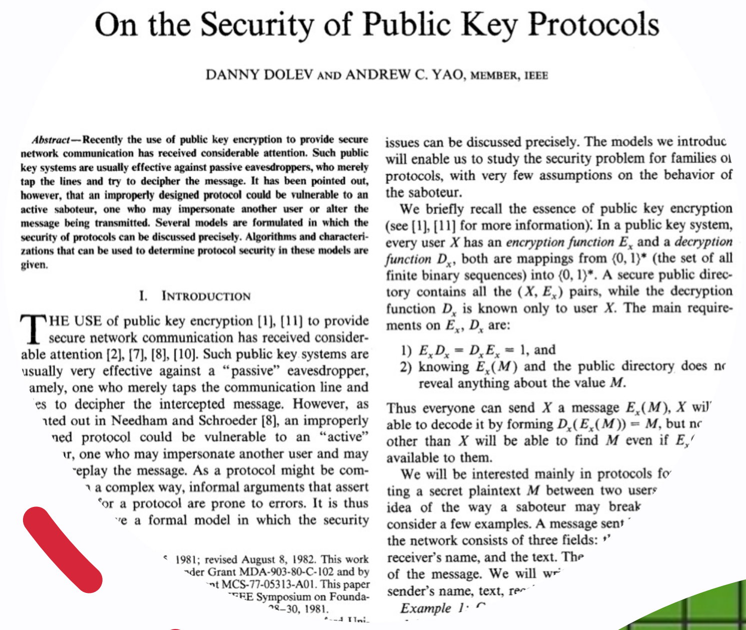
University of Udine

Formalizing Social Engineering attacks in the Symbolic Model

July 2023 – D’Ambrosi Denis

The Dolev Yao Model

Introduction



On the Security of Public Key Protocols

DANNY DOLEV AND ANDREW C. YAO, MEMBER, IEEE

Abstract—Recently the use of public key encryption to provide secure network communication has received considerable attention. Such public key systems are usually effective against passive eavesdroppers, who merely tap the lines and try to decipher the message. It has been pointed out, however, that an improperly designed protocol could be vulnerable to an active saboteur, one who may impersonate another user or alter the message being transmitted. Several models are formulated in which the security of protocols can be discussed precisely. Algorithms and characterizations that can be used to determine protocol security in these models are given.

I. INTRODUCTION

THE USE of public key encryption [1], [11] to provide secure network communication has received considerable attention [2], [7], [8], [10]. Such public key systems are usually very effective against a "passive" eavesdropper, amely, one who merely taps the communication line and es to decipher the intercepted message. However, as nted out in Needham and Schroeder [8], an improperly ned protocol could be vulnerable to an "active" r, one who may impersonate another user and may eplay the message. As a protocol might be com- a complex way, informal arguments that assert for a protocol are prone to errors. It is thus e a formal model in which the security

1981; revised August 8, 1982. This work der Grant MDA-903-80-C-102 and by t MCS-77-05313-A01. This paper EE Symposium on Founda- 8–30, 1981.

issues can be discussed precisely. The models we introduc will enable us to study the security problem for families o protocols, with very few assumptions on the behavior of the saboteur.

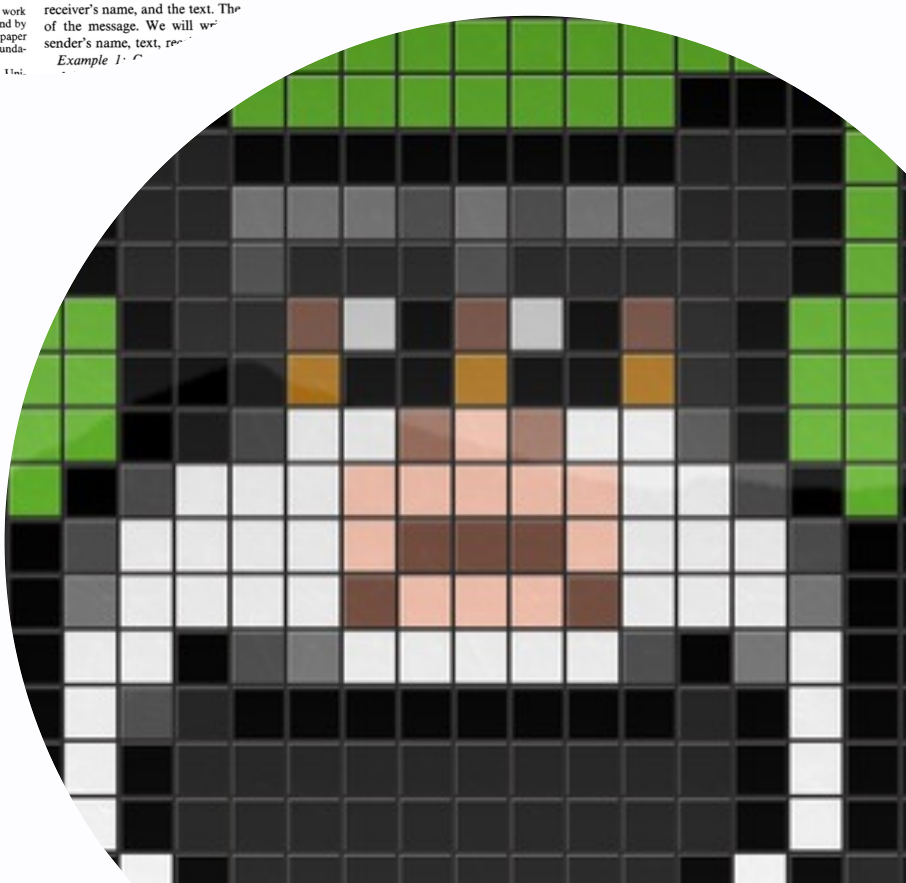
We briefly recall the essence of public key encryption (see [1], [11] for more information). In a public key system, every user X has an *encryption function* E_x and a *decryption function* D_x, both are mappings from $\{0,1\}^*$ (the set of all finite binary sequences) into $\{0,1\}^*$. A secure public directory contains all the (X, E_x) pairs, while the decryption function D_x is known only to user X. The main requirements on E_x, D_x are:

1) $E_x D_x = D_x E_x = 1$, and
2) knowing $E_x(M)$ and the public directory does n reveal anything about the value M.

Thus everyone can send X a message $E_x(M)$, X wil able to decode it by forming $D_x(E_x(M)) = M$, but n other than X will be able to find M even if E_x available to them.

We will be interested mainly in protocols fo ting a secret plaintext M between two users idea of the way a saboteur may break consider a few examples. A message sent the network consists of three fields: receiver's name, and the text. The of the message. We will wr sender's name, text, re

Example 1:



- Cryptographic protocols are complex to verify
- We need:
    1. A threat model
    2. A (possibly automated) tool to verify the protocol in said model

The Dolev Yao Model

- D. Dolev and A. Yao. On the security of public key protocols. IEEE Transactions on Information Theory, 1983.
- M. Barbosa, G. Barthe, K. Bhargavan, B. Blanchet, C. Cremers, K. Liao, and B. Parno. Sok: Computer-aided cryptography. Cryptology ePrint Archive, 2019.
- S. Even and O. Goldreich. On the security of multi-party ping-pong protocols. In 24th Annual Symposium on Foundations of Computer Science, 1983.
- N Durgin, P. Lincoln, and J. Mitchell. Multiset rewriting and the complexity of bounded security protocols. Journal of Computer Security, 2004.

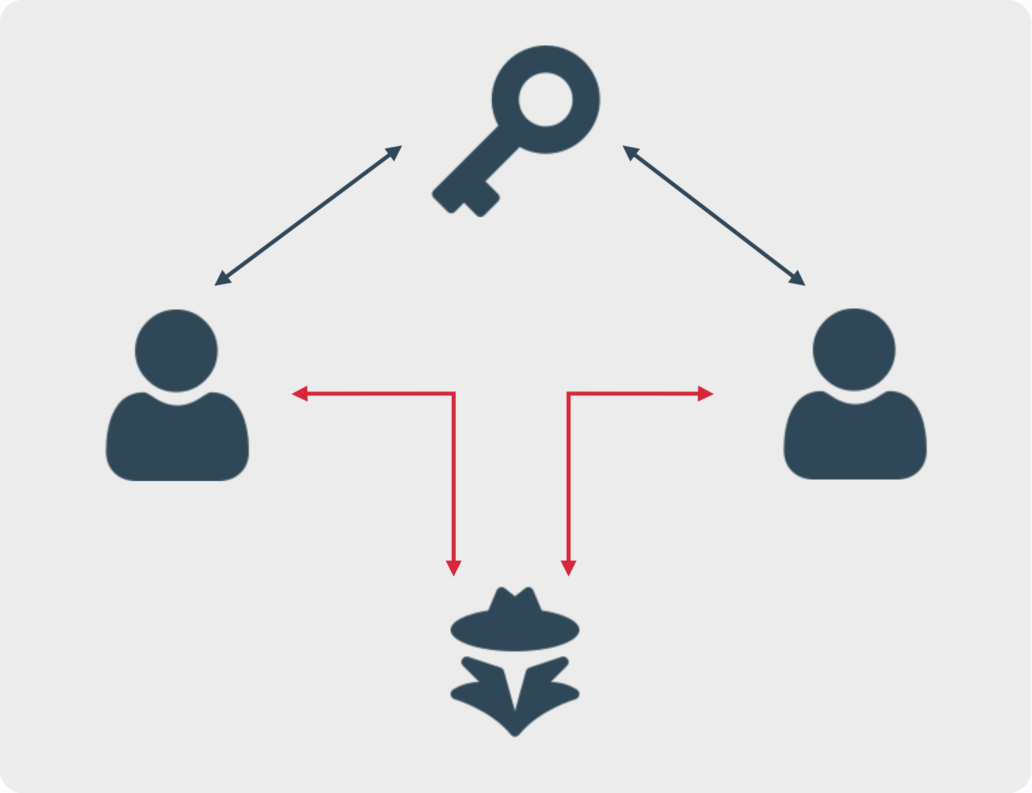
The Dolev Yao Model

Main features:

- Attacker-controlled network
- Term algebras for cryptography
- Perfect cryptography assumption

$$sdec_k\ senc_k = 1$$
$$adec_{pr}\ senc_{pub} = adec_{pub}\ aenc_{pr} = 1$$

Perfect cryptography identities



Attacker controlled network with trusted party

The Dolev Yao Model

What can be specified:

Trace properties

- Confidentiality
- Integrity
- Authentication
- etc ...

Observational equivalence properties:

- Privacy
- Indistinguishability

What can be verified:

Sources of infinity

1. Unbounded sessions
2. Unbounded nonces
3. Unbounded messages

Undecidability

Infinite state space

Limit nonces and messages ⟶ DEXP-complete

Limit sessions ⟶ NP-complete

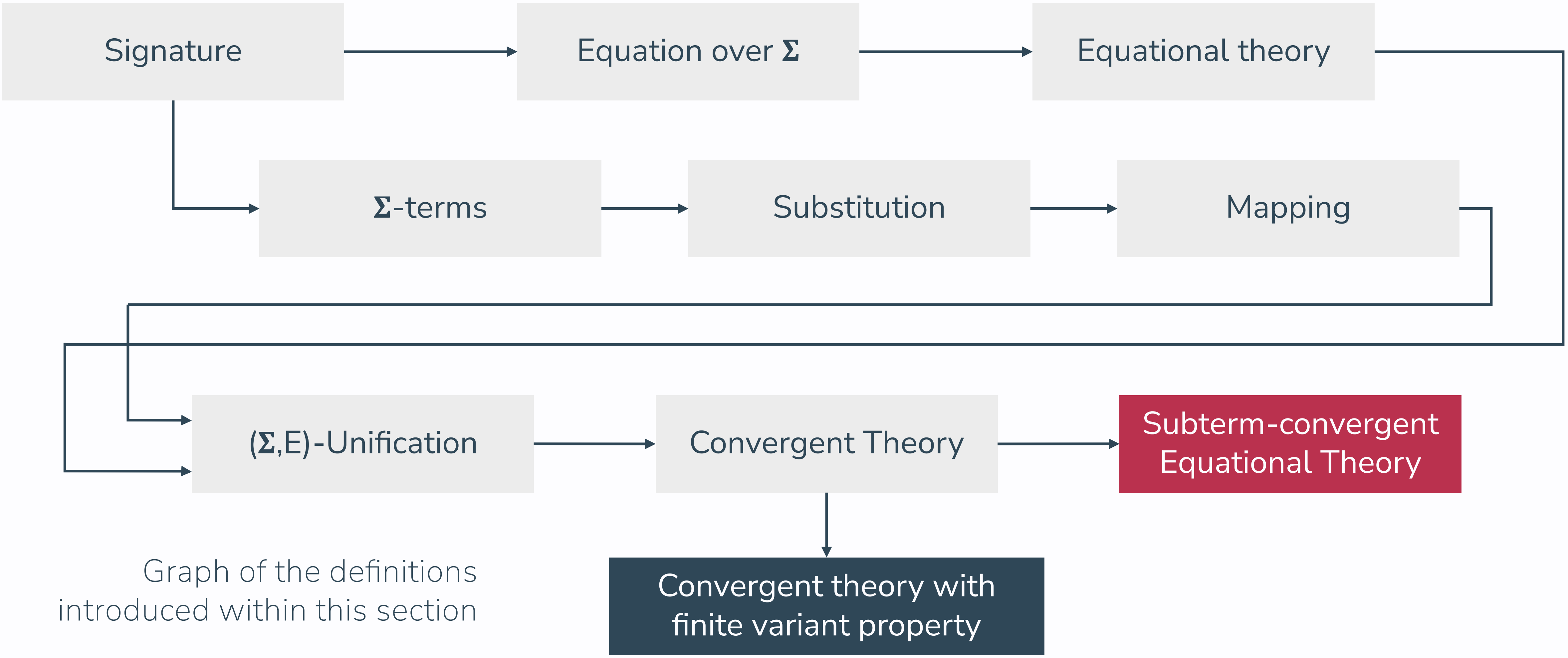
... or try to address undecidability without loosing generality ⟶ Tamarin, Proverlf

Tamarin prover overview

- S. Meier. Advancing automated security protocol verification. PhD thesis, ETH, 2013.
- B. Schmidt. Formal analysis of key exchange protocols and physical protocols. PhD thesis, ETH, 2012.
- The Tamarin Team. Tamarin-Prover Manual: Security Protocol Analysis in the Symbolic Model, 2023.
- J. Dreier, C. Dumenil, S. Kremer, and R. Sasse. Beyond subterm-convergent equational theories in automated verification of stateful protocols. Principles of Security and Trust, 2017

Tamarin Prover overview

Term-algebra



Graph of the definitions introduced within this section

Tamarin Prover overview

Equational theories

Intuitive definitions of the equational theories accepted by Tamarin

Convergent theory with finite variant property

A set of equations Σ that ensure that any term t belonging to the relative term algebra constructed on Σ and its function symbols has a finite amount of possible normal terms (modulo Σ)

Subterm Convergent Equational Theory

A set of equations in the form

$$lhs = rhs$$

such that

1. Each term in lhs and rhs has a normal form
2. If a term t can be rewritten as t_1 and t_2, then it is possible to reach a fourth term t' from t_1 and t_2 in a finite amount of steps
3. rhs is either ground and in normal form or a proper subterm of lhs

Tamarin Prover overview

Protocols as sets of multiset rewriting rules

State evolution in Tamarin: application of rewriting rules to multisets of facts

Facts

- Intuitively, they represent **true predicates**
- Fixed arity
- Made up of terms

Linear facts

- Represent ephemeral information
- Consumed just once

Persistent facts

- Represent enduring knowledge

Multiset rewriting rules

- Starting state: $\Gamma_t = \{\{F_0, \ldots, F_n\}\}$
- Current trace: $tr_t = \langle a_0, \ldots, a_{t-1} \rangle$
- Rewriting rule: $RR = L \xrightarrow{A} R$
- Ground istance of RR: $rr = l \xrightarrow{a} r,\ l \subseteq^{\#} \Gamma_t$

→ New state: $\Gamma_{t+1} = \Gamma_t \setminus^{\#} lin(l) \cup^{\#} r$

→ New trace: $tr_{t+1} = \langle a_0, \ldots, a_{t-1}, a \rangle$

Additional definitions

- Observable trace: $tr_{obs,t} = \langle A_i | A_i \in tr_t \wedge A_i \neq \emptyset^{\#} \rangle$
- Labelled rules: $R_i = (Name, RR)$

Tamarin Prover overview

Protocols as sets of multiset rewriting rules

Definition of protocol rules

A protocol rule is a multiset rewriting rule $l \xrightarrow{a} r$ such that

1. l, a, r do not contain fresh names
2. l does not contain $\mathbf{K}$ and $\mathbf{Out}$ facts
3. r does not contain $\mathbf{Fr}$ and $\mathbf{In}$ facts
4. The argument of any $\mathbf{Fr}$ fact belongs to the set of fresh terms
5. r does not contain the function symbol $*$
6. $l \xrightarrow{a} r$ satisfies:
   - $vars(r) \subseteq vars(l) \cup V_{pub}$
   - l only contains irreducible function symbols from the given signature or it is an instance of a rule that satisfies both conditions

Tamarin Prover overview

Protocols as sets of multiset rewriting rules

State evolution in Tamarin: application of rewriting rules to multisets of facts

Multiset rewriting rules

- Starting state: $\Gamma_t = \{\{F_0, \dots, F_n\}\}$
- Current trace: $tr_t = \langle a_0, \dots, a_{t-1} \rangle$
- Rewriting rule: $RR = L \xrightarrow{A} R$
- Ground istance of RR: $rr = l \xrightarrow{a} r,\ l \subseteq^{\#} \Gamma_t$

→ New state: $\Gamma_{t+1} = \Gamma_t \setminus^{\#}\ lin(l)\ \cup^{\#}\ r$
→ New trace: $tr_{t+1} = \langle a_0, \dots, a_{t-1}, a \rangle$

Additional definitions

- Observable trace: $tr_{obs,t} = \langle A_i | A_i \in tr_t \wedge A_i \neq \emptyset^{\#} \rangle$
- Labelled rules: $R_i = (Name_i, RR_i)$

Example

$$P = \{\ (N_1,\ \emptyset^{\#} -[\ \{\{Init(0)\}\}\ ] \rightarrow \{\{A(0)\}\}),$$
$$(N_2, \{\{A(x)\}\} - [\ \emptyset^{\#}\ ] \rightarrow \{\{B(x)\}\}),$$
$$(N_3, \{\{B(x)\}\} - [\ \{\{Concl(x)\}\}\ ] \rightarrow \emptyset^{\#})\ \}$$

Apply N_1, N_2, N_1, N_3 to an initial empty state

$$\Gamma_0 = \emptyset^{\#}$$

$$tr_0 = \langle\ \rangle$$
$$tr_{obs,0} = \langle\ \rangle$$

Tamarin Prover overview

Protocols as sets of multiset rewriting rules

State evolution in Tamarin: application of rewriting rules to multisets of facts

Multiset rewriting rules

- Starting state: $\Gamma_t = \{\{F_0, \dots, F_n\}\}$
- Current trace: $tr_t = \langle a_0, \dots, a_{t-1} \rangle$
- Rewriting rule: $RR = L \xrightarrow{A} R$
- Ground istance of RR: $rr = l \xrightarrow{a} r,\ l \subseteq^{\#} \Gamma_t$

→ New state: $\Gamma_{t+1} = \Gamma_t \setminus^{\#} \ lin(l)\ \cup^{\#}\ r$

→ New trace: $tr_{t+1} = \langle a_0, \dots, a_{t-1}, a \rangle$

Additional definitions

- Observable trace: $tr_{obs,t} = \langle A_i | A_i \in tr_t \wedge A_i \neq \emptyset^{\#} \rangle$
- Labelled rules: $R_i = (Name_i, RR_i)$

Example

$$P = \{ \left(N_1, \emptyset^{\#} - [\ \{\{Init(0)\}\}\ ] \rightarrow \{\{A(0)\}\}\right),$$
$$\left(N_2, \{\{A(x)\}\} - [\ \emptyset^{\#}\ ] \rightarrow \{\{B(x)\}\}\right),$$
$$\left(N_3, \{\{B(x)\}\} - [\ \{\{Concl(x)\}\}\ ] \rightarrow \emptyset^{\#}\right) \}$$

Apply N_1, N_2, N_1, N_3 to an initial empty state

$$\Gamma_0 = \emptyset^{\#}$$
$$\Gamma_1 = \emptyset^{\#}\ \setminus^{\#}\ \emptyset^{\#} \cup^{\#}\ \{\{A(0)\}\} = \{\{A(0)\}\}$$
$$tr_1 = \langle Init(0) \rangle$$
$$tr_{obs,1} = \langle Init(0) \rangle$$

Tamarin Prover overview

Protocols as sets of multiset rewriting rules

State evolution in Tamarin: application of rewriting rules to multisets of facts

Multiset rewriting rules

- Starting state: $\Gamma_t = \{\{F_0, \ldots, F_n\}\}$
- Current trace: $tr_t = \langle a_0, \ldots, a_{t-1} \rangle$
- Rewriting rule: $RR = L \xrightarrow{A} R$
- Ground istance of RR: $rr = l \xrightarrow{a} r,\ l \subseteq^{\#} \Gamma_t$

→ New state: $\Gamma_{t+1} = \Gamma_t \setminus^{\#}\ lin(l)\ \cup^{\#}\ r$

→ New trace: $tr_{t+1} = \langle a_0, \ldots, a_{t-1}, a \rangle$

Additional definitions

- Observable trace: $tr_{obs,t} = \langle A_i | A_i \in tr_t \wedge A_i \neq \emptyset^{\#} \rangle$
- Labelled rules: $R_i = (Name_i, RR_i)$

Example

$$P = \{\ (N_1, \emptyset^{\#} - [\ \{\{Init(0)\}\}\ ] \rightarrow \{\{A(0)\}\}),$$
$$(N_2, \{\{A(x)\}\} - [\ \emptyset^{\#}\ ] \rightarrow \{\{B(x)\}\}),$$
$$(N_3, \{\{B(x)\}\} - [\ \{\{Concl(x)\}\}\ ] \rightarrow \emptyset^{\#})\ \}$$

Apply N_1, N_2, N_1, N_3 to an initial empty state

$$\Gamma_1 = \{\{A(0)\}\}$$
$$\Gamma_2 = \{\{A(0)\}\} \setminus^{\#} \{\{A(0)\}\} \cup^{\#}\ \{\{B(0)\}\} = \{\{B(0)\}\}$$
$$tr_2 = \langle Init(0), \emptyset^{\#} \rangle$$
$$tr_{obs,2} = \langle Init(0) \rangle$$

Tamarin Prover overview

Protocols as sets of multiset rewriting rules

State evolution in Tamarin: application of rewriting rules to multisets of facts

Multiset rewriting rules

- Starting state: $\Gamma_t = \{\{F_0, \dots, F_n\}\}$
- Current trace: $tr_t = \langle a_0, \dots, a_{t-1} \rangle$
- Rewriting rule: $RR = L \xrightarrow{A} R$
- Ground istance of RR: $rr = l \xrightarrow{a} r,\ l \subseteq^{\#} \Gamma_t$

→ New state: $\Gamma_{t+1} = \Gamma_t \setminus^{\#} \ lin(l) \ \cup^{\#} \ r$
→ New trace: $tr_{t+1} = \langle a_0, \dots, a_{t-1}, a \rangle$

Additional definitions

- Observable trace: $tr_{obs,t} = \langle A_i | A_i \in tr_t \wedge A_i \neq \emptyset^{\#} \rangle$
- Labelled rules: $R_i = (Name_i, RR_i)$

Example

$$P = \{ \left(N_1, \emptyset^{\#} -[\{\{Init(0)\}\}] \rightarrow \{\{A(0)\}\}\right),$$
$$\left(N_2, \{\{A(x)\}\} - [\emptyset^{\#}] \rightarrow \{\{B(x)\}\}\right),$$
$$\left(N_3, \{\{B(x)\}\} - [\{\{Concl(x)\}\}] \rightarrow \emptyset^{\#}\right) \}$$

Apply N_1, N_2, N_1, N_3 to an initial empty state

$$\Gamma_2 = \{\{B(0)\}\}$$
$$\Gamma_3 = \{\{B(0)\}\} \setminus^{\#} \emptyset^{\#} \ \cup^{\#} \ \{\{A(0)\}\} = \{\{A(0), B(0)\}\}$$
$$tr_3 = \langle Init(0), \emptyset^{\#}, Init(0) \rangle$$
$$tr_{obs,3} = \langle Init(0), Init(0) \rangle$$

Tamarin Prover overview

Protocols as sets of multiset rewriting rules

State evolution in Tamarin: application of rewriting rules to multisets of facts

Multiset rewriting rules

- Starting state: $\Gamma_t = \{\{F_0, \dots, F_n\}\}$
- Current trace: $tr_t = \langle a_0, \dots, a_{t-1} \rangle$
- Rewriting rule: $RR = L \xrightarrow{A} R$
- Ground istance of RR: $rr = l \xrightarrow{a} r,\ l \subseteq^{\#} \Gamma_t$

→ New state: $\Gamma_{t+1} = \Gamma_t \setminus^{\#} \ lin(l) \ \cup^{\#} \ r$
→ New trace: $tr_{t+1} = \langle a_0, \dots, a_{t-1}, a \rangle$

Additional definitions

- Observable trace: $tr_{obs,t} = \langle A_i | A_i \in tr_t \wedge A_i \neq \emptyset^{\#} \rangle$
- Labelled rules: $R_i = (Name_i, RR_i)$

Example

$$P = \{ (N_1, \emptyset^{\#} - [\ \{\{Init(0)\}\}\ ] \rightarrow \{\{A(0)\}\}),$$
$$(N_2, \{\{A(x)\}\} - [\ \emptyset^{\#}\ ] \rightarrow \{\{B(x)\}\}),$$
$$(N_3, \{\{B(x)\}\} - [\ \{\{Concl(x)\}\}\ ] \rightarrow \emptyset^{\#})\ \}$$

Apply N_1, N_2, N_1, N_3 to an initial empty state

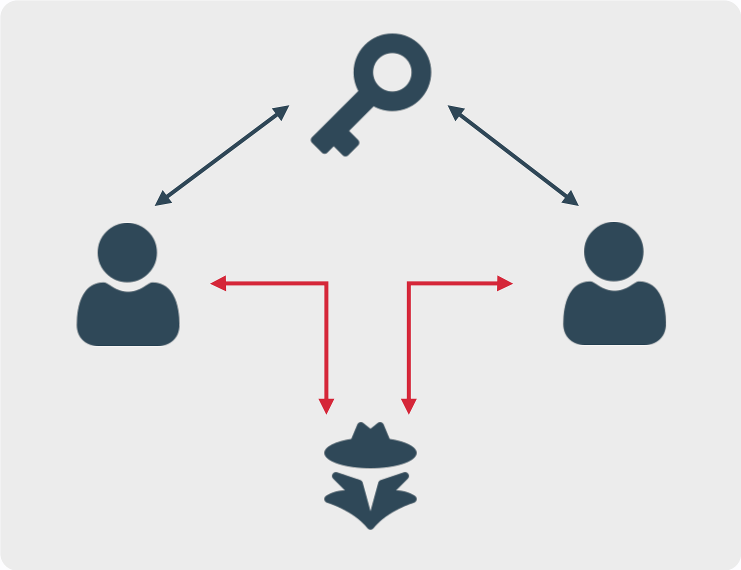
$$\Gamma_3 = \{\{A(0), B(0)\}\}$$
$$\Gamma_4 = \{\{A(0), B(0)\}\} \setminus^{\#} \{\{B(0)\}\} \ \cup^{\#} \emptyset^{\#} = \{\{A(0)\}\}$$
$$tr_4 = \langle Init(0),\ \emptyset^{\#}, Init(0), Concl(0) \rangle$$
$$tr_{obs,4} = \langle Init(0), Init(0), Concl(0) \rangle$$

Tamarin Prover overview

Protocols as sets of multiset rewriting rules

Dolev-Yao rules

1. Term generation: $[\,] \xrightarrow{\mathrm{Fresh}(\sim msg)} \rightarrow [\,\mathrm{Fr}(\sim msg)\,]$
2. Term generation (attacker): $[\,\mathrm{Fr}(\sim msg)\,] \rightarrow [\,\mathrm{K}(\sim msg)\,]$
3. Sending to the network: $[\,\mathrm{Out}(msg)\,] \rightarrow [\,\mathrm{K}(msg)\,]$
4. Receiving from the network: $[\,\mathrm{K}(msg)\,] \xrightarrow{\mathrm{K}(msg)} [\,\mathrm{In}(msg)\,]$
5. Knowledge of public names: $[\,] \rightarrow [\,\mathrm{K}(\$x)\,]$
6. Use of non-private functions: $[\,\mathrm{K}(x_1, \dots, x_n)\,] \rightarrow [\,\mathrm{K}(f(x_1, \dots, x_n))\,]$



Attacker controlled network with trusted party

Tamarin Prover overview

Protocols as sets of multiset rewriting rules

Confidential channel rules

$$[\ \mathrm{Out}_{conf}(msg)\ ] \rightarrow [\ \mathrm{In}_{conf}(x)\ ]$$
$$[\ \mathrm{K}(msg)\ ] \rightarrow [\ \mathrm{In}_{conf}(x)\ ]$$

Authenticated channel rules

$$[\ \mathrm{Out}_{auth}(msg)\ ] \xrightarrow{\mathrm{K}(msg)} [\ \mathrm{In}_{auth}(msg), \mathrm{K}(msg)\ ]$$

Confidential channel (with id) rules

$$\left[\ \mathrm{Out}_{\langle conf, channel\rangle}(msg, channel)\ \right] \rightarrow \left[\ \mathrm{In}_{\langle conf, channel\rangle}(msg, channel)\ \right]$$
$$[\ \mathrm{K}(msg), \mathrm{K}(channel)\ ] \rightarrow \left[\mathrm{Out}_{\langle conf, channel\rangle}(msg, channel)\ \right]$$

Tamarin Prover overview

Security properties

Properties can be formalized through guarded fragments of first order logic

Formulas can be specified through the following constructs:

- False: $\perp$
- Logical operators: $\neg,\ \vee,\ \wedge,\ \Longrightarrow$
- Quantifiers and variables: $\forall, \exists, a, b, c, \ldots$
- Term equalities: $t_1 \approx t_2$
- Time-point ordering and equalities: $i < j,\ i \approx j$
- Action facts at time points: $F\ @\ i$

Example of definition of Secrecy

$$\forall m, t_1 : Secret(m)@t_1 \Longrightarrow \neg\exists t_2 : K(m)@t_2$$

Tamarin Prover overview

Security properties

Traces of a protocol

Given a set of labelled rewriting rules P, we define the set of possible traces generated by P as

$$traces(P) = \{\langle A_1, \dots, A_n \rangle \mid \exists\, S_1, \dots, S_n \,.\, \emptyset^{\#} \xrightarrow{A_1} S_1 \xrightarrow{A_2} \dots \xrightarrow{A_n} S_n \text{ and no ground istance of Fresh( ) is used twice}\}$$

where A_i is the action fact of the i^{th} rule applied

Correctness

A protocol P is said correct with regards to a formula (security property) ϕ if:

$$P \vDash \phi \iff traces(P) \subseteq traces(\phi)$$

All traces belonging to $traces(P) \backslash traces(\phi)$ represent valid attacks

Traces of a formula

Given a property formula ϕ, $traces(\phi)$ is the set of traces that satisfy it

Tamarin Prover overview

Observational Equivalences

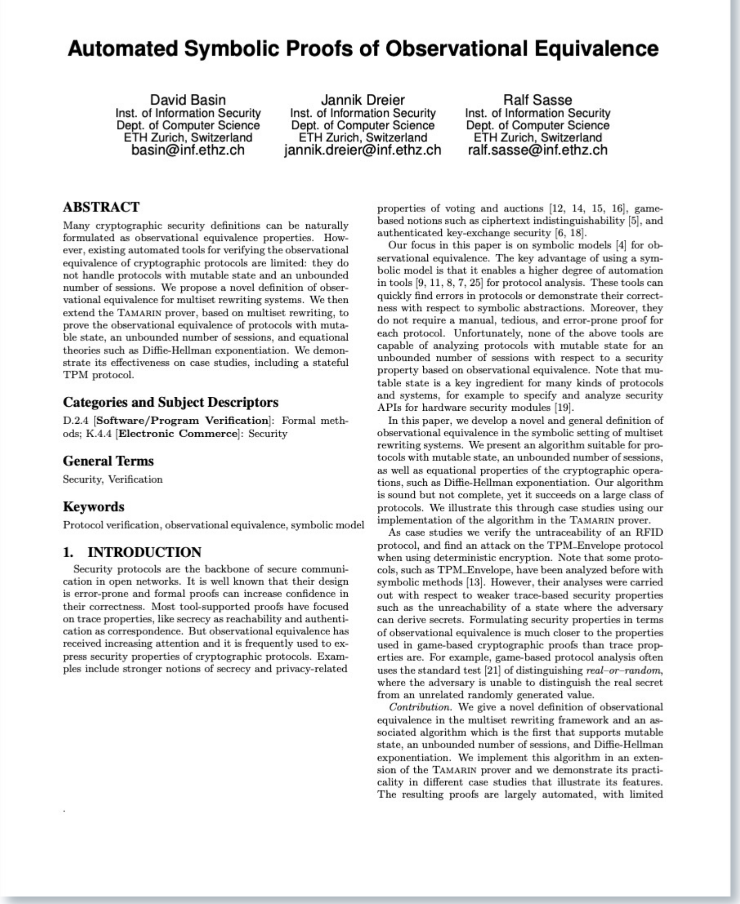
Trace equivalence

Two different protocols P_1, P_2 are **trace equivalent** if an only if for each trace of P_1 exists a trace of P_2 so that the messages exchanged during the two executions are indistinguishable

Diff equivalence

Two protocols $P_1,\ P_2$ are **diff-equivalent** if and only if they have the same structure and differ only by the message exchanged

Introductory paper for the Observational Equivalence extension in Tamarin


Automated Symbolic Proofs of Observational Equivalence

David Basin
Inst. of Information Security
Dept. of Computer Science
ETH Zurich, Switzerland
basin@inf.ethz.ch

Jannik Dreier
Inst. of Information Security
Dept. of Computer Science
ETH Zurich, Switzerland
jannik.dreier@inf.ethz.ch

Ralf Sasse
Inst. of Information Security
Dept. of Computer Science
ETH Zurich, Switzerland
ralf.sasse@inf.ethz.ch

ABSTRACT

Many cryptographic security definitions can be naturally formulated as observational equivalence properties. However, existing automated tools for verifying the observational equivalence of cryptographic protocols are limited: they do not handle protocols with mutable state and an unbounded number of sessions. We propose a novel definition of observational equivalence for multiset rewriting systems. We then extend the TAMARIN prover, based on multiset rewriting, to prove the observational equivalence of protocols with mutable state, an unbounded number of sessions, and equational theories such as Diffie-Hellman exponentiation. We demonstrate its effectiveness on case studies, including a stateful TPM protocol.

Categories and Subject Descriptors

D.2.4 [**Software/Program Verification**]: Formal methods; K.4.4 [**Electronic Commerce**]: Security

General Terms

Security, Verification

Keywords

Protocol verification, observational equivalence, symbolic model

1. INTRODUCTION

Security protocols are the backbone of secure communication in open networks. It is well known that their design is error-prone and formal proofs can increase confidence in their correctness. Most tool-supported proofs have focused on trace properties, like secrecy as reachability and authentication as correspondence. But observational equivalence has received increasing attention and it is frequently used to express security properties of cryptographic protocols. Examples include stronger notions of secrecy and privacy-related properties of voting and auctions [12, 14, 15, 16], game-based notions such as ciphertext indistinguishability [5], and authenticated key-exchange security [6, 18].

Our focus in this paper is on symbolic models [4] for observational equivalence. The key advantage of using a symbolic model is that it enables a higher degree of automation in tools [9, 11, 8, 7, 25] for protocol analysis. These tools can quickly find errors in protocols or demonstrate their correctness with respect to symbolic abstractions. Moreover, they do not require a manual, tedious, and error-prone proof for each protocol. Unfortunately, none of the above tools are capable of analyzing protocols with mutable state for an unbounded number of sessions with respect to a security property based on observational equivalence. Note that mutable state is a key ingredient for many kinds of protocols and systems, for example to specify and analyze security APIs for hardware security modules [19].

In this paper, we develop a novel and general definition of observational equivalence in the symbolic setting of multiset rewriting systems. We present an algorithm suitable for protocols with mutable state, an unbounded number of sessions, as well as equational properties of the cryptographic operations, such as Diffie-Hellman exponentiation. Our algorithm is sound but not complete, yet it succeeds on a large class of protocols. We illustrate this through case studies using our implementation of the algorithm in the TAMARIN prover.

As case studies we verify the untraceability of an RFID protocol, and find an attack on the TPM_Envelope protocol when using deterministic encryption. Note that some protocols, such as TPM_Envelope, have been analyzed before with symbolic methods [13]. However, their analyses were carried out with respect to weaker trace-based security properties such as the unreachability of a state where the adversary can derive secrets. Formulating security properties in terms of observational equivalence is much closer to the properties used in game-based cryptographic proofs than trace properties are. For example, game-based protocol analysis often uses the standard test [21] of distinguishing *real–or–random*, where the adversary is unable to distinguish the real secret from an unrelated randomly generated value.

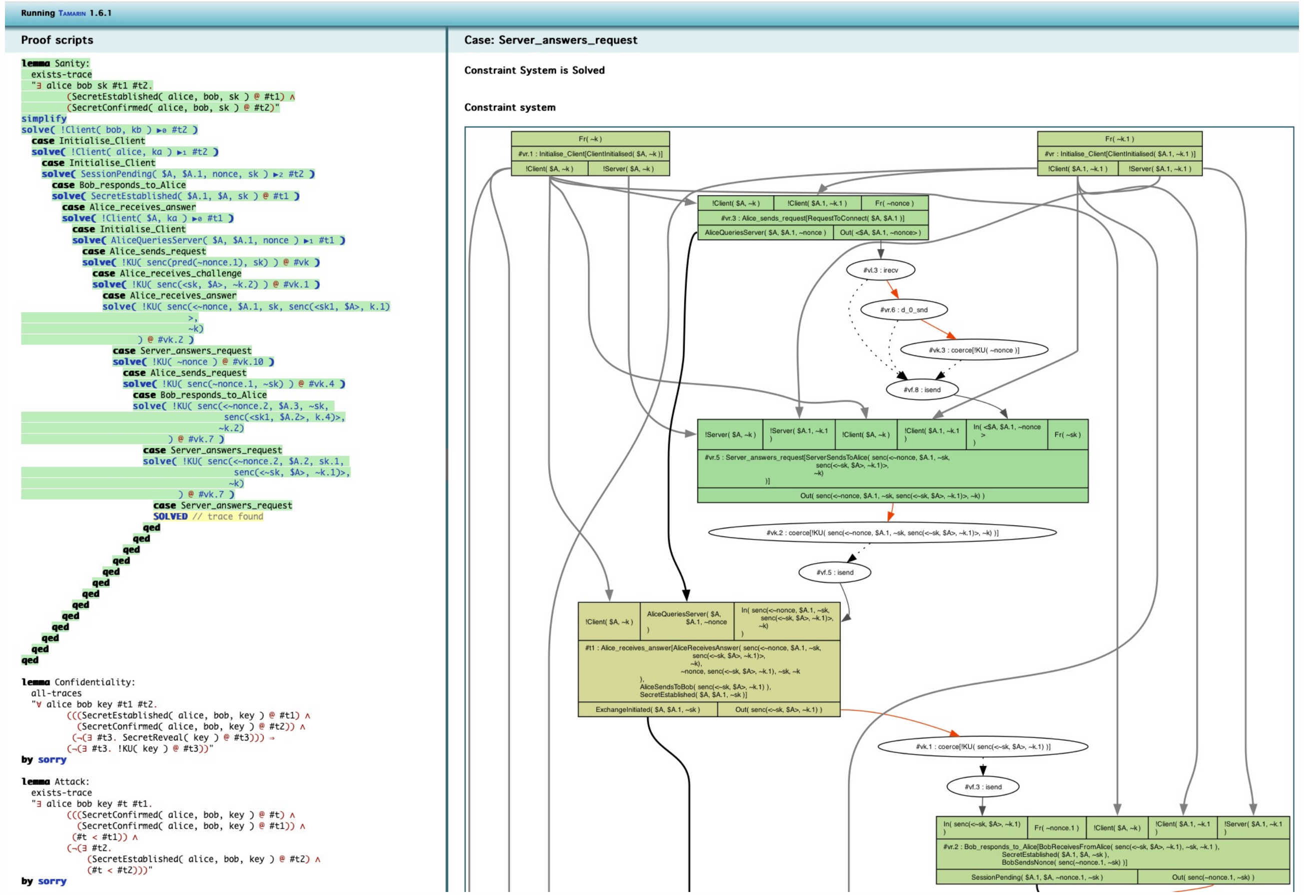
Contribution. We give a novel definition of observational equivalence in the multiset rewriting framework and an associated algorithm which is the first that supports mutable state, an unbounded number of sessions, and Diffie-Hellman exponentiation. We implement this algorithm in an extension of the TAMARIN prover and we demonstrate its practicality in different case studies that illustrate its features. The resulting proofs are largely automated, with limited

Tamarin Prover overview

Aiding termination

Tools provided to aid termination:

- Source lemmas
- Oracles
- Interactive mode
- Restrictions
- Re-use lemmas



Tamarin's interactive mode

Formalizing S-E attacks

- D. Basin, S. Radomirovic, and L. Schmid. Modeling human errors in security protocols. In 2016 IEEE 29th Computer Security Foundations Symposium (CSF, 2016.
- G. Lowe. A hierarchy of authentication specifications. In Proceedings of the 10th IEEE Workshop on Computer Security Foundations, CSFW '97, 1997.
- D. Basin, S. Radomirovic, and M. Schlapfer. A complete characterization of secure human-server communication. 2015 IEEE 28th Computer Security Foundations Symposium, 2015.

Formalizing S-E Attacks

Introduction

Social Engineering is a prevalent threat, with 90% of data breaches having social engineering components and 62% of businesses experiencing attacks

Gitnux report of 2023

Phishing	Baiting	Pretexting	Summary
Ransomware	Impersonation on Help Desk	Diversion Theft	Dumpster Diving
Shoulder Surfing	Quid Pro Quo	Pop-up Windows	Robocalls
Reverse Social Engineering	Online Social Engineering	Phone Social Engineering	Stealing important documents
Fake Software	Pharming	SMSishing	Whitelisting flow

Types of Social Engineering Attacks

Formalizing S-E Attacks

Including human knowledge

We assume that a human can:

- Send and receive messages on specified channels
- Concatenate messages
- Split concatenated messages

And cannot (unaided by a computer):

- ✗ Encrypt messages
- ✗ Decrypt messages

We also define an infinite human knowledge database through persistent facts

$$!\,\mathrm{HK}(H, \langle t, x\rangle)$$

Where

- H is the human agent
- t is the tag for the data stored
- x is the information known by H

Along with predicate $\vdash_H$:

$$\begin{aligned}\langle t, m\rangle \vdash_H \langle t', m'\rangle \Leftrightarrow \exists i, k\colon 1 \le i \le k \wedge \\ t = \langle t_1, \dots, t_k\rangle \wedge \\ m = \langle m_1, \dots, m_k\rangle \wedge \\ t' = t_i \wedge m' = m_i\end{aligned}$$

Formalizing S-E Attacks

Formalizing human errors

Turning Alice&Bob notation into role scripts for various actors

Alice&Bob notation

0	$S\colon \mathrm{knows}(R)$
0	$R\colon \mathrm{knows}(S, m_2)$
1	$S \xrightarrow{ins} R\colon \mathrm{fresh}(m_1).\, m_1$
2	$R \xrightarrow{sec} S\colon m_2$

Rewriting rules for the role script

$$[\,] \xrightarrow{\mathrm{Start}(S,R)} [\, AgSt(S, 0, R) \,]$$

$$[\, \mathrm{AgSt}(S, 0, R), \mathrm{Fr}(\sim m_1) \,] \xrightarrow{\mathrm{Fresh}(S,\sim m_1)} [\, \mathrm{AgSt}(S, 1, \langle R, \sim m_1 \rangle) \,]$$

$$[\, \mathrm{AgSt}(S, 1, \langle R, m_1 \rangle) \,] \xrightarrow{\mathrm{Send}(S,ins,R,m_1)} [\, \mathrm{AgSt}(S, 2, \langle R, m_1 \rangle), \mathrm{Out}_{ins}(\langle S, R, m_1 \rangle) \,]$$

$$[\, \mathrm{AgSt}(S, 2, \langle R, m_1 \rangle), \mathrm{In}_{sec}(\langle R, S, m_2 \rangle) \,] \xrightarrow{\mathrm{Receive}(S,sec,R,m_2)} [\, \mathrm{AgSt}(S, 3, \langle R, m_1, m_2 \rangle) \,]$$

Role script of role S

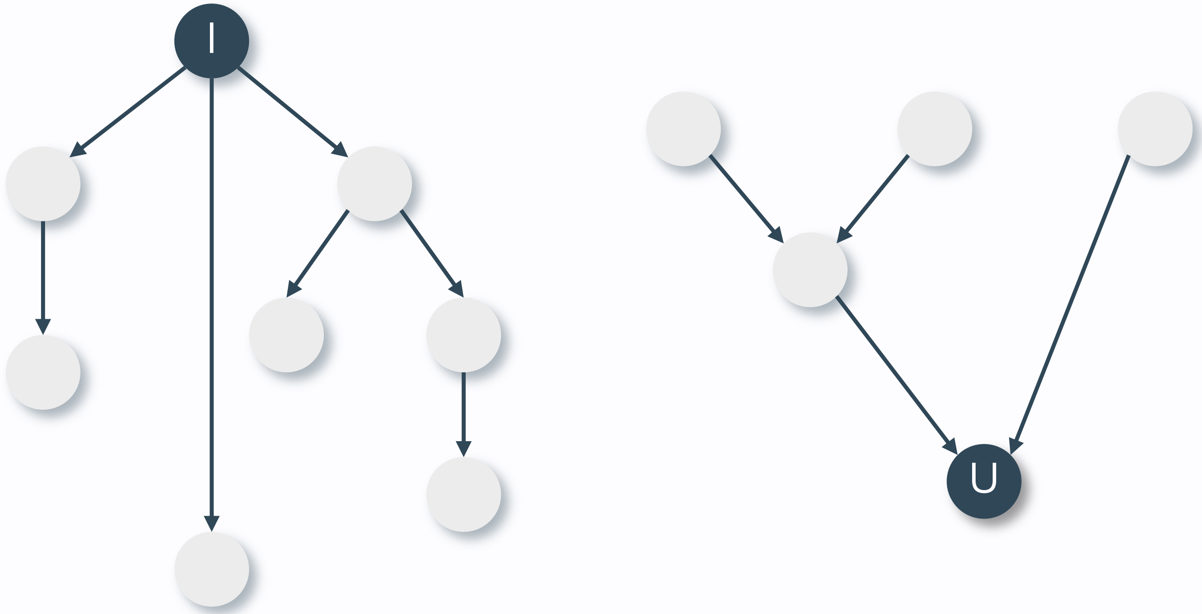
$$\langle \mathrm{Start}(S, R), \mathrm{Fresh}(S, m_1), \mathrm{Send}(S, ins, R, m_1), \mathrm{Receive}(S, sec, R, m_2) \rangle$$

Formalizing S-E Attacks

Formalizing human errors

Human error

A human error is any deviation of a human from his role script



Approaches to introducing fallible humans:

- On the left, the skilled human approach
- On the right, the rule-based human approach

Formalizing S-E Attacks

Rule-based human approach

First we need to introduce the formalization of an untrained human…

(This agent must be able to follow any possible behavior)

1. $[\ \mathrm{Fr}(\sim x)\ ] \xrightarrow{\mathrm{Fresh}(H,\langle t,\sim x\rangle)} [\ !\,\mathrm{HK}(H,\langle t,\sim x\rangle)\ ]$
2. $[\ !\,\mathrm{HK}(H,\langle t,x\rangle)\ ] \xrightarrow{\mathrm{Send}(H,ins,P,\langle t,x\rangle)} [\ \mathrm{Out}_{ins}(\langle H,P,\langle t,x\rangle\rangle)\ ]$
3. $[\ \mathrm{In}_{ins}(\langle P,H,\langle t,x\rangle\rangle)\ ] \xrightarrow{\mathrm{Receive}(H,ins,P,\langle t,x\rangle)} [\ !\,\mathrm{HK}(H,\langle t,x\rangle)\ ]$

Rewriting rules for the untrained human

Formalizing S-E Attacks

Rule-based human approach

Untrained human rules

1. $[\, \mathrm{Fr}(\sim x) \,] \xrightarrow{\mathrm{Fresh}(H,\langle t,\sim x\rangle)} [\, !\,\mathrm{HK}(H,\langle t,\sim x\rangle) \,]$
2. $[\, !\,\mathrm{HK}(H,\langle t,x\rangle) \,] \xrightarrow{\mathrm{Send}(H,ins,P,\langle t,x\rangle)} [\, \mathrm{Out}_{ins}(\langle H,P,\langle t,x\rangle\rangle) \,]$
3. $[\, \mathrm{In}_{ins}(\langle P,H,\langle t,x\rangle\rangle) \,] \xrightarrow{\mathrm{Receive}(H,ins,P,\langle t,x\rangle)} [\, !\,\mathrm{HK}(H,\langle t,x\rangle) \,]$

Avoiding leakage of private information

$$\mathrm{NoTell}(H,tag) \coloneqq \forall \mathrm{Send}(H,l,P,\langle t,m\rangle) \in tr \in traces(P), t',m' : \langle t,m\rangle \vdash_H \langle t',m'\rangle \implies t' \neq tag$$

Formalizing S-E Attacks

Rule-based human approach

Untrained human rules

1. $[\,\mathrm{Fr}(\sim x)\,] \xrightarrow{\mathrm{Fresh}(H,\langle t,\sim x\rangle)} [\,!\,\mathrm{HK}(H,\langle t,\sim x\rangle)\,]$
2. $[\,!\,\mathrm{HK}(H,\langle t,x\rangle)\,] \xrightarrow{\mathrm{Send}(H,ins,P,\langle t,x\rangle)} [\,\mathrm{Out}_{ins}(\langle H,P,\langle t,x\rangle\rangle)\,]$
3. $[\,\mathrm{In}_{ins}(\langle P,H,\langle t,x\rangle\rangle)\,] \xrightarrow{\mathrm{Receive}(H,ins,P,\langle t,x\rangle)} [\,!\,\mathrm{HK}(H,\langle t,x\rangle)\,]$

Safe comunications of credentials

$$\mathrm{NoTellExcept}(H,tag,rtag) \coloneqq \forall \mathrm{Send}(H,l,P,\langle t,m\rangle) \in tr \in traces(P), m', R:$$
$$\mathrm{InitK}(H,\langle rtag,R\rangle) \in tr \wedge \langle t,m\rangle \vdash_H \langle tag,m'\rangle$$
$$\Longrightarrow P = R \wedge (l = sec \vee l = conf)$$

Formalizing S-E Attacks

Rule-based human approach

Untrained human rules

1. $[\,\mathrm{Fr}(\sim x)\,] \xrightarrow{\mathrm{Fresh}(H,\langle t,\sim x\rangle)} [\,!\,\mathrm{HK}(H,\langle t,\sim x\rangle)\,]$
2. $[\,!\,\mathrm{HK}(H,\langle t,x\rangle)\,] \xrightarrow{\mathrm{Send}(H,ins,P,\langle t,x\rangle)} [\,\mathrm{Out}_{ins}(\langle H,P,\langle t,x\rangle\rangle)\,]$
3. $[\,\mathrm{In}_{ins}(\langle P,H,\langle t,x\rangle\rangle)\,] \xrightarrow{\mathrm{Receive}(H,ins,P,\langle t,x\rangle)} [\,!\,\mathrm{HK}(H,\langle t,x\rangle)\,]$

Protecting information integrity

$$\mathrm{NoGet}(H,tag) \coloneqq \forall \mathrm{Receive}(H,l,P,\langle t,m\rangle) \in tr \in traces(P), t',m' : \langle t,m\rangle \vdash_H \langle t',m'\rangle \implies t' \neq tag$$

Formalizing S-E Attacks

Rule-based human approach

Untrained human rules

1. $[\,\mathrm{Fr}(\sim x)\,] \xrightarrow{\mathrm{Fresh}(H,\langle t,\sim x\rangle)} [\,!\,\mathrm{HK}(H,\langle t,\sim x\rangle)\,]$
2. $[\,!\,\mathrm{HK}(H,\langle t,x\rangle)\,] \xrightarrow{\mathrm{Send}(H,ins,P,\langle t,x\rangle)} [\,\mathrm{Out}_{ins}(\langle H,P,\langle t,x\rangle\rangle)\,]$
3. $[\,\mathrm{In}_{ins}(\langle P,H,\langle t,x\rangle\rangle)\,] \xrightarrow{\mathrm{Receive}(H,ins,P,\langle t,x\rangle)} [\,!\,\mathrm{HK}(H,\langle t,x\rangle)\,]$

Enforce important security checks

$$\mathrm{ICompare}(H,tag) \coloneqq \forall \mathrm{Receive}(H,l,P,\langle t,m\rangle) \in tr \in traces(P), m':$$
$$\langle t,m\rangle \vdash_H \langle tag,m'\rangle \Longrightarrow \mathrm{InitK}(H,\langle tag,m'\rangle) \in tr$$

Formalizing S-E Attacks

Rule-based human approach

Positive sides to this approach

1. We can easily specify the level of training of the user
2. Assumptions on the possible mistakes are translated into restrictions → no need to rewrite property lemmas
3. We can analyze the security properties of a distinguished trained human H while allowing others less-trained agents in the network

Conclusions

What have we seen today?

1. How is the Dolev Yao model useful for protocol verification
2. An introduction to the Tamarin prover from a user's perspective
3. How to formalize Social Engineering attacks and human errors within Tamarin's multiset rewriting system

Are there other Tamarin extensions available?

- 📖 Automated analysis of security protocols with global state
- 📖 Distance-Bounding Protocols: Verification without Time and Location
- 📖 Alice and Bob Meet Equational Theories

… and (hopefully) more to come