The Journey towards a Reference Implementation of IPSec

Automatic Security Analysis with Tamarin-Prover

Eike Stadtländer

July 12, 2018

Outline

Motivation

Motivation

Modern proofs are error-prone, they become more complex and they are created faster than they can be verified.

Motivation

Modern proofs are error-prone, they become more complex and they are created faster than they can be verified.

- Gerhard Opfer. "An Analytic Approach to the Collatz 3n+1 Problem". In: *Hamburger Beiträge zur Angewandten Mathematik* 9 (2011)

Motivation

Modern proofs are error-prone, they become more complex and they are created faster than they can be verified.

- Gerhard Opfer. “An Analytic Approach to the Collatz 3n+1 Problem”. In: *Hamburger Beiträge zur Angewandten Mathematik* 9 (2011)
- Norbert Blum. *A Solution of the P versus NP Problem*. Aug. 11, 2017. arXiv: `1708.03486v2 [cs.CC]`

Motivation

Modern proofs are error-prone, they become more complex and they are created faster than they can be verified.

- Gerhard Opfer. "An Analytic Approach to the Collatz 3n+1 Problem". In: *Hamburger Beiträge zur Angewandten Mathematik* 9 (2011)
- Norbert Blum. *A Solution of the P versus NP Problem*. Aug. 11, 2017. arXiv: `1708.03486v2 [cs.CC]`
- Sinichi Mochizuki. *Inter-universal Teichmüller Theory I-IV*. . June 28, 2018 (recently updated: 602 pages)

Motivation

Modern proofs are error-prone, they become more complex and they are created faster than they can be verified.

- Gerhard Opfer. "An Analytic Approach to the Collatz 3n+1 Problem". In: *Hamburger Beiträge zur Angewandten Mathematik* 9 (2011)
- Norbert Blum. *A Solution of the P versus NP Problem*. Aug. 11, 2017. arXiv: `1708.03486v2 [cs.CC]`
- Sinichi Mochizuki. *Inter-universal Teichmüller Theory I-IV*. . June 28, 2018 (recently updated: 602 pages)

Security of network protocols is critical:

Motivation

Modern proofs are error-prone, they become more complex and they are created faster than they can be verified.

- Gerhard Opfer. "An Analytic Approach to the Collatz 3n+1 Problem". In: *Hamburger Beiträge zur Angewandten Mathematik* 9 (2011)
- Norbert Blum. *A Solution of the P versus NP Problem*. Aug. 11, 2017. arXiv: `1708.03486v2 [cs.CC]`
- Sinichi Mochizuki. *Inter-universal Teichmüller Theory I-IV*. . June 28, 2018 (recently updated: 602 pages)

Security of network protocols is critical:

- corporate espionage (Germany: 55bn. €/a, bitkom 2017)

Motivation

Modern proofs are error-prone, they become more complex and they are created faster than they can be verified.

- Gerhard Opfer. “An Analytic Approach to the Collatz 3n+1 Problem”. In: *Hamburger Beiträge zur Angewandten Mathematik* 9 (2011)
- Norbert Blum. *A Solution of the P versus NP Problem*. Aug. 11, 2017. arXiv: 1708.03486v2 [cs.CC]
- Sinichi Mochizuki. *Inter-universal Teichmüller Theory I-IV*. . June 28, 2018 (recently updated: 602 pages)

Security of network protocols is critical:

- corporate espionage (Germany: 55bn. €/a, bitkom 2017)

The security proofs are not always trustworthy (Halevi 2005; Bellare and Rogaway 2004). Automatic security analysis aims to improve trustworthiness of security proofs.

Tamarin-Prover

Tamarin-Prover

Overview

- Developed by Simon Meier and Benedikt Schmidt as part of their PhD theses. (Meier 2013; Schmidt 2012).

Tamarin-Prover

Overview

- Developed by Simon Meier and Benedikt Schmidt as part of their PhD theses. (Meier 2013; Schmidt 2012). We will follow Meier 2013, Chapters 7,8.

Tamarin-Prover

Overview

- Developed by Simon Meier and Benedikt Schmidt as part of their PhD theses. (Meier 2013; Schmidt 2012). We will follow Meier 2013, Chapters 7,8.
- Security protocol verification tool
    - based on labeled multiset rewriting

Tamarin-Prover

Overview

- Developed by Simon Meier and Benedikt Schmidt as part of their PhD theses. (Meier 2013; Schmidt 2012). We will follow Meier 2013, Chapters 7,8.
- Security protocol verification tool
    - based on labeled multiset rewriting
- Symbolic model
    - messages are not bitstrings but terms
    - relations between terms are given by equational theories

Tamarin-Prover

Overview

- Developed by Simon Meier and Benedikt Schmidt as part of their PhD theses. (Meier 2013; Schmidt 2012). We will follow Meier 2013, Chapters 7,8.
- Security protocol verification tool
  - based on labeled multiset rewriting
- Symbolic model
  - messages are not bitstrings but terms
  - relations between terms are given by equational theories
- Dolev-Yao attacker
  - cryptographic primitives are handled as black-boxes
  - active attacker has complete control over the network
  - access to a corrupt oracle

Term Algebras and Cryptographic Messages I

Definition

Example (Cryptographic messages)

, ,

Term Algebras and Cryptographic Messages I

Definition

A order-sorted signature is a triple $(S, \leq, \Sigma)$

Example (Cryptographic messages)

, ,

Term Algebras and Cryptographic Messages I

Definition

A order-sorted signature is a triple $(S, \leq, \Sigma)$ where $(S, \leq)$ is partially-ordered set of sorts

Example (Cryptographic messages)

, ,

Term Algebras and Cryptographic Messages I

Definition

A order-sorted signature is a triple $(S, \leq, \Sigma)$ where $(S, \leq)$ is partially-ordered set of sorts

Example (Cryptographic messages)

$S = \{\text{msg}, \quad\quad \}$, ,

Term Algebras and Cryptographic Messages I

Definition

A order-sorted signature is a triple $(S, \leq, \Sigma)$ where $(S, \leq)$ is partially-ordered set of sorts

Example (Cryptographic messages)

$S = \{\text{msg}, \text{fresh}, \text{pub}\}$, ,

Term Algebras and Cryptographic Messages I

Definition

A order-sorted signature is a triple $(S, \leq, \Sigma)$ where $(S, \leq)$ is partially-ordered set of sorts

Example (Cryptographic messages)

$S = \{\mathsf{msg}, \mathsf{fresh}, \mathsf{pub}\}$, $\mathsf{fresh} \leq \mathsf{msg}, \mathsf{pub} \leq \mathsf{msg}$,

Term Algebras and Cryptographic Messages I

Definition

A order-sorted signature is a triple $(S, \leq, \Sigma)$ where $(S, \leq)$ is partially-ordered set of sorts and Σ is a set of function symbols associated with the sorts

Example (Cryptographic messages)

$S = \{\text{msg}, \text{fresh}, \text{pub}\}$, $\text{fresh} \leq \text{msg}, \text{pub} \leq \text{msg}$,

Term Algebras and Cryptographic Messages I

Definition

A order-sorted signature is a triple $(S, \leq, \Sigma)$ where $(S, \leq)$ is partially-ordered set of sorts and Σ is a set of function symbols associated with the sorts

Example (Cryptographic messages)

$S = \{\mathsf{msg}, \mathsf{fresh}, \mathsf{pub}\}$, $\mathsf{fresh} \leq \mathsf{msg}, \mathsf{pub} \leq \mathsf{msg}$,

$$\Sigma_{\mathsf{PHS}} = \{\langle\cdot,\cdot\rangle, \mathsf{fst}(\cdot), \mathsf{snd}(\cdot), \mathsf{h}(\cdot), \mathsf{senc}(\cdot,\cdot), \mathsf{sdec}(\cdot,\cdot)\}$$

Term Algebras and Cryptographic Messages I

Definition

A order-sorted signature is a triple $(S, \leq, \Sigma)$ where $(S, \leq)$ is partially-ordered set of sorts and Σ is a set of function symbols associated with the sorts such that

1. For every $s \in S$ the connected component C of s contains a top sort $\mathsf{top}(s)$ satisfying $\forall c \in C : c \leq \mathsf{top}(s)$.

Example (Cryptographic messages)

$S = \{\mathsf{msg}, \mathsf{fresh}, \mathsf{pub}\}$, $\mathsf{fresh} \leq \mathsf{msg}, \mathsf{pub} \leq \mathsf{msg}$,

$$\Sigma_{\mathsf{PHS}} = \{\langle\cdot,\cdot\rangle, \mathsf{fst}(\cdot), \mathsf{snd}(\cdot), \mathsf{h}(\cdot), \mathsf{senc}(\cdot,\cdot), \mathsf{sdec}(\cdot,\cdot)\}$$

Term Algebras and Cryptographic Messages I

Definition

A order-sorted signature is a triple $(S, \leq, \Sigma)$ where $(S, \leq)$ is partially-ordered set of sorts and Σ is a set of function symbols associated with the sorts such that

1. For every $s \in S$ the connected component C of s contains a top sort $\mathrm{top}(s)$ satisfying $\forall c \in C : c \leq \mathrm{top}(s)$.
2. For every k-ary function symbol $f : s_1 \times \cdots \times s_k \to s \in \Sigma$, we also have $f : \mathrm{top}(s_1) \times \cdots \times \mathrm{top}(s_k) \to \mathrm{top}(s) \in \Sigma$.

Example (Cryptographic messages)

$S = \{\mathrm{msg}, \mathrm{fresh}, \mathrm{pub}\}$, $\mathrm{fresh} \leq \mathrm{msg}, \mathrm{pub} \leq \mathrm{msg}$,

$$\Sigma_{\mathrm{PHS}} = \{\langle \cdot, \cdot \rangle, \mathrm{fst}(\cdot), \mathrm{snd}(\cdot), \mathrm{h}(\cdot), \mathrm{senc}(\cdot, \cdot), \mathrm{sdec}(\cdot, \cdot)\}$$

Term Algebras and Cryptographic Messages II

Definition

Example (Cryptographic messages)

Term Algebras and Cryptographic Messages II

Definition

Given a order-sorted signature $\mathbf{\Sigma} = (S, \leq, \Sigma)$.

Example (Cryptographic messages)

Term Algebras and Cryptographic Messages II

Definition

Given a order-sorted signature $\mathbf{\Sigma} = (S, \leq, \Sigma)$. For every sort $s \in S$ we assume there are countably infinite sets of variables $\mathcal{V}_s$ and constants $\mathcal{C}_s$

Example (Cryptographic messages)

Term Algebras and Cryptographic Messages II

Definition

Given a order-sorted signature $\mathbf{\Sigma} = (S, \leq, \Sigma)$. For every sort $s \in S$ we assume there are countably infinite sets of variables $\mathcal{V}_s$ and constants $\mathcal{C}_s$ where $\mathcal{C}_s \cap \mathcal{V}_s = \emptyset$ and $\mathcal{V}_s \cap \mathcal{V}_t = \emptyset = \mathcal{C}_s \cap \mathcal{C}_t$ if $s, t \in S, s \neq t$.

Example (Cryptographic messages)

Term Algebras and Cryptographic Messages II

Definition

Given a order-sorted signature $\mathbf{\Sigma} = (S, \leq, \Sigma)$. For every sort $s \in S$ we assume there are countably infinite sets of variables $\mathcal{V}_s$ and constants $\mathcal{C}_s$ where $\mathcal{C}_s \cap \mathcal{V}_s = \emptyset$ and $\mathcal{V}_s \cap \mathcal{V}_t = \emptyset = \mathcal{C}_s \cap \mathcal{C}_t$ if $s, t \in S, s \neq t$.

Then given $A \subseteq \bigcup_{s \in S} \mathcal{C}_s \cup \bigcup_{s \in S} \mathcal{V}_s$, $\mathcal{T}_\Sigma(A)$ denotes the set of all well-sorted terms constructed over $\Sigma \cup A$.

Example (Cryptographic messages)

Term Algebras and Cryptographic Messages II

Definition

Given a order-sorted signature $\mathbf{\Sigma} = (S, \leq, \Sigma)$. For every sort $s \in S$ we assume there are countably infinite sets of variables $\mathcal{V}_s$ and constants $\mathcal{C}_s$ where $\mathcal{C}_s \cap \mathcal{V}_s = \emptyset$ and $\mathcal{V}_s \cap \mathcal{V}_t = \emptyset = \mathcal{C}_s \cap \mathcal{C}_t$ if $s, t \in S, s \neq t$.

Then given $A \subseteq \bigcup_{s \in S} \mathcal{C}_s \cup \bigcup_{s \in S} \mathcal{V}_s$, $\mathcal{T}_\Sigma(A)$ denotes the set of all well-sorted terms constructed over $\Sigma \cup A$.

Example (Cryptographic messages)

Given $\mathbf{\Sigma}_{\mathsf{PHS}} = (\{\mathsf{msg}, \mathsf{fresh}, \mathsf{pub}\}, \leq, \Sigma_{\mathsf{PHS}})$ we have, for instance, the following well-sorted terms

Term Algebras and Cryptographic Messages II

Definition

Given a order-sorted signature $\mathbf{\Sigma} = (S, \leq, \Sigma)$. For every sort $s \in S$ we assume there are countably infinite sets of variables $\mathcal{V}_s$ and constants $\mathcal{C}_s$ where $\mathcal{C}_s \cap \mathcal{V}_s = \emptyset$ and $\mathcal{V}_s \cap \mathcal{V}_t = \emptyset = \mathcal{C}_s \cap \mathcal{C}_t$ if $s, t \in S, s \neq t$.

Then given $A \subseteq \bigcup_{s \in S} \mathcal{C}_s \cup \bigcup_{s \in S} \mathcal{V}_s$, $\mathcal{T}_\Sigma(A)$ denotes the set of all well-sorted terms constructed over $\Sigma \cup A$.

Example (Cryptographic messages)

Given $\mathbf{\Sigma}_{\mathrm{PHS}} = (\{\mathrm{msg}, \mathrm{fresh}, \mathrm{pub}\}, \leq, \Sigma_{\mathrm{PHS}})$ we have, for instance, the following well-sorted terms

$$m \in \mathcal{V}_{\mathrm{msg}},$$

Term Algebras and Cryptographic Messages II

Definition

Given a order-sorted signature $\mathbf{\Sigma} = (S, \leq, \Sigma)$. For every sort $s \in S$ we assume there are countably infinite sets of variables $\mathcal{V}_s$ and constants $\mathcal{C}_s$ where $\mathcal{C}_s \cap \mathcal{V}_s = \emptyset$ and $\mathcal{V}_s \cap \mathcal{V}_t = \emptyset = \mathcal{C}_s \cap \mathcal{C}_t$ if $s, t \in S, s \neq t$.

Then given $A \subseteq \bigcup_{s \in S} \mathcal{C}_s \cup \bigcup_{s \in S} \mathcal{V}_s$, $\mathcal{T}_\Sigma(A)$ denotes the set of all well-sorted terms constructed over $\Sigma \cup A$.

Example (Cryptographic messages)

Given $\mathbf{\Sigma}_{\mathsf{PHS}} = (\{\mathsf{msg}, \mathsf{fresh}, \mathsf{pub}\}, \leq, \Sigma_{\mathsf{PHS}})$ we have, for instance, the following well-sorted terms

$$m \in \mathcal{V}_{\mathsf{msg}}, \qquad \mathsf{fst}(\langle m, n \rangle),$$

Term Algebras and Cryptographic Messages II

Definition

Given a order-sorted signature $\mathbf{\Sigma} = (S, \leq, \Sigma)$. For every sort $s \in S$ we assume there are countably infinite sets of variables $\mathcal{V}_s$ and constants $\mathcal{C}_s$ where $\mathcal{C}_s \cap \mathcal{V}_s = \emptyset$ and $\mathcal{V}_s \cap \mathcal{V}_t = \emptyset = \mathcal{C}_s \cap \mathcal{C}_t$ if $s, t \in S, s \neq t$.

Then given $A \subseteq \bigcup_{s \in S} \mathcal{C}_s \cup \bigcup_{s \in S} \mathcal{V}_s$, $\mathcal{T}_\Sigma(A)$ denotes the set of all well-sorted terms constructed over $\Sigma \cup A$.

Example (Cryptographic messages)

Given $\mathbf{\Sigma}_{\mathsf{PHS}} = (\{\mathsf{msg}, \mathsf{fresh}, \mathsf{pub}\}, \leq, \Sigma_{\mathsf{PHS}})$ we have, for instance, the following well-sorted terms

$$m \in \mathcal{V}_{\mathsf{msg}}, \quad \mathsf{fst}(\langle m, n \rangle), \quad \mathsf{senc}(m, k), \quad \mathsf{sdec}(k_2, \mathsf{senc}(k_1, m))$$

Equational Theories and Cryptographic Primitives

Definition

Example (Cryptographic primitives)

Equational Theories and Cryptographic Primitives

Definition

Let $\mathbf{\Sigma}$ be a order-sorted signature. A pair $\{s, t\}$ of terms $s, t \in \mathcal{T}_{\Sigma}(\mathcal{V})$ is called an equation, we write $s = t$.

Example (Cryptographic primitives)

Equational Theories and Cryptographic Primitives

Definition

Let $\mathbf{\Sigma}$ be a order-sorted signature. A pair $\{s, t\}$ of terms $s, t \in \mathcal{T}_{\Sigma}(\mathcal{V})$ is called an equation, we write $s = t$.

Example (Cryptographic primitives)

Equational Theories and Cryptographic Primitives

Definition

Let $\mathbf{\Sigma}$ be a order-sorted signature. A pair $\{s, t\}$ of terms $s, t \in \mathcal{T}_{\Sigma}(\mathcal{V})$ is called an equation, we write $s = t$.

Example (Cryptographic primitives)

Given $\mathbf{\Sigma}_{\mathsf{PHS}}$ as before. We define

$$E_{\mathsf{PHS}} = \{\mathsf{fst}(\langle x, y\rangle) = x, \mathsf{snd}(\langle x, y\rangle) = y, \mathsf{sdec}(k, \mathsf{senc}(k, m)) = m\}$$

Equational Theories and Cryptographic Primitives

Definition

Let $\boldsymbol{\Sigma}$ be a order-sorted signature. A pair $\{s, t\}$ of terms $s, t \in \mathcal{T}_{\Sigma}(\mathcal{V})$ is called an equation, we write $s = t$.
The equational theory defined by E is the smallest congruence relation $=_E$ containing all instances of equations in E.

Example (Cryptographic primitives)

Given $\boldsymbol{\Sigma}_{\mathsf{PHS}}$ as before. We define

$$E_{\mathsf{PHS}} = \{\mathsf{fst}(\langle x, y\rangle) = x, \mathsf{snd}(\langle x, y\rangle) = y, \mathsf{sdec}(k, \mathsf{senc}(k, m)) = m\}$$

Labeled Multiset Rewriting and Protocol Specification I

Definition

Example

Labeled Multiset Rewriting and Protocol Specification I

Definition

Let Σ_{Fact} be an unsorted signature partitioned into linear and persistent fact symbols.

Example

Labeled Multiset Rewriting and Protocol Specification I

Definition

Let Σ_{Fact} be an unsorted signature partitioned into linear and persistent fact symbols.

Given a order-sorted term algebra $\mathcal{T}$, we define the set of all facts by

$$\mathcal{F} = \{F(t_1, \ldots, t_k) \mid t_1, \ldots, t_k \in \mathcal{T}, F \in \Sigma_{\mathsf{Fact}}, \mathsf{arity}(F) = k\}$$

Example

Labeled Multiset Rewriting and Protocol Specification I

Definition

Let Σ_{Fact} be an unsorted signature partitioned into linear and persistent fact symbols.

Given a order-sorted term algebra $\mathcal{T}$, we define the set of all facts by

$$\mathcal{F} = \{F(t_1, \ldots, t_k) \mid t_1, \ldots, t_k \in \mathcal{T}, F \in \Sigma_{\mathsf{Fact}}, \mathsf{arity}(F) = k\}$$

Example

In our example from before: $\mathsf{Secret}(m, k)$

Labeled Multiset Rewriting and Protocol Specification I

Definition

Let Σ_{Fact} be an unsorted signature partitioned into linear and persistent fact symbols.

Given a order-sorted term algebra $\mathcal{T}$, we define the set of all facts by

$$\mathcal{F} = \{F(t_1, \ldots, t_k) \mid t_1, \ldots, t_k \in \mathcal{T}, F \in \Sigma_{\mathsf{Fact}}, \mathsf{arity}(F) = k\}$$

A (labeled) multiset rewriting rule is a triple (p, a, c) of finite sequences $p, a, c \in \mathcal{F}^*$, written $p \,{-}[\, a \,]{\rightarrowtail}\, c$.

Example

In our example from before: $\mathsf{Secret}(m, k)$

Labeled Multiset Rewriting and Protocol Specification I

Definition

Let Σ_{Fact} be an unsorted signature partitioned into linear and persistent fact symbols.

Given a order-sorted term algebra $\mathcal{T}$, we define the set of all facts by

$$\mathcal{F} = \{F(t_1, \ldots, t_k) \mid t_1, \ldots, t_k \in \mathcal{T}, F \in \Sigma_{\mathsf{Fact}}, \mathsf{arity}(F) = k\}$$

A (labeled) multiset rewriting rule is a triple (p, a, c) of finite sequences $p, a, c \in \mathcal{F}^*$, written $p \dashv\!\![\, a \,]\!\!\rightarrowtail c$.

Example

In our example from before: $[\mathsf{Secret}(m, k)] \dashv\!\![\, \mathsf{Encrypted}(m) \,]\!\!\rightarrowtail [\mathsf{Out}(\mathsf{senc}(k, m))]$

Labeled Multiset Rewriting and Protocol Specification I

Definition

Let Σ_{Fact} be an unsorted signature partitioned into linear and persistent fact symbols. Furthermore, assume there is a designated fact symbol $\mathsf{Fr} \in \Sigma_{\mathsf{Fact}}$ modelling freshness.

Given a order-sorted term algebra $\mathcal{T}$, we define the set of all facts by

$$\mathcal{F} = \{F(t_1, \ldots, t_k) \mid t_1, \ldots, t_k \in \mathcal{T}, F \in \Sigma_{\mathsf{Fact}}, \mathsf{arity}(F) = k\}$$

A (labeled) multiset rewriting rule is a triple (p, a, c) of finite sequences $p, a, c \in \mathcal{F}^*$, written $p \dashv\![\, a \,]\!\rightarrowtail c$.

Example

In our example from before: $[\mathsf{Secret}(m, k)] \dashv\![\, \mathsf{Encrypted}(m) \,]\!\rightarrowtail [\mathsf{Out}(\mathsf{senc}(k, m))]$

Labeled Multiset Rewriting and Protocol Specification II

Definition

Let E define an equational theory $=_E$. Let R be a finite set of multiset rewriting rules.

Example

Labeled Multiset Rewriting and Protocol Specification II

Definition

Let E define an equational theory $=_E$. Let R be a finite set of multiset rewriting rules. Then we call R a (labeled) multiset rewriting system if it satisfies the following properties

Example

Labeled Multiset Rewriting and Protocol Specification II

Definition

Let E define an equational theory $=_E$. Let R be a finite set of multiset rewriting rules. Then we call R a (labeled) multiset rewriting system if it satisfies the following properties

1. No rule in R contains a fresh name ($\mathcal{V}_{\mathsf{fresh}}$).

Example

Labeled Multiset Rewriting and Protocol Specification II

Definition

Let E define an equational theory $=_E$. Let R be a finite set of multiset rewriting rules. Then we call R a (labeled) multiset rewriting system if it satisfies the following properties

1. No rule in R contains a fresh name ($\mathcal{V}_{\mathsf{fresh}}$).
2. No conclusion of a rule in R contains a Fr fact.

Example

Labeled Multiset Rewriting and Protocol Specification II

Definition

Let E define an equational theory $=_E$. Let R be a finite set of multiset rewriting rules. Then we call R a (labeled) multiset rewriting system if it satisfies the following properties

1. No rule in R contains a fresh name ($\mathcal{V}_{\mathsf{fresh}}$).
2. No conclusion of a rule in R contains a Fr fact.
3. No conclusion of a rule instance in R contains a fresh name which does not occur in one of its premises (modulo E).

Example

Labeled Multiset Rewriting and Protocol Specification II

Definition

Let E define an equational theory $=_E$. Let R be a finite set of multiset rewriting rules. Then we call R a (labeled) multiset rewriting system if it satisfies the following properties

1. No rule in R contains a fresh name ($\mathcal{V}_{\text{fresh}}$).
2. No conclusion of a rule in R contains a Fr fact.
3. No conclusion of a rule instance in R contains a fresh name which does not occur in one of its premises (modulo E).

Example

Message deduction rules (means of the attacker):

$$\mathsf{MD}_\Sigma := \left\{ \qquad\qquad\qquad \right\}$$

Labeled Multiset Rewriting and Protocol Specification II

Definition

Let E define an equational theory $=_E$. Let R be a finite set of multiset rewriting rules. Then we call R a (labeled) multiset rewriting system if it satisfies the following properties

1. No rule in R contains a fresh name ($\mathcal{V}_{\mathsf{fresh}}$).
2. No conclusion of a rule in R contains a Fr fact.
3. No conclusion of a rule instance in R contains a fresh name which does not occur in one of its premises (modulo E).

Example

Message deduction rules (means of the attacker):

$$\mathsf{MD}_\Sigma := \left\{ \quad \mathsf{Out}(x) \longrightarrow \mathsf{K}(x), \quad \right\}$$

Labeled Multiset Rewriting and Protocol Specification II

Definition

Let E define an equational theory $=_E$. Let R be a finite set of multiset rewriting rules. Then we call R a (labeled) multiset rewriting system if it satisfies the following properties

1. No rule in R contains a fresh name ($\mathcal{V}_{\mathsf{fresh}}$).
2. No conclusion of a rule in R contains a Fr fact.
3. No conclusion of a rule instance in R contains a fresh name which does not occur in one of its premises (modulo E).

Example

Message deduction rules (means of the attacker):

$$\mathsf{MD}_\Sigma := \left\{ \quad \mathsf{Out}(x) \longrightarrow \mathsf{K}(x), \qquad \mathsf{K}(x) -[\, \mathsf{K}(x) \,]\!\rightarrow \mathsf{In}(x), \right\}$$

Labeled Multiset Rewriting and Protocol Specification II

Definition

Let E define an equational theory $=_E$. Let R be a finite set of multiset rewriting rules. Then we call R a (labeled) multiset rewriting system if it satisfies the following properties

1. No rule in R contains a fresh name ($\mathcal{V}_{\mathsf{fresh}}$).
2. No conclusion of a rule in R contains a Fr fact.
3. No conclusion of a rule instance in R contains a fresh name which does not occur in one of its premises (modulo E).

Example

Message deduction rules (means of the attacker):

$$\mathsf{MD}_\Sigma := \left\{ \begin{array}{ccc} \mathsf{Out}(x) \longrightarrow \mathsf{K}(x), & & \mathsf{K}(x) \;\text{-}[\, \mathsf{K}(x) \,]\!\!\rightarrow \mathsf{In}(x), \\ \mathsf{Fr}(x : \mathsf{fresh}) \longrightarrow \mathsf{K}(x : \mathsf{fresh}) & & \end{array} \right\}$$

Labeled Multiset Rewriting and Protocol Specification II

Definition

Let E define an equational theory $=_E$. Let R be a finite set of multiset rewriting rules. Then we call R a (labeled) multiset rewriting system if it satisfies the following properties

1. No rule in R contains a fresh name ($\mathcal{V}_{\mathsf{fresh}}$).
2. No conclusion of a rule in R contains a Fr fact.
3. No conclusion of a rule instance in R contains a fresh name which does not occur in one of its premises (modulo E).

Example

Message deduction rules (means of the attacker):

$$\mathsf{MD}_\Sigma := \left\{ \begin{array}{cc} \mathsf{Out}(x) \longrightarrow \mathsf{K}(x), & \mathsf{K}(x) \dashv[\, \mathsf{K}(x) \,]\!\!\rightarrow \mathsf{In}(x), \\ \mathsf{Fr}(x : \mathsf{fresh}) \longrightarrow \mathsf{K}(x : \mathsf{fresh}) & \longrightarrow \mathsf{K}(x : \mathsf{pub}) \end{array} \right\}$$

Labeled Multiset Rewriting and Protocol Specification II

Definition

Let E define an equational theory $=_E$. Let R be a finite set of multiset rewriting rules. Then we call R a (labeled) multiset rewriting system if it satisfies the following properties

1. No rule in R contains a fresh name ($\mathcal{V}_{\mathsf{fresh}}$).
2. No conclusion of a rule in R contains a Fr fact.
3. No conclusion of a rule instance in R contains a fresh name which does not occur in one of its premises (modulo E).

Example

Message deduction rules (means of the attacker):

$$\mathsf{MD}_\Sigma := \left\{ \begin{array}{rlrl} \mathsf{Out}(x) \longrightarrow & \mathsf{K}(x), & \mathsf{K}(x) \;-\![\, \mathsf{K}(x) \,]\!\rightarrow & \mathsf{In}(x), \\ \mathsf{Fr}(x:\mathsf{fresh}) \longrightarrow & \mathsf{K}(x:\mathsf{fresh}) & \longrightarrow & \mathsf{K}(x:\mathsf{pub}) \end{array} \right\}$$

$$\cup \, \{\mathsf{K}(x_1), \ldots, \mathsf{K}(x_k) \longrightarrow \mathsf{K}(f(x_1, \ldots, x_k)) \mid f \in \Sigma, \mathsf{arity}(f) = k\}$$

Traces and Security Properties

Given a multiset rewriting system R and a equational theory by E. This yields a transition relation $\Longrightarrow_{R,E}$ modelling the application of rewriting rules to multisets of facts.

Traces and Security Properties

Given a multiset rewriting system R and a equational theory by E. This yields a transition relation $\Longrightarrow_{R,E}$ modelling the application of rewriting rules to multisets of facts.

$$\mathrm{traces}_E(R) = \{[A_1, A_2, \ldots, A_n] \mid \exists S_1, \ldots, S_n : \emptyset \overset{A_1}{\Longrightarrow}_{R,E} S_1 \overset{A_2}{\Longrightarrow}_{R,E} \ldots \overset{A_n}{\Longrightarrow}_{R,E} S_n\}$$

Traces and Security Properties

Given a multiset rewriting system R and a equational theory by E. This yields a transition relation $\Longrightarrow_{R,E}$ modelling the application of rewriting rules to multisets of facts.

$$\begin{aligned}\mathrm{traces}_E(R) = \{[A_1, A_2, \ldots, A_n] \mid \exists S_1, \ldots, S_n : \emptyset \overset{A_1}{\Longrightarrow}_{R,E} S_1 \overset{A_2}{\Longrightarrow}_{R,E} \ldots \overset{A_n}{\Longrightarrow}_{R,E} S_n \\ \wedge \text{ uniqueness condition for freshness}\}\end{aligned}$$

Traces and Security Properties

Given a multiset rewriting system R and a equational theory by E. This yields a transition relation $\Longrightarrow_{R,E}$ modelling the application of rewriting rules to multisets of facts.

$$\text{traces}_E(R) = \{[A_1, A_2, \ldots, A_n] \mid \exists S_1, \ldots, S_n : \emptyset \overset{A_1}{\Longrightarrow}_{R,E} S_1 \overset{A_2}{\Longrightarrow}_{R,E} \ldots \overset{A_n}{\Longrightarrow}_{R,E} S_n$$
$$\wedge \text{ uniqueness condition for freshness}\}$$

Security properties can then be formulated as first-order formulas on traces

Traces and Security Properties

Given a multiset rewriting system R and a equational theory by E. This yields a transition relation $\Longrightarrow_{R,E}$ modelling the application of rewriting rules to multisets of facts.

$$\mathrm{traces}_E(R) = \{[A_1, A_2, \ldots, A_n] \mid \exists S_1, \ldots, S_n : \emptyset \overset{A_1}{\Longrightarrow}_{R,E} S_1 \overset{A_2}{\Longrightarrow}_{R,E} \ldots \overset{A_n}{\Longrightarrow}_{R,E} S_n$$
$$\wedge \text{ uniqueness condition for freshness}\}$$

Security properties can then be formulated as first-order formulas on traces, e.g. secrecy properties:

$$\forall I, x : \mathsf{K}(x) \wedge \mathsf{Id}(I, x) \Rightarrow \mathsf{Corrupt}(I, x)$$

Theoretical Outlook

Let R be a multiset rewriting system (with conditions) and $=_E$ an equational theory.

Theoretical Outlook

Let R be a multiset rewriting system (with conditions) and $=_E$ an equational theory.

- Trace formulas φ can be R, E-valid or R, E-satisfiable (or neither).

Theoretical Outlook

Let R be a multiset rewriting system (with conditions) and $=_E$ an equational theory.

- Trace formulas φ can be R, E-valid or R, E-satisfiable (or neither).
- Every R, E-validity claim can be converted into a R, E-satisfiability claim.

Theoretical Outlook

Let R be a multiset rewriting system (with conditions) and $=_E$ an equational theory.

- Trace formulas φ can be R, E-valid or R, E-satisfiable (or neither).
- Every R, E-validity claim can be converted into a R, E-satisfiability claim.
    - φ is R, E-valid iff. $\neg\varphi$ is not R, E-satisfiable.

Theoretical Outlook

Let R be a multiset rewriting system (with conditions) and $=_E$ an equational theory.

- Trace formulas φ can be R, E-valid or R, E-satisfiable (or neither).
- Every R, E-validity claim can be converted into a R, E-satisfiability claim.
  - φ is R, E-valid iff. $\neg\varphi$ is not R, E-satisfiable.
- Security protocol verification boils down to searching R, E-satisfying traces.

Theoretical Outlook

Let R be a multiset rewriting system (with conditions) and $=_E$ an equational theory.

- Trace formulas φ can be R, E-valid or R, E-satisfiable (or neither).
- Every R, E-validity claim can be converted into a R, E-satisfiability claim.
  - φ is R, E-valid iff. $\neg\varphi$ is not R, E-satisfiable.
- Security protocol verification boils down to searching R, E-satisfying traces.
- Constraint systems are used to incrementally construct a satisfying trace by solving constraints.

Theoretical Outlook

Let R be a multiset rewriting system (with conditions) and $=_E$ an equational theory.

- Trace formulas φ can be R, E-valid or R, E-satisfiable (or neither).
- Every R, E-validity claim can be converted into a R, E-satisfiability claim.
  - φ is R, E-valid iff. $\neg\varphi$ is not R, E-satisfiable.
- Security protocol verification boils down to searching R, E-satisfying traces.
- Constraint systems are used to incrementally construct a satisfying trace by solving constraints.
- The constraint reduction rules are a heuristic giving rise to a verification algorithm. When the algorithm terminates it arrived either
  - at a trivially unsolvable constraint and the claim is falsified or
  - at a constraint system for which a trivial solution can be easily found and the claim is verified.

Theoretical Outlook

Let R be a multiset rewriting system (with conditions) and $=_E$ an equational theory.

- Trace formulas φ can be R, E-valid or R, E-satisfiable (or neither).
- Every R, E-validity claim can be converted into a R, E-satisfiability claim.
  - φ is R, E-valid iff. $\neg\varphi$ is not R, E-satisfiable.
- Security protocol verification boils down to searching R, E-satisfying traces.
- Constraint systems are used to incrementally construct a satisfying trace by solving constraints.
- The constraint reduction rules are a heuristic giving rise to a verification algorithm. When the algorithm terminates it arrived either
  - at a trivially unsolvable constraint and the claim is falsified or
  - at a constraint system for which a trivial solution can be easily found and the claim is verified.
- The underlying satisfiability problem is undecidable, the solver does not always terminate.

Overview of the Theoretical Part

Notion	Model

Overview of the Theoretical Part

Notion	Model
Terms	Cryptographic messages
Equational Theories	Semantics of cryptographic primitives

Overview of the Theoretical Part

Notion	Model
Terms	Cryptographic messages
Equational Theories	Semantics of cryptographic primitives
Rules	State transitions of protocol instances, Oracles
Action facts	Protocol transcript

Overview of the Theoretical Part

Notion	Model
Terms	Cryptographic messages
Equational Theories	Semantics of cryptographic primitives
Rules	State transitions of protocol instances, Oracles
Action facts	Protocol transcript
Rewriting Systems	Protocol Specification, Means of the Attacker

Overview of the Theoretical Part

Notion	Model
Terms	Cryptographic messages
Equational Theories	Semantics of cryptographic primitives
Rules	State transitions of protocol instances, Oracles
Action facts	Protocol transcript
Rewriting Systems	Protocol Specification, Means of the Attacker
Traces	Parallel executions of the protocol

Overview of the Theoretical Part

Notion	Model
Terms	Cryptographic messages
Equational Theories	Semantics of cryptographic primitives
Rules	State transitions of protocol instances, Oracles
Action facts	Protocol transcript
Rewriting Systems	Protocol Specification, Means of the Attacker
Traces	Parallel executions of the protocol
Trace Formulas	Security properties (e.g. executability, secrecy, authenticity)

Tamarin-Prover in Practice

(Short) Demo ☺

Reference Implementation of IPSec

Building Blocks for IPSec

Building Blocks for IPSec

- Random choices

Building Blocks for IPSec

- Random choices

```
rule gen_nonce:
    [ Fr(~n) ] --> [ State(~n) ]
```

Building Blocks for IPSec

- Random choices ✓

```
rule gen_nonce:
    [ Fr(~n) ] --> [ State(~n) ]
```

Building Blocks for IPSec

- Random choices ✓
- Cryptographic primitives

Building Blocks for IPSec

- Random choices ✓
- Cryptographic primitives
    - Diffie-Hellman exponentiation

```
rule dh_calc:
    let
        gab = ga ^ ~b
    in
    [ Fr(~b), In(<A, ga>) ] --> [ Out(<B, gab>) ]
```

Building Blocks for IPSec

- Random choices ✓
- Cryptographic primitives
  - Diffie-Hellman exponentiation (built-in)

```
rule dh_calc:
    let
        gab = ga ^ ~b
    in
    [ Fr(~b), In(<A, ga>) ] --> [ Out(<B, gab>) ]
```

Building Blocks for IPSec

- Random choices ✓
- Cryptographic primitives
  - Diffie-Hellman exponentiation (built-in)
  - Pseudo-random functions

Building Blocks for IPSec

- Random choices ✓
- Cryptographic primitives
  - Diffie-Hellman exponentiation (built-in)
  - Pseudo-random functions (function symbols, no collisions)

```
functions: prf/1
rule use_prf:
    let SKEYSEED = prf(<Ni, Nr, DH>)
    in
    [ State(Ni, Nr, DH) ] --> [ State(Ni, Nr, DH, SKEYSEED) ]
```

Building Blocks for IPSec

- Random choices ✓
- Cryptographic primitives
  - Diffie-Hellman exponentiation (built-in)
  - Pseudo-random functions (function symbols, no collisions)
  - Signature schemes (cf. demo)

Building Blocks for IPSec

- Random choices ✓
- Cryptographic primitives
  - Diffie-Hellman exponentiation (built-in)
  - Pseudo-random functions (function symbols, no collisions)
  - Signature schemes (cf. demo)
  - Authenticated encryption schemes

Building Blocks for IPSec

- Random choices ✓
- Cryptographic primitives
  - Diffie-Hellman exponentiation (built-in)
  - Pseudo-random functions (function symbols, no collisions)
  - Signature schemes (cf. demo)
  - Authenticated encryption schemes (use EtA for now)

```
rule use_aeenc:
let ct = senc(~secret, key_e)
    tag = mac(ct, key_a)
    hdr = < '120', ... >
in [ Fr(~secret), State(key_e, key_a) ] --> [ Out(<hdr, ct, tag>)]
```

Building Blocks for IPSec

- Random choices ✓
- Cryptographic primitives (✓)
  - Diffie-Hellman exponentiation (built-in)
  - Pseudo-random functions (function symbols, no collisions)
  - Signature schemes (cf. demo)
  - Authenticated encryption schemes (use EtA for now)

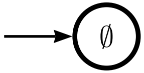
Building Blocks for IPSec

- Random choices ✓
- Cryptographic primitives (✓)
  - Diffie-Hellman exponentiation (built-in)
  - Pseudo-random functions (function symbols, no collisions)
  - Signature schemes (cf. demo)
  - Authenticated encryption schemes (use EtA for now)
- Certificates

Building Blocks for IPSec

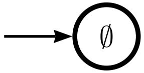
- Random choices ✓
- Cryptographic primitives (✓)
  - Diffie-Hellman exponentiation (built-in)
  - Pseudo-random functions (function symbols, no collisions)
  - Signature schemes (cf. demo)
  - Authenticated encryption schemes (use EtA for now)
- Certificates (use identifier and signature(s) for now ✓)

Finite State Machine



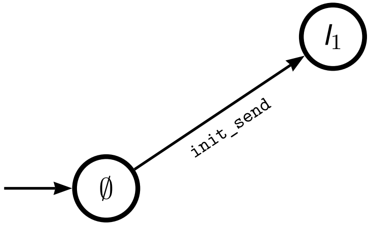
Finite State Machine

Init Phase



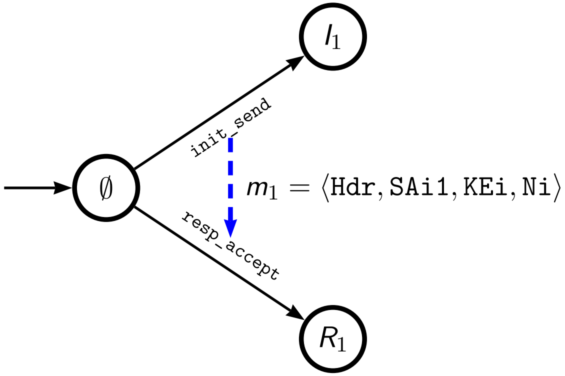
Finite State Machine

Init Phase


$$\begin{tikzcd} & I_1 \\ \arrow[r] & \emptyset \arrow[u, "\texttt{init\_send}"] \end{tikzcd}$$

Finite State Machine

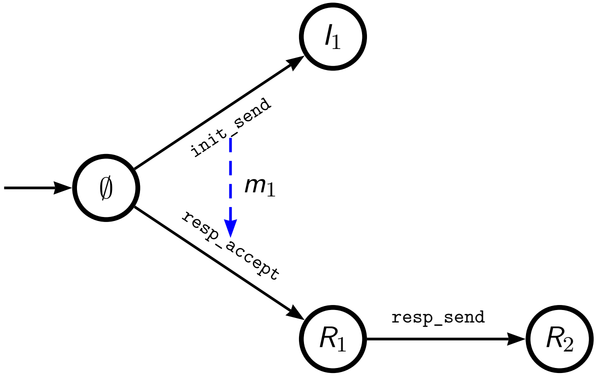
Init Phase



$\emptyset$
I_1
R_1
init_send
resp_accept
$m_1 = \langle \texttt{Hdr}, \texttt{SAi1}, \texttt{KEi}, \texttt{Ni} \rangle$

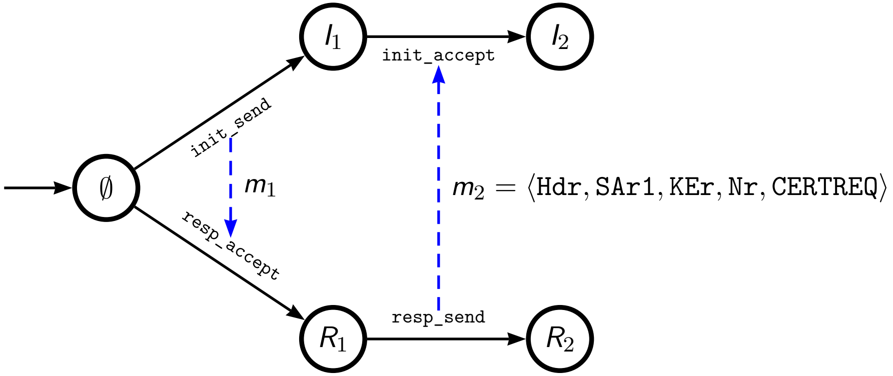
Finite State Machine

Init Phase



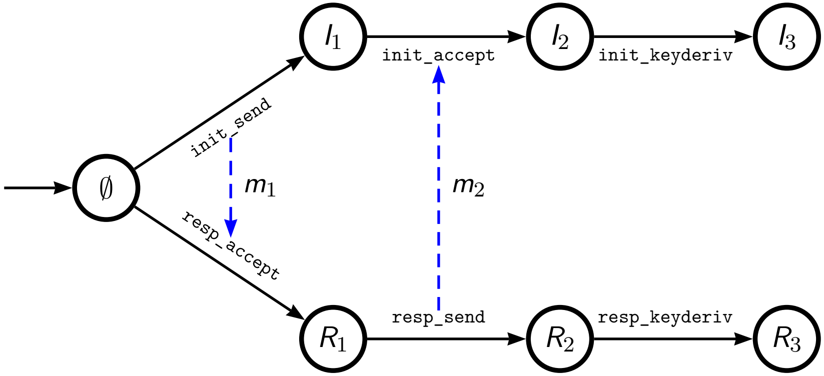
Finite State Machine

Init Phase


$\emptyset$
I_1
I_2
R_1
R_2
init_send
resp_accept
init_accept
resp_send
m_1
$m_2 = \langle \mathtt{Hdr}, \mathtt{SAr1}, \mathtt{KEr}, \mathtt{Nr}, \mathtt{CERTREQ} \rangle$

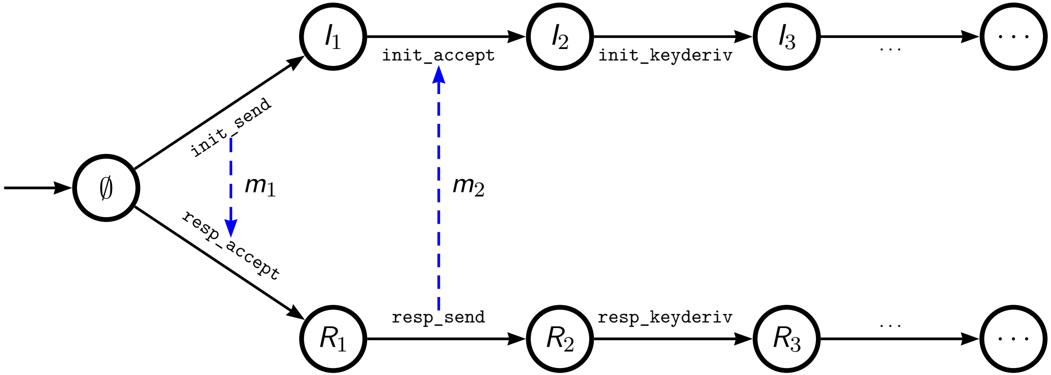
Finite State Machine

Init Phase



Finite State Machine

Init Phase



Code Walkthrough ☺

Lab-Goals Reflection

Goals for the Lab - Revisited

Goals for the Lab - Revisited

- Theory of Tamarin-Prover

- Practical Application

Goals for the Lab - Revisited

- Theory of Tamarin-Prover
  - mathematical foundation, in particular
    - order-sorted term algebras
    - equational theories
    - operations: substitution, replacements, unification, matching, rewriting modulo equational theories
  - How is the language of Tamarin-Prover reflecting those notions?
  - What are the limitations of Tamarin-Prover?
- Practical Application

Goals for the Lab - Revisited

- Theory of Tamarin-Prover
  - mathematical foundation, in particular ☺
    - order-sorted term algebras
    - equational theories
    - operations: substitution, replacements, unification, matching, rewriting modulo equational theories
  - How is the language of Tamarin-Prover reflecting those notions?
  - What are the limitations of Tamarin-Prover?
- Practical Application

Goals for the Lab - Revisited

- Theory of Tamarin-Prover
  - mathematical foundation, in particular ☺
    - order-sorted term algebras
    - equational theories
    - operations: substitution, replacements, unification, matching, rewriting modulo equational theories
  - How is the language of Tamarin-Prover reflecting those notions? ☺
  - What are the limitations of Tamarin-Prover?
- Practical Application

Goals for the Lab - Revisited

- Theory of Tamarin-Prover
  - mathematical foundation, in particular ☺
    - order-sorted term algebras
    - equational theories
    - operations: substitution, replacements, unification, matching, rewriting modulo equational theories
  - How is the language of Tamarin-Prover reflecting those notions? ☺
  - What are the limitations of Tamarin-Prover? 😐
- Practical Application

Goals for the Lab - Revisited

- Theory of Tamarin-Prover
  - mathematical foundation, in particular ☺
    - order-sorted term algebras
    - equational theories
    - operations: substitution, replacements, unification, matching, rewriting modulo equational theories
  - How is the language of Tamarin-Prover reflecting those notions? ☺
  - What are the limitations of Tamarin-Prover? 😐
- Practical Application
  - Implementing small toy examples to learn the language
  - Working on (parts of) the IPSec protocol

Goals for the Lab - Revisited

- Theory of Tamarin-Prover
  - mathematical foundation, in particular ☺
    - order-sorted term algebras
    - equational theories
    - operations: substitution, replacements, unification, matching, rewriting modulo equational theories
  - How is the language of Tamarin-Prover reflecting those notions? ☺
  - What are the limitations of Tamarin-Prover? 😐
- Practical Application
  - Implementing small toy examples to learn the language ☺
  - Working on (parts of) the IPSec protocol

Goals for the Lab - Revisited

- Theory of Tamarin-Prover
  - mathematical foundation, in particular ☺
    - order-sorted term algebras
    - equational theories
    - operations: substitution, replacements, unification, matching, rewriting modulo equational theories
  - How is the language of Tamarin-Prover reflecting those notions? ☺
  - What are the limitations of Tamarin-Prover? 😐
- Practical Application
  - Implementing small toy examples to learn the language ☺
  - Working on (parts of) the IPSec protocol 😐

References

References I

[bit17] bitkom, ed. *Spionage, Sabotage, Datendiebstahl: Deutscher Wirtschaft entsteht jährlich ein Schaden von 55 Milliarden Euro*. July 21, 2017. URL: `https://www.bitkom.org/Presse/Presseinformation/Spionage-Sabotage-Datendiebstahl-Deutscher-Wirtschaft-entsteht-jaehrlich-ein-Schaden-von-55-Milliarden-Euro.html` (visited on 06/30/2018).

[Blu17] Norbert Blum. *A Solution of the P versus NP Problem*. Aug. 11, 2017. arXiv: `1708.03486v2 [cs.CC]`.

[BR04] Mihir Bellare and Phillip Rogaway. *Code-Based Game-Playing Proofs and the Security of Triple Encryption*. Cryptology ePrint Archive, Report 2004/331. 2004. URL: `https://eprint.iacr.org/2004/331` (visited on 05/11/2018).

References II

[Hal05] Shai Halevi. *A plausible approach to computer-aided cryptographic proofs.* Cryptology ePrint Archive, Report 2005/181. 2005. URL: `https://eprint.iacr.org/2005/181` (visited on 05/11/2018).

[Mei13] Simon Meier. "Advancing automated security protocol verification". PhD thesis. ETH Zürich, 2013. DOI: `10.3929/ethz-a-009790675`.

[Moc18] Sinichi Mochizuki. *Inter-universal Teichmüller Theory I-IV*. June 28, 2018.

[Opf11] Gerhard Opfer. "An Analytic Approach to the Collatz 3n+1 Problem". In: *Hamburger Beiträge zur Angewandten Mathematik* 9 (2011).

[Sch12] Benedikt Schmidt. "Formal analysis of key exchange protocols and physical protocols". PhD thesis. ETH Zürich, 2012. DOI: `10.3929/ethz-a-009898924`.

Thank you for your attention!