CISPA
HELMHOLTZ CENTER FOR
INFORMATION SECURITY

Formal Analysis of Real-World Security Protocols

Lecture 4: Verification Theory (Part 1)

Aleksi Peltonen | 18.11.2024

This lecture

Tamarin Workflow

Dependency Graphs

Constraint Systems

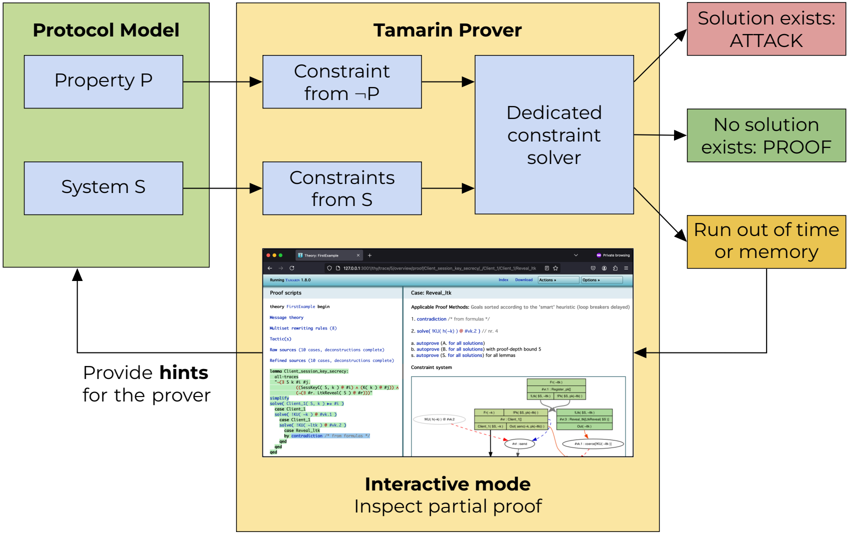
Tamarin Workflow

The Tamarin prover

Theorem prover

Constraint solver

Tamarin workflow



Semantics

- **Transition relation**

  $S \dashv\!\![\; a \;]\!\!\!\rightarrow_R ((S \setminus^{\#} l) \cup^{\#} r)$, where

  - $l \dashv\!\![\; a \;]\!\!\!\rightarrow r$ is a ground instance of a rule in R, and
  - $l \subseteq^{\#} S$ wrt the equational theory

- **Executions**

  - $execs(R) = \{ [\,] \dashv\!\![\; a_1 \;]\!\!\!\rightarrow \ldots \dashv\!\![\; a_n \;]\!\!\!\rightarrow S_n \mid \forall n.\ Fr(n)$ appears only once on the right-hand side of the rule $\}$

- **Traces**

  - $traces(R) = \{ [a_1, \ldots, a_n] \mid [\,] \dashv\!\![\; a_1 \;]\!\!\!\rightarrow \ldots \dashv\!\![\; a_n \;]\!\!\!\rightarrow S_n \in execs(R) \}$

- Question: Can we reach a specific state (encoded by actions)?

Example

MSR	Alternative syntax

`rule r1:`
`    [ Fr(~a), Fr(~k) ]`
`    -->`
`    [ St(~a, ~k)`
`    , Out(enc(~a,~k))`
`    , Key(~k) ]`
	$\dfrac{\mathrm{Fr(a)} \quad \mathrm{Fr(k)}}{\mathrm{St(a,k)} \quad \mathrm{Out(enc(a,k))} \quad \mathrm{Key(k)}}$

`rule r2:`
`    [ St(a, k)`
`    , In(<a, a>) ]`
`  --[ Fin(a, k) ]->`
`    [ ]`
	$\dfrac{\mathrm{St(a,k)} \quad \mathrm{In(\langle a, a\rangle)}}{}[\mathrm{Fin(a,k)}]$

`rule r3:`
`    [ Key(k) ]`
`  --[ Rev(k) ]->`
`    [ Out(k) ]`
	$\dfrac{\mathrm{Key(k)}}{\mathrm{Out(k)}}[\mathrm{Rev(k)}]$

```
// Fin(a, k) is reachable
lemma trace: exists-trace
    " Ex a k #i . Fin(a, k)@i "
```

$\exists a, k(Fin(a, k))$

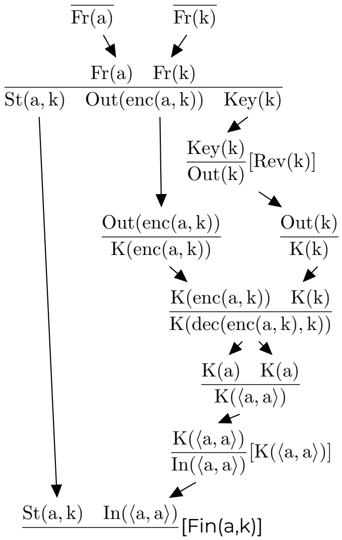
Example

MSR Instances	Resulting State
$\dfrac{}{\mathrm{Fr}(a)}$	$\{\mathrm{Fr}(a)\}^\sharp$
$\dfrac{}{\mathrm{Fr}(k)}$	$\{\mathrm{Fr}(a), \mathrm{Fr}(k)\}^\sharp$
$\dfrac{\mathrm{Fr}(a) \quad \mathrm{Fr}(k)}{\mathrm{St}(a, k) \quad \mathrm{Out}(\mathrm{enc}(a, k)) \quad \mathrm{Key}(k)}$	$\{\mathrm{St}(a,k), \mathrm{Out}(\mathrm{enc}(a,k)), \mathrm{Key}(k)\}^\sharp$
$\dfrac{\mathrm{Key}(k)}{\mathrm{Out}(k)}[\mathrm{Rev}(k)]$	$\{\mathrm{St}(a,k), \mathrm{Out}(\mathrm{enc}(a,k)), \mathrm{Out}(k)\}^\sharp$
$\dfrac{\mathrm{Out}(k)}{\mathrm{K}(k)}$	$\{\mathrm{St}(a,k), \mathrm{Out}(\mathrm{enc}(a,k)), \mathrm{K}(k)\}^\sharp$
$\dfrac{\mathrm{Out}(\mathrm{enc}(a, k))}{\mathrm{K}(\mathrm{enc}(a, k))}$	$\{\mathrm{St}(a,k), \mathrm{K}(\mathrm{enc}(a,k)), \mathrm{K}(k)\}^\sharp$
$\dfrac{\mathrm{K}(\mathrm{enc}(a, k)) \quad \mathrm{K}(k)}{\mathrm{K}(a)}$	$\{\mathrm{St}(a,k), \mathrm{K}(\mathrm{enc}(a,k)), \mathrm{K}(k), \mathrm{K}(a)\}^\sharp$
$\dfrac{\mathrm{K}(a) \quad \mathrm{K}(a)}{\mathrm{K}(\langle a, a\rangle)}$	$\{\mathrm{St}(a,k), \mathrm{K}(\mathrm{enc}(a,k)), \mathrm{K}(k), \mathrm{K}(a), \mathrm{K}(\langle a, a\rangle)\}^\sharp$
$\dfrac{\mathrm{K}(\langle a, a\rangle)}{\mathrm{In}(\langle a, a\rangle)}[\mathrm{K}(\langle a, a\rangle)]$	$\{\mathrm{St}(a,k), \mathrm{K}(\mathrm{enc}(a,k)), \mathrm{K}(k), \mathrm{K}(a), \mathrm{K}(\langle a, a\rangle), \mathrm{In}(\langle a, a\rangle)\}^\sharp$
$\dfrac{\mathrm{St}(a, k) \quad \mathrm{In}(\langle a, a\rangle)}{}[\mathrm{Fin}(a,k)]$	$\{\mathrm{K}(\mathrm{enc}(a,k)), \mathrm{K}(k), \mathrm{K}(a), \mathrm{K}(\langle a, a\rangle)\}^\sharp$

Dependency Graphs

Dependency graph intuition

- **Constraints** represent the minimal requirements for a valid solution
- **Dependency graphs** are used to abstractly represent constraints on traces
  - Each node instance is a rule
  - Edges connecting nodes represent facts being consumed
- Tamarin tries to prove that *at least one* trace instantiates the graph and produce a counterexample



Dependency graph definition

Let E be an equational theory and R be a set of multiset rewriting rules. We say that $dg = (I, D)$ is a dependency graph modulo E for R if $I \in (ginsts(R \cup \{\text{FRESH}\}))^*$, $D \subseteq \mathbb{N}^2 \times \mathbb{N}^2$, and dg satisfies the following conditions:

DG1 For every edge $(i, u) \rightarrowtail (j, v) \in D$, it holds that $i < j$ and the conclusion fact of (i, u) is equal modulo E to the premise fact of (j, v).

DG2 Every premise of dg has exactly one incoming edge.

DG3 Every linear conclusion of dg has at most one outgoing edge.

DG4 The FRESH rule instances in I are unique.

Dependency graphs and traces

- Dependency graphs provide us with an alternative formulation of the multiset rewriting semantics given in the previous lectures
- We exploit this alternative semantic in our backwards reachability analysis by incrementally constructing dependency graphs instead of (action-)traces.
- Theorem: *for every multiset rewriting system R and every equational theory E, holds that:*

$$traces_E(R) =_E trace(dg) \mid dg \in dgraphs_E(R)$$

Finding traces

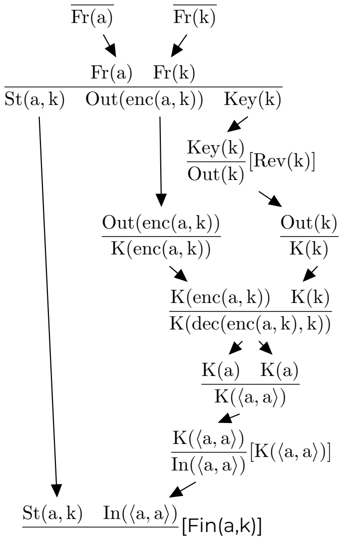
- **Goal:** Derive constraints from the multiset rewriting rules and properties to prove that *at least one* possible trace exists
- **Constraint solving** (intuition):
  1. Create an empty constraint system
  2. Add node constraints corresponding to the actions in the formula
     - e.g., not Ex A(x): add rules that contain A(x)
     - If variables are used, consider them free
  3. Add premise constraints for the nodes we added in the previous step
  4. Continue adding node and edge constraints (with equal variables) until we can construct a trace

Dependency graph example

$$PE := \left\{ \frac{Fr(a) \quad Fr(k)}{St(a,k) \quad Out(enc(a,k)) \quad Key(k)} \right\} \cup \left\{ \frac{St(a,k) \quad In(\langle a,a \rangle)}{}[Fin(a,k)] \right\} \cup \left\{ \frac{Key(k)}{Out(k)}[Rev(k)] \right\}$$

$$MD := \left\{ \frac{Out(x)}{K(x)}, \quad \frac{K(x)}{In(x)}[K(x)], \quad \frac{}{K(x:pub)} \right\} \cup \left\{ \frac{Fr(x:fresh)}{K(x:fresh)} K(x:fresh) \right\} \cup \left\{ \frac{K(x_1)\dots K(x_k)}{K(f(x_1\dots x_k))} \mid f \in \Sigma^k \right\}$$

$$SR := \left\{ \frac{}{Fr(x)} \right\}$$



Does this always work?

- There is only one possible source for `St(a,k)`
- There is no rule that produces `Out(`$\langle \mathrm{a}, \mathrm{a} \rangle$`)` so it must come from the attacker. **How did the attacker construct it?** Multiple options!

$$\frac{\mathrm{K(a)} \quad \mathrm{K(a)}}{\mathrm{K}(\langle \mathrm{a}, \mathrm{a} \rangle)}$$

$$\frac{\mathrm{Fr(a)} \quad \mathrm{Fr(k)}}{\mathrm{Out(enc(a,k))} \quad \mathrm{Key(k)} \quad \mathrm{St(a,k)}} \qquad \frac{\mathrm{K}(\langle \mathrm{a}, \mathrm{a} \rangle)}{\mathrm{In}(\langle \mathrm{a}, \mathrm{a} \rangle)}[\mathrm{K}(\langle \mathrm{a}, \mathrm{a} \rangle)]$$

$$\frac{\mathrm{St(a,k)} \quad \mathrm{In}(\langle \mathrm{a}, \mathrm{a} \rangle)}{}\,\text{[Fin(a,k)]}$$

Does this always work?

- There is only one possible source for `St(a,k)`
- There is no rule that produces `Out`$(\langle a, a\rangle)$ so it must come from the attacker. **How did the attacker construct it?** Multiple options!

$$\dots$$

$$\dots$$

$$\frac{K(enc(enc(\langle a, a\rangle, k_1), k_2)) \quad K(k_2)}{K(dec(enc(enc(\langle a, a\rangle, k_1), k_2), k_2))}$$

$$\frac{K(enc(\langle a, a\rangle, k_1)) \quad K(k_1)}{K(dec(enc(\langle a, a\rangle, k_1), k_1))}$$

$$\frac{Fr(a) \quad Fr(k)}{Out(enc(a, k)) \quad Key(k) \quad St(a, k)} \qquad \frac{K(\langle a, a\rangle)}{In(\langle a, a\rangle)}[K(\langle a, a\rangle)]$$

$$\frac{St(a, k) \quad In(\langle a, a\rangle)}{}\text{[Fin(a,k)]}$$

Constraint Systems

How Can an Attacker Use Compound Terms?
Construct Them:
The attacker can create a compound term using what they know.
For example, if the attacker knows m and k, they can construct enc(m,k)

Learn Them:
The attacker can obtain a compound term directly from the network.
for eg the value of enc(m,k)

Construction and deconstruction

A compound term is a term that combines multiple elements eg. enc(m,k)
It's non-atomic, meaning it's built from smaller pieces eg m and k

We need to prevent the attacker from performing **unnecessary** steps

- No need to first construct, then deconstruct
  - e.g., attacker learns `m` and `k`, then applies `enc(m, k)`, then applies `dec(enc(m, k), k)`
- For compound (non-atomic) terms, the attacker can either
  - construct them, or
  - learn them from an `Out()` fact
- We can use this!
  - When considering the possibility that a term was deconstructed, there must be a **chain** from an `Out()` to the `K()` fact

Attacker deduction through (de)construction

Construction rules:

$$\frac{}{\mathrm{K}^{\uparrow}(\mathrm{x}:\mathrm{pub})}[\mathrm{K}^{\uparrow}(\mathrm{x}:\mathrm{pub})] \qquad \frac{\mathrm{Fr}(\mathrm{x}:\mathrm{fresh})}{\mathrm{K}^{\uparrow}(\mathrm{x}:\mathrm{fresh})}[\mathrm{K}^{\uparrow}(\mathrm{x}:\mathrm{fresh})] \qquad \frac{\mathrm{K}^{\uparrow}(\mathrm{x})}{\mathrm{K}^{\uparrow}(\mathrm{h}(\mathrm{x}))}[\mathrm{K}^{\uparrow}(\mathrm{h}(\mathrm{x}))]$$

$$\frac{\mathrm{K}^{\uparrow}(\mathrm{x}) \quad \mathrm{K}^{\uparrow}(\mathrm{y})}{\mathrm{K}^{\uparrow}(\mathrm{enc}(\mathrm{x},\mathrm{y}))}[\mathrm{K}^{\uparrow}(\mathrm{enc}(\mathrm{x},\mathrm{y}))] \qquad \frac{\mathrm{K}^{\uparrow}(\mathrm{x}) \quad \mathrm{K}^{\uparrow}(\mathrm{y})}{\mathrm{K}^{\uparrow}(\mathrm{dec}(\mathrm{x},\mathrm{y}))}[\mathrm{K}^{\uparrow}(\mathrm{dec}(\mathrm{x},\mathrm{y}))]$$

$$\frac{\mathrm{K}^{\uparrow}(\mathrm{x}) \quad \mathrm{K}^{\uparrow}(\mathrm{y})}{\mathrm{K}^{\uparrow}(\langle\mathrm{x},\mathrm{y}\rangle)}[\mathrm{K}^{\uparrow}(\langle\mathrm{x},\mathrm{y}\rangle)] \qquad \frac{\mathrm{K}^{\uparrow}(\mathrm{x})}{\mathrm{K}^{\uparrow}(\mathrm{fst}(\mathrm{x}))}[\mathrm{K}^{\uparrow}(\mathrm{fst}(\mathrm{x}))] \qquad \frac{\mathrm{K}^{\uparrow}(\mathrm{x})}{\mathrm{K}^{\uparrow}(\mathrm{snd}(\mathrm{x}))}[\mathrm{K}^{\uparrow}(\mathrm{snd}(\mathrm{x}))]$$

Deconstruction rules:

$$\frac{\mathrm{K}^{\downarrow}(\langle\mathrm{x},\mathrm{y}\rangle)}{\mathrm{K}^{\downarrow}(\mathrm{x})} \qquad \frac{\mathrm{K}^{\downarrow}(\langle\mathrm{x},\mathrm{y}\rangle)}{\mathrm{K}^{\downarrow}(\mathrm{y})} \qquad \frac{\mathrm{K}^{\downarrow}(\mathrm{enc}(\mathrm{x},\mathrm{y})) \quad \mathrm{K}^{\uparrow}(\mathrm{y})}{\mathrm{K}^{\downarrow}(\mathrm{x})}$$

K |^ (x)The attacker knows x knowledge the attacker possesses).

K |v (x): The attacker deduces x (knowledge the attacker derives).

Message deduction: (de)construction

Communication rules:

$$\textsc{IRecv}\frac{\mathrm{Out}(x)}{\mathrm{K}^{\downarrow}(x)} \qquad \textsc{ISend}\frac{\mathrm{K}^{\uparrow}(x)}{\mathrm{In}(x)}[\mathrm{K}(x)]$$

Coerce rule:

$$\textsc{Coerce}\frac{\mathrm{K}^{\downarrow}(x)}{\mathrm{K}^{\uparrow}(x)}[\mathrm{K}^{\uparrow}(x)]$$

Summary

Next lecture

- We now know how to model..
  - ..protocol behavior as **multiset rewriting rules**
  - ..protocol properties as **first-order logic formulas**
- We also have an intuition of Tamarin's workflow and how to represent the system as **dependency graphs**
- In the next lecture, we will talk more about constraint solving

Reading material

Recommended reading: [Bas+24, Ch. 7.2-7.3], [Sch+12]

[Bas+24] D. Basin, C. Cremers, J. Dreier, and R. Sasse. **Modeling and Analyzing Security Protocols with Tamarin: A Comprehensive Guide.** Draft v0.5. Sept. 2024.

[Sch+12] B. Schmidt, S. Meier, C. Cremers, and D. Basin. **Automated Analysis of Diffie-Hellman Protocols and Advanced Security Properties.** In: 2012 IEEE 25th Computer Security Foundations Symposium. 2012.