mr(a), mr(a), mr(b), crypt(pk(kb), c(nonce(c(na, XTime)), mr(a))))],
  wk(2, mr(b), mr(a), nonce(c(na, XTime)), 1, true, Xc),
  inknw(mr(a), mr(a), 1, Xc),
  inknw(mr(a), mr(b), 2, Xc),
  inknw(mr(a), pk(ka), 3, Xc),
  inknw(mr(a), primed(pk(ka)), 4, Xc),
  inknw(mr(a), pk(kb), 5, Xc)],
  [wk(0, mr(intruder), mr(a), etc, 1, true, Xc)]).

```

Remark: The language specified by a SATE specification can be infinite. This is usually not the case in STRIPS languages (e.g. PDDL).

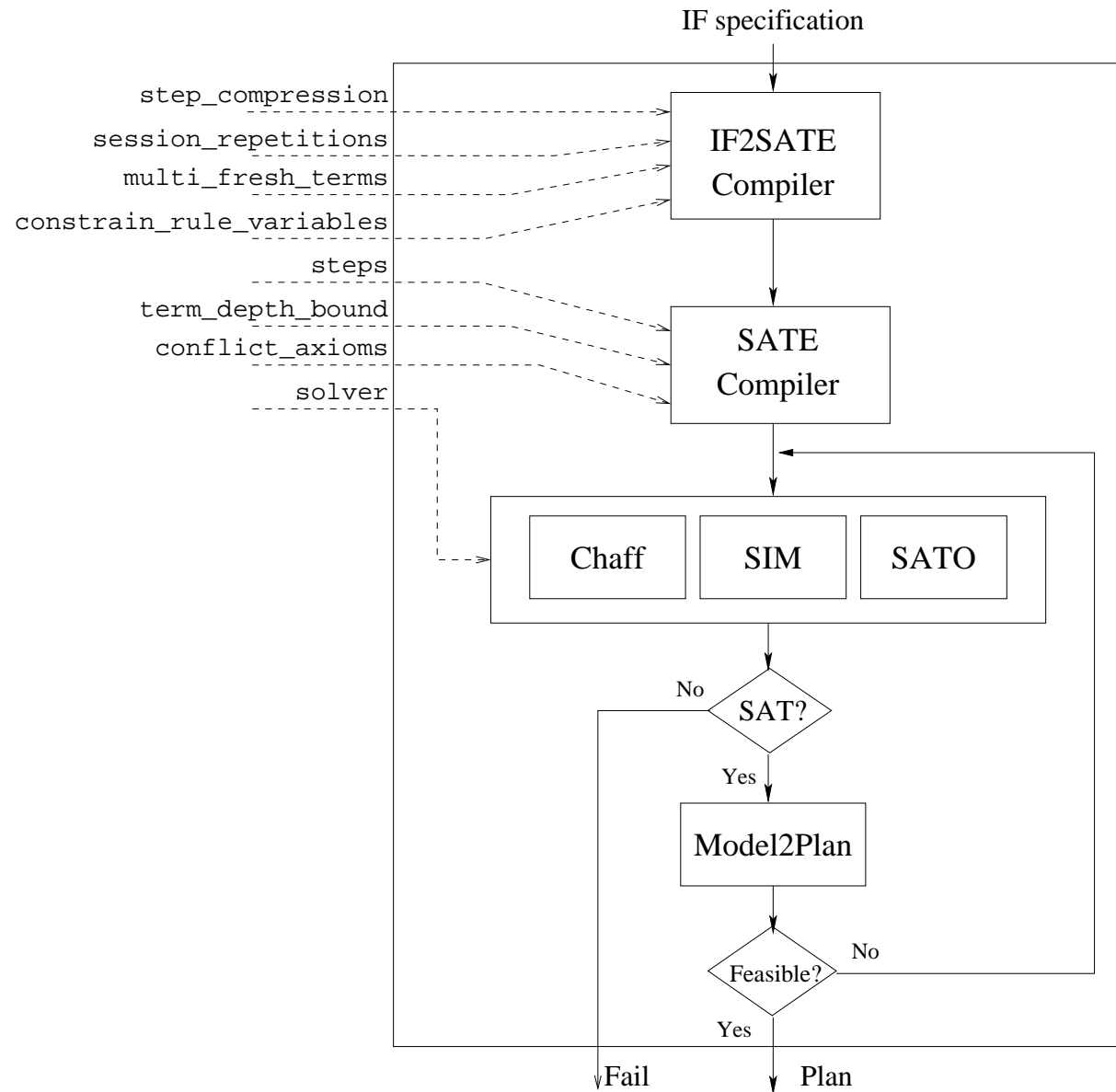
Performance of the back-ends

Experiments on Pentium III, 1.4GHz, 512Mb.

Protocol	Attack	OFMC	CL	SATMC
<i>ISO symm. key 1-pass unilateral auth.</i>	Replay	0.01	1.98	0.18/0.00
<i>ISO symm. key 2-pass mutual auth.</i>	Replay	0.01	3.86	0.43/0.01
<i>Andrew Secure RPC</i>	Type flaw	0.03	4.26	NA
	Replay	0.05	32.74	80.57/2.65
<i>ISO CCF 1-pass unilateral auth.</i>	Replay	0.00	2.23	0.17/0.00
<i>ISO CCF 2-pass mutual auth.</i>	Replay	0.01	4.55	0.46/0.01
<i>Needham-Schroeder Conventional Key</i>	Replay STS	0.31	63.43	29.25/0.39
<i>Denning-Sacco (symmetric)</i>	Type flaw	0.02	15.98	NA
<i>Otway-Rees</i>	Type flaw	0.02	10.71	NA
<i>Needham-Schroeder Signature</i>	Man-in-the-middle	0.13	53.88	3.77/0.05
<i>Neuman Stubblebine initial part</i>	Type flaw	0.03	6.19	NA
<i>Neuman Stubblebine repeated part</i>	Replay STS	0.03	3.54	15.17/0.21
<i>Neuman Stubblebine (complete)</i>	Type flaw	0.04	46.78	NA
<i>SPLICE/AS auth.</i>	Replay	4.02	352.42	5.48/0.05
<i>Needham-Schroeder Public Key</i>	Man-in-the-middle	0.04	12.91	1.77/0.05

Protocol	Attack	OFMC	CL	SATMC
<i>Yahalom with Lowe's alteration</i>	Type flaw	0.02	44.08	NA
<i>Woo-Lam Π_1</i>	Type flaw	0.01	0.81	NA
<i>Woo-Lam Π_2</i>	Type flaw	0.01	0.80	NA
<i>Woo-Lam Π_3</i>	Type flaw	0.00	0.82	NA
<i>Woo-Lam Π</i>	Parallel-session	0.24	1074.95	3.31/0.04
<i>Woo-Lam Mutual auth.</i>	Parallel-session	0.27	245.56	1024.08/7.95
<i>Kao Chow Repeated auth. , 1</i>	Replay STS	0.45	76.82	16.34/0.17
<i>Kao Chow Repeated auth. , 2</i>	Replay STS	0.48	45.25	339.70/2.11
<i>Kao Chow Repeated auth. , 3</i>	Replay STS	0.49	50.09	1288/MO
<i>ISO public key 1-pass unilateral auth.</i>	Replay	0.02	4.23	0.32/0.00
<i>ISO public key 2-pass mutual auth.</i>	Replay	0.01	11.06	1.18/0.01
<i>Needham-Schroeder Public Key with key server</i>	Man-in-the-middle	1.12	TO	4.29/0.04
<i>Needham-Schroeder with Lowe's fix</i>	Type flaw	0.01	31.12	NA
<i>Kehne Langendorfer Schoenwalder (repeated part)</i>	Parallel-session	0.20	199.43	MO/-
<i>Hwang and Chen's modified SPLICE</i>	Man-in-the-middle	0.02	13.10	NS
<i>Denning Sacco Key Distribution with Public Key</i>	Man-in-the-middle	0.52	936.90	NS
<i>Shamir Rivest Adelman Three Pass</i>	Type flaw	0.03	0.70	NA
<i>Encrypted Key Exchange</i>	Parallel-session	0.10	240.77	75.39/1.78
<i>Davis Swick Private Key Certificates, 1</i>	Type flaw	0.05	106.15	NA
	Replay	1.19	TO	1.37/0.02
<i>Davis Swick Private Key Certificates, 2</i>	Type flaw	0.16	348.49	NA
	Replay	0.86	TO	2.68/0.03
<i>Davis Swick Private Key Certificates, 3</i>	Replay	0.03	2.68	1.50/0.02
<i>Davis Swick Private Key Certificates, 4</i>	Replay	0.04	35.97	8.18/0.13

Architecture of SATMC



Optimizations: Fluent Splitting

- Since $\langle j, s, r, c \rangle$ is a **key** for $w(j, s, r, ak, ik, c)$, **replace** $w(j, s, r, ak, ik, c)$ with the conjunction of (new predicates) $wk(j, s, r, ak, c)$ and $inknow(j, s, r, ik, c)$.
- Similar considerations allow us to simplify $inknow(j, s, r, ik, c)$ to $inknow(r, ik, c)$.
- Replace $wk(j, s, r, [ak_1, \dots, ak_l], c)$ with:

$$wk(j, s, r, ak_1, 1, c), \dots, wk(j, s, r, ak_l, l, c)$$

The number of wk terms reduces from $O(|\text{text}|^l)$ to $O(l * |\text{text}|)$.

Optimizations: Reducing the Number of Conflict Exclusion Axioms

Problem: The number of Conflict Exclusion Axioms grows quadratically in the number of actions.

Solution: exploit the monotonicity of the intruder knowledge. Since a monotonic fluent never appears in the delete list of some action, then it cannot be a cause of a conflict.

Example: IF rewrite rules of the form:

$$i(\langle xM1, xM2 \rangle) \Rightarrow i(xM1) \cdot i(xM2)$$

are replaced by:

$$i(\langle xM1, xM2 \rangle) \Rightarrow i(xM1) \cdot i(xM2) \cdot i(\langle xM1, xM2 \rangle)$$