

Towards Verification with no False Attack of Security Protocols in First-order Logic

Reynald Affeldt and Hubert Comon-Lundh

Research Center for Information Security (RCIS),
National Institute of Advanced Industrial Science and Technology (AIST)

JSIAM-FAIS

September 18, 2008

Motivation

Successful automatic verifications of protocols in first-order logic:

- With general-purpose theorem provers (e.g., SPASS [C. Weidenbach, 1999])
- With specialized tools (e.g., ProVerif [B. Blanchet, 2001])

Problem: False attacks because of modeling approximations

- Known issue
- Sound approximations w.r.t. the freshness of nonces (in the case of verification for an unbounded number of sessions)
- Sound approximations w.r.t. the execution order of protocols rules (this cannot easily be fixed by encoding state information)

Our goal: Security proofs with standard theorem provers,
discarding all false attacks,
with termination for a bounded number of sessions

How to Avoid False Attacks while Using Standard Theorem Provers?

<p>Our approach: Use <i>rigid variables</i> [P. Andrews, 1981]</p> <p>$\left. \begin{array}{l} \text{our} \\ \text{result} \end{array} \right\} \begin{array}{l} + \text{ translation to first-order logic} \\ + \text{ complete and terminating resolution strategy} \end{array}$</p>

In first-order logic protocol models, the intruder instantiates variables:

- with the name of agents it wants to attack,
 - with made-up messages, etc.
- as many times as it wants
- ⇒ This enables construction/decomposition of arbitrary messages
- ⇒ But this also allows arbitrary replays of protocol rules!

With rigid variables:

- The intruder can still instantiate a variable with an arbitrary message
- But it has to commit to this one message

Rigidity has already been applied to verification of protocols:

- Decision procedure for rigid clauses in [DeLaune, Lin, and Lynch, LPAR 2007]

Outline

1. **False attacks in first-order logic models of protocols**
2. Rigid variables to avoid false attacks
3. Rigid resolution implemented with standard techniques

First-order Model of Protocols

The Intruder Model

Logical formulation of Dolev-Yao:

- Function symbols to build messages: $\langle \cdot, \cdot \rangle$, $[\cdot]$. (sym. encr.), *etc.*
- A predicate “ I ” to model the knowledge of the intruder
- Deduction rules for the intruder:

$$\text{Pairing/projections} \begin{cases} \forall x,y. I(x) \wedge I(y) \rightarrow I(\langle x, y \rangle) \\ \forall x,y. I(\langle x, y \rangle) \rightarrow I(x) \end{cases}$$

$$\text{Symetric encryption} \begin{cases} \forall x,y. I(x) \wedge I(y) \rightarrow I([x]_y) \\ \forall x,y. I([x]_y) \wedge I(y) \rightarrow I(x) \end{cases}$$

etc.

First-order Model of a Sample Protocol (1/3)

In Alice-and-Bob notation:

$$\begin{array}{l} A \rightarrow B : [A, N_0]_{K_{AB}} \\ B \rightarrow A : [B, N_0, N_1]_{K_{AB}}, [B, N_0, N_2]_{K_{AB}} \\ A \rightarrow B : N_1 \end{array}$$

Is $N_1 \oplus N_2$ kept secret?

First-order Model of a Sample Protocol (1/3)

In Alice-and-Bob notation:

$$\begin{array}{l} A \rightarrow B : [A, N_0]_{K_{AB}} \\ B \rightarrow A : [B, N_0, N_1]_{K_{AB}}, [B, N_0, N_2]_{K_{AB}} \\ A \rightarrow B : N_1 \end{array}$$

Is $N_1 \oplus N_2$ kept secret?

The same as a set of rules:

$$\text{A's role) } \mid \rightarrow I([A, N_0]_{K(A,B)})$$

First-order Model of a Sample Protocol (1/3)

In Alice-and-Bob notation:

$$\begin{array}{l} A \rightarrow B : [A, N_0]_{K_{AB}} \\ B \rightarrow A : [B, N_0, N_1]_{K_{AB}}, [B, N_0, N_2]_{K_{AB}} \\ A \rightarrow B : N_1 \end{array}$$

Is $N_1 \oplus N_2$ kept secret?

The same as a set of rules:

$$\begin{array}{l} \text{A's role)} \quad \left| \quad \rightarrow I \left([A, N_0]_{K(A,B)} \right) \right. \\ \text{B's role)} \quad \left| \quad I \left([A, x]_{K(A,B)} \right) \rightarrow I \left([B, x, N_1]_{K(A,B)}, [B, x, N_2]_{K(A,B)} \right) \right. \end{array}$$

First-order Model of a Sample Protocol (1/3)

In Alice-and-Bob notation:

$$\begin{array}{l} A \rightarrow B : [A, N_0]_{K_{AB}} \\ B \rightarrow A : [B, N_0, N_1]_{K_{AB}}, [B, N_0, N_2]_{K_{AB}} \\ A \rightarrow B : N_1 \end{array}$$

Is $N_1 \oplus N_2$ kept secret?

The same as a set of rules:

A's role)	$\rightarrow I([A, N_0]_{K(A,B)})$
B's role)	$I([A, x]_{K(A,B)}) \rightarrow I([B, x, N_1]_{K(A,B)}, [B, x, N_2]_{K(A,B)})$
A's role)	$I([B, N_0, y]_{K(A,B)}, [B, N_0, z]_{K(A,B)}) \rightarrow I(y)$

First-order Model of a Sample Protocol (1/3)

In Alice-and-Bob notation:

$$\begin{array}{l} A \rightarrow B : [A, N_0]_{K_{AB}} \\ B \rightarrow A : [B, N_0, N_1]_{K_{AB}}, [B, N_0, N_2]_{K_{AB}} \\ A \rightarrow B : N_1 \end{array}$$

Is $N_1 \oplus N_2$ kept secret?

The same as a set of rules:

$$\begin{array}{l|l} \text{A's role)} & \rightarrow I([A, N_0]_{K(A,B)}) \\ \text{B's role)} & I([A, x]_{K(A,B)}) \rightarrow I([B, x, N_1]_{K(A,B)}, [B, x, N_2]_{K(A,B)}) \\ \text{A's role)} & I([B, N_0, y]_{K(A,B)}, [B, N_0, z]_{K(A,B)}) \rightarrow I(y) \end{array}$$

Proof *ab absurdo* by assuming $\neg I(N_1, N_2)$

First-order Model of a Sample Protocol (2/3)

In Alice-and-Bob notation:

$$\begin{array}{l} A \rightarrow B : [A, N_0]_{K_{AB}} \\ B \rightarrow A : [B, N_0, N_1]_{K_{AB}}, [B, N_0, N_2]_{K_{AB}} \\ A \rightarrow B : N_1 \end{array}$$

Is $N_1 \oplus N_2$ kept secret?

The same as a set of rules:

(generalization: the names of agents are replaced with variables)

$$\begin{array}{l} \text{A's role) } \mid I(a) \wedge I(b) \rightarrow I([a, N_0]_{K(a,b)}) \\ \text{B's role) } \mid I([a, x]_{K(a,b)}) \rightarrow I([b, x, N_1]_{K(a,b)}, [b, x, N_2]_{K(a,b)}) \\ \text{A's role) } \mid I([b, N_0, y]_{K(a,b)}, [b, N_0, z]_{K(a,b)}) \rightarrow I(y) \end{array}$$

Proof *ab absurdo* by assuming $\neg I(N_1, N_2)$

First-order Model of a Sample Protocol (3/3)

In Alice-and-Bob notation:

$$\begin{array}{l} A \rightarrow B : [A, N_0]_{K_{AB}} \\ B \rightarrow A : [B, N_0, N_1]_{K_{AB}}, [B, N_0, N_2]_{K_{AB}} \\ A \rightarrow B : N_1 \end{array}$$

Is $N_1 \oplus N_2$ kept secret?

The same as a set of rules:

(approximation: freshness of nonces abstracted with dependencies)

$$\text{A's role) } I(a) \wedge I(b) \rightarrow I([a, N_0(a, b)]_{K(a, b)})$$

$$\text{B's role) } I([a, x]_{K(a, b)}) \rightarrow I([b, x, N_1(a, x, b)]_{K(a, b)}, [b, x, N_2(a, x, b)]_{K(a, b)})$$

$$\text{A's role) } I([b, N_0(a, b), y]_{K(a, b)}, [b, N_0(a, b), z]_{K(a, b)}) \rightarrow I(y)$$

Proof *ab absurdo* by assuming $\neg I(N_1(a, i, b), N_2(a, i, b))$

Tentative Verification of our Sample Protocol

In ProVerif:

```
pred I/1 elimVar,decompData. fun senc/2.
query I:(N1[a,i,b],N2[a,i,b]). reduc
(***) the intruder (***)
I:x & I:y          -> I:senc(x,y)      ;
I:x & I:senc(y, x) -> I:y              ;
(***) the protocol (***)
I:a & I:b          -> I:senc((a,N0[a,b]), K[a,b])      ; (* rule 1 *)
I:senc((a,x), K[a,b]) -> I:(senc((b,x,N1[a,x,b]),K[a,b]), K[a,b]) ; (* rule 2 *)
                      senc((b,x,N2[a,x,b]),K[a,b])) ;
I:(senc((b,N0[a,b],y),K[a,b]),senc((b,N0[a,b],z),K[a,b])) (* rule 3 *)
-> I:y
.
```

A potential attack is found by applying, in this order:

(* rule 1 *), (* rule 2 *), (* rule 3 *), and... (* rule 3 *)!

This is a false attack!

(* rule 3 *) has been played twice in the same session

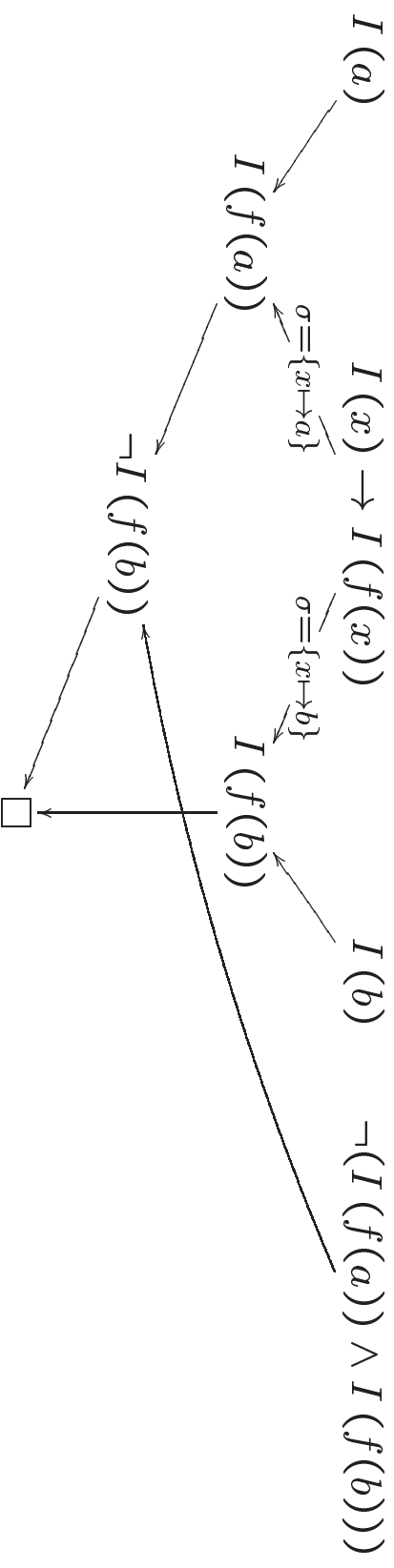
Outline

1. False attacks in first-order logic models of protocols
2. **Rigid variables to avoid false attacks**
3. Rigid resolution implemented with standard techniques

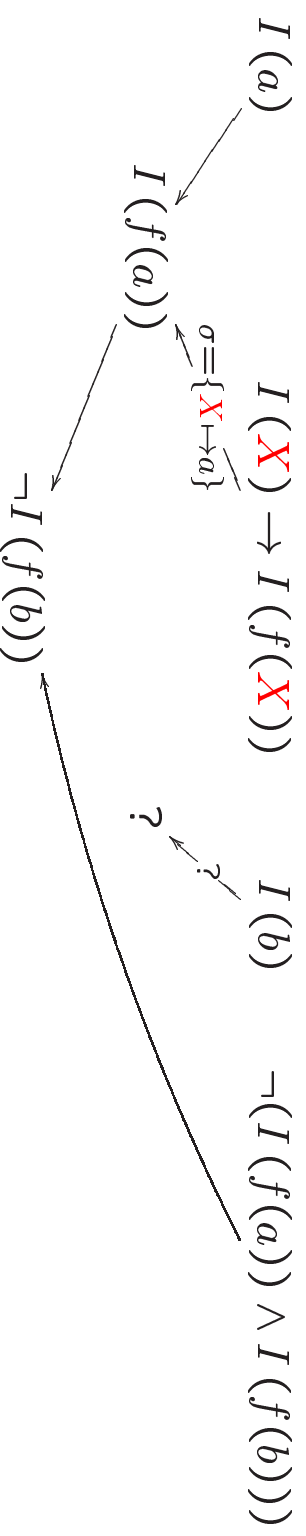
Flexible Variables vs. Rigid Variables

Consider $\forall x. \{I(a), I(x) \rightarrow I(f(x)), I(b), \neg(I(f(a)) \wedge I(f(b)))\}$

- With flexible variables: \square derivable



- With rigid variables: \square not derivable



Our Sample Protocol with Rigid Variables

Our sample protocol in first-order logic:

$$\begin{aligned}
 & \rightarrow I \left([A, N_0]_{K_{AB}} \right) \\
 & \rightarrow I \left(\langle [B, x, N_1]_{K_{AB}}, [B, x, N_2]_{K_{AB}} \rangle \right) \\
 I \left(\langle [B, N_0, y]_{K_{AB}}, [B, N_0, z]_{K_{AB}} \rangle \right) & \rightarrow I (y) \\
 I \left(\langle N_1, N_2 \rangle \right) & \rightarrow
 \end{aligned}$$

Just replace flexible variables with rigid variables:

$$\begin{aligned}
 & \rightarrow I \left([A, N_0]_{K_{AB}} \right) \\
 & \rightarrow I \left(\langle [B, X, N_1]_{K_{AB}}, [B, X, N_2]_{K_{AB}} \rangle \right) \\
 I \left(\langle [B, N_0, Y]_{K_{AB}}, [B, N_0, Z]_{K_{AB}} \rangle \right) & \rightarrow I (Y) \\
 I \left(\langle N_1, N_2 \rangle \right) & \rightarrow
 \end{aligned}$$

\Rightarrow The 3rd rule cannot be played twice anymore

\Rightarrow The previous false attack has disappeared

Difficulty in Implementing Rigid Resolution

Direct implementation of rigid resolution requires backtracking (hence the complications in [Delaune, Lin, and Lynch, LPAR 2007])

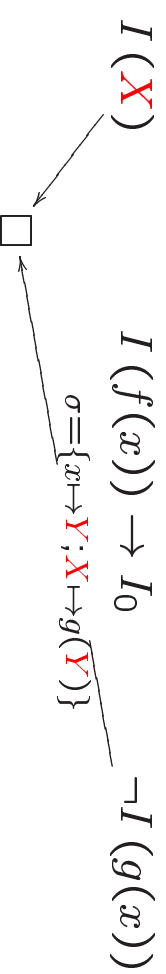
Consider $\{I(X), I(f(x)) \rightarrow I_0, \neg I(g(x))\}$

- 1st tentative: we cannot conclude



X has been assigned a f -headed term and $I(X)$ cannot be used anymore, \Rightarrow backtracking required

- 2nd tentative: we can conclude



Outline

1. False attacks in first-order logic models of protocols
2. Rigid variables to avoid false attacks
3. **Rigid resolution implemented with standard techniques**

Translation to First-order Logic

Idea: Replace rigid variables with flexible ones and prepend a vector of these flexible variables in I

E.g.: 3 rigid variables X, Y, Z in

$$\begin{array}{l}
 \rightarrow I([A, N_0]_{K_{AB}}) \\
 \rightarrow I(\langle [B, X, N_1]_{K_{AB}}, [B, X, N_2]_{K_{AB}} \rangle) \\
 \rightarrow I(Y) \\
 \rightarrow I(\langle N_1, N_2 \rangle) \\
 \rightarrow I(x) \wedge I(y) \\
 \rightarrow I(\langle x, y \rangle) \\
 \rightarrow I(x) \\
 \dots
 \end{array}$$

become 3 flexible variables x, y, z and a vector x, y, z in

$$\begin{array}{l}
 \rightarrow I(x, y, z, [A, N_0]_{K_{AB}}) \\
 \rightarrow I(x, y, z, \langle [B, x, N_1]_{K_{AB}}, [B, x, N_2]_{K_{AB}} \rangle) \\
 \rightarrow I(x, y, z, y) \\
 \rightarrow I(x, y, z, \langle N_1, N_2 \rangle) \\
 \rightarrow I(x, y, z, x') \wedge I(x, y, z, y') \\
 \rightarrow I(x, y, z, x') \\
 \dots
 \end{array}$$

Theorem: Both problems are equivalent w.r.t. satisfiability

A Decidable Fragment of First-order Logic

Overview

- Rules with atoms of the form $I(\bar{x}, t)$ such as:

protocol rules: $I(\bar{x}, s) \rightarrow I(\bar{x}, t)$ with $\text{Var}(t) \subseteq \text{Var}(s) \subseteq \bar{x}$

intruder rules: $I(\bar{x}, y_0), \dots, I(\bar{x}, y_{n-1}) \rightarrow I(\bar{x}, f(y_0, \dots, y_{n-1}))$
with $\bar{x} \cap \{y_0, \dots, y_{n-1}\} = \emptyset$

- Resolution with free selection for Horn clauses:

$$\frac{C \rightarrow A \quad A', C'' \rightarrow C'}{C \rightarrow C'}$$

with A and A' subterm-maximal atoms (preferentially negative)
whose rightmost term is not a variable

- Elimination of redundant clauses

Theorem: The above strategy is complete and terminating

Mechanized Illustration (1/3)

One session of our sample protocol with rigid variables (\textcircled{x} , \textcircled{y} , \textcircled{z}):

```

~ I[m] \ / ~ I[k] \ / I[senc{m,k}] //
~ I[senc{m,k}] \ / ~ I[k] \ / I[m] //
~ I[m] \ / ~ I[k] \ / I[pair{m,k}] //
~ I[m] \ / ~ I[k] \ / ~ I[1] \ / I[triple{m,k,1}] //
~ I[pair{m,k}] \ / I[m] //
~ I[pair{m,k}] \ / I[k] //
~ I[triple{m,k,1}] \ / I[m] //
~ I[triple{m,k,1}] \ / I[k] //
~ I[triple{m,k,1}] \ / I[1] //
--
/**** I([A, N0]KAB) ****/
-> I[senc{pair{A,  $\textcircled{x}$ }, %KAB}] /\

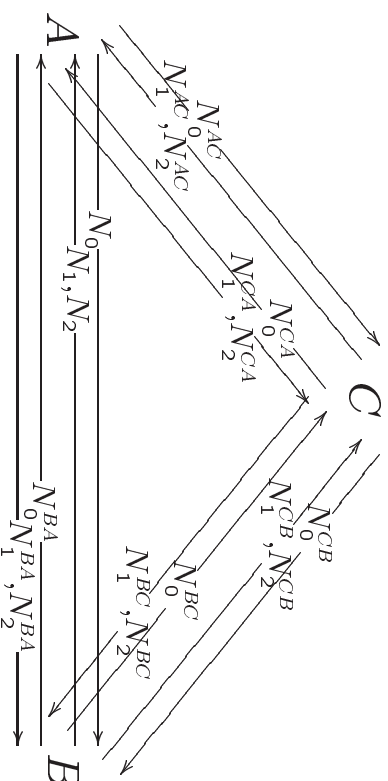
/**** I([A, X]KAB) -> I([B, X, N1]KAB, [B, X, N2]KAB) ****/
I[senc{pair{A,  $\textcircled{x}$ }, %KAB}] ->
    I[pair{senc{triple{B,  $\textcircled{x}$ , %N1}, %KAB}, senc{triple{B,  $\textcircled{x}$ , %N2}, %KAB}}] /\

/**** I([B, N0, Y]KAB, [B, N0, Z]KAB) -> I(Y) ****/
I[pair{senc{triple{B, %N0,  $\textcircled{y}$ }, %KAB}, senc{triple{B, %N0,  $\textcircled{z}$ }, %KAB}}] -> I[ $\textcircled{y}$ ]
--
/**** ~I(N1, N2) ****/
~ I[pair{N1{ }, N2{ }}]

```

Mechanized Illustration (2/3)

With two honest agents A,B and one corrupted agent C:



```

/** the attacker knows the keys of the corrupted agent:  $I(K_{BC}) \wedge I(K_{AC})$  */
I[KBC{}] \ I[KAC{}]
--
/** session A  $\rightarrow$  B */
A := A{}, NO := NO{}, KAB := KAB{}; B := B{}, N1 := N1{}, N2 := N2{};
/** session B  $\rightarrow$  A */
A := B{}, NO := NOBA{}, KAB := KAB{}; B := A{}, N1 := N1BA{}, N2 := N2BA{};
/** session A  $\rightarrow$  C */
A := C{}, NO := NOAC{}, KAB := KAC{}; B := C{}, N1 := N1AC{}, N2 := N2AC{};
/** session C  $\rightarrow$  A */
A := C{}, NO := NOCA{}, KAB := KAC{}; B := A{}, N1 := N1CA{}, N2 := N2CA{};
/** session B  $\rightarrow$  C */
A := B{}, NO := NOBC{}, KAB := KBC{}; B := C{}, N1 := N1BC{}, N2 := N2BC{};
/** session C  $\rightarrow$  B */
A := C{}, NO := NOCB{}, KAB := KBC{}; B := B{}, N1 := N1CB{}, N2 := N2CB{};

```

Mechanized Illustration (3/3)

Results obtained with a home-made theorem prover*

Resolution on our sample protocol (6-sessions case):

- Does not terminate with traditional strategies
(standard binary resolution, positive, ordered with subterm and lpo)
- Terminates with our strategy (after working off approx. 200 clauses)

Other results:

- Insecurity of Otway Rees, Needham-Schroeder public key
- Security of Yahalom, Lowe-Needham-Schroeder public key
- etc.

*around 2000 lines of OCaml

Conclusion

- Useful observation (the translation rigid \rightarrow flexible) for encoding the problem of security for a bounded number of sessions:
- It leads to decidable fragments of first-order logic
 - It extends to public-key encryption
 - Adding parameters for ordering, it avoids all false attacks

Future work:

- Evaluate complexity of resolution
- Extend to other cryptographic constructs

blank slide

Non-Termination Issue

Problem: An inappropriate strategy may cause looping

E.g.: Consider subterm-ordered resolution:

