COL876: SPECIAL TOPICS IN FORMAL METHODS

Formal verification of security protocols

Lecture 2, 27 July 2023

MORE LOGISTICS

- Register (potentially via a General Request) if you have not already
- Join the Teams channel for the course and check it regularly
- All announcements (including those for assignments and deadlines) will be made only on the Teams channel
- Lecture notes will be uploaded to the channel

SYMBOLIC MODEL: DOLEV & YAO, 1983

- Split each communication into a send and a receive
- I is essentially the network
  - Each send captured by I
  - Each receive assumed to come from I
- A send action need not have a corresponding receive action!

DOLEV-YAO INTRUDER

- Intruder I cannot break encryption, but, on the public channel, can
  - see any message sent on the channel
  - block any message from reaching the intended recipient
  - re-route any message to any principal
  - masquerade as any principal and send messages in their name
  - initiate new communication according to the protocol
  - generate messages — according to some rules

MODELLING MESSAGES

- Split each communication into a send and a (potential) receive
- But what about the messages sent and received?
- Messages are not bitstrings
  - Ignore extraneous details like headers, metadata &c.
- Modelled as symbolic terms from a *term algebra*.

$$t := m \mid \mathsf{f}(t_1, \ldots, t_k)$$

MORE ABOUT MESSAGES

- When can an agent/intruder send a particular message term?
- When they can generate it, according to particular rules.
- Will only consider "well-formed" protocols
    - Honest principals can always generate whatever messages they need to send according to the protocol
    - Need to check correct generation only for the intruder

PROOF RULES FOR MESSAGES

$\dfrac{}{X \vdash m}\ \mathbf{ax}(m \in X)$	
$\dfrac{X \vdash (t_1, t_2)}{X \vdash t_i}\ \mathbf{split}$	$\dfrac{X \vdash t \quad X \vdash u}{X \vdash (t, u)}\ \mathbf{pair}$
$\dfrac{X \vdash \mathsf{enc}(t, \mathsf{pk}(k)) \quad X \vdash k}{X \vdash t}\ \mathbf{adec}$	$\dfrac{X \vdash t \quad X \vdash \mathsf{pk}(k)}{X \vdash \mathsf{enc}(t, \mathsf{pk}(k))}\ \mathbf{aenc}$

Proof system for a term algebra
with pairing and asymmetric encryption

VERIFYING PROPERTIES

- Many properties involve looking for a proof using these rules
- *Passive intruder problem*: can the intruder violate some desired property just by observing traffic on the network?
- *Active intruder problem*: can the intruder violate some desired property by orchestrating DY-allowable behaviours and then observing the resulting network traffic?

LOOKING FOR PROOFS

- Is this always easy? Is it even always *doable*?
- Passive intruder problem: Fixed X, fixed t, try to find a proof
- Active intruder problem: Come up with an X and a suitable mapping for variables (what variables? we'll see later!) in X and t such that there is a proof of t from X

PASSIVE INTRUDER PROBLEM

- Given an X and a t, check if X derives t

RECALL: PROOF SYSTEM

$\dfrac{}{X \vdash m}\ \mathbf{ax}(m \in X)$	
$\dfrac{X \vdash (t_1, t_2)}{X \vdash t_i}\ \mathbf{split}$	$\dfrac{X \vdash t \quad X \vdash u}{X \vdash (t, u)}\ \mathbf{pair}$
$\dfrac{X \vdash \mathsf{enc}(t, \mathsf{pk}(k)) \quad X \vdash k}{X \vdash t}\ \mathbf{adec}$	$\dfrac{X \vdash t \quad X \vdash \mathsf{pk}(k)}{X \vdash \mathsf{enc}(t, \mathsf{pk}(k))}\ \mathbf{aenc}$

EXAMPLE 1

$X = \{$

$\text{aenc}(\ m,\ pk(k_1)\ ),$

$\text{pair}(\ k_2,\ \text{aenc}(\ \text{pair}(n, k_1),\ pk(k_3)\ )\ ),$

$\text{pair}(\ n,\ \text{aenc}(\ k_3,\ pk(k_2)\ )\ )$

$\}$

Is there a proof of $X \vdash m$?

EXAMPLE 2

$X = \{$

$\mathsf{aenc}(\, m,\ pk(k_I)\,),$

$\mathsf{pair}(\, k_2,\ \mathsf{aenc}(\, \mathsf{pair}(m, k_I),\ pk(k_3)\,)\,),$

$\mathsf{aenc}(\, k_3,\ pk(k_2)\,)$

$\}$

Is there a proof of $X \vdash m$?

EXAMPLE 3

$X = \{$

$\text{aenc}(\, m,\ pk(k_I)\,),$

$\text{pair}(\, k_2,\ \text{aenc}(\, \text{pair}(m, k_I),\ pk(k_3)\,)\,),$

$\text{aenc}(\, k_3,\ pk(k_2)\,)$

$\}$

Is there a proof of $X \vdash \text{aenc}(\, m,\ pk(k_3)\,)$?

PROOF SEARCH

- At first glance, not easy at all! Have to search "upwards" from the intended conclusion.
  - Which rules do we apply?
  - In what sequence?
  - Which terms do we apply them to? &c. &c.
- Want an efficient algorithm which, given X and t, can check if X derives t.

MORE ABOUT PROOF SEARCH

- No rule in our system changes the X on the LHS
- Constructor rules lead to "bigger" terms on the right
- Destructor rules lead to "smaller" terms on the right
- Can create arbitrarily large terms while searching for a proof of a tiny one

MINOR DETOUR: SIZES OF TERMS

- How do we measure the size of a term?
- Treat it like a tree, count the number of nodes
- Each subtree is called a “subterm”
- Define it inductively

$$size(t) = \begin{cases} 1, & \text{if } t \text{ is atomic} \\ 1 + \sum_{i=1}^{n} size(t_i), & \text{if there is an } f \text{ such that } t = f(t_1, \ldots, t_n) \end{cases}$$

ALL PROOFS ARE EQUAL…

- But some are more equal than others
- Want the shortest proof, with no “unnecessary detours”
  - Should not build up a new term only to break it down
- Can we always get such a proof?

NORMAL PROOFS

- A “normal” proof is one where no such detours happen
- A constructor rule is never “followed by” a destructor rule
- End result:
  - Can always first break down terms from the set on the LHS, before we start building new ones
  - Proof has structure and enjoys some interesting properties
  - Gives us a good handle on how to go about searching for proofs!

MORE ABOUT NORMAL PROOFS

- *Normalization theorem*: Can convert any proof into a normal one
- A normal proof also enjoys the *subterm property*, stated as follows

For any normal proof π of $X \vdash t$, every term occurring in π on the RHS is a subterm of $X \cup \{t\}$. In particular, if π ends in a destructor rule, every such term is a subterm of X alone.

PTIME ALGORITHM FOR CHECKING DERIVABILITY

- Given an X and a t, check if X derives t.
- Denote by *st* the (finite) set of all subterms of $X \cup \{t\}$. Let $N = |st|$.
- Start with $A := \emptyset$ and $B := X$. As long as $A \neq B$, set

$$A := B, \text{ and}$$

$$B := \{u \mid u \in st \text{ is derivable from } A \text{ using one application of any proof rule}\}$$

- Each iteration of the loop needs N^2 many steps: examine all pairs of terms in A and see if one can derive a new term in *st* using them
- At most N iterations in all: the loop stops if one cannot add any new terms to B