COL876: SPECIAL TOPICS IN FORMAL METHODS

Formal verification of security protocols

Lecture 7, 17 August 2023

QUIZ!

I. Write out the following protocol in the applied-pi calculus.

$$A \to B : \mathsf{enc}((A, \mathsf{enc}(m, \mathsf{pk}(B))), \mathsf{pk}(B))$$
$$B \to A : \mathsf{enc}(m, \mathsf{pk}(A))$$

II. Write the equational theory for XOR with identity 0. XOR is associative and commutative.

III. When can one get multiple reduction sequences from a single configuration? List out all possible scenarios.

BONUS: Is there a bound on the number of such sequences? If yes, what is it?

RECAP

- Saw how to use ProVerif for automated verification
- Wrote a simple protocol and properties in ProVerif
- ProVerif tries to prove a property; if it cannot be proved, tries to produce an attack trace
- Saw that ProVerif can often produce bizarre error traces, and needs some help to produce a “reasonable” error trace

PROVERIF: SYNTAX

- Crypto operations specified using equations or *rewrite rules*
- Declare types, constructor functions, and reduction rules for destructors before starting a protocol description
- Declare the desired property using the **`query`** keyword
- Enriched terms: allow one to include **`new`**, **`if then else`** etc in *term* syntax

ENRICHED TERM SYNTAX

$M, N ::=$	enriched terms
a, b, c, k, m, n, s	names
x, y, z	variables
$(M_1, \ldots, M_j)$	tuple
$h(M_1, \ldots, M_j)$	constructor/destructor application
i	natural number ($i \in \mathbb{N}$)
$M + i$	addition ($i \in \mathbb{N}$)
$i + M$	addition ($i \in \mathbb{N}$)
$M - i$	subtraction ($i \in \mathbb{N}$)
$M > N$	greater
$M < N$	smaller
$M >= N$	greater or equal
$M <= N$	smaller or equal
$M = N$	term equality
$M <> N$	term disequality
$M \;\&\&\; M$	conjunction
$M \;\|\|\; M$	disjunction
$\mathbf{not}(M)$	negation
$\mathbf{new}\ a : t; M$	name restriction
$\mathbf{if}\ M\ \mathbf{then}\ N\ \mathbf{else}\ N'$	conditional
$\mathbf{let}\ T = M\ \mathbf{in}\ N\ \mathbf{else}\ N'$	term evaluation
$\mathbf{event}\ e(M_1, \ldots, M_n); M$	event

PROVERIF: NATURAL NUMBERS

- ProVerif has a type **`nat`** to represent natural numbers
- One can add and subtract a number i from a term t (t+i, i+t, t-i)
- One can also test for order on terms (>, <, >=, <=, =, <>)
- Constructors cannot have type **`nat`**, but destructors can

```
type key.

fun enc5(nat, key) : bitstring.

fun dec5(bitstring, key) : nat.

reduc forall x:nat, y:key; dec5(enc5(x+5, y), y) = y.
```

PROVERIF: EQUATIONS

- Suppose I add multiplication over objects of some new type **`prd`**
- Express that multiplication is commutative? Use **`equation`**
- Equations need to be convergent (terminating and confluent rewriting) and linear (variables occur at most once in LHS and at most once in RHS)

```
type prd.

fun mult(prd, prd) : prd.

equation forall x:prd, y:prd; mult(x, y) = mult(y, x).
```

PROVERIF: STORAGE

- Can model persistent storage using **`table`**
- Can populate (**`insert`**) and access (**`get`**) entries, but not delete
- Tables are not accessible to the attacker

```
table d(t1,…,tn).

insert d(M1,…,Mn); P.

get d(u1,…,un) in P else Q.

get d(u1,…,un) suchthat M in P else Q.
```

PROVERIF: PROPERTIES

- Secrecy specified as intruder knowledge: **`query attacker(t)`**
- Properties can involve events
- Correspondence queries specified as implications over events:
  `query x:t, y:t; event(x, y) ==> event(y, x)`
- What about an event Place-order(t) and an event Cust-pays(t)?
-

PROVERIF: PROPERTIES

- Secrecy specified as intruder knowledge: **`query attacker(t)`**
- Can check for any reachability property this way
- Is term M sent on channel c? **`query mess(c, M)`**
- Is (t1,…,tn) present in table d? **`query table(d(t1,…,tn))`**
- Does ev occur? **`query x1:t1,…,xn:tn; event (ev(x1,…,xn))`**

PROVERIF: PROPERTIES

- Correspondence queries specified as implications over events: **`query x:t; ev1(x) ==> ev2(x)`**

- If ev1 has happened, then ev2 has happened (for the same x)

- What about an event Place-order(x) and an event Cust-pays(x)?

- **`inj-event`** allows us to specify a one-to-one correspondence

- **`query x:t; inj-event(order(x)) ==> inj-event(paid(x))`**

PROVERIF: PROPERTIES

- But what about temporal order?
- If an order has been placed, the payment was done *before*

- ProVerif has type **`time`**; can mark events with timestamp
- ```
  query x:t, i:time, i':time;
  inj-event(order(x))@i ==>
  (inj-event(paid(x))@i' && i' < i)
  ```
  ```