Advanced Solver Technology

Helmut Seidl

TUM + DTU

2006
Part 2

Beyond Recognizability:
One-Variable and Flat Clauses
Outline of Part 2:

- Protocols with single blind copying
- One-variable Clauses
- Flat Clauses
- One-variable or Flat Clauses
- ... and Beyond :-)

... and Beyond :-)

3
1. Protocols with Single Blind Copying

Rules for exchanging messages:

{\{Alice, Na\}}_{pub(Bob)}

{\{Na, Nb\}}_{pub(Alice)}

{\{Nb\}}_{pub(Bob)}

Properties to be verified: secrecy, authenticity, ...
The Dolev-Yao Model:

- Messages are terms:

<table>
<thead>
<tr>
<th>representation</th>
<th>encrypt(m, k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>{m}_k</td>
<td>\langle m_1, m_2 \rangle</td>
</tr>
</tbody>
</table>

\[ \Rightarrow \quad \text{Distinct terms represent distinct messages} \quad \Rightarrow \]

\[ \Rightarrow \quad \text{perfect cryptography}. \quad \text{Thus, e.g.,} \]

\[ \{m\}_k = \{m'\}_{k'} \quad \text{iff} \quad m = m' \quad \text{and} \quad k = k' \]

- Intruder has full control over the network:

All messages are sent to the intruder and received from the intruder.
Single blind Copying:

The Needham-Schroeder public key example:

1. $A \rightarrow B : \{A, N_a\}_{K_b}$
2. $B \rightarrow A : \{N_a, N_b\}_{K_a}$
3. $A \rightarrow B : \{N_b\}_{K_b}$

Abstraction:

- Unbounded number of sessions !!
- Nonces may be non-fresh ??
Idea:

Model intruder’s knowledge with Horn clauses ...

1. $A \rightarrow B : \{A, N_a\}_{K_b}$ known($\{a, n_a\}_{k_b}$) $\Leftarrow$
2. $B \rightarrow A : \{N_a, N_b\}_{K_a}$ known($\{X, n_b\}_{k_a}$) $\Leftarrow$ known($\{a, X\}_{k_b}$)
3. $A \rightarrow B : \{N_b\}_{K_b}$ known($\{X\}_{k_b}$) $\Leftarrow$ known($\{n_a, X\}_{k_a}$)

Secrecy of $N_b$: $\Leftarrow$ known($n_b$).

$\Rightarrow$ One-variable clauses $\therefore$
Discussion:

- We abstracted all nonces to finitely many.
- Less severe (still safe) abstractions are possible ...

1. $A \rightarrow B : \{A, N_a\}_{K_b}$ ... 
2. $B \rightarrow A : \{N_a, N_b\}_{K_a}$ known($\{X, n_b(X)\}_{k_a}$) $\Leftarrow$ known($\{a, X\}_{k_b}$)
3. $A \rightarrow B : \{N_b\}_{K_b}$ ... 

The generated nonce is a function of the received nonce :-) 

Blanchet 2001

This is still one-variable !!!
Other Abilities of the Intruder:

\[
\begin{align*}
\text{known}(\text{encrypt}(X, Y)) & \iff \text{known}(X) \land \text{known}(Y) \\
& \quad \text{// Intruder can encrypt messages} \\
\text{known}(\langle X, Y \rangle) & \iff \text{known}(X) \land \text{known}(Y) \\
& \quad \text{// Intruder can form pairs} \\
\text{known}(X) & \iff \text{known}(\text{encrypt}(X, Y)) \land \text{known}(Y) \\
& \quad \text{// Intruder can decrypt messages} \\
\text{known}(X) & \iff \text{known}(\langle X, Y \rangle) \\
\text{known}(Y) & \iff \text{known}(\langle X, Y \rangle) \\
& \quad \text{// Intruder can unpair messages}
\end{align*}
\]

\[\implies \text{Flat clauses} !!!\]
2. One-variable Clauses

- Every predicate is monadic.
- Every clause contains at most one variable. ☺

Example:

\[ p(a(b(X), b(b(X)))) \iff q_1(c(c(d(X, X)))) \land q_2(b(X)) \]
Observation:

Terms containing exactly one variable behave like strings ...

\[ p(a(b(X), b(b(X)))) \iff q_1(c(c(d(X, X)))) \land q_2(b(X)) \]
Observation:

Terms containing exactly one variable behave like strings ...

\[ p(a(b(X), b(b(X)))) \iff q_1(c(c(d(X, X)))) \land q_2(b(X)) \]
Fact:

- The set of all non-ground one-variable trees forms a free monoid!
- Each such tree can be uniquely factored into primitive trees ...

\[ a(b(X), b(b(X))) = a(X, b(X)) \bullet b(X) \]
Fact:

- The set of all non-ground one-variable trees forms a free monoid!
- Each such tree can be uniquely factored into primitive trees ...

\[ a(b(X), b(b(X))) = a(X, b(X)) \bullet b(X) \]

- Unification of two primitive terms maps variable to variable or both variables to ground subterms ...

\[
\begin{align*}
\text{unify}(d(X, a(X)), d(Y, a(Y))) &= \{ X \mapsto Y \} \\
\text{unify}(d(X, a(b)), d(Y, Y)) &= \{ X \mapsto a(b), Y \mapsto a(b) \}
\end{align*}
\]
Idea:

Goal: Automata clauses:

\[ p(t) \iff q_1(X) \land \ldots \land q_k(X) \]

// \( t \) primitive

\[ p(t) \iff \quad \quad \quad \quad \quad // \quad t \text{ ground} \]

\[ p(X) \iff \]
Idea:

Goal: Automata clauses:

\[ p(t) \iff q_1(X) \land \ldots \land q_k(X) \]

//  \( t \) primitive

\[ p(t) \iff // \  t \  \text{ground} \]

\[ p(X) \iff \]

Technique:

- Introduce auxiliary predicates to factor complex heads ...
- normalize  :-(
(0) Factorization of heads:

\[ p(a(\text{\textcolor{blue}{b(X)}}, \text{\textcolor{blue}{b(X)}})) \Leftarrow q_1(c(c(d(X, X)))) \land q_2(b(X)) \]

is simulated by:

\[
\begin{align*}
p(a(Z, b(Z))) & \Leftarrow p_1(Z) \\
p_1(\text{\textcolor{blue}{b(X)}}) & \Leftarrow q_1(c(c(d(X, X)))) \land q_2(b(X))
\end{align*}
\]
(1) Resolution Step:

\[
p_1(b(X)) \iff q_1(c(c(d(X, X)))) \land q_2(b(X))
\]

\[
q_1(c(X)) \iff r(X) \land s(X)
\]

resolves into:

\[
p_1(b(X)) \iff r(c(d(X, X))) \land s(c(d(X, X))) \land q_2(b(X))
\]

- The terms in the new body get smaller :-)

18
(2) Resolution Step:

\[ h(X) \iff s_1(d(X, X)) \land s_2(X) \]

\[ s_1(d(b(a), X)) \iff p(X) \]

resolves into:

\[ h(b(a)) \iff p(b(a)) \land s_2(b(a)) \]

- The terms in the new body get **ground**  :-)}
(3) Resolution Step:

\[ h(X) \iff s_1(d(X, X)) \land s_2(X) \]

\[ s_1(d([a], [a])) \iff \]

resolves into:

\[ h([a]) \iff s_2([a]) \]

- The terms in the new body get ground :-)

Theorem

- Normalization of one-variable clauses is $\text{DEXPTIME}$-complete.
- Satisfiability of one-variable queries is is $\text{DEXPTIME}$-complete.
3. Flat Clauses

- Every predicate is monadic.
- Every literal contains at most one constructor ...

Example:

\[ p(a(X, X, Y)) \iff q(Z) \land p_1(f(Z, Y, Y)) \land p_2(g(X, Y, Y)) \]
Observation:

- These clauses generalize the Horn fragments of:
  - the Bernays-Schoenfinkel Class;
  - the Monadic Class.
Observation:

• These clauses generalize the Horn fragments of:
  ⇒ the Bernays-Schoenfinkel Class;
  ⇒ the Monadic Class.

• They are related to finite tree automata with equality constraints:

\[
p(f(Z_1, \ldots, Z_k)) \iff p_1(X_1) \land \ldots \land p_n(X_n)
\]

  // the \(Z_i\) not necessarily distinct
  // the \(X_j\) among the \(Z_i\)
Theorem:

- Normalization of flat clauses is $\text{DEXPTIME}$-complete.
- Satisfiability of flat queries is $\text{DEXPTIME}$-complete.

Idea:

Just normalize the set of clauses ...
One Typical Case:

\[
\begin{align*}
  & h(a(X, Y)) \iff p(f(X, Y, Z)) \land q(g(X, X)) \\
  & p(f(X, X, Z)) \iff s_1(X) \land s_2(Z)
\end{align*}
\]

resolves into:

\[
\begin{align*}
  & h(a(X, X)) \iff s_1(X) \land s_2(Z) \land q(g(X, X))
\end{align*}
\]

- The resolved complex goal disappears \( :-) \)
- Variables may be instantiated with variables \( :-)) \)
4. One-variable or Flat Clauses

Idea:

- Allow automata clauses which are either one-variable or flat :-)  
- Ensure that resolution between flat and one-variable clauses are flat or one-variable — whenever one is an automata clause ...
Observation:

Unification of a flat term $f(Z_1, \ldots, Z_k)$ with a one-variable term $t$ ...

- maps all variables to ground terms or
- maps $X$ to $X$ and all variables $Z_i$ to one-variable subterms ...

\[
\text{unify}(f(Z, Z, Y), f(a(b, X), a(X, b), c(X))) = \\
\{ X \mapsto b, Y \mapsto c(b), Z \mapsto a(b, b) \}
\]

\[
\text{unify}(f(Z, Z, Y), f(a(b, X), a(b, X), c(X))) = \\
\{ X \mapsto X, Y \mapsto c(X), Z \mapsto a(b, X) \}
\]
Example:

\[ p(X) \iff q(a(X, \boxed{b(X)})) \land p'(X) \]
\[ q(a(X, Y)) \iff q_1(X) \land q_2(Y) \]

resolves into:

\[ p(X) \iff q_1(X) \land q_2(b(X)) \land p'(X) \]

- If the automata clause is flat, the result is one-variable :-)
Example (cont.):

\[ p(X) \iff q(a(X, Y)) \land h(f(Y, Z)) \]
\[ q(a(X, \boxed{b(X)})) \iff q_1(X) \land q_2(X) \]

resolves into:

\[ p(X) \iff q_1(X) \land q_2(X) \land h(f(\boxed{b(X)}, Z)) \]

- If the automata clause is one-variable, the result need neither be one-variable nor flat :-(

30
Restriction:

- Every literal in the body containing a constructor also contains all variables of the clause :-)

Thus, e.g.,

\[
\begin{align*}
p(X) & \iff q(a(X, Y)) \land h(f(X, X, Y)) \\
q(a(X, b(X))) & \iff q_1(X) \land q_2(X)
\end{align*}
\]

resolves into:

\[
p(X) \iff q_1(X) \land q_2(X) \land h(f(X, X, b(X)))
\]

- If the automata clause is one-variable, the result then is also one-variable :-)

31
Warning:

- The new one-variable heads need not be primitive ...

\[
p(f(Y)) \iff q(a(X, Y)) \land h(X)
\]

\[
q(a(X, b(X))) \iff q_1(X) \land q_2(X)
\]

resolves into:

\[
p(f(b(X))) \iff q_1(X) \land q_2(X) \land h(X)
\]

- ... but they consist at most of a constructor applied to a proper subterm of a primitive term :-)
- ... which we will factor again :-))
Theorem: Verma, S. 2005

- Normalization of restricted flat or one-variable clauses is DEXPTIME-complete.
- Satisfiability of flat or one-variable queries is DEXPTIME-complete.

Corollary:

Secrecy for protocols with single blind copying is in DEXPTIME.
5. ... and Beyond?

Discussion:

- Single blind copying is weak :-(
- One-variable and restricted flat Horn clauses are contained in the clausal format of the initially extended Skolem class $S^+$ :-)
- In the class $S^+$, every functional non-ground term contains all variables ...
An Example Tree:

\[ f \]

\[ X \]

\[ a \]

\[ g \]

\[ X \]

\[ h \]

\[ X \]

\[ Y \]

\[ X \]

\[ h \]

... which again can be factored:
Warning:

Instantiation may introduce further factorization ...

Instantiating $X$ with $a$ yields ...
**Warning:**

Instantiation may introduce further factorization ...

[Diagram of factorization trees]
Observation:

Unification of extended regular terms over disjoint sets of variables can be performed in two phases ... 

\[ t_1 = f(a, X_1, Y_1) \quad t_2 = f(Y_2, X_2, g(X_2, Y_2)) \]

**Phase 1:** In each term some of the variables are equated or replaced by ground terms:

- \( Y_2 \mapsto a \)
- \( t'_1 = f(a, X_1, Y_1) \quad t'_2 = f(a, X_2, g(X_2, a)) \)

**Phase 2:** Variables of one term are mapped to extended regular suffices of the other ... 

- \( X_1 \mapsto X_2 \quad Y_1 \mapsto g(X_2, a) \)
Idea:

- Proceed similar to the one-variable or flat case :-)
- Use factorization of heads of automata clauses — but
- Do this factorization only when needed !
- For the factorization, automata clauses may additionally contain literals like \( p(X, Y, Z) \) ...

\[
\begin{align*}
p(f(X, Y, Z), Y, g(b)) & \iff p(X, Y, Z) \\
p(a(X, Y), f(X, Y, X), Y) & \iff q(X, Y) \\
q(X, Y) & \iff q_1(X) \land q_2(Y)
\end{align*}
\]

We obtain:
Theorem:

- For restricted flat and $k$-variable Horn clauses ($k$ fix), normalization can be performed in \( \text{DEXPTIME} \).
- For arbitrary $S^+$ Horn clauses, normalization can be performed in \( \text{DEXP}^2\text{TIME} \).

Corollary:

Secrecy for protocols with joint copying is decidable \( :-) \)
Summary:

\[ S^+ \]

\[ k + \text{flat} \]

\[ k \text{ vars} \quad \text{flat} \]

\[ PD \quad \mathcal{H}_1 \quad \mathcal{H}_3 \]