REWRITING TECHNIQUES APPLIED TO SECURITY PROTOCOLS

Hubert Comon-Lundh

h.comon-lundh@aist.go.jp

INTRODUCTION

- Security protocols and their formal analysis: a brief summary of the past 20 years of research.
  - Automatic verification tools
  - Formal models and operational semantics
  - Decision results
  - The role of rewriting techniques
- Goals of the lectures:
  - A rewriting-centered view
  - Results and open questions in rewriting related to security protocols.

SUMMARY

Part 1: Protocols: examples and semantics. Intruder deduction systems. Algebraic theories. Locality.

Part 2: Bounded number of sessions: deducibility constraints. The small attack property. Solving deducibility constraints: the Dolev-Yao case.

Part 3: Solving deducibility constraints in equational theories. The finite variant property and examples. The small attack property: constraint solving methodology. Combination problems and one-step deducibility constraints.

1. PROTOCOLS: EXAMPLES AND SEMANTICS

SUMMARY 1

Basic examples and definitions Security protocols, operational semantics.

Algebraic properties of security primitives Examples, relevant equational theories

Properties of intruder systems (1)

- Recognizability preservation
- Locality and locality proofs

The most popular example

How it is usually described:

$$\begin{aligned} A \rightarrow B: &\quad \{A, N_A\}_{pub(B)} \\ B \rightarrow A: &\quad \{N_A, N_B\}_{pub(A)} \\ A \rightarrow B: &\quad \{N_B\}_{pub(B)} \end{aligned}$$

where

- A, B are two agents,
- N_A, N_B are newly generated random number: *nonces*.
- $\{m\}_k$ models the encryption of the message m with the (public) key k
- $pub(X)$ is the public key of X, which is supposed to be now by all other agents.
- The security goal is a mutual authentication: if a, b are two agents running the protocol, at the end, a holds a nonce n_b generated by b and b holds a nonce n_a generated by agent a. These two nonces must also be unknown to the outside.

THE MOST POPULAR EXAMPLE (2)

A protocol is a finite set of *roles* (pattern-matching or spi-calculus version):

$$
\begin{array}{llcl}
A(a,b):\ \nu N_A & & \rightarrow & \{a, N_A\}_{pub(b)} \\
 & \{N_A, y\}_{pub(a)} & \rightarrow & \{y\}_{pub(b)}
\end{array}
$$

$$
\begin{array}{llcl}
B(a,b):\ \nu N_B & \{a, x\}_{pub(b)} & \rightarrow & \{x, N_B\}_{pub(a)} \\
 & \{N_B\}_{pub(b)} & \rightarrow &
\end{array}
$$

THE MOST POPULAR EXAMPLE (2)

A protocol is a finite set of *roles* (pattern-matching or spi-calculus version):

$$
\begin{array}{rrcl}
A(a,b) : \nu N_A & & \rightarrow & \{a, N_A\}_{pub(b)} \\
 & \{N_A, y\}_{pub(a)} & \rightarrow & \{y\}_{pub(b)}
\end{array}
$$

$$
\begin{array}{rrcl}
B(a,b) : \nu N_B & \{a, x\}_{pub(b)} & \rightarrow & \{x, N_B\}_{pub(a)} \\
 & \{N_B\}_{pub(b)} & \rightarrow &
\end{array}
$$

Applied π-calculus/ explicit destructors version:

$$
\begin{array}{rrcl}
A(a,b) : \nu N_A & & \rightarrow & \{a, N_A\}_{pub(b)} \\
 & y & \rightarrow & \text{if } \pi_1(\mathsf{dec}(y, priv(a))) = N_A \text{ then } \{\pi_2(\mathsf{dec}(y, priv(a)))\}_{pub(b)}
\end{array}
$$

$$
\begin{array}{rrcl}
B(b) : \nu N_B & x & \rightarrow & \{\pi_2(\mathsf{dec}(x, priv(b))), N_B\}_{pub(\pi_1(\mathsf{dec}(x, priv(b))))} \\
 & z & \rightarrow & \text{if } \mathsf{dec}(z, priv(b)) = N_B \text{ then } OK
\end{array}
$$

Implicitly, if some projection or decryption attempt fails, the message is not sent (and the process aborts)

THE MOST POPULAR EXAMPLE (3)

Any number of copies (the ! construction) of any instances of the roles may run concurrently:

$$\begin{aligned} P = \quad & A(a_1, b_1)! \parallel A(a_2, b_2)! \parallel \cdots \parallel A(a_n, b_n)! \parallel \cdots \\ & B(a_1)! \parallel B(a_2)! \parallel \cdots \parallel B(a_n)! \parallel \cdots \end{aligned}$$

THE MOST POPULAR EXAMPLE (3)

Any number of copies (the ! construction) of any instances of the roles may run concurrently:

$$\begin{aligned} P = \quad & A(a_1, b_1)! \parallel A(a_2, b_2)! \parallel \cdots \parallel A(a_n, b_n)! \\ & B(a_1)! \parallel B(a_2)! \parallel \cdots \parallel B(a_n)! \end{aligned}$$

For most security properties, a fixed number of instances is sufficient: 2 agents for secrecy, 3 agents for authentication... ([CLC03]).

The most popular example (3)

Any number of copies (the ! construction) of any instances of the roles may run concurrently:

$$\begin{aligned} P = \quad & A(a_1, b_1)! \parallel A(a_2, b_2)! \parallel \cdots \parallel A(a_n, b_n)! \\ & B(a_1)! \parallel B(a_2)! \parallel \cdots \parallel B(a_n)! \end{aligned}$$

For most security properties, a fixed number of instances is sufficient: 2 agents for secrecy, 3 agents for authentication... ([CLC03]).
The attackers controls the network: (s)he may intercept, delay, forge messages.

$$\forall I .\, P \parallel I$$

Defines a (infinitely branching, non-terminating) transition system.

ATTACKER'S CAPABILITIES (SIMPLE CASE)

From a set of messages T, I may forge any message that can be obtained using the rules

$$\frac{x \quad y}{[x]_y}\ \mathtt{Symenc} \qquad \frac{x \quad y}{< x, y >}\ \mathtt{Pair} \qquad \frac{x \quad y}{\{x\}_{pub(y)}}\ \mathtt{PubEnc}$$

$$\frac{[x]_y \quad y}{x}\ \mathtt{Dec} \qquad \frac{< x, y >}{x}\ \pi_1 \qquad \frac{< x, y >}{y}\ \pi_2 \qquad \frac{\{x\}_{pub(y)} \quad priv(y)}{x}\ \mathtt{Pdec}$$

Equivalently the second set of rules can be replaced by:

$$\frac{x \quad y}{\mathtt{Dec}(x, y)} \qquad \frac{x}{\pi_1(x)} \qquad \frac{x}{\pi_2(x)} \qquad \frac{x \quad y}{\mathtt{Pdec}(x, y)}$$

Together with the rewrite system

$$\begin{array}{rclcrclcrcl} \mathtt{Dec}([x]_y, y) & \rightarrow & x & & \pi_1(< x, y >) & \rightarrow & x & & \pi_2(< x, y >) & \rightarrow & y \\ \mathtt{PDec}(\{x\}_{pub(y)}, priv(y)) & \rightarrow & x & & & & & & & & \end{array}$$

INTRUDER CAPABILITIES (2)

I can *forge* t using T, if $T \vdash t$ using the intruder deduction system.

Equivalently, intruder capabilities are described by a convergent term rewriting system (modulo associativity and commutativity, see later).
I can forge t from T if

$$\exists \zeta. \ \ \zeta[T] \downarrow = t$$

ζ is the *recipe*: any term built with public symbols and data from corrupted agents with as many variables as elements of T. $\zeta[T] = \zeta\{x_1 \mapsto t_1; \ldots ; x_n \mapsto t_n\}$ if $T = \{t_1, \ldots, t_n\}$

Example:
$T = \{a, b, [s]_{<a,b>}\}$
$\zeta = \text{dec}(x_3, < x_1, x_2 >)$

$$\zeta[T] \downarrow =$$

Public symbols usually consist of all function symbols, except the constants representing private data and the symbol $priv()$ which builds private decryption keys.

OPERATIONAL SEMANTICS (EXAMPLE)

$$\begin{array}{lcll} A(a,b):\ \nu N_A & & \rightarrow & \{a, N_A\}_{pub(b)} \\ & y & \rightarrow & \text{if } \pi_1(\mathsf{dec}(y, priv(a))) = N_A \text{ then } \{\pi_2(\mathsf{dec}(y, priv(a)))\}_{pub(b)} \end{array}$$

$$\begin{array}{lcll} B(b):\ \nu N_B & x & \rightarrow & \text{let } x_1 = \pi_1(\mathsf{dec}(x, priv(b))), x_2 = \pi_2(\mathsf{dec}(x, priv(b))) \text{ in } \{x_1, N_B\}_{pub(x_2)} \\ & z & \rightarrow & \text{if } \mathsf{dec}(z, priv(b)) = N_B \text{ then } OK \end{array}$$

Consider $A(a,c) \| B(b)$. a, b honest and c is corrupted.

$$\left(\begin{array}{ll} a: & pub(c), priv(agenta), \ldots \\ b: & pub(a), priv(b), \ldots \\ I: & pub(b), priv(c), \ldots \end{array}\right) \xrightarrow{\{a, n_a\}_{pub(c)}} \left(\begin{array}{ll} a: & pub(c), priv(agenta), n_A, \\ & \mathtt{stage}1 \\ b: & pub(a), priv(b) \\ I: & pub(b), priv(c), \{a, n_A\}_{pub(c)} \end{array}\right)$$

$$\xrightarrow{\{a, n_a\}_{pub(b)}} \left(\begin{array}{ll} a: & pub(c), priv(agenta), n_A, \\ & \mathtt{stage}1 \\ b: & pub(a), priv(b), n_B \\ & x_1 = n_A, x_2 = a, \mathtt{stage}1 \\ I: & pub(b), priv(c), \{a, n_A\}_{pub(c)} \end{array}\right)$$

Operational semantics (Example cntd)

$$A(a,b):\ \nu N_A \quad \rightarrow \quad \{a, N_A\}_{pub(b)}$$
$$y \quad \rightarrow \quad \text{if } \pi_1(\mathsf{dec}(y, priv(a))) = N_A \text{ then } \{\pi_2(\mathsf{dec}(y, priv(a)))\}_{pub(b)}$$

$$B(b):\ \nu N_B \quad x \quad \rightarrow \quad \text{let } x_1 = \pi_1(\mathsf{dec}(x, priv(b))), x_2 = \pi_2(\mathsf{dec}(x, priv(b))) \text{ in } \{x_1, N_B\}_{pub(x_2)}$$
$$z \quad \rightarrow \quad \text{if } \mathsf{dec}(z, priv(b)) = N_B \text{ then } OK$$

$$\begin{pmatrix} a: & pub(c), priv(agenta), n_A, \\ & \mathtt{stage}1 \\ b: & pub(a), priv(b), n_B \\ & x_1 = n_A, x_2 = a, \mathtt{stage}1 \\ I: & pub(b), priv(c), \{a, n_A\}_{pub(c)} \end{pmatrix} \xrightarrow{\{n_A n_B\}_{pub(a)}} \begin{pmatrix} a: & pub(c), priv(agenta), n_A, \\ & \mathtt{stage}1 \\ b: & pub(a), priv(b), n_B \\ & x_1 = n_A, x_2 = a, \mathtt{stage}1 \\ I: & pub(b), priv(c), \{a, n_A\}_{pub(c)} \\ & \{n_A n_B\}_{pub(a)} \end{pmatrix}$$

$$\xrightarrow{\{n_A n_B\}_{pub(a)}} \begin{pmatrix} a: & pub(c), priv(agenta), n_A, \\ & \mathtt{stage}2, y = \{n_B\}_{pub(a)} \\ b: & pub(a), priv(b), n_B \\ & x_1 = n_A, x_2 = a, \mathtt{stage}1 \\ I: & pub(b), priv(c), \{a, n_A\}_{pub(c)} \\ & \{n_A n_B\}_{pub(a)} \end{pmatrix}$$

$$\xrightarrow{\{n_B\}_{pub(c)}} \begin{pmatrix} a: & pub(c), priv(agenta), n_A, \\ & \mathtt{stageend}, y = \{n_B\}_{pub(a)} \\ b: & pub(a), priv(b), n_B \end{pmatrix}$$

Security properties

Secrecy:
- If n_A is generated in an instance $A(a, b)$ in which both a and b are honest, then there is no state in which I can deduce n_A.
- If n_b is generated in an instance $B(b)$ in which both b and x_1 are honest, then there is no state in which I can deduce n_b.

Agreement: For each instance $A(a, b)$ in which both a and b are honest and a (resp. b) reached the end state, then $x_1 = n_A$ and dec$(y, priv(n_A)) = n_B$.

To be precise we should index variables, agent names, nonces, with the role instances. Agreement properties then require a mapping of roles; there are several possible definitions.

Equivalence properties For instance privacy (anonymity):

$$P(a, b) \sim_o P(b, a)$$

THE PRESERVATION OF RECOGNIZABILITY THEOREM

Theorem: Alternating two-way automata only accept recognizable languages.

THE PRESERVATION OF RECOGNIZABILITY THEOREM

Theorem: Alternating two-way automata only accept recognizable languages.

$$q(f(x_1, \ldots, x_n)) \quad \leftarrow \quad q_1(x_1), \cdots, q_n(x_n)$$

THE PRESERVATION OF RECOGNIZABILITY THEOREM

Theorem: Alternating two-way automata only accept recognizable languages.

$$\begin{array}{rcl} q(f(x_1,\ldots,x_n)) & \leftarrow & q_1(x_1),\cdots,q_n(x_n) \\ q(x_i) & \leftarrow & q_1(f(x_1,\ldots,x_n)) \end{array}$$

The Preservation of Recognizability Theorem

Theorem: Alternating two-way automata only accept recognizable languages.

$$
\begin{array}{rcl}
q(f(x_1,\ldots,x_n)) & \leftarrow & q_1(x_1),\cdots,q_n(x_n) \\
q(x_i) & \leftarrow & q_1(f(x_1,\ldots,x_n)) \\
q(x) & \leftarrow & q_1(x), q_2(x)
\end{array}
$$

THE PRESERVATION OF RECOGNIZABILITY THEOREM

Theorem: Alternating two-way automata only accept recognizable languages.

$$
\begin{array}{rcl}
q(f(x_1, \ldots, x_n)) & \leftarrow & q_1(x_1), \cdots, q_n(x_n) \\
q(x_i) & \leftarrow & q_1(f(x_1, \ldots, x_n)) \\
q(x) & \leftarrow & q_1(x), q_2(x)
\end{array}
$$

Application:
If L is a recognizable tree language, then the set of terms deducible from L by a DY intruder is also recognizable.

THE PRESERVATION OF RECOGNIZABILITY THEOREM

Theorem: Alternating two-way automata only accept recognizable languages.

$$\begin{aligned} q(f(x_1, \ldots, x_n)) &\leftarrow q_1(x_1), \cdots, q_n(x_n) \\ q(x_i) &\leftarrow q_1(f(x_1, \ldots, x_n)) \\ q(x) &\leftarrow q_1(x), q_2(x) \end{aligned}$$

Application:
If L is a recognizable tree language, then the set of terms deducible from L by a DY intruder is also recognizable.

$$\begin{aligned} I(\{x\}_y) &\leftarrow I(x), I(y) \\ I(< x, y >) &\leftarrow I(x), I(y) \\ I(x) &\leftarrow I(\{x\}_y), I(y) \\ I(x) &\leftarrow I(< x, y >) \\ I(y) &\leftarrow I(< x, y >) \end{aligned}$$

Tree automata and intruder deductions (cntd)

A generalization of the preservation theorem.

The H1 class [Nielson,Nielson, Seidl 2003; Goubault-Larrecq 2005].

$$\begin{array}{rcl} Q(f(x_1,\ldots,x_n)) & \leftarrow & Q_1(t_1),\ldots,Q_n(t_n) \\ Q(x) & \leftarrow & Q_1(t_1),\ldots,Q_n(t_n) \end{array}$$

Emptiness is DEXPTIME-complete.

$\rightarrow$ a protocol verification tool.

Tree automata with one memory

$$a \xrightarrow[b]{\top} q_a \qquad f(q_a, q_a) \xrightarrow[\lambda x_1 . h(x_1)]{1=2} q$$

$$g(q) \xrightarrow[\lambda x_1 . x_1]{\top} q \qquad g(q_a) \xrightarrow[\lambda x_1 . h(x_1)]{\top} q$$

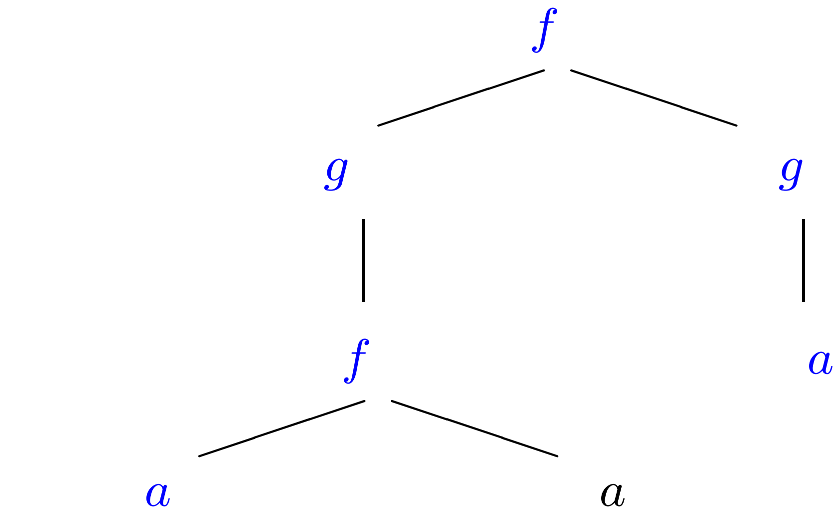
$$f(q, q) \xrightarrow[\lambda h(x_1) . x_1]{1=2} q$$

TREE AUTOMATA WITH ONE MEMORY

$$a \xrightarrow[b]{\top} q_a \qquad f(q_a, q_a) \xrightarrow[\lambda x_1.h(x_1)]{1=2} q$$

$$g(q) \xrightarrow[\lambda x_1.x_1]{\top} q \qquad g(q_a) \xrightarrow[\lambda x_1.h(x_1)]{\top} q$$

$$f(q, q) \xrightarrow[\lambda h(x_1).x_1]{1=2} q$$

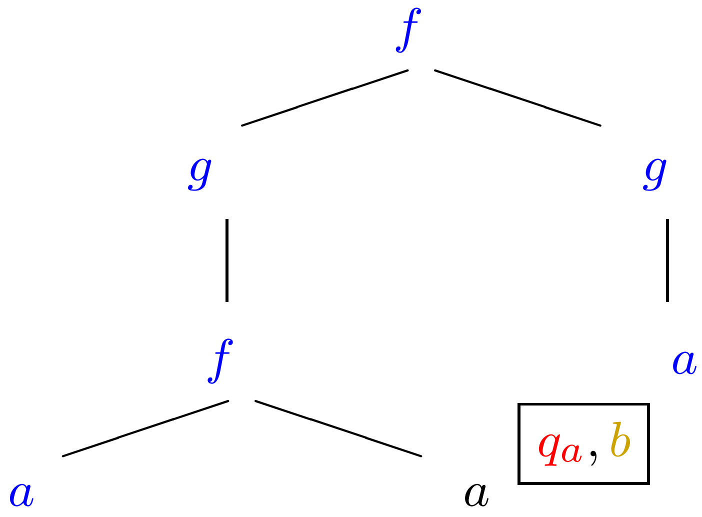


TREE AUTOMATA WITH ONE MEMORY

$$a \xrightarrow[b]{\top} q_a \qquad\qquad f(q_a, q_a) \xrightarrow[\lambda x_1 . h(x_1)]{1=2} q$$

$$g(q) \xrightarrow[\lambda x_1 . x_1]{\top} q \qquad\qquad g(q_a) \xrightarrow[\lambda x_1 . h(x_1)]{\top} q$$

$$f(q, q) \xrightarrow[\lambda h(x_1) . x_1]{1=2} q$$

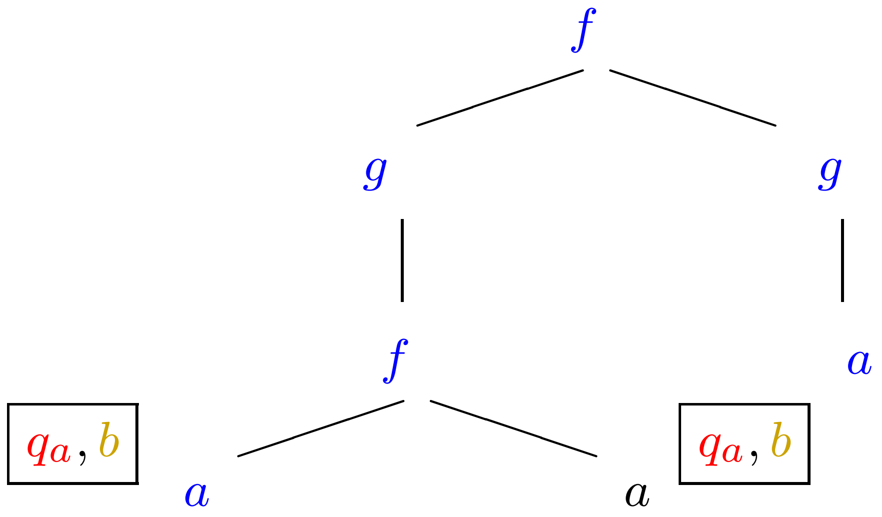


Tree automata with one memory

$$a \xrightarrow[b]{\top} q_a \qquad f(q_a, q_a) \xrightarrow[\lambda x_1.h(x_1)]{1=2} q$$

$$g(q) \xrightarrow[\lambda x_1.x_1]{\top} q \qquad g(q_a) \xrightarrow[\lambda x_1.h(x_1)]{\top} q$$

$$f(q, q) \xrightarrow[\lambda h(x_1).x_1]{1=2} q$$

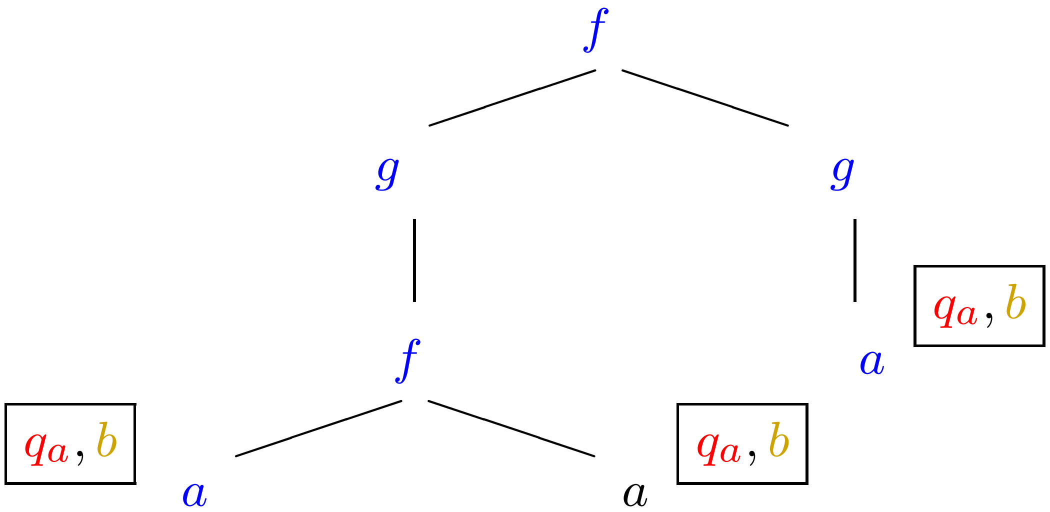


TREE AUTOMATA WITH ONE MEMORY

$$a \xrightarrow[b]{\top} q_a \qquad f(q_a, q_a) \xrightarrow[\lambda x_1.h(x_1)]{1=2} q$$

$$g(q) \xrightarrow[\lambda x_1.x_1]{\top} q \qquad g(q_a) \xrightarrow[\lambda x_1.h(x_1)]{\top} q$$

$$f(q, q) \xrightarrow[\lambda h(x_1).x_1]{1=2} q$$

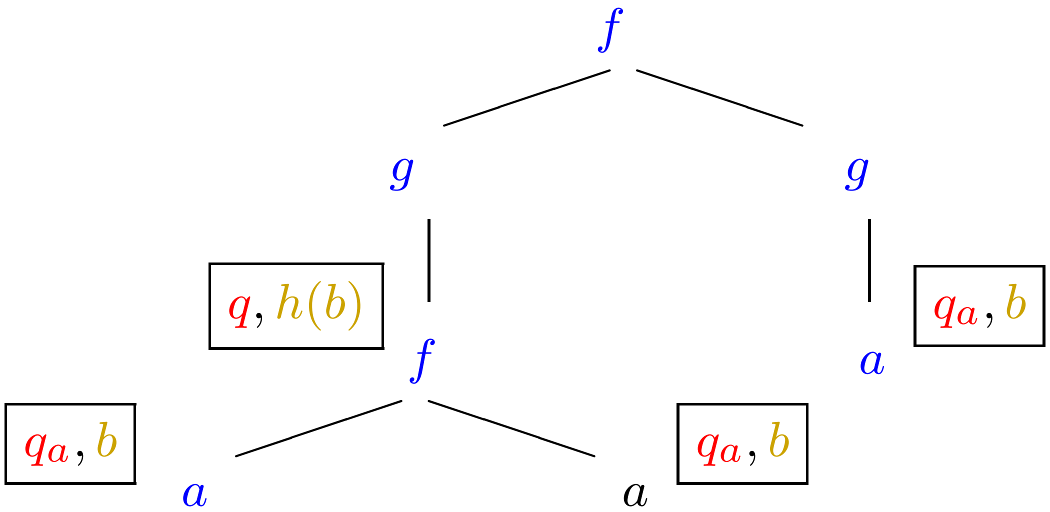


Tree automata with one memory

$$a \xrightarrow[b]{\top} q_a \qquad f(q_a, q_a) \xrightarrow[\lambda x_1.h(x_1)]{1=2} q$$

$$g(q) \xrightarrow[\lambda x_1.x_1]{\top} q \qquad g(q_a) \xrightarrow[\lambda x_1.h(x_1)]{\top} q$$

$$f(q, q) \xrightarrow[\lambda h(x_1).x_1]{1=2} q$$

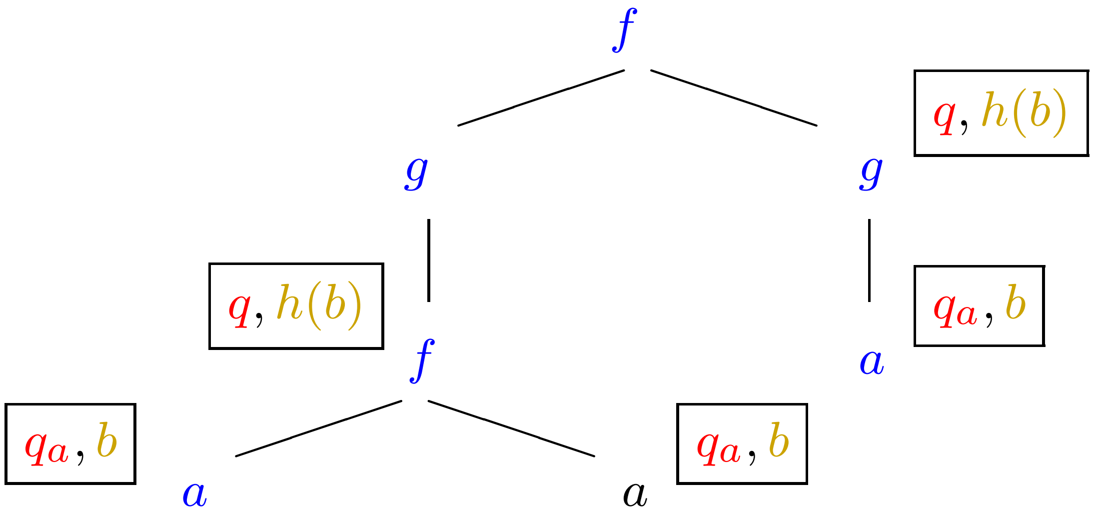


Tree automata with one memory

$$a \xrightarrow[b]{\top} q_a \qquad f(q_a, q_a) \xrightarrow[\lambda x_1 . h(x_1)]{1=2} q$$

$$g(q) \xrightarrow[\lambda x_1 . x_1]{\top} q \qquad g(q_a) \xrightarrow[\lambda x_1 . h(x_1)]{\top} q$$

$$f(q, q) \xrightarrow[\lambda h(x_1) . x_1]{1=2} q$$

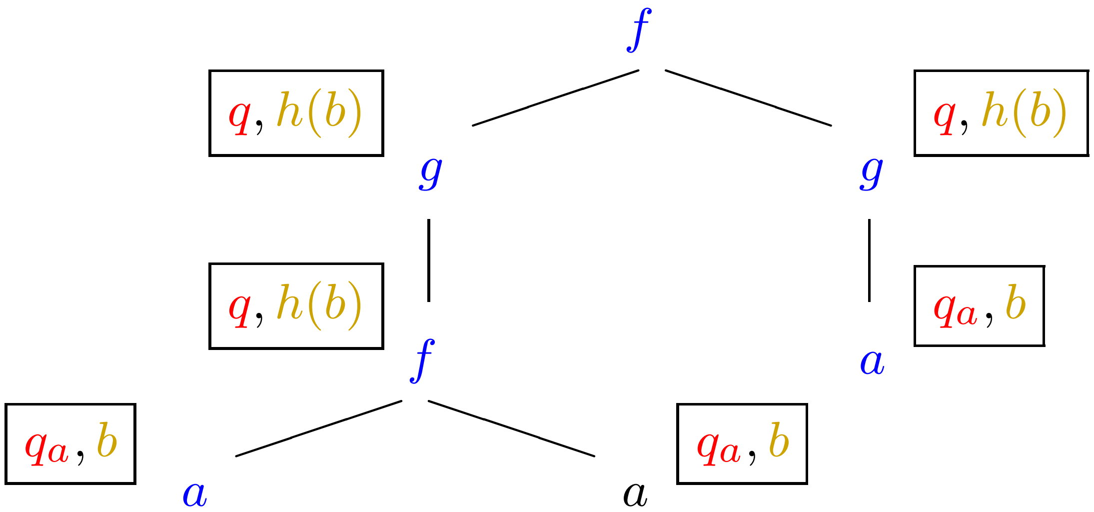


Tree automata with one memory

$$a \xrightarrow[b]{\top} q_a \qquad f(q_a, q_a) \xrightarrow[\lambda x_1.h(x_1)]{1=2} q$$

$$g(q) \xrightarrow[\lambda x_1.x_1]{\top} q \qquad g(q_a) \xrightarrow[\lambda x_1.h(x_1)]{\top} q$$

$$f(q, q) \xrightarrow[\lambda h(x_1).x_1]{1=2} q$$

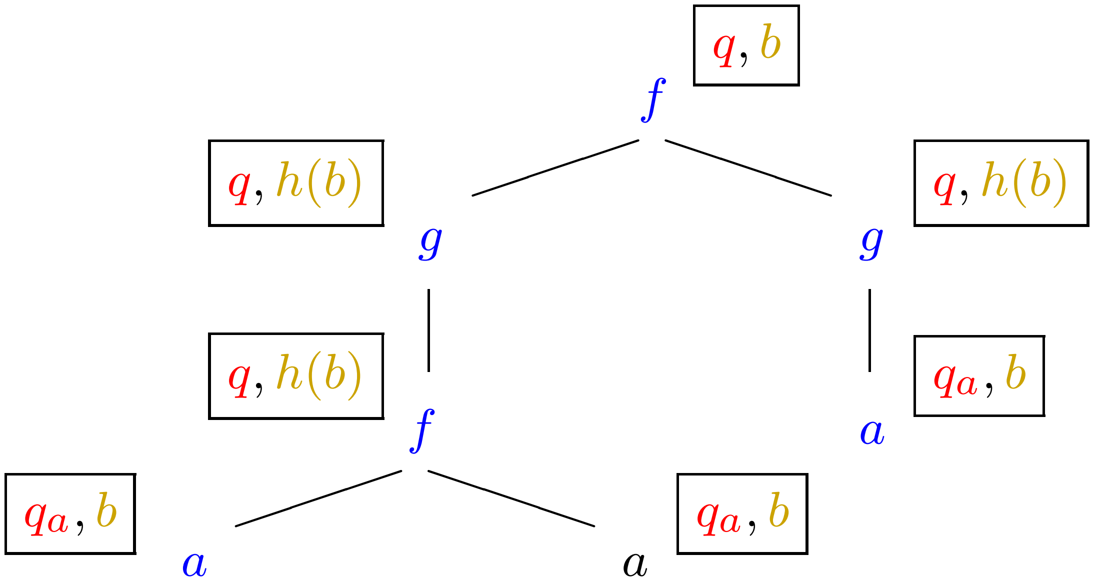


TREE AUTOMATA WITH ONE MEMORY

$$a \xrightarrow[b]{\top} q_a \qquad f(q_a, q_a) \xrightarrow[\lambda x_1 . h(x_1)]{1=2} q$$

$$g(q) \xrightarrow[\lambda x_1 . x_1]{\top} q \qquad g(q_a) \xrightarrow[\lambda x_1 . h(x_1)]{\top} q$$

$$f(q, q) \xrightarrow[\lambda h(x_1) . x_1]{1=2} q$$



ONE COPY ONLY: MORE TREE AUTOMATA

[CL,Cortier 2001–2004]

$$
\begin{array}{lrcl}
A(a,b):\ \nu N_A & & \rightarrow & \{a, N_A\}_{pub(b)} \\
 & \{N_A, y\}_{pub(a)} & \rightarrow & \{y\}_{pub(b)} \\
 & & & \\
B(a,b):\ \nu N_B & \{a, x\}_{pub(b)} & \rightarrow & \{x, N_B\}_{pub(a)} \\
 & \{N_B\}_{pub(b)} & \rightarrow &
\end{array}
$$

Theorem: emptyness of tree automata with one memory is DEXPTIME-complete.

A PROTOCOL EXAMPLE

b, r are two public positive integers. $b^s \mod r$ is the public key of EP and s is the associated private key.
In a first authentication phase, the two parties agree on a session nonce N_s and S owes the certified public key $b^s \mod r$.

$$\begin{array}{lll}
1. & EP \rightarrow S: & \nu N.\, \text{hash}(b^N \mod r, S, N_s, X) \\
2. & S \rightarrow EP: & \nu N_c.\, N_c \\
3. & EP \rightarrow S: & N - s \times N_c, X
\end{array}$$

Then S checks that the first message x and the last message y satisfy

$$x = \text{hash}((b^s)^{N_c} \times b^y \mod r, S, N_s, X)$$

The security property states that this verification is OK only if EP sent $N - s \times N_c, X$ at step 3.

ALGEBRAIC PROPERTIES

See also [Cortier,Delaune, Lafourcade 2005, *Journal of Computer Security*]

DY:

$$\begin{array}{rcl} \mathtt{Dec}([x]_y, y) & \rightarrow & x \\ \mathtt{PDec}(\{x\}_{pub(y)}, priv(y)) & \rightarrow & x \end{array} \qquad \pi_1(< x, y >) \rightarrow x \qquad \pi_2(< x, y >) \rightarrow y$$

Signatures (Sign)

$$v(\mathsf{sign}(x,k), pub(k)) \rightarrow 1 \qquad u(\mathsf{sign}(x,k), pub(k)) \rightarrow x$$

decryption confusion (DC): DY +

$$[\mathtt{Dec}(x,k)]_k \rightarrow x$$

Exercise: Find a simple protocol which is secure (e.g. keeps secrecy) for 1 session DY, but is insecure

with this additional rule.

ALGEBRAIC PROPERTIES (2)

Homomorphic encryption (ECB): DY +

$$[< x, y >]_k \rightarrow < [x]_k, [y]_k >$$

Prefix (CBC): DY +

$$p([< x, y >]_k) \rightarrow [x]_k$$

Blind signatures: `Sign` + DY +

$$\mathsf{unblind}(\mathsf{blind}(x, k), k) \rightarrow x \qquad \mathsf{unblind}(\mathsf{sign}(\mathsf{blind}(x, r), k), r) \rightarrow \mathsf{sign}(x, k)$$

Exclusive or

$$\begin{array}{rclcrclcrcl} x \oplus 0 & = & x & \quad & x \oplus x & = & 0 & \quad & x \oplus (y \oplus z) & = & (x \oplus y) \oplus z \\ x \oplus y & = & y \oplus x & & & & & & & & \end{array}$$

ALGEBRAIC PROPERTIES (3)

Abelian Groups

$$
\begin{array}{rclcrclcrcl}
x + 0 & = & x & \quad & x + (-x) & = & 0 & \quad & x + (y + z) & = & (x + y) + z \\
x + y & = & y + x & & & & & & & &
\end{array}
$$

EP: $AG(+), AG(\times)$ +

$$
(z^x)^y = z^{x \times y} \qquad z^x \times z^y = z^{x+y}
$$

Combination of theories.

Locality: the DY case

Theorem: If $T \vdash t$, then there is a proof whose all nodes are in St(T, t).

Equivalenty: Let T, t be in normal form. If there is a recipe ζ such that $\zeta[T]\downarrow = t$, then there is a recipe ζ_0 such that $\zeta_0[T]\downarrow = t$ and, for every subterms ζ_1 of ζ_0, $\zeta_1[T]\downarrow \in$ St(T, t).

Corollary: Given T, t, whether t can be deduced from T is decidable in linear time (and is PTIME-complete).

LOCALITY: THE DY CASE: PROOF

We prove that for any recipe in normal form,

- either $\zeta_0[T]\downarrow$ is a subterm of some $u \in T$ (Decomposition)
- or else $\zeta_0[T] = f(\zeta_1[T], \ldots, \zeta_n[T])$ and $\zeta_0[T]\downarrow = f(\zeta_1[T]\downarrow, \ldots, \zeta_n[T]\downarrow)$ (Composition)

By induction on ζ_0 (base case straightforward):

- Use an innermost rewriting strategy.
- Case analysis depending on the top symbol of ζ_0:

  $top(\zeta_0)$ **is a pair or an encryption** : we fall in the second case

  $top(\zeta_0)$ **is a projection** : $\zeta_0[T] = \pi_i(\zeta_1[T])$. The top symbol of ζ_1 is not a pair. Apply the induction hypothesis on ζ_1: if there is no top redex we have a composition, otherwise $\zeta_1[T]\downarrow \in$ St(T) and $\zeta_0[T]\downarrow \in$ St(T).

  $top(\zeta_0)$ **is a decryption** : $\zeta_0 =$ dec$(\zeta_1[T], \zeta_2[T])$. If there is no top redex, then we get a composition. Otherwise, apply the induction hypothesis to ζ_1:
  - if ζ_1 is not empty and $\zeta_1[T]\downarrow$ is a composition there cannot be any top redex by irreducibility of ζ_0
  - otherwise, $\zeta_1[T]\downarrow = \{u\}_k \in$ St(T) (symmetric key case) and $\zeta_0[T]\downarrow = u \in$ St(T)

LOCALITY (GENERAL CASE)

- Almost all equational theories relevant to protocol verification can be presented by a finite convergent rewrite system (possibly modulo AC).
- Locality property:
  If there is a recipe ζ such that $\zeta[t_1, \ldots, t_n] \downarrow = t$, then there is such a small context ζ_0.

  There is a (efficiently computable) function F from finite sets of terms to finite sets of terms such that

$$\forall t_1, \ldots, t_n, t, \; \forall \zeta, \exists \zeta_0, \\ t = \zeta[t_1, \ldots, t_n] \downarrow \Longrightarrow \forall \zeta_1 \in \mathsf{St}(\zeta_0), \zeta_1[t_1, \ldots, t_n] \downarrow \in F(t_1, \ldots, t_n, t)$$

  Examples

  - DY, `xor` theory: $F(T)$ is the set of subterms of T: given $t_1, \ldots, t_n, t$ (in normal form), it is decidable in polynomial time whether there is a ζ such that ζ such that $\zeta[t_1, \ldots, t_n] \downarrow = t$
  - EP, homomorphic encryption, combined theories: requires a semantic notion of subterms.
    $F(t)$ is obtained by (possibly) adding a fixed context on the top of a subterm of t. ($F(T)$ computable in linear time).

LOCALITY: THE CASE OF AC-SYMBOLS

In case of AC-symbols, terms are flattened.

Equivalently, we need an unbounded number of inference rules, indexed by n:

$$\frac{u_1 \quad \cdots \quad u_n}{(u_1 + \cdots + u_n)\downarrow}$$

Exercise:Prove the locality theorem for the `xor` theory.
Hint: consider two versions of the n-premisses rule above: one in which $(u_1 + \cdots + u_n)\downarrow$ has not $+$ as a top symbol (decomposition) and one in which $+$ is the top symbol of the resulting term (composition)

LOCALITY: THE CASE OF AC-SYMBOLS

In case of AC-symbols, terms are flattened.

Equivalently, we need an unbounded number of inference rules, indexed by n:

$$\frac{u_1 \quad \cdots \quad u_n}{(u_1 + \cdots + u_n)\downarrow}$$

Exercise:Prove the locality theorem for the `xor` theory.
Hint: consider two versions of the n-premisses rule above: one in which $(u_1 + \cdots + u_n)\downarrow$ has not $+$ as a top symbol (decomposition) and one in which $+$ is the top symbol of the resulting term (composition)

Infinite number of rules: the complexity of deducibility depends on the complexity of one-step deducibility.

If, given $t_1, \ldots, t_n, t$, the solvability of $\lambda_1 t_1 + \cdots + \lambda_n t_n = t$ is in PTIME and the deduction system is F-local, where F is a PTIME function, then deducibility is in PTIME.

Typical examples: `xor`, Abelian groups: linear systems over $\mathbb{F}_2$ or $\mathbb{Z}$ are solvable in PTIME.

More generally: solving linear systems + locality $\Rightarrow$ decision of intruder

LOCALITY (CNTD)

- For Abelian Groups and some more complex theories, we need to replace "Subterm" by a semantic notion of subterms. A typical example is the combination of theories: subterms are alien subterms. Then F might add a (pure) context on top of the terms.

  Example: for AG, we use a rule

$$\frac{u_1 \quad \cdots \quad u_n \quad v_1 \quad \cdots \quad v_m}{(u_1 + \cdots + u_n - v_1 \cdots - v_m) \downarrow}$$

- Open question: nice sufficient conditions on the rewrite system for locality. (This is solved only when there is no AC symbols: [Basin & Ganzinger 2001], [CL, Treinen 2003]).
  When there is no AC-symbol, a sufficient condition is the saturation property (see also: finite variant property in part 2) of the intruder deductions, *w.r.t. an ordering isomorphic to* ω.

2. Bounding the number of sessions: deducibility constraints

SUMMARY 2

- The formal model for a bounded number of sessions (consistent with many previous works by J. Millen, V. Shmatikov, Y. Chevalier, R. Küsters, M. Rusinowitch, M. Turuani, M. Baudet, S. Delaune etc...
- Solving deducibility constraints in the DY case
- General deducibility constraints: splitting the problem in 4 parts: 4 important properties of the rewrite system
- Narrowing and the finite variant property
- Conservativity and the small attack theorem

BOUNDED NUMBER OF SESSIONS

- Consider a fixed number of instances of each role.
  E.g. $A(a, c) \parallel B(b)$
- Nonces are distinct constants
- Guess an interleaving of the rules
  E.g. 1 of A; 1 of B; 2 of A; 2 of B.

The question is: for this sequence of actions, is there any attack ?

Difficulty: there is no a priori bound on the size of the attack.

DEDUCIBILITY CONSTRAINTS

For instance: [CL & Shmatikov 2003], [Millen & Shmatikov 2001,2004], [Baudet 2005,2006], [Chevalier & Rusinowitch 2005, 2006], [Delaune et al 2006], [Bursuc et al. 2007].

T_0 is the initial intruder knowledge. For each (guessed) interleaving of actions $x_1 \rightarrow t_1$ if $u_1 = v_1, \ldots, x_n \rightarrow t_n$ if $u_n = v_n$:

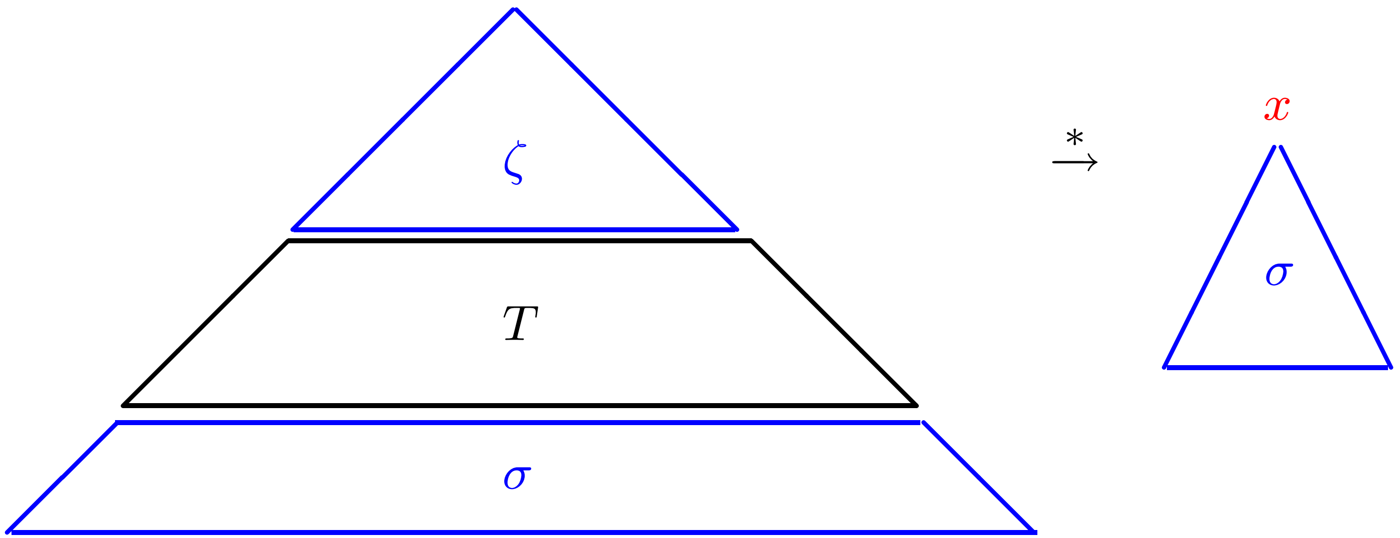
$$\begin{array}{rcll} T_0 & \Vdash & x_1 & \quad u_1 = v_1 \\ T_0, t_1 & \Vdash & x_2 & \quad u_2 = v_2 \\ & \vdots & & \\ T_0, t_1, \ldots, t_{n-1} & \Vdash & x_n & \quad u_n = v_n \end{array}$$

A valid trace instance is a substitution σ such that there are public contexts (*recipes*) $\zeta_1, \ldots, \zeta_n$ such that

$$\forall i. \quad \zeta_i[T_0, t_1\sigma, \ldots, t_{i-1}\sigma] =_E x_i\sigma \text{ and } u_i\sigma =_E v_i\sigma$$

SOLUTION OF A DEDUCIBILITY CONSTRAINT

A solution of the deducility constraint $T \Vdash u$ actually consists in two parts, generalizing unification problems:



Examples (1)

$$\begin{array}{llrclcl} A(a,b): & \nu N_A & 1. & & \rightarrow & \{a, N_A\}_{pub(b)} \\ & & 2. & \{N_A, y\}_{pub(a)} & \rightarrow & \{y\}_{pub(b)} \end{array}$$

$$\begin{array}{llrclcl} B(a,b): & \nu N_B & 1. & \{a, x\}_{pub(b)} & \rightarrow & \{x, N_B\}_{pub(a)} \\ & & 2. & \{N_B\}_{pub(b)} & \rightarrow & \end{array}$$

Consider $A(a,c) \parallel B(b)$ with the interleaving $A1;\ B1;\ A2;\ B2$.

$T_0 = \{pub(a), pub(b), pub(c), priv(c), a\}$.

$$\begin{array}{rcllcl} T_1 = T_0, \{a, n_a\}_{pub(c)} & \Vdash & x_1 & & x_1 = \{a, x\}_{pub(b)} \\ T_2 = T_1, \{\pi_2(\mathsf{dec}(x_1, priv(b))), n_b\}_{pub(a)} & \Vdash & x_2 & & x_2 = \{n_a, y\}_{pub(a)} \\ T_3 = T_2, \{\pi_2(\mathsf{dec}(x_2, priv(a)))\}_{pub(c)} & \Vdash & x_3 & & x_3 = \{n_b\}_{pub(b)} \end{array}$$

$$\pi_1(x,y) = x \quad \pi_2(x,y) = y \quad \mathsf{dec}(\{x\}_{pub(y)}, priv(y)) = x$$

$x = n_a$, $y = n_b$ yields a solution.

Examples (2)

$$\begin{array}{lccl} 1. & A \rightarrow S & : & B, \{Ka\}_{pub(S)} \\ 2. & S \rightarrow B & : & A \\ 3. & B \rightarrow S & : & A, \{Kb\}_{pub(S)} \\ 4. & S \rightarrow A & : & B, Kb \oplus Ka \end{array}$$

$$\begin{array}{llcl} A(a,b,s) : \nu K_a & & \longrightarrow & b, \{K_a\}_{pub(s)} \\ & b, x & \longrightarrow & \\ B(a,b,s) : \nu K_b & a & \longrightarrow & a, \{K_b\}_{pub(s)} \\ S(a,b,s) : & b, \{y_1\}_{pub(s)} & \longrightarrow & a \\ & a, \{y_2\}_{pub(s)} & \longrightarrow & b, y_1 \oplus y_2 \end{array}$$

$$x \oplus (y \oplus z) = (x \oplus y) \oplus z \quad x \oplus y = y \oplus x \quad x \oplus 0 = x \quad x \oplus x = 0 \quad \{x \star \{y\}_{pub(z)}\}_{pub(z)} = \{x \star y\}_{pub(z)}$$

$B(a,b,s) \| S(c,d,s)$ (c is corrupted).

$$\begin{array}{rclcrcl} T_0, a & \Vdash & x_1 & \quad & x_1 & = & a \\ T_1, \{K_b\}_{pub(s)} & \Vdash & x_2 & & x_2 & = & d, \{x\}_{pub(s)} \\ T_2, c & \Vdash & x_3 & & x_3 & = & c, \{y\}_{pub(s)} \\ T_3, d, x \oplus y & \Vdash & x_4 & & x_4 & = & K_b \end{array}$$

EXAMPLES (3)

$$
\begin{array}{llll}
EP(x,y,b,X): & \nu n_1. & \rightarrow & h(b^{n_1}, y, N(x,y), X) \\
 & z & \rightarrow & n_1 - s(x) \times z, X
\end{array}
$$

$$
\begin{array}{lllll}
S(x,y,b): & \nu n_2, n_3. & z_1 & \rightarrow & n_2 \\
 & & z_2 & \rightarrow & \text{if } h((b^{s(x)})^{n_2} \times b^{\pi_1(z_2)}, y, N(x,y), \pi_2(z_2)) = z_1 \text{ then } n_3
\end{array}
$$

$$T_0 = \{b_0, b_0^{s(A)}, A, S, h(b_0^{n_1}, S, N(A,S), X_1)\}$$

$$
\begin{array}{rcl}
T_0 & \Vdash & z_1 \\
T_0, n_2 & \Vdash & z \\
T_0, n_2, n_1 - s(A) \times z, X_1 & \Vdash & z_2
\end{array}
$$

$$
\begin{array}{rcl}
h((b^{s(A)})^{n_2} \times b_0^{\pi_1(z_2)}, S, N(A,S), \pi_2(z_2)) & = & z_1 \\
z_2 & \neq & n_1 - s(A) \times z, X_1
\end{array}
$$

SECURITY PROPERTIES

The security property is expressed using the solutions of the constraint system(s): for any solution σ

Secrecy $T_0, t_1\sigma, \ldots, t_n\sigma \not\Vdash \texttt{secret}$

Authentication (agreement) $C_1[u_1, \ldots, u_m]\sigma =_E C_2[v_1, \ldots, v_k]\sigma$

Trace equivalence

$$C_1[t_1, \ldots, t_n]\sigma =_E C_2[t_1, \ldots, t_n]\sigma \Leftrightarrow C_1[u_1, \ldots, u_n]\sigma =_E C_2[u_1, \ldots, u_n]\sigma$$

In this case solutions include the recipes, which should be the same in both constraint systems

ADDITIONAL PROPERTIES OF THE CONSTRAINTS

Constraints:

- A deduction part, $T_0 \Vdash x_1, \ldots, T_{n-1} \Vdash x_n$
  (A restricted second-order unification problem).
- An equational part: a conjunction of equations
- A part expressing the negation of a security property: a deduction constraint (secrecy) a conjunction of equalities/disequalities (authentication), membership constraints...

Monotonicity The attacker's knowledge is increasing: $T_0 \subseteq T_1 \subseteq \ldots \subseteq T_n$

Origination If $x \in Var(T_i)$, then there is $j < i$ such that $x = x_j$.

Example

$$\begin{array}{rcl} T_0 & \Vdash & x_1 \\ T_0, x & \Vdash & x_2 \end{array} \qquad x_1 = h(x)$$

is *not* a constraint

Solving deducibility constraints in the DY case (1)

- Observe that, for any term in normal form s and any normalized substitution σ, $\mathsf{St}(s\sigma\downarrow) \subseteq \mathsf{St}(s)\sigma\downarrow \cup \mathsf{St}(\sigma)$.

  Example $s = < \mathsf{dec}(x, k), x >$ and $\sigma = \{x \mapsto [< a, n_a >]_k$.
  $s\sigma\downarrow = << a, n_a >, [< a, n_a >]_k >$ ...

  Exercise: complete the proof. Which property of the rewrite system are we using here ?

- let E be a unification problem E is equivalent (modulo DY) to a finite disjunction $\bigvee_i E_i$ where each E_i is a conjunction of equations $x_{i,j} = u_{i,j}$ such that
  - $\forall i, j, l, x_{i,j} \notin Var(u_{i,l})$,
  - $\forall i, j, u_{i,j} \in \mathsf{St}(E)$. (or $pub(\mathsf{St}(E))$ in case of asymetric encryption)

  Exercise: Complete the proof. Which properties of the rewrite system are we using here ?

Solving deducibility constraints in the DY case (2): the small attack property

The small attack property: Let σ be a solution of C. Then the substitution θ obtained by replacing any $v \in \mathrm{St}(\sigma) \setminus \mathrm{St}(C)\sigma\downarrow$ by an arbitrary public subterm is also a solution of C.

Proof idea: First, using the previous observation, we can consider w.l.o.g. pure deducibility constraints (without equations).
Let x be a rhs such that $v \in \mathrm{St}(x\sigma)$ and $T_x \Vdash x \in C$, T_x minimal. Let $\zeta_0[T_x\sigma]\downarrow = x\sigma$.

By locality,

- either $x\sigma \in \mathrm{St}(T_x\sigma)$: then $v \in \mathrm{St}(T_x)\sigma$ by minimality
- Or $\zeta_0[T_x] = f(\zeta_1[T_x], \ldots, \zeta_n[T_x])$ and $x\sigma = f(\zeta_1[T_x\sigma]\downarrow, \ldots, \zeta_n[T_x\sigma]\downarrow)$

If $v \notin \mathrm{St}(C)\sigma$, let $x\theta = x\sigma[v \mapsto v']$.

If $\zeta[T\sigma]\downarrow = y\sigma$ then

- Replace any $\zeta' = \zeta|_p$ such that $\zeta'[T]\downarrow$ is $x\sigma$ or one of its direct subterms, with the corresponding ζ_i
- The resulting recipe $\overline{\zeta}$ is such that $\overline{\zeta}[T\theta]\downarrow = y\theta$

Solving deducibility constraints in the DY case (3)

Non deterministic algorithm:

- For each $s \in \mathrm{St}C$, guess if its instance is deducible and at which step: insert $T_i \Vdash x_s \wedge x_s = s$, adding x_s to all T_j, $j > i$.
- By locality and the small attack property, C has a solution iff there is one of the above systems which has a one-step solution θ: recipes consist only in a single function symbol
- Turn non-deterministically each deduction constraint into an equation.
- Solve the equation system

Theorem [Rusinowitch, Turuani, 2001]: In the DY case, deducibility constraints are NP-complete.

Example

$$\left\{\begin{array}{rcll} a & \Vdash & x & \\ a, [k]_{<a,x>} & \Vdash & y & \quad y = k \end{array}\right.$$

Insert a guessed deducible subterm:

$$\left\{\begin{array}{rclrcl} a & \Vdash & x & & & \\ a & \Vdash & z & z & = & < a, x > \\ a, [k]_{<a,x>}, z & \Vdash & y & y & = & k \end{array}\right.$$

Turn the constraints into equations:

$$\left\{\begin{array}{rclrcll} a & = & x & & & & (\zeta_1 = x_1) \\ < a, a > & = & z & z & = & < a, x > & (\zeta_2 =< x_1, x_1 >) \\ \text{dec}([k]_{<a,x>}, z) & = & y & y & = & k & (\zeta_2 = \text{dec}(x_2, x_3)) \end{array}\right.$$

SOLVING DEDUCIBILITY CONSTRAINTS: THE DY CASE (4)

R_1	$C \wedge T \Vdash u$	$\rightsquigarrow$	C	if $T \cup \{x \mid (T' \Vdash x) \in C, T' \subsetneq T\} \vdash u$
R_2	$C \wedge T \Vdash u$	$\rightsquigarrow_\sigma$	$C\sigma \wedge T\sigma \Vdash u\sigma$	if $\sigma = mgu(t, u)$, $t \in$ St(T), $t \neq u,\ t, u$ not variables
R_3	$C \wedge T \Vdash u$	$\rightsquigarrow_\sigma$	$C\sigma \wedge T\sigma \Vdash u\sigma$	if $\sigma = mgu(t_1, t_2)$, $t_1, t_2 \in$ St(T), $t_1 \neq t_2,\ t_1, t_2$ not variables
R_3'	$C \wedge T \Vdash u$	$\rightsquigarrow_\sigma$	$C\sigma \wedge T\sigma \Vdash u\sigma$	if $\sigma = mgu(t_2, t_3)$, $\{t_1\}_{t_2}, priv(t_3) \in$ St(T), $t_2 \neq t_3,\ t_2$ or t_3 (or both) is a variable
R_4	$C \wedge T \Vdash u$	$\rightsquigarrow$	$\perp$	if $var(T, u) = \emptyset$ and $T \nvdash u$
R_f	$C \wedge T \Vdash f(u, v)$	$\rightsquigarrow$	$C \wedge T \Vdash u \wedge T \Vdash v$	for $f \in \{\langle\rangle, [], \{\}\}$

Theorem [CL, Cortier, Zalinescu, 2007] This system is correct, complete and terminating in polynomial time.

3. Solving deducibility constraints in equational theories

FOUR KEY PROPERTIES OF THE REWRITE SYSTEM

Assume E is described by a finite AC-convergent rewrite system.

- **Finite variant property**
- **Locality (small proofs)**
- **Conservativity (small attacks)**
- **One-Step deducibility constraints**

The finite variant propery

Convergent Rewriting Systems (possibly modulo AC). Every term u has a normal form $u\downarrow$.

Finite variant property: [CL, Delaune 2005],
For every term t, there is a finite (computable) set of substitutions $\theta_1, \ldots, \theta_n$ such that

$$\forall \sigma.\ \exists i, \exists \tau.;\ t\sigma\downarrow =_{AC} (t\theta_i\downarrow)\tau$$

Examples
Abelian Groups

$$\begin{array}{rcl}
x + 0 & \to & x \\
-(-(x)) & \to & x \\
-0 & \to & 0
\end{array}
\qquad
\begin{array}{rcl}
x + (-x) & \to & 0 \\
(-x) + x + y & \to & y \\
(-x) + (-y) & \to & -(x + y)
\end{array}
\qquad
\begin{array}{rcl}
-(x) + (-y) + z & \to & (-(x + y)) + z \\
(-(x + y)) + x & \to & -y \\
(-(x + y)) + (x + z) & \to & (-y) + z \\
-((-x) + y) & \to & x + (-y)
\end{array}$$

Exercise

Show that orienting the red rules in the other direction, while we still get a convergent rewrite system, we do not get the finite variant property.

THE FINITE VARIANT PROPERTY (2)

A sufficient criterion

If, for every function symbol f there is $c_f \in \mathbb{N}$ s.t. for any terms $t_1, \ldots, t_n$ in normal form, $f(t_1, \ldots, t_n)$ can be normalized in at most c_f steps, then the rewrite system has the FVP

Which theories satisfy finite variant property ?

Most relevant examples (enc-dec,`xor`, modular exponentiation, EP,...)

Proofs of FVP [Escobar,Meseguer,Sasse 2008]

Unification

Note that the FVP implies that unification is decidable and finitary.

Example

What are the variants of $x + y + a$ in the case of Abelian groups ?

CONSERVATIVITY

There is a (effective) function F from finite sets of terms to finite sets of terms such that for every constraint C and every solution θ, there is a solution σ such that

$$\mathsf{St}(C\sigma\downarrow) \subseteq F(\mathsf{St}(C)\sigma\downarrow)$$

σ is built out of pieces of C.

Examples

- DY theory, `xor`-theory: F is the identity.
  Intuition: if $x\sigma$ is constructed from its direct subterms, then every time we look "inside" $x\sigma$, we could use its (constuctible) subterms instead. Therefore, $x\sigma$ could be replaced by any other value.
  Now, if $x\sigma$ is not constructed from its direct subterms, it must be obtained by rewriting a context applied to pieces of the constraint. Hence being itself a subterm of the constraint, since the rewrite rules only yield subterms of the left sides.
- EP: F may add or remove one (and only one) top exponential

EXAMPLE

$\theta = \{x \mapsto a+b; y \mapsto c^{a+b+c}; z \mapsto b^a\}$ is a solution of

$$\begin{array}{rcll} c, a+b & \Vdash & x & \\ c, a+b, x+a & \Vdash & y & \\ c, a+b, x+a, y^a & \Vdash & z & \qquad z = b^a \end{array}$$

Example

$\theta = \{x \mapsto a+b; y \mapsto c^{a+b+c}; z \mapsto b^a\}$ is a solution of

$$\begin{array}{rcll} c, a+b & \Vdash & x & \\ c, a+b, x+a & \Vdash & y & \\ c, a+b, x+a, y^a & \Vdash & z & \qquad z = b^a \end{array}$$

θ is a solution: $(2(a+b)-(x+a))^{x+a-(a+b)} = z$

EXAMPLE

$\theta = \{x \mapsto a+b; y \mapsto c^{a+b+c}; z \mapsto b^a\}$ is a solution of

$$\begin{array}{rcl} c, a+b & \Vdash & x \\ c, a+b, x+a & \Vdash & y \\ c, a+b, x+a, y^a & \Vdash & z \qquad\qquad z = b^a \end{array}$$

θ is a solution: $(2(a+b)-(x+a))^{x+a-(a+b)} = z$
it is not a conservative one

$\sigma = \{x \mapsto a+b; y \mapsto c; z \mapsto b^a\}$ is a conservative one

ONE-STEP DEDUCIBILITY CONSTRAINTS

Only recipes whose subterms are variables are considered.
Example there is no one-step solution to $a, b \Vdash x \quad \wedge x = (a + b) \star a$

One-step deducibility constraints can be non-deterministically turned into equations (guess the top symbol and the arguments).
In case of AC-symbols, this requires in addition to introduce integer variables counting the number of times each term is used. This yields possible additional difficulties (avoid non-linear Diophantine systems !!)

Instances

- One step deducibility constraints are straightforward for DY.
- One step deducibility constraints are decidable in polynomial time for Abelian Groups, `xor`: they reduce to linear equations systems.
- Decidable for EP in NP. (More complex decision procedure)
- Open question for AC

Example

$$
\begin{array}{rcl}
a + b & \Vdash & x \\
a + b, x + a, 2x + c & \Vdash & y \qquad\qquad y = 2z + c \\
& \ldots &
\end{array}
$$

$$
\left\{
\begin{array}{rcl}
x & = & \lambda.(a + b) \\
y & = & \lambda_1(a + b) + \lambda_2(x + 3a) + \lambda_3(2x + c) \\
2z + c & = & y
\end{array}
\right.
$$

$$
\Longrightarrow
\left\{
\begin{array}{rcl}
x & = & \lambda.(a + b) \\
y & = & \lambda'_1(a + b) + 3\lambda'_2 a + \lambda'_3 c \\
y & = & 2z + c
\end{array}
\right.
$$

$$
\Longrightarrow
\left\{
\begin{array}{rclrcl}
 & & & \lambda & = & \lambda_{x,a} \\
 & & & \lambda & = & \lambda_{x,b} \\
 & & & \lambda_{x,c} & = & 0 \\
x & = & \lambda_{x,a} a + \lambda_{x,b} b + \lambda_{x,c} c & \lambda_{y,a} & = & \lambda'_1 + 3\lambda'_2 \\
y & = & \lambda_{y,a} a + \lambda_{y,b} b + \lambda_{y,c} c & \lambda_{y,b} & = & \lambda'_1 \\
z & = & \lambda_{z,a} a + \lambda_{z,b} b + \lambda_{z,c} c & \lambda_{y,c} & = & \lambda'_3 \\
 & & & 2\lambda_{z,a} & = & \lambda_{y,a} \\
 & & & 2\lambda_{z,b} & = & \lambda_{y,b} \\
 & & & \lambda_{z,c} & = & 2\lambda_{y,c}
\end{array}
\right.
$$

A TEMPLATE FOR DECISION PROCEDURES

Theorem: Conservativity, locality, and the finite variant property allow, altogether, to reduce deduction constraints to *one-step* deduction constraint.

Conservativity : $\mathsf{St}(C\sigma\downarrow) \subseteq F_1(\mathsf{St}(C)\sigma\downarrow)$

Finite variant property: $\mathsf{St}(C) \rightarrow \mathsf{St}(C)\theta_1\downarrow, \ldots, \mathsf{St}(C)\theta_n\downarrow$

$F_1(\mathsf{St}(C)\sigma\downarrow) \subseteq F_1(\mathsf{St}(C)\theta_i\downarrow\sigma_1) \subseteq F_2(\mathsf{St}(C)\theta_i\downarrow)\sigma_2$

Guess deducible terms in $F_2(\mathsf{St}(C)\theta_i\downarrow)$, and in which order. Insert the appropriate constraints.

Locality each step requires only a pure recipe.

Corollary: decidability in NP for DY, xor, exponentiation.
Also: the EP case, disjoint and hierarchical combinations (**)

(**) For combinations: only conservativity has been proved to be combinable in general

IF THERE IS STILL SOME TIME LEFT...

The combination problems:

- Protocols may rely on several primitives yielding combined equational theories (not necessarily disjoint)
- Use a *semantic subterm* notion (subterm= alien subterm) Then follow the same procedure, yielding pure systems instead of one-step systems.
- Yields decision procedures for *well-moded systems* [Chevalier, Rusinowitch 2006]

Open questions

- General criteria for locality in presence of AC symbols
- Combination results for locality, conservativity, finite variant property when there are AC symbols
- Procedures preserving the set of solutions (as in the DY case); allowing the decision of wider classes of security properties
- A good proof-theoretic explanation of the small attack property