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?

Our approach: Use *rigid variables* [P. Andrews, 1981]

our result $\left\{ \begin{array}{l} + \text{ translation to first-order logic} \\ + \text{ complete and terminating resolution strategy} \end{array} \right.$

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

$\Rightarrow$ This enables construction/decomposition of arbitrary messages

$\Rightarrow$ 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: $\langle \cdot, \cdot \rangle$, $[\cdot]_\cdot$ (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:

$$\begin{array}{lcl} A \to B & : & [A, N_0]_{K_{AB}} \\ B \to A & : & [B, N_0, N_1]_{K_{AB}}, [B, N_0, N_2]_{K_{AB}} \\ A \to B & : & N_1 \end{array}$$

Is $N_1 \oplus N_2$ kept secret?

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

In Alice-and-Bob notation:

$$\begin{array}{rcl} A \to B & : & [A, N_0]_{K_{AB}} \\ B \to A & : & [B, N_0, N_1]_{K_{AB}}, [B, N_0, N_2]_{K_{AB}} \\ A \to B & : & N_1 \end{array}$$

Is $N_1 \oplus N_2$ kept secret?

The same as a set of rules:

A's role) $\Big|$ $\quad \to I\left([A, N_0]_{K(A,B)}\right)$

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

In Alice-and-Bob notation:

$$
\begin{array}{rcl}
A \to B & : & [A, N_0]_{K_{AB}} \\
B \to A & : & [B, N_0, N_1]_{K_{AB}}, [B, N_0, N_2]_{K_{AB}} \\
A \to B & : & N_1
\end{array}
$$

Is $N_1 \oplus N_2$ kept secret?

The same as a set of rules:

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

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

In Alice-and-Bob notation:

$$\begin{array}{rcl} A \to B & : & [A, N_0]_{K_{AB}} \\ B \to A & : & [B, N_0, N_1]_{K_{AB}}, [B, N_0, N_2]_{K_{AB}} \\ A \to B & : & N_1 \end{array}$$

Is $N_1 \oplus N_2$ kept secret?

The same as a set of rules:

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

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

In Alice-and-Bob notation:

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

Is $N_1 \oplus N_2$ kept secret?

The same as a set of rules:

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

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:

$$\begin{array}{rcl} A \to B & : & [A, N_0]_{K_{AB}} \\ B \to A & : & [B, N_0, N_1]_{K_{AB}}, [B, N_0, N_2]_{K_{AB}} \\ A \to B & : & N_1 \end{array}$$

Is $N_1 \oplus N_2$ kept secret?

The same as a set of rules:

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

$$\begin{array}{r|l} \text{A's role)} & I(a) \wedge I(b) \to I\left([a, N_0]_{K(a,b)}\right) \\ \text{B's role)} & I\left([a, x]_{K(a,b)}\right) \to I\left([b, x, N_1]_{K(a,b)}, [b, x, N_2]_{K(a,b)}\right) \\ \text{A's role)} & I\left([b, N_0, y]_{K(a,b)}, [b, N_0, z]_{K(a,b)}\right) \to I(y) \end{array}$$

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:

$$\begin{array}{rcl} A \to B & : & [A, N_0]_{K_{AB}} \\ B \to A & : & [B, N_0, N_1]_{K_{AB}}, [B, N_0, N_2]_{K_{AB}} \\ A \to B & : & N_1 \end{array}$$

Is $N_1 \oplus N_2$ kept secret?

The same as a set of rules:

(approximation: freshness of nonces abstracted with dependencies)

A's role)	$I(a) \wedge I(b) \to I\left([a, N_0(a,b)]_{K(a,b)}\right)$
B's role)	$I\left([a, x]_{K(a,b)}\right) \to I\left([b, x, N_1(a,x,b)]_{K(a,b)}, [b, x, N_2(a,x,b)]_{K(a,b)}\right)$
A's role)	$I\left([b, N_0(a,b), y]_{K(a,b)}, [b, N_0(a,b), z]_{K(a,b)}\right) \to I(y)$

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

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]),   (* 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 $\quad {}^{\forall x.}\{I\,(a)\,,\;\; I\,(x) \rightarrow I\,(f(x))\,,\;\; I\,(b)\,,\;\; \neg(I\,(f(a)) \wedge I\,(f(b)))\}$

- With flexible variables: $\square$ derivable

$$
\begin{tikzcd}
I\,(a) \arrow[dr] & & I\,(x) \rightarrow I\,(f(x)) \arrow[dl, "\sigma=\{x\mapsto a\}"'] \arrow[dr, "\sigma=\{x\mapsto b\}"] & & I\,(b) \arrow[dl] & \neg(I\,(f(a)) \wedge I\,(f(b))) \arrow[ddlll] \\
& I\,(f(a)) \arrow[dr] & & I\,(f(b)) \arrow[dd] & & \\
& & \neg I\,(f(b)) \arrow[dr] & & & \\
& & & \square & &
\end{tikzcd}
$$

- With rigid variables: $\square$ not derivable

$$
\begin{tikzcd}
I\,(a) \arrow[dr] & & I\,(X) \rightarrow I\,(f(X)) \arrow[dl, "\sigma=\{X\mapsto a\}"'] & & I\,(b) \arrow[dl, "?"'] & \neg(I\,(f(a)) \wedge I\,(f(b))) \arrow[ddlll] \\
& I\,(f(a)) \arrow[dr] & & ? & & \\
& & \neg I\,(f(b)) & & &
\end{tikzcd}
$$

Our Sample Protocol with Rigid Variables

Our sample protocol in first-order logic:

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

Just replace flexible variables with rigid variables:

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

$\Rightarrow$ The 3$^{\text{rd}}$ 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 $\quad \{I(X), \quad I(f(x)) \to I_0, \quad \neg I(g(x))\}$

- 1^{st} tentative: we cannot conclude

$$\begin{tikzcd}
I(X) \arrow[dr] & & I(f(x)) \to I_0 \arrow[dl, "\sigma=\{x\mapsto Y;X\mapsto f(Y)\}"'] & \neg I(g(x)) \arrow[d, dashed, "?"] \\
& I(f(Y)),\ I_0 & & ?
\end{tikzcd}$$

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

- 2^{nd} tentative: we can conclude

$$\begin{tikzcd}
I(X) \arrow[dr] & & I(f(x)) \to I_0 & \neg I(g(x)) \arrow[dll, "\sigma=\{x\mapsto Y;X\mapsto g(Y)\}"] \\
& \square & &
\end{tikzcd}$$

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

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

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

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

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

A Decidable Fragment of First-order Logic

Overview

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

  protocol rules: $I\,(\overline{x}, s) \to I\,(\overline{x}, t)$ with $Var(t) \subseteq Var(s) \subseteq \overline{x}$

  intruder rules: $I\,(\overline{x}, y_0)\,, \ldots, I\,(\overline{x}, y_{n-1}) \to I\,(\overline{x}, f\,(y_0, \ldots, y_{n-1}))$
  with $\overline{x} \cap \{y_0, \ldots, y_{n-1}\} = \emptyset$

- Resolution with free selection for Horn clauses:

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

  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 (@x,@y,@z):

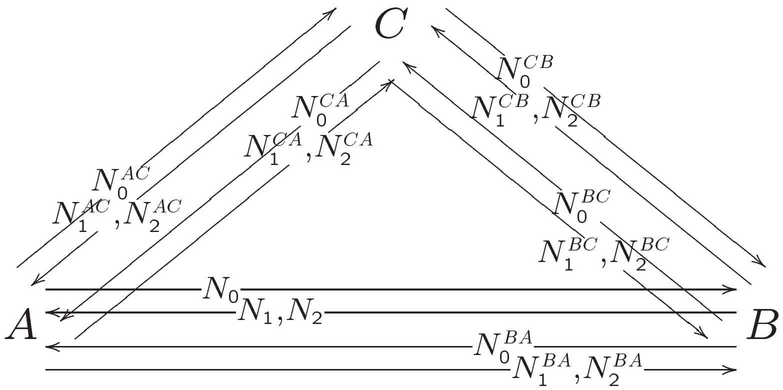
```
~ I[m] \/ ~ I[k] \/ I[senc{m,k}]                              /\
~ I[senc{m,k}] \/ ~ I[k] \/ I[m]                              /\
~ I[m] \/ ~ I[k] \/ I[pair{m,k}]                              /\
~ I[m] \/ ~ I[k] \/ ~ I[l] \/ I[triple{m,k,l}]                /\
~ I[pair{m,k}] \/ I[m]                                        /\
~ I[pair{m,k}] \/ I[k]                                        /\
~ I[triple{m,k,l}] \/ I[m]                                    /\
~ I[triple{m,k,l}] \/ I[k]                                    /\
~ I[triple{m,k,l}] \/ I[l]
--
/*** $I([A, N_0]_{K_{AB}})$ ***/
-> I[senc{pair{%A,%N0},%KAB}] /\

/*** $I([A, X]_{K_{AB}}) \rightarrow I([B, X, N_1]_{K_{AB}}, [B, X, N_2]_{K_{AB}})$ ***/
I[senc{pair{%A,@x},%KAB}] ->
        I[pair{senc{triple{%B,@x,%N1},%KAB},senc{triple{%B,@x,%N2},%KAB}}] /\

/*** $I([B, N_0, Y]_{K_{AB}}, [B, N_0, Z]_{K_{AB}}) \rightarrow I(Y)$ ***/
I[pair{senc{triple{%B,%N0,@y},%KAB},senc{triple{%B,%N0,@z},%KAB}}] -> I[@y]
--
/*** $\neg I(N1, N2)$ ***/
~ 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) ∧ I(K_AC) ***/
I[KBC{}] /\ I[KAC{}]
--
/*** session A → B ***/
A := A{}, N0 := N0{}, KAB := KAB{}, B := B{}, N1 := N1{}, N2 := N2{} ;
/*** session B → A ***/
A := B{}, N0 := N0BA{}, KAB := KAB{}, B := A{}, N1 := N1BA{}, N2 := N2BA{} ;
/*** session A → C ***/
A := C{}, N0 := N0AC{}, KAB := KAC{}, B := C{}, N1 := N1AC{}, N2 := N2AC{} ;
/*** session C → A ***/
A := C{}, N0 := N0CA{}, KAB := KAC{}, B := A{}, N1 := N1CA{}, N2 := N2CA{} ;
/*** session B → C ***/
A := B{}, N0 := N0BC{}, KAB := KBC{}, B := C{}, N1 := N1BC{}, N2 := N2BC{} ;
/*** session C → B ***/
A := C{}, N0 := N0CB{}, 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$\to$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:

$$I(y,u) \wedge I(y,v) \rightarrow \underline{I(y,\langle u,v\rangle)} \qquad\qquad \underline{I(x,x)} \rightarrow I(x,N)$$

$$\sigma = \{x \mapsto \langle u,v\rangle; y \mapsto \langle u,v\rangle\}$$

$$\underline{I(\langle u_1,v_1\rangle,u_1)} \wedge I(\langle u_1,v_1\rangle,v_1) \rightarrow I(\langle u_1,v_1\rangle,N)$$

$$\sigma = \{y \mapsto \langle\langle u,v\rangle,v_1\rangle; u_1 \mapsto \langle u,v\rangle\}$$

$$\underline{I(\langle\langle u_2,v_2\rangle,v_1\rangle,u_2)} \wedge I(\langle\langle u_2,v_2\rangle,v_1\rangle,v_2) \wedge I(\langle\langle u_2,v_2\rangle,v_1\rangle,v_1) \rightarrow I(\langle\langle u_2,v_2\rangle,v_1\rangle,N)$$

$$\sigma = \{\cdots\}$$

$$\vdots$$