Symbolic Verification of Security Protocols with Tamarin

Nicola Vitacolonna

University of Udine

Department of Mathematics, Computer Science, and Physics

May 25, 2021

References

(Schmidt, 2012b) (especially Ch. 3) and (Meier, 2013)

- PhD theses laying the foundations for Tamarin

(Schmidt, et al., 2012a)

- Main paper on Tamarin's foundations

Online Resources

- **Summer School on Verification Technology, Systems & Applications (David Basin' Slides)**
  - The present slides are ~~copied~~ heavily inspired by Basin's presentation
- **Teaching Materials for the Tamarin Prover**
- **Tamarin's manual**

Motivation

- Many good cryptographic primitives: RSA, DSA, ElGamal, AES, SHA3, ...
- How can we construct *secure distributed applications* with them?
  - E-commerce
  - E-banking
  - E-voting
  - Mobile communication
  - Digital contract signing
- Even if cryptography is hard to break, this is not a trivial task

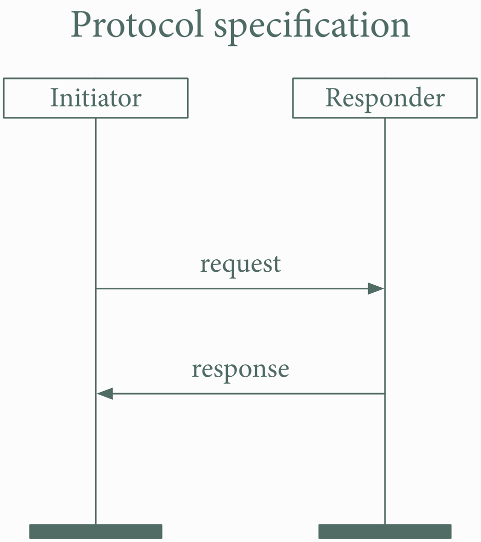
What Is a Protocol?

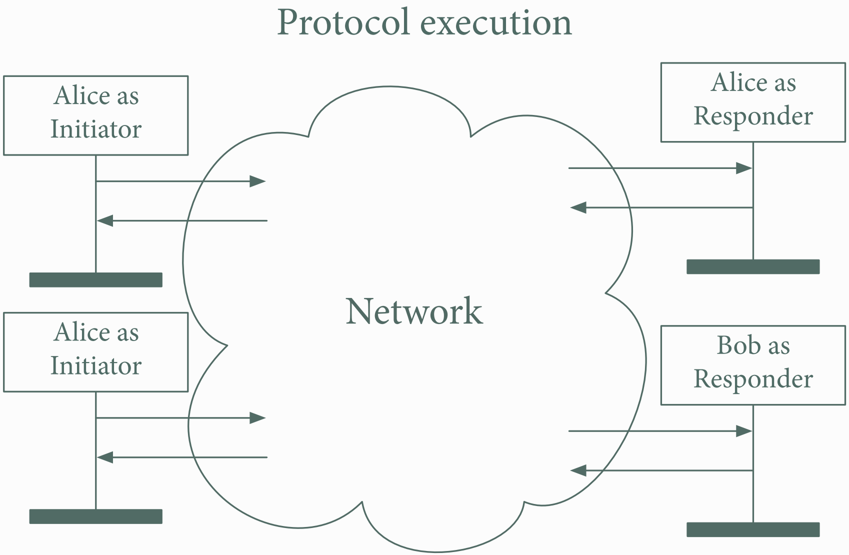
- A *protocol* consists of a set of rules (conventions) that determine the exchange of messages between two or more principals
- In short, a *distributed algorithm* with emphasis on communication
- *Security* (or *cryptographic* protocols use cryptographic mechanisms to achieve their *security goals* against a given *threat model*
  - Entity or message authentication, message secrecy, key establishment, integrity, non-repudiation, etc.
- Small recipes, but nontrivial to design and understand
- "Three-line programs that people still get wrong"

Preliminaries

- In a concrete execution of a protocol, the roles are played by *agents* (or *principals*)
- Security goals:
  - *Key secrecy*: at the end of a run of the protocol between A and B, the session key is known only to A and B
  - *Key freshness*: A and B know that the key is freshly generated
- How do we formalize the protocol steps and goals?
- How about "knowledge", "secrecy", "freshness"?

Protocol Specification vs Protocol Execution





Building a Key Establishment Protocol

- Let's try to design a cryptographic protocol from first principles
- Common scenario:
  - A *set of users*, any two of who may wish to establish a new *session key* for subsequent secure communication
  - Users are not necessarily honest
  - There is an *honest server*: it never cheats and never gives out secrets
- We thus design a protocol with three roles: initiator role A, responder role B, and server role S

First Attempt (Alice & Bob Notation)

$$
\begin{aligned}
A \to S: &\quad A, B \\
S \to A: &\quad k_{AB} \\
A \to B: &\quad k_{AB}, A
\end{aligned}
$$

- Issue? Secrecy: k_{AB} is sent in clear
- The session key k_{AB} must be transported to A and B, but readable by no other parties

Threat assumption 1: The adversary is able to eavesdrop on all sent messages

⇒ Use cryptography

Second Attempt

Assume that S initially shared a key $k_S(X)$ with each agent X

$$\begin{aligned} A \rightarrow S: &\quad A, B \\ S \rightarrow A: &\quad \{k_{AB}\}_{k_S(A)}, \{k_{AB}\}_{k_S(B)} \\ A \rightarrow B: &\quad \{k_{AB}\}_{k_S(B)}, A \end{aligned}$$

- Is eavesdropping an issue? No, messages are encrypted
- *Perfect cryptography assumption:* encrypted messages may be deciphered only by agents who have the required decryption key

Threat assumption 2: The adversary may also capture and modify messages

Dolev-Yao Model (Dolev & Yao, 1983)

The adversary is able to intercept messages on the network and send to anybody (under any sender name) modified or new messages based on any information available

- The adversary has complete control over the network
- We assume the *worst-case* network adversary
  - Although only a few messages are exchanged in a legitimate session, there are infinitely many variations where the adversary can participate
  - These variations involve an unbounded number of messages and each must satisfy the protocol's security requirements

A Binding Attack on the Second Attempt

Let I be an adversary (intruder)

$$\begin{aligned} A \rightarrow I: &\quad A, B \\ I \rightarrow S: &\quad A, I \\ S \rightarrow I: &\quad \{k_{AI}\}_{k_{S(A)}}, \{k_{AI}\}_{k_{S(I)}} \\ I \rightarrow A: &\quad \{k_{AI}\}_{k_{S(A)}}, \{k_{AI}\}_{k_{S(I)}} \\ A \rightarrow I: &\quad \{k_{AI}\}_{k_{S(I)}}, A \end{aligned} \tag{1}$$

Threat assumption 3: The adversary may be a legitimate protocol participant (an insider), or an external party (an outsider), or a combination of both

Third Attempt

$$
\begin{aligned}
A \to S: &\quad A, B \\
S \to A: &\quad \{k_{AB}, B\}_{k_S(A)}, \{k_{AB}, A\}_{k_S(B)} \\
A \to B: &\quad \{k_{AB}, A\}_{k_S(B)}
\end{aligned}
$$

- The previous attack now fails
- But old keys can be *replayed* at a later time…

Threat assumption 4: The adversary is able to obtain the value of a session key used in any "sufficiently old" previous run of the protocol

Replay Attack and Session Key Compromise

Suppose that the intruder knows $\{k_{AB'}, B\}_{k_S(A)}$ and $\{k_{AB'}, A\}_{k_S(B)}$ from an old session between A and B, and was able to discover $k_{AB'}$ (**key compromise**)

Then, I masquerades as S and replays $k_{AB'}$

$$\begin{aligned} A \rightarrow I: &\quad A, B \\ I \rightarrow A: &\quad \{k_{AB'}, B\}_{k_S(A)}, \{k_{AB'}, A\}_{k_S(B)} \\ A \rightarrow B: &\quad \{k_{AB'}, A\}_{k_S(B)} \end{aligned}$$

After the protocol has run, the adversary can decrypt, modify, or inject messages encrypted with $k_{AB'}$ (no confidentiality or integrity)

Thwarting the Replay Attack

- The replay attack can still be regarded as successful even if the adversary has not obtained the value of $k_{AB'}$
  - Adversary gets A and B to accept an old session key!
  - I can therefore replay (encrypted) messages sent in the previous session
  - Various techniques may be used to guard against replay of session key, such as incorporating **challenge-response**
- A **nonce** ("a number used only once") is a random value generated by one principal and returned to that principal to show that a message is newly generated

Fourth Attempt: NSCK

Let N_X denote a nonce generated by X

$$
\begin{aligned}
A \rightarrow S: &\quad A, B, N_A \\
S \rightarrow A: &\quad \{k_{AB}, B, N_A, \{k_{AB}, A\}_{k_S(B)}\}_{k_S(A)} \\
A \rightarrow B: &\quad \{k_{AB}, A\}_{k_S(B)} \\
B \rightarrow A: &\quad \{N_B\}_{k_{AB}} \\
A \rightarrow B: &\quad \{N_B - 1\}_{k_{AB}}
\end{aligned}
$$

- *Needham-Schroeder with Conventional Keys (1978)*
- Assumes that only A can form correct reply to message 4 from B

Attack on NSCK

The adversary masquerades as A and *convinces* B *to use old key* $k_{AB'}$

$$
\begin{aligned}
A \rightarrow S: &\quad A, B, N_A \\
S \rightarrow A: &\quad \{k_{AB}, B, N_A, \{k_{AB}, A\}_{k_S(B)}\}_{k_S(A)} \\
I \rightarrow B: &\quad \{k_{AB'}, A\}_{k_S(B)} \\
B \rightarrow I: &\quad \{N_B\}_{k_{AB'}} \\
I \rightarrow B: &\quad \{N_B - 1\}_{k_{AB'}}
\end{aligned}
$$

Attack found by Dennis and Sacco

Fifth (and Final) Attempt (1)

$$
\begin{aligned}
B \to A: &\quad A, B, N_B \\
A \to S: &\quad A, B, N_A, N_B \\
S \to A: &\quad \{k_{AB}, B, N_A\}_{k_S(A)}, \{k_{AB}, A, N_B\}_{k_S(B)} \\
A \to B: &\quad \{k_{AB}, A, N_B\}_{k_S(B)}
\end{aligned}
$$

- The protocol is now initiated by B who sends his nonce N_B first to A
- A adds her nonce N_A and sends both to S, who now sends K_{AB} in separate messages for A and B, which can be verified as fresh by the respective recipients

Fifth (and Final) Attempt (2)

$$B \to A: \quad A, B, N_B$$

$$A \to S: \quad A, B, N_A, N_B$$

$$S \to A: \quad \{k_{AB}, B, N_A\}_{k_S(A)}, \{k_{AB}, A, N_B\}_{k_S(B)}$$

$$A \to B: \quad \{k_{AB}, A, N_B\}_{k_S(B)}$$

- In NSCK, A can verify that her communication partner actually possesses the key (**key confirmation**), thanks to the last two messages
- In above protocol, neither A nor B can deduce at the end of a successful run that the partner actually has k_{AB} (is this an issue?)

Fifth (and Final) Attempt (3)

$$
\begin{aligned}
B \to A: &\quad A, B, N_B \\
A \to S: &\quad A, B, N_A, N_B \\
S \to A: &\quad \{k_{AB}, B, N_A\}_{k_S(A)}, \{k_{AB}, A, N_B\}_{k_S(B)} \\
A \to B: &\quad \{k_{AB}, A, N_B\}_{k_S(B)}
\end{aligned}
$$

- This protocol avoids all the attacks shown so far
- Under the assumptions of perfect cryptography and honesty of S
- So, is it correct? (What does it mean to be "correct"?)

Summary: Adversary, Attacks and Defense

The **adversary** must be expected to

- eavesdrop on messages (but cannot break cryptography)
- completely control the network
  - immediately intercept, modify, drop, and fake messages
  - compose/decompose messages with the available keys
  - participate in the protocol (as insider or outsider)
  - be able to obtain old session keys
- **Attacks and defenses:**
  - *Eavesdropping:* encrypt session keys using long-term keys
  - *Binding attack:* cryptographically bind names to session keys
  - *Replay attack:* use challenge-response based on nonces

(Informally Stated) Types of Protocol Attacks

- *Intruder-in-the-middle attack:* $A \leftrightarrow I \leftarrow B$
- *Replay (or freshness) attack:* reuse parts of previous messages
- *Masquerading attack:* pretend to be another principal
- *Reflection attack:* send transmitted information back to originator
- *Oracle attack:* take advantage of normal protocol responses as encryption and decryption "services"
- *Binding attack:* using messages in a different context/for a different purpose than originally intended
- *Type flaw attack:* substitute a different type of message field

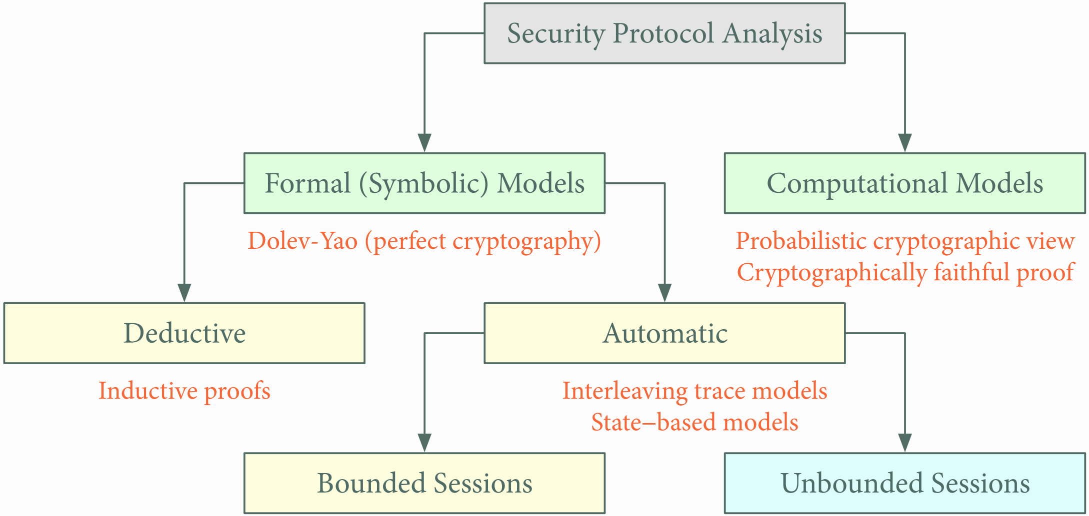
Prudent Engineering of Security Protocols

From (Abadi & Needham, 1996)

- Every message should say what it means
- Specify clear conditions for a message to be acted on
- Mention names explicitly if they are essential to the meaning
- Be clear about the purpose of encryption: confidentiality, message authentication, binding of messages, etc.
- Be explicit on what properties you are assuming
- Beware of clock variations (for timestamps)
- Etc... Is the protocol secure then? Is it optimal/minimal?

Formal Analysis of Security Protocol

Goal: formally model protocols and their properties and provide a mathematically sound means for reasoning about these models



Why Is Security Protocol Analysis Difficult?

Infinite state space

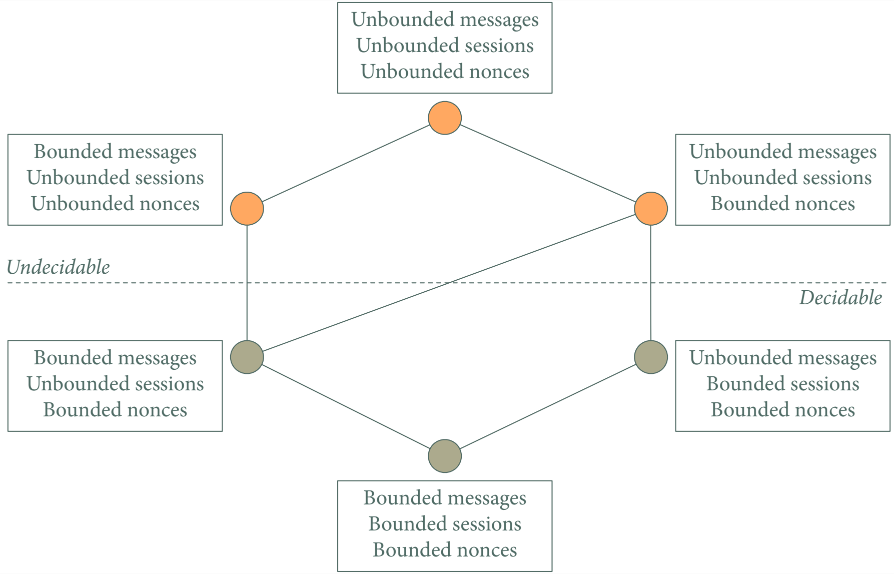
- *Messages:* adversary can produce messages of arbitrary size
- *Sessions:* unbounded number of parallel sessions
- *Nonces:* unbounded number of nonces (if sessions unbounded)

Undecidability

- Secrecy problem for security protocols is undecidable (Even & Goldreich, 1983)
- Even if the number of nonces or the message size is bounded

(Un)decidability: The Complete Picture

Bottom line: need at least two bounded parameters for decidability



Tamarin: High Level Picture

Modeling protocol and adversary with **multiset rewriting**

- Specifies a **labelled transition system**
- Induces a set of **traces**

Property verification using a guarded fragment of first-order logic

- Specifies "good" traces

TAMARIN tries to prove that all traces are good, or to find a counterexample trace (**attack**)

Verification algorithm is sound and complete; termination is not guaranteed

Security Protocol Model

Security protocol ≡ Labelled transition system

State:

- Adversary's knowledge
- Messages on the network
- Information about freshly generated values
- Protocol's state

The adversary and the protocol interact by updating network messages and freshness information

Transition Rules

Adversary capabilities and protocols are specified jointly as a set of (labeled) **multiset rewriting rules**

Basic ingredients:

- **terms** (think "messages")
- **facts** (think "sticky notes on the fridge")
- Special facts: $\mathsf{Fr}(t)$, $\mathsf{In}(t)$, $\mathsf{Out}(t)$, $\mathsf{K}(t)$
- State of the system $\equiv$ multiset of facts
- **Transition rules**: $L \dashv[\ A\ ]\!\!\rightarrow R$, with L, A, R are multisets of facts

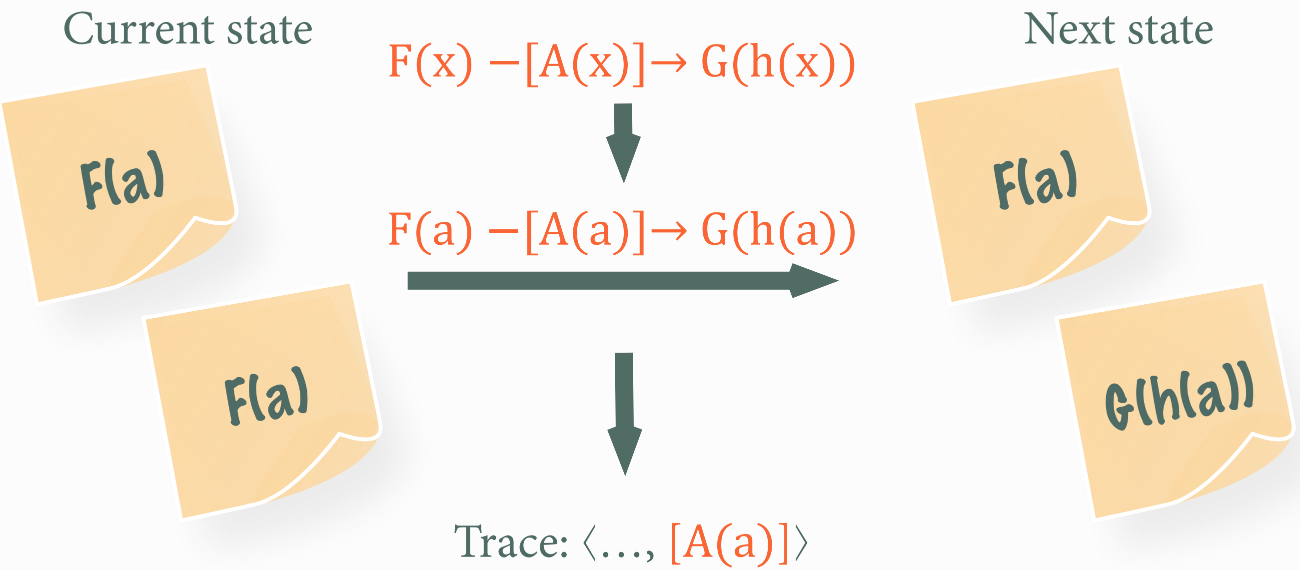
Informal Semantics of Transitions

- Let S be the current state of the system
- Let $L \to R$ be a transition rule
- Let $l \to r$ be a ground instance of the rule (ground $\equiv$ no variables)
- Applying $l \to r$ to S yields the new state:

  $$S \setminus^{\#} l \cup^{\#} r$$

  where $\setminus^{\#}$ and $\cup^{\#}$ are multiset difference and union, respectively
- For labelled rules of the form $l \dashv\![\ a\ ]\!\mapsto r$, a is added to the trace of the execution

Transition Rules: Example



The Model at 10,000 Feet (1)

Term algebra

- $\mathsf{enc}/2, \mathsf{dec}/2, h/1, \_\cdot\_, \_^{-1}, \ldots$

Equational Theory

- $\mathsf{dec}(\mathsf{enc}(m,k),k) \simeq m, x \cdot y \simeq y \cdot x, {x^{-1}}^{-1} \simeq x, \ldots$

Facts

- $F(t_1, \ldots, t_n)$

The Model at 10,000 Feet (2)

Transition system

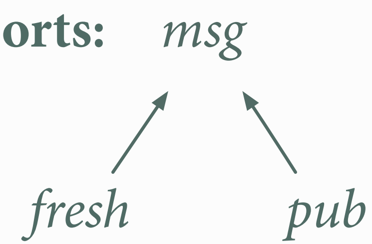
- State: multiset of facts
- (Labelled) Rules: $L \dashv[\ A\ ]\!\!\rightarrow R$

TAMARIN-**specific**

- Built-in Dolev-Yao attacker rules
  - $\textsf{\textbf{Out}}(x) \dashv[]\!\!\rightarrow \textsf{\textbf{K}}(x), \textsf{\textbf{K}}(x) \dashv[\ \textsf{\textbf{K}}(x)\ ]\!\!\rightarrow \textsf{\textbf{In}}(x), \ldots$
- Special **fresh** rule:
  - $[\ ] \dashv[]\!\!\rightarrow \textsf{\textbf{Fr}}(x)$

Sorts and Signatures

Sorts:



We assume there are two countably infinite sets *FN* and *PN* of fresh and public names (i.e., constant symbols), and a countably infinite set $\mathcal{V}_s$ of variables for each sort s

Signature[1] Σ_{DH}: enc/2, dec/2, h/1, $\langle .,. \rangle$, fst/1, snd/1, _^_, $\_^{-1}$, _*_, 1

The set of cryptographic messages $\mathcal{M}$ is modelled as the set $\mathcal{T}$ of well-sorted ground terms over Σ_{DH}

1 User-defined signatures are supported, but for simplicity we stick to a fixed signature.

Term Algebra

Let Σ be a signature and $\mathcal{X}$ be a set of variables, with $\Sigma \cap \mathcal{X} = \emptyset$

The set of Σ-**terms** $\mathcal{T}_\Sigma(\mathcal{X})$ is the least set such that

- $\mathcal{X} \subseteq \mathcal{T}_\Sigma(\mathcal{X})$
- if $t_1, \ldots, t_n \in \mathcal{T}_\Sigma(\mathcal{X})$ and $f \in \Sigma$ is a function symbol of arity n then $f(t_1, \ldots, t_n) \in \mathcal{T}_\Sigma(\mathcal{X})$

The set of **ground terms** $\mathcal{T}_\Sigma$ is $\mathcal{T}_\Sigma(\emptyset)$, i.e., it consists of terms built without variables

The (Σ)-**term algebra** has domain $\mathcal{T}_\Sigma(\mathcal{X})$ and interprets each term as itself

Substitutions

- A **substitution** is a function $\sigma\colon \mathcal{X} \to \mathcal{T}_\Sigma(\mathcal{X})$
- A substitution can be extended to a mapping[2] $\sigma\colon \mathcal{T}_\Sigma(\mathcal{X}) \to \mathcal{T}_\Sigma(\mathcal{X})$ in an straightforward way:

$$f(t_1, ..., t_n)\sigma = f(t_1\sigma, ...t_n\sigma)$$

Example:

- Let $s = f(e, x)$ and $t = f(y, f(x, y))$
- Let $\sigma = \{x \mapsto i(y), y \mapsto e\}$
- Then $s\sigma = f(e, i(y))$ and $t\sigma = f(e, f(i(y), e))$

---

[2] With abuse of notation, we keep calling it σ

Equational Theories

Equation (over Σ)
A pair of terms (t, u), with $t, u \in \mathcal{T}_\Sigma(\mathcal{X})$, written $t \simeq u$

(Σ, E)-equational presentation
A set of equations E over a signature Σ

Equations can be oriented, written as $t \rightarrow u$ (**rewriting**), for use in simplifying terms

Algebraic Properties

- A set of equation E induces a congruence relation $=_E$ on terms (**equational theory**) and thus equivalence classes $[t]_E$
- The **quotient algebra** $\mathcal{T}_\Sigma(\mathcal{X})/_{=_E}$ interprets each term by its equivalence class
- Terms t and u are **equal (modulo E)**, written $t =_E u$, iff $[t]_E = [u]_E$

Example

- Let $\Sigma = \{s/1, +/2, 0\}$
- Let $E = \{X + 0 \simeq X, X + s(Y) \simeq s(X + Y)\}$
- Then, $s(s(0)) + s(0) =_E s(s(s(0)) + 0) =_E s(s(s(0)))$

Unification

- $t, u \in \mathcal{T}_\Sigma(\mathcal{X})$ are (Σ, E)-**unifiable** if there is σ such that $t\sigma =_E t'\sigma$
- For syntactic unification (i.e., when $E = \emptyset$) there is a **most general unifier**, and unification is decidable
- Unification modulo theories ($E \neq \emptyset$) is undecidable in general
- $\Rightarrow$ Restrictions on the form of the acceptable equational theories

The Equational Theory E_{DH}

(1) $\mathsf{dec}(\mathsf{enc}(m, k), k) \simeq m$	(6) $x * 1 \simeq x$
(2) $\mathsf{fst}(\langle x, y \rangle) \simeq x$	(7) $x * x^{-1} \simeq 1$
(3) $\mathsf{snd}(\langle x, y \rangle) \simeq y$	(8) $(x^{-1})^{-1} \simeq x$
(4) $x * (y * z) \simeq (x * y) * z$	(9) $(x^y)^z \simeq x^{y*z}$
(5) $x * y \simeq y * x$	(10) $x^1 \simeq x$

The theory can be extended with any *subterm-convergent* rewriting theory, which permits, for instance, to model asymmetric encryption, signatures, etc.

The Equational Theory E_{DH}: Example

By equation (9), the term

$$((g^a)^b)^{a^{-1}}$$

is **equal (modulo E_{DH}) to**

$$g^{((a*b)*a^{-1})}$$

and can be further simplified to

$$g^b$$

using Equations (4–7)

Subterm-Convergent Rewriting

- **Termination:** it is always the case that after finitely many rule applications no more rules can be applied—i.e., each term has a **normal form** (or, is **reduced**)
- **Confluence:** if a given term t can be rewritten (in an arbitrary number of steps) to t_1 and t_2, then there is t' such that both t_1 and t_2 can be rewritten to t'
- A confluent and terminating theory is **convergent**
- A **subterm-convergent** theory is convergent and, for each rule $L \rightarrow R$ of the theory, R is a proper subterm of L, or R is ground and in normal form

Facts

- The states of the transition system are finite multisets of **facts**
- A fixed set of fact symbols (**In**(), **Out**(), **K**(), **Fr**()) is used to encode the adversary's knowledge, freshness information, and the messages on the network
- The remaining fact symbols are used to represent the protocol state
- Facts can be:
  - **linear:** they model resources that can be only consumed once
  - **persistent:** they model inexhaustible resources that can be consumed arbitrarily often

Special Facts

- **K**(m) (persistent): m is known to the adversary
- **Out**(m) (linear): message m has been sent, and can be received by the adversary
- **In**(m) (linear): the adversary has sent message m, and m can be received by the protocol
- **Fr**(n) (linear): the new name n was freshly generated

Labelled Multiset Rewriting

Labeled multiset rewriting rule

- A triple (L, A, R), denoted $L \dashv[\ A\ ]\mapsto R$, with
  - L: multiset of facts called **premises**
  - A: multiset of facts called **actions**
  - R: multiset of facts called **conclusions**
- Three types of rules:
  - A rule for fresh name generation
  - Message deduction rules
  - Protocol rules

Fresh Name Generation

All fresh names are created with the following built-in rule:

$$\textsc{Fresh}: \quad [\,] \dashv[]\mapsto \mathbf{Fr}(x : \mathit{fresh})$$

- This is the only rule that produces **Fr**() facts
- Ground instances of this rules are assumed to be unique, i.e., the same fresh name is never generated twice

Message Deduction Rules (1)

$\mathsf{Out}(x) \dashv [] \mapsto \mathsf{K}(x)$

- Allows an adversary to receive the messages sent by the protocol

$\mathsf{K}(x) \dashv [\ \mathsf{K}(x)\ ] \mapsto \mathsf{In}(x)$

- Allows the protocol to receive a message from the adversary
- Messages sent by the adversary are observable in the trace

Message Deduction Rules (2)

$[\,] \dashv[]\mapsto \mathsf{K}(x : pub)$

- The adversary knows all public names

$\mathsf{Fr}(x) \dashv[]\mapsto \mathsf{K}(x)$

- The adversary can generate and use fresh names

For every k-ary function symbol f:

$\mathsf{K}(x_1), ..., \mathsf{K}(x_k) \dashv[]\mapsto \mathsf{K}(f(x_1, ..., x_k))$

- The adversary can apply any function to the known messages

Protocol Rules

A **protocol rule** is a multiset rewriting rule $L \dashv\![\ A\ ]\!\mapsto R$ such that

- there is no occurrence of $\mathsf{K}()$ anywhere in the rule
- $\mathsf{Out}()$ can appear only in the conclusions
- $\mathsf{In}()$ and $\mathsf{Fr}()$ can appear only in the premises
- all non-public variables in the conclusions must occur in the premises

A **protocol** is a finite set of protocol rules

Transition Relation

- Let S be the current state
- Let $l \; -\!\![\; a \;]\!\!\rightarrow \; r$ be a ground instance of a rule in P, a message deduction rule, or a fresh name generating rule
- Let $lin(l)$ be the multiset of linear facts in l
- Let $pers(l)$ be the set of persistent facts in l
- Assume that $lin(l) \subseteq^{\#} S$ and $pers(l) \subseteq S$ (note that $\subseteq^{\#}$ is multiset inclusion, and equality is modulo the equational theory)
- Then, compute the new state $S' \doteq S \setminus^{\#} lin(l) \cup^{\#} r$
- Append a to the end of the current trace

Traces

- **Trace:** a sequence $\langle A_1, ..., A_n \rangle$ of sets of ground facts denoting the sequence of actions that happened during a protocol's execution
- $\mathit{traces}(P)$ denotes the set of all traces generated by all possible executions of the protocol P

$$\mathit{traces}(P) \doteq \{\, \langle A_1 ..., A_n \rangle \mid \exists S_1 ... \exists S_n .\ \emptyset^{\#} \xrightarrow{A_1} S_1 \xrightarrow{A_2} \cdots \xrightarrow{A_{n-1}} S_{n-1} \xrightarrow{A_n} S_n$$

$$\textit{and no ground instance of } \mathrm{F{\scriptsize RESH}} \textit{ is used twice} \,\}$$

Observable trace: trace $\langle A_1, ..., A_n \rangle$ in which the empty A_i's are removed

Executions and Traces: Example

Rule 1: $[\,] \dashv[\mathsf{Init}()]\!\to \mathsf{A}(5)$

Rule 2: $\mathsf{A}(x) \dashv[\mathsf{Step}(\mathsf{x})]\!\to \mathsf{B}(x)$

Example of execution:

Current state	Ground rule	Next state	Trace
$\emptyset^{\#}$	$[\,] \dashv[\mathsf{Init}()]\!\to \mathsf{A}(5)$	$[\mathsf{A}(5)]$	$\langle [\mathsf{Init}()] \rangle$
$[\mathsf{A}(5)]$	$[\,] \dashv[\mathsf{Init}()]\!\to \mathsf{A}(5)$	$[\mathsf{A}(5), \mathsf{A}(5)]$	$\langle [\mathsf{Init}()], [\mathsf{Init}()] \rangle$
$[\mathsf{A}(5), \mathsf{A}(5)]$	$\mathsf{A}(5) \dashv[\mathsf{Step}(5)]\!\to \mathsf{B}(5)$	$[\mathsf{A}(5), \mathsf{B}(5)]$	$\langle [\mathsf{Init}()], [\mathsf{Init}()], [\mathsf{Step}(5)] \rangle$

Executions and Traces: Example (Persistent Facts)

Rule 1 (R1): $[\,] \;-\![\; \mathsf{I}() \;]\!\!\rightarrow\; !\mathsf{C}(a), \mathsf{D}(1)$

Rule 2 (R2): $!\mathsf{C}(x), \mathsf{D}(y) \;-\![\; \mathsf{S}(x,y) \;]\!\!\rightarrow\; \mathsf{D}(h(y))$

Example of execution:

Current state	Ground rule	Next state	Trace
$\emptyset^{\#}$	R1	$[!\mathsf{C}(a), \mathsf{D}(1)]$	$\langle \mathsf{I}() \rangle$
$[!\mathsf{C}(a), \mathsf{D}(1)]$	$\mathrm{R2}[x/a, y/1]$	$[!\mathsf{C}(a), \mathsf{D}(h(1))]$	$\langle \mathsf{I}(), \mathsf{S}(a, 1) \rangle$
$[!\mathsf{C}(a), \mathsf{D}(h(1))]$	$\mathrm{R2}[x/a, y/h(1)]$	$[!\mathsf{C}(a), \mathsf{D}(h(h(1)))]$	$\langle \mathsf{I}(), \mathsf{S}(a, 1), \mathsf{S}(a, h(1)) \rangle$

Modeling Public-Key Infrastructure

A pre-distributed PKI with asymmetric keys for each party can be modeled by a single rule that generates a key for a party

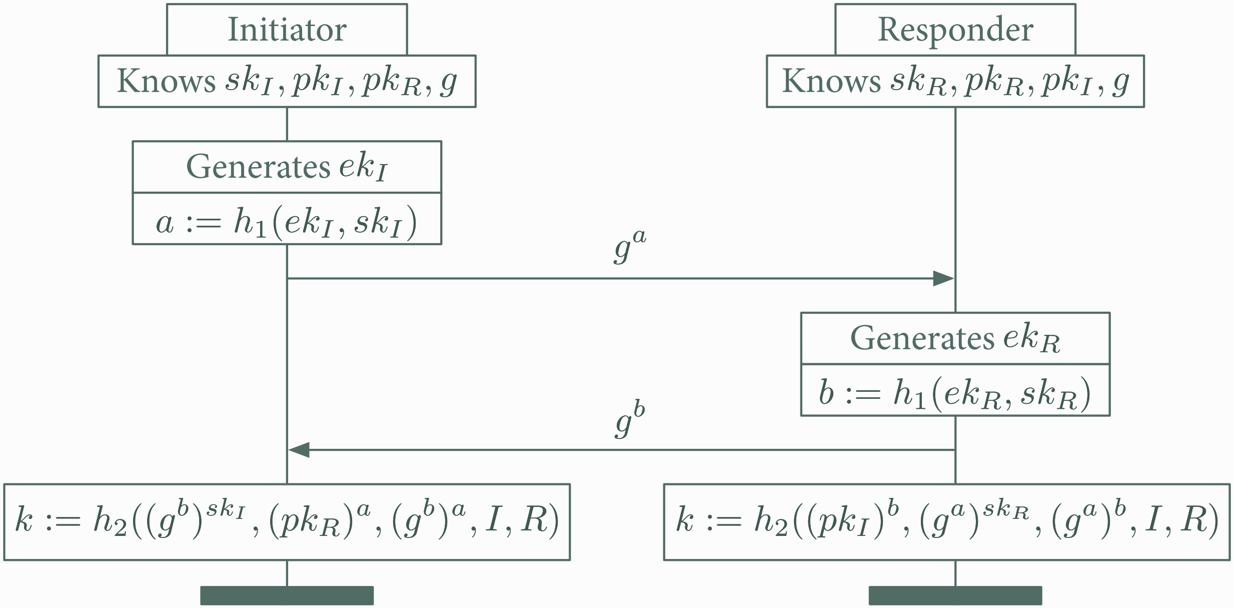
$$\mathbf{Fr}(y) \;{-}[]{\rightarrow}\; !\mathsf{Sk}(X, y),\ !\mathsf{Pk}(X, \mathsf{pk}(y)),\ \mathbf{Out}(\mathsf{pk}(y))$$

- $!\mathsf{Sk}(X : pub, y : fresh)$: y is a private key of agent X
- $!\mathsf{Pk}(X : pub, y : fresh)$: y is a public key of agent X
- $\mathsf{pk}(x)$: denotes the public key corresponding to private key x

For Diffie-Hellman-style key pairs (g is a constant, i.e., a 0-ary function):

$$\mathbf{Fr}(y) \;{-}[]{\rightarrow}\; !\mathsf{Sk}(X, y),\ !\mathsf{Pk}(X, \mathsf{g}^y),\ \mathbf{Out}(\mathsf{g}^y)$$

The NAXOS Protocol



Formalizing the NAXOS Protocol (1)

Generate long-term keypair:

$$\frac{\mathsf{Fr}(lk_X)}{!\mathsf{Ltk}(\mathcal{X} : pub, lk_X),\ !\mathsf{Pk}(\mathcal{X}, \mathsf{g}^{lk_X}),\ \mathbf{Out}(\mathsf{g}^{lk_X})}$$

Initiator step 1:

$$\frac{\mathsf{Fr}(ek_I),\ !\mathsf{Ltk}(\mathcal{I}, lk_I)}{\mathsf{Start}(ek_I, \mathcal{I}, \mathcal{R} : pub, lk_I, \mathsf{g}^a),\ !\mathsf{Ephk}(ek_I, ek_I),\ \mathbf{Out}(\mathsf{g}^a)}$$

where $a = h_1(ek_I, lk_I)$

Formalizing the NAXOS Protocol (2)

Initiator step 2:

$$\frac{\mathsf{Start}(ek_I, \mathcal{I}, \mathcal{R}, lk_I, \mathtt{g}^a),\ !\mathsf{Pk}(\mathcal{R}, \mathsf{pk}_R),\ \mathbf{In}(Y)}{!\mathsf{Sessk}(ek_I, k_I)} \left[\begin{array}{c} \mathsf{Accept}(ek_I, \mathcal{I}, \mathcal{R}, k_I), \\ \mathsf{Sid}(ek_I, \langle \mathtt{Init}, \mathcal{I}, \mathcal{R}, \mathtt{g}^a, Y \rangle), \\ \mathsf{Match}(ek_I, \langle \mathtt{Resp}, \mathcal{R}, \mathcal{I}, \mathtt{g}^a, Y \rangle) \end{array} \right]$$

where $k_I = h_2(Y^{lk_I}, \mathsf{pk}_R{}^a, Y^a, \mathcal{I}, \mathcal{R})$

Formalizing the NAXOS Protocol (3)

Responder step:

$$\frac{\mathsf{Fr}(ek_R),\ !\mathsf{Ltk}(\mathcal{R}, lk_R),\ !\mathsf{Pk}(\mathcal{I}, \mathsf{pk}_I),\ \mathbf{In}(X)}{!\mathsf{Sessk}(ek_R, k_R),\ !\mathsf{Ephk}(ek_R, ek_R),\ \mathbf{Out}(\mathtt{g}^b)} \left[\begin{array}{c} \mathsf{Accept}(ek_R, \mathcal{R}, \mathcal{I}, k_R), \\ \mathsf{Sid}(ek_R, \langle \mathtt{Resp}, \mathcal{R}, \mathcal{I}, X, \mathtt{g}^b \rangle), \\ \mathsf{Match}(ek_R, \langle \mathtt{Init}, \mathcal{I}, \mathcal{R}, X, \mathtt{g}^b \rangle) \end{array} \right]$$

where

- $b = h_1(ek_R, lk_R)$
- $k_R = h_2(\mathsf{pk}_I{}^b, X^{lk_R}, X^b, \mathcal{I}, \mathcal{R})$

Formalizing Additional Attacker's Capabilities

The session key and the ephemeral key of a principal can be exfiltrated:

$$\frac{!\mathsf{Sessk}(s,k)}{\mathbf{Out}(k)}[\,\mathsf{SesskRev}(s)\,]$$

$$\frac{!\mathsf{Ephk}(s,ek_X)}{\mathbf{Out}(ek_X)}[\,\mathsf{EphkRev}(s)\,]$$

The long term secret of a principal can be exfiltrated:

$$\frac{!\mathsf{Ltk}(X,lk_X)}{\mathbf{Out}(lk_X)}[\,\mathsf{LtkRev}(X)\,]$$

Protocol Goals

A **security goal** defines what the protocol is intended to achieve

- Authenticate messages, binding them to their originator
- Ensure timeliness of messages (recent, fresh, ...)
- Guarantee secrecy of certain items (e.g., generated keys)

Most common goals:

- secrecy (many forms)
- authentication (many forms)

Other goals: anonymity, non-repudiation (of receipt, submission, delivery), key confirmation, fairness, availability,...

Protocol Properties and Correctness

Properties

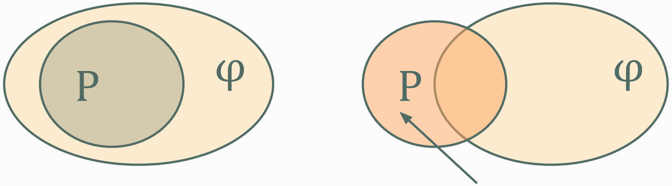
- Semantics of protocol P is $traces(P)$
- A security goal/property φ also denotes a set of traces $traces(\varphi)$

Correctness has an exact meaning:

$P \models \varphi \quad \text{iff} \quad traces(P) \subseteq traces(\varphi)$

Attack traces are those in

$traces(P) \setminus traces(\varphi)$



Ok, no attacks *Attacks*

Security Properties

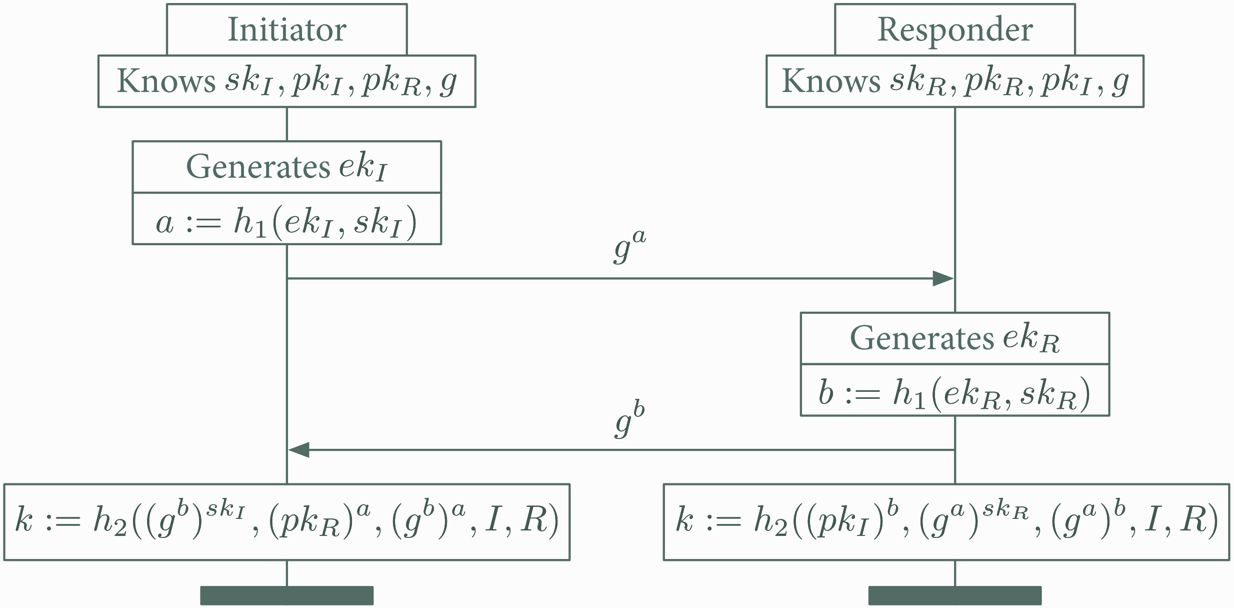
- Many-sorted first-order logic is used to specify security properties
- The logic supports quantification over both messages and time points
- Formulas are interpreted over traces (the temporal domain is $\mathbb{Q}$)
- **Trace atoms:**
  - $\perp$ (false)
  - Term equality: $t_1 \approx t_2$
  - Time point ordering and equality: $i \lessdot j$ and $i \doteq j$
  - Actions at time points: $F@i$, for a fact F and a time point i
- **Trace formula:** a first-order formula over trace atoms

Semantics of Trace Formulas

For a trace $T = \langle A_1, ..., A_n \rangle$ and sort-respecting valuation θ:

$(T, \theta) \vDash F@i$	iff $1 \leq \theta(i) \leq n$ and $\theta(F) \in T[\theta(i)]$
$(T, \theta) \vDash i \lessdot j$	iff $\theta(i) < \theta(j)$
$(T, \theta) \vDash i \doteq j$	iff $\theta(i) = \theta(j)$
$(T, \theta) \vDash t_1 \approx t_2$	iff $\theta(t_1) \simeq \theta(t_2)$
$(T, \theta) \vDash \neg\varphi$	iff it is not the case that $(T, \theta) \vDash \varphi$
$(T, \theta) \vDash \varphi \wedge \psi$	iff $(T, \theta) \vDash \varphi$ and $(T, \theta) \vDash \psi$
$(T, \theta) \vDash \exists x{:}\, s.\, \varphi$	iff there is $v \in \mathcal{D}_s$ such that $(T, \theta[x \mapsto v]) \vDash \varphi$

The NAXOS Protocol



The NAXOS Protocol: Formalizing Secrecy (1)

"If A accepts key k in a test session s with B, and the adversary learns k, then… something bad has happened"

$\forall s\, A\, B\, k\, i_1\, i_2.\, (\mathsf{Accept}(s, A, B, k)@i_1 \wedge \mathbf{K}(k)@i_2) \rightarrow (\text{Bad things}\ldots)$

The NAXOS Protocol: Formalizing Secrecy (2)

Which bad things?

- The session key of test session s was revealed

  $\exists i_3.\,\mathsf{SesskRev}(s)@i_3$

- Or, a session key for a matching session was revealed

  $\exists s'\,\mathrm{sid}\,i_3\,i_4.\left(\mathsf{Sid}(s',\mathrm{sid})@i_3 \wedge \mathsf{Match}(s,\mathrm{sid})@i_4 \wedge \exists i_5.\,\mathsf{SesskRev}(s')@i_5\right)$

- Or… long term secrets and ephemeral keys were revealed (can be formalized similarly—see (Schmidt, et al., 2012a))

Issues with Multiset Rewriting

- Incrementally constructing attacks is difficult with (action-)traces
  - No history of past states
  - No causal dependencies between steps
- Symbolic reasoning modulo an equational theory is difficult, because of cancellation equations
  - If the adversary knows $t \approx n * x$ for some nonce n, we cannot conclude that n has been used to construct t, as x could be n^{-1}
- Message deduction rules may be applied redundantly
  - Encrypt some plaintext m then decrypt m and send it, instead of just sending m

Dependency Graphs

A **dependency graph** consists of

- nodes labelled with rule instances
- edges represent the dependencies between nodes

Dependency graphs are used to represent protocol executions together with their causal dependencies

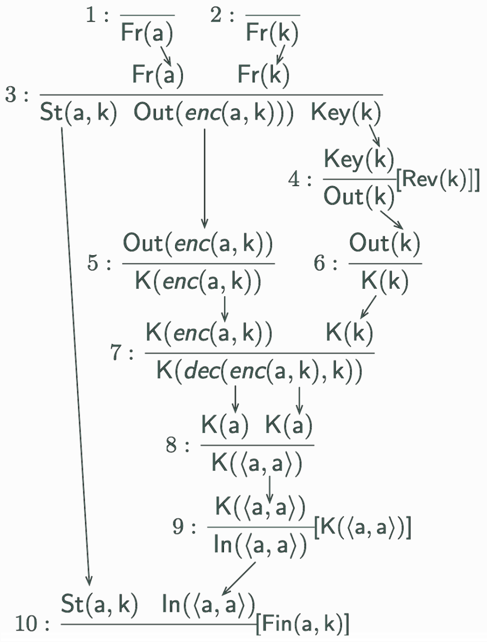
Dependency Graph: Example

$$\frac{\mathbf{Fr}(x), \mathbf{Fr}(k)}{\mathsf{St}(x,k), \mathbf{Out}(\mathsf{enc}(x,k)), \mathsf{Key}(k)}[\,]$$

$$\frac{\mathsf{St}(x,k), \mathbf{In}(\langle x, x\rangle)}{\emptyset^{\#}}[\,\mathsf{Fin}(x,k)\,]$$

$$\frac{\mathsf{Key}(k)}{\mathbf{Out}(k)}[\,\mathsf{Rev}(k)\,]$$

- Node indexes denote rule application order
- The **trace of the graph** is the trace of the execution
- An edge $(i, F) \rightarrow (j, G)$ denotes that $F =_E G$ and F is generated by i and G is consumed by j
- Other technical conditions... (Schmidt, et al., 2012a)



The Equational Theory AC

The **equational theory** AC is the theory generated by the following equations:

$x * (y * z) \simeq (x * y) * z$ **(Associativity)**

$x * y \simeq y * x$ **(Commutativity)**

The Rewriting system DH

$$(1)\quad \mathrm{dec}(\mathrm{enc}(m,k),k) \to m$$

$$(2)\quad \mathrm{fst}(\langle x,y\rangle) \to x$$

$$(3)\quad \mathrm{snd}(\langle x,y\rangle) \to y$$

$$(9)\quad (x^y)^z \to x^{y*z}$$

$$(10)\quad x^1 \to x$$

$$(a)\quad (x^{-1} * y)^{-1} \to x * y^{-1}$$

$$(b)\quad x^{-1} * y^{-1} \to (x * y)^{-1}$$

$$(c)\quad x * (x * y)^{-1} \to y^{-1}$$

$$(d)\quad x^{-1} * (y^{-1} * z) \to (x * y)^{-1} * z$$

$$(e)\quad (x * y)^{-1} * (y * z) \to x^{-1} * z$$

$$(f)\quad 1^{-1} \to 1$$

$$(g)\quad x * 1 \to x$$

$$(h)\quad (x^{-1})^{-1} \to x$$

$$(i)\quad x * (x^{-1} * y) \to y$$

$$(j)\quad x * x^{-1} \to 1$$

Dependency Graph Modulo AC

- It can be proved that any term t has a (unique) normal form $t{\downarrow_{DH}}$ with respect to AC,DH-rewriting
- In particular, $t \simeq s$ iff $t{\downarrow_{DH}} =_{AC} s{\downarrow_{DH}}$
- A dependency graph is **$\downarrow_{DH}$-normal** if all its rule instances are $\downarrow_{DH}$-normal
- Informally speaking, by reducing the rules to their AC,DH-normal form, we obtain a graph that is "equivalent" modulo AC to the original graph
- By switching to the simpler theory AC we get rid of cancellation equations

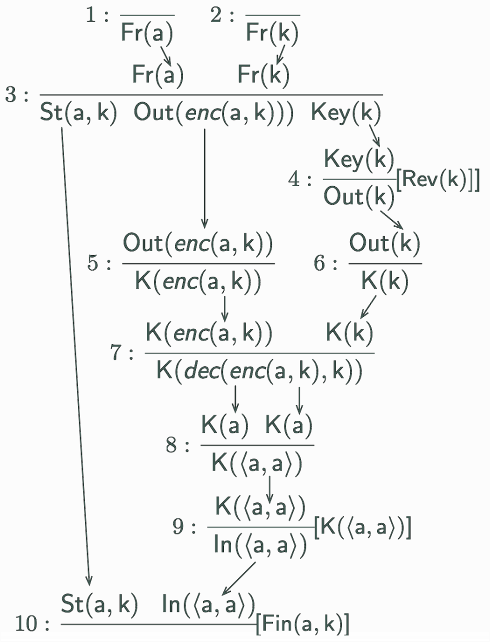
Dependency Graph Normalization: Example

To normalize the graph on the right, replace rule 7

$$\frac{\mathbf{K}(\mathrm{enc}(a,k)) \quad \mathbf{K}(k)}{\mathbf{K}(\mathrm{dec}(\mathrm{enc}(a,k),k))}$$

with

$$\frac{\mathbf{K}(\mathrm{enc}(a,k)) \qquad \mathbf{K}(k)}{\mathbf{K}(a)}$$



Star-Restricted Protocols

A protocol P is $*$-**restricted** if no rule performs multiplication of the exponents, or introduces products by other means

For every rule $L \dashv[\, A \,]\!\mapsto R$:

- L does not contain $*$, $\wedge$, $^{-1}$, fst, snd, and dec
- R does not contain $*$

Protocols that use multiplication in the group of exponents can usually be specified by using repeated exponentiation

Preventing Loops and Redundant Derivation

$$\frac{\mathsf{K}(\langle a, b\rangle)}{\mathsf{K}(a)} \rightarrow \frac{\mathsf{K}(a) \quad \mathsf{K}(c)}{\mathsf{K}(\langle a, c\rangle)} \rightarrow \frac{\mathsf{K}(\langle a, c\rangle)}{\mathsf{K}(a)} \rightarrow \frac{\mathsf{K}(a) \quad \mathsf{K}(d)}{\mathsf{K}(\langle a, d\rangle)} \rightarrow \cdots$$

- Idea: split adversary knowledge into $\mathsf{K}^{\uparrow}$ and $\mathsf{K}^{\downarrow}$
- Distinguish between **construction rules** and **deconstruction rules**
- Tag $\downarrow$ means "deconstruction allowed"
- Tag $\uparrow$ means "deconstruction forbidden"
- Using a deconstruction rule to deconstruct the result of a construction rule is forbidden

Construction and Deconstruction Rules: Example

Deconstruction rules:

$$\frac{\mathbf{K}^{\downarrow}(\langle x, y\rangle)}{\mathbf{K}^{\downarrow}(x)} \qquad \frac{\mathbf{K}^{\downarrow}(\langle x, y\rangle)}{\mathbf{K}^{\downarrow}(y)}$$

Construction rule:

$$\frac{\mathbf{K}^{\uparrow}(x) \quad \mathbf{K}^{\uparrow}(y)}{\mathbf{K}^{\uparrow}(\langle x, y\rangle)}$$

Coerce rule:

$$\frac{\mathbf{K}^{\downarrow}(x)}{\mathbf{K}^{\uparrow}(x)}$$

Now:

$$\frac{\mathbf{K}^{\downarrow}(\langle a, b\rangle)}{\mathbf{K}^{\downarrow}(a)} \to \frac{\mathbf{K}^{\downarrow}(a)}{\mathbf{K}^{\uparrow}(a)} \to \frac{\mathbf{K}^{\uparrow}(a) \quad \mathbf{K}^{\uparrow}(c)}{\mathbf{K}^{\uparrow}(\langle a, c\rangle)} \not\to \frac{\mathbf{K}^{\downarrow}(\langle a, c\rangle)}{\mathbf{K}^{\downarrow}(a)}$$

Preventing Repeated Exponentiation

$$\frac{\mathsf{K}(g^a) \quad \mathsf{K}(a^{-1} * b)}{\mathsf{K}(g^b)} \quad \rightarrow \quad \frac{\mathsf{K}(g^b) \quad \mathsf{K}(b^{-1} * c)}{\mathsf{K}(g^c)}$$

- Tags (exp/noexp) are also used to prevent repeated exponentiation, which can always be replaced by a single exponentiation with the product of all exponents
- A conclusion with a noexp-tag cannot be used with a premise that requires an exp-tag

Preventing Repeated Exponentiation: Example (1)

An exponentiation rule:

$$\frac{\mathsf{K}^{\downarrow}_{\mathsf{exp}}(x^{y}) \quad \mathsf{K}^{\uparrow}_{e}(y^{-1} * z)}{\mathsf{K}^{\downarrow}_{\mathsf{noexp}}(x^{z})}$$

Now:

$$\frac{\mathsf{K}^{\downarrow}_{\mathsf{exp}}(g^{a}) \quad \mathsf{K}^{\uparrow}_{e_1}(a^{-1} * b)}{\mathsf{K}^{\downarrow}_{\mathsf{noexp}}(g^{b})} \quad \not\rightarrow \quad \frac{\mathsf{K}^{\downarrow}_{\mathsf{exp}}(g^{b}) \quad \mathsf{K}^{\uparrow}_{e_2}(b^{-1} * c)}{\mathsf{K}^{\downarrow}_{\mathsf{noexp}}(g^{c})}$$

Preventing Repeated Exponentiation: Example (2)

What you can do instead (using suitable exponentiation rules):

$$\frac{\mathsf{K}^{\uparrow}_{e_1}(a^{-1} * b) \quad \mathsf{K}^{\uparrow}_{e_2}(b^{-1} * c)}{\mathsf{K}^{\uparrow}_{\mathsf{exp}}(a^{-1} * c)} \quad \rightarrow \quad \frac{\mathsf{K}^{\downarrow}_{\mathsf{exp}}(g^{a}) \quad \mathsf{K}^{\uparrow}_{\mathsf{exp}}(a^{-1} * c)}{\mathsf{K}^{\downarrow}_{\mathsf{noexp}}(g^{c})}$$

Normal Deduction Rules

Coerce rule: $\textsc{Coerce}\ \dfrac{\mathsf{K}_e^{\downarrow}(x)}{\mathsf{K}_e^{\uparrow}(x)}$ *Communication rules:* $\textsc{IRecv}\ \dfrac{\mathsf{Out}(x)}{\mathsf{K}_{\mathsf{exp}}^{\downarrow}(x)}$ $\textsc{ISend}\ \dfrac{\mathsf{K}_e^{\uparrow}(x)}{\mathsf{In}(x)}[\mathsf{K}(x)]$

Construction rules:

$$\frac{\mathsf{K}_{\mathsf{exp}}^{\uparrow}(x)\quad \mathsf{K}_e^{\uparrow}(y)}{\mathsf{K}_{\mathsf{noexp}}^{\uparrow}(x \,\hat{}\, y)} \qquad \frac{}{\mathsf{K}_{\mathsf{exp}}^{\uparrow}(x{:}pub)} \qquad \frac{\mathsf{Fr}(x{:}fresh)}{\mathsf{K}_{\mathsf{exp}}^{\uparrow}(x{:}fresh)} \qquad \frac{\mathsf{K}_e^{\uparrow}(x)}{\mathsf{K}_{\mathsf{exp}}^{\uparrow}(x^{-1})} \qquad \frac{}{\mathsf{K}_{\mathsf{exp}}^{\uparrow}(1)} \qquad \frac{\mathsf{K}_{e_1}^{\uparrow}(x)\quad \mathsf{K}_{e_2}^{\uparrow}(y)}{\mathsf{K}_{\mathsf{exp}}^{\uparrow}(\mathrm{enc}(x,y))} \qquad \frac{\mathsf{K}_{e_1}^{\uparrow}(x)\quad \mathsf{K}_{e_2}^{\uparrow}(y)}{\mathsf{K}_{\mathsf{exp}}^{\uparrow}(\mathrm{dec}(x,y))}$$

$$\frac{\mathsf{K}_e^{\uparrow}(x)}{\mathsf{K}_{\mathsf{exp}}^{\uparrow}(\mathrm{h}(x))} \qquad \frac{\mathsf{K}_e^{\uparrow}(x)}{\mathsf{K}_{\mathsf{exp}}^{\uparrow}(\mathrm{fst}(x))} \qquad \frac{\mathsf{K}_e^{\uparrow}(x)}{\mathsf{K}_{\mathsf{exp}}^{\uparrow}(\mathrm{snd}(x))} \qquad \frac{\mathsf{K}_{e_1}^{\uparrow}(x)\quad \mathsf{K}_{e_2}^{\uparrow}(y)}{\mathsf{K}_{\mathsf{exp}}^{\uparrow}(\langle x,y\rangle)} \qquad \frac{\mathsf{K}_{e_1}^{\uparrow}(x_1)\ \ldots\ \mathsf{K}_{e_n}^{\uparrow}(x_n) \qquad \mathsf{K}_{e_{n+1}}^{\uparrow}(x_{n+1})\ \ldots\ \mathsf{K}_{e_l}^{\uparrow}(x_l)}{\mathsf{K}_{\mathsf{exp}}^{\uparrow}((x_1 * \ldots * x_n) * (x_{n+1} * \ldots * x_l)^{-1})}$$

Deconstruction rules:

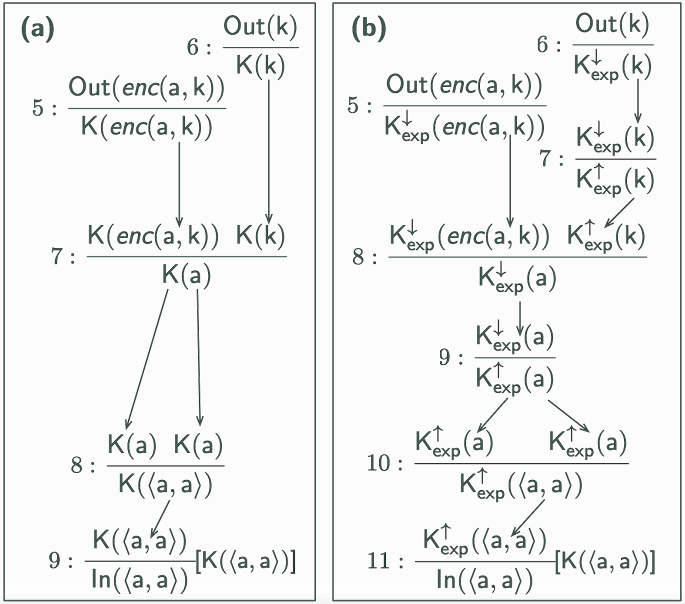
$$\frac{\mathsf{K}_{\mathsf{exp}}^{\downarrow}(x \,\hat{}\, y)\quad \mathsf{K}_e^{\uparrow}(y^{-1})}{\mathsf{K}_{\mathsf{noexp}}^{\downarrow}(x)} \qquad \frac{\mathsf{K}_{\mathsf{exp}}^{\downarrow}(x \,\hat{}\, y^{-1})\quad \mathsf{K}_e^{\uparrow}(y)}{\mathsf{K}_{\mathsf{noexp}}^{\downarrow}(x)} \qquad \frac{\mathsf{K}_{\mathsf{exp}}^{\downarrow}(x \,\hat{}\, (y * z^{-1}))\quad \mathsf{K}_e^{\uparrow}(y^{-1} * z)}{\mathsf{K}_{\mathsf{noexp}}^{\downarrow}(x)}$$

$$\frac{\mathsf{K}_e^{\downarrow}(\langle x,y\rangle)}{\mathsf{K}_{\mathsf{exp}}^{\downarrow}(x)} \qquad \frac{\mathsf{K}_e^{\downarrow}(\langle x,y\rangle)}{\mathsf{K}_{\mathsf{exp}}^{\downarrow}(y)} \qquad \frac{\mathsf{K}_e^{\downarrow}(x^{-1})}{\mathsf{K}_{\mathsf{exp}}^{\downarrow}(x)} \qquad \frac{\mathsf{K}_{e_1}^{\downarrow}(\mathrm{enc}(x,y))\quad \mathsf{K}_{e_2}^{\uparrow}(y)}{\mathsf{K}_{\mathsf{exp}}^{\downarrow}(x)}$$

Exponentiation rules:

$$\frac{\mathsf{K}_{\mathsf{exp}}^{\downarrow}(x \,\hat{}\, y)\quad \mathsf{K}_e^{\uparrow}(z)}{\mathsf{K}_{\mathsf{noexp}}^{\downarrow}(x \,\hat{}\, (y * z))} \qquad \frac{\mathsf{K}_{\mathsf{exp}}^{\downarrow}(x \,\hat{}\, y)\quad \mathsf{K}_e^{\uparrow}(y^{-1} * z)}{\mathsf{K}_{\mathsf{noexp}}^{\downarrow}(x \,\hat{}\, z)} \qquad \ldots \qquad \frac{\mathsf{K}_{\mathsf{exp}}^{\downarrow}(x \,\hat{}\, (y * z^{-1}))\quad \mathsf{K}_e^{\uparrow}(a * b^{-1})}{\mathsf{K}_{\mathsf{noexp}}^{\downarrow}(x \,\hat{}\, (y * a * (z * b)^{-1}))}$$

Normal Dependency Graph



a. Dependency graph modulo *AC*

b. Normal dependency graph

Lemma. For a large class of protocols *P* ("∗-restricted"), the set of normalized traces of executions of *P* and the set of traces of the normal dependency graph of *P* are equal modulo *AC*

Verification Strategy at 10,000 Feet

Backward reachability analysis –searching for insecure states

- Negate security property, search for *solutions*

Constraint solving

- Constraint systems are used to represent the intermediate states of the search
- Dependency graphs denote the solutions of the constraint systems
- Uses normal dependency graphs for state-space reduction
- Efficient in practice, despite undecidability

Tamarin's Constraint Solving Procedure

```
function Solve(P ⊨_{E_DH} φ)
  φ̂ ← ¬φ rewritten into negation normal form
  Ω ← {{φ̂}}
  while Ω ≠ ∅ and solved(Ω) = ∅ do
    choose Γ ⇝_P {Γ_1, ..., Γ_k} such that Γ ∈ Ω
    Ω ← (Ω \ {Γ}) ∪ {Γ_1, ..., Γ_k}
  if solved(Ω) ≠ ∅
    then return "attack(s) found: ", solved(Ω)
    else return "verification successful"
```

Guarded Trace Formulas

- In negation normal form
  - Negation is only applied to trace atoms
  - all logical connectives are $\wedge$, $\vee$, $\forall$, $\exists$
- All quantifiers are of the form:

  $$\exists \vec{x}. (F@i) \wedge \psi \quad \text{or} \quad \forall \vec{x}. \neg(F@i) \vee \psi$$

  where ψ is guarded and all $\vec{x}$ appear in $F@i$
- Terms can only be built out of the quantified variables and public names
- A **guarded trace property** is a closed guarded trace formula

Guarded Trace Properties

- The set of guarded trace properties is closed under negation
- They support quantifier alternation and comparison of time points
- Guarded trace properties are invariant under $\downarrow_{DH}$-normalization of traces
- $\Rightarrow$ Verification if multiset rewriting semantics modulo E_{DH} $\equiv$ verification in a dependency graph semantics modulo AC

Theorem. For every ($*$-restricted) protocol P and every guarded trace property φ:

$$P \vDash_{E_{DH}} \varphi \quad \text{iff} \quad \{\, traces(G) \mid G \text{ is a normal d.g. for } P \,\} \vDash_{AC} \varphi$$

Guarded Trace Properties: Example

"Fresh values (nonces) are all distinct"

$$\forall n : \mathit{fresh}, i, j.\, \mathsf{Act}(n)@i \land \mathsf{Act}(n)@j \rightarrow i = j$$

The above is equivalent to a guarded formula, which can be obtained by pushing quantifiers and negation inwards as far as possible:

$$\forall n, i, j.\, \neg\mathsf{Act}(n)@i \lor \neg\mathsf{Act}(n)@j \lor i = j$$

$$\forall n, i.\, \neg\mathsf{Act}(n)@i \lor \left(\forall j(\neg\mathsf{Act}(n)@j \lor i = j)\right)$$

This property is trivially valid, given the definition of traces

How to Ensure Guardedness in Tamarin

For universally quantified variables:

- they must occur in an action atom right after the quantifier
- the outermost logical operator inside the quantifier is an implication

For existentially quantified variables:

- they all occur in an action atom right after the quantifier
- the outermost logical operator inside the quantifier is a conjunction

Constraints

Graph constraints

- A node $i : r$ (where r is a rule instance with index i)
- An edge $(i, F) \rightarrow (j, G)$
- A “deconstruction chain”
- An “implicit construction”

A **constraint** is either a graph constraint or a guarded trace formula

- Constraints are evaluated with respect to a d.g. (and a valuation)
- **Constraint system:** set of constraints

The Constraint Reduction Relation $\rightsquigarrow_P$

- A normal dependency graph for a protocol P with a valuation that satisfies each constraint of a constraint system is called a ***P*-solution**
- So, to find a counterexample for a guarded trace property φ, one tries to find a P-solution to $\{\hat{\varphi}\}$ (i.e., $\neg\varphi$ in negated normal form)
- Intuitively, $\rightsquigarrow_P$ is used to refine the initial constraint system $\{\hat{\varphi}\}$ until it either encounters a solved system or all systems contain (trivially) contradictory constraints
- There are 27 reduction rules for $\rightsquigarrow_P$
- A solved constraint system is one that is irreducible w.r.t. $\rightsquigarrow_P$

Trace Formula Reduction Rules

$\mathbf{S}_{\approx}$:	$\Gamma \rightsquigarrow_P \|_{\sigma \in unify_{AC}(t_1,t_2)}(\Gamma\sigma)$	if $(t_1 \approx t_2) \in \Gamma$ and $t_1 \neq_{AC} t_2$
$\mathbf{S}_{\doteq}$:	$\Gamma \rightsquigarrow_P \Gamma\{i/j\}$	if $(i \doteq j) \in \Gamma$ and $i \neq j$
$\mathbf{S}_{@}$:	$\Gamma \rightsquigarrow_P \|_{ri \in [P]^{DH} \cup \{\text{ISEND}\}} \|_{f' \in acts(ri)}(i : ri\ ,\ f \approx f', \Gamma)$	if $(f@i) \in \Gamma$ and $(f@i) \notin_{AC} as(\Gamma)$
$\mathbf{S}_{\perp}$:	$\Gamma \rightsquigarrow_P \perp$	if $\perp \in \Gamma$
$\mathbf{S}_{\neg,\approx}$:	$\Gamma \rightsquigarrow_P \perp$	if $\neg(t \approx t) \in_{AC} \Gamma$
$\mathbf{S}_{\neg,\doteq}$:	$\Gamma \rightsquigarrow_P \perp$	if $\neg(i \doteq i) \in \Gamma$
$\mathbf{S}_{\neg,@}$:	$\Gamma \rightsquigarrow_P \perp$	if $\neg(f@i) \in \Gamma$ and $(f@i) \in as(\Gamma)$
$\mathbf{S}_{\neg,\lessdot}$:	$\Gamma \rightsquigarrow_P (i \lessdot j, \Gamma) \parallel (\Gamma\{i/j\})$	if $\neg(j \lessdot i) \in \Gamma$ and neither $i \lessdot_\Gamma j$ nor $i = j$
$\mathbf{S}_{\vee}$:	$\Gamma \rightsquigarrow_P (\phi_1, \Gamma) \parallel (\phi_2, \Gamma)$	if $(\phi_1 \vee \phi_2) \in_{AC} \Gamma$ and $\{\phi_1, \phi_2\} \cap_{AC} \Gamma = \varnothing$
$\mathbf{S}_{\wedge}$:	$\Gamma \rightsquigarrow_P (\phi_1, \phi_2, \Gamma)$	if $(\phi_1 \wedge \phi_2) \in_{AC} \Gamma$ and not $\{\phi_1, \phi_2\} \subseteq_{AC} \Gamma$
$\mathbf{S}_{\exists}$:	$\Gamma \rightsquigarrow_P (\phi\{y/x\}, \Gamma)$	if $(\exists x{:}s.\ \phi) \in \Gamma$, $\phi\{w/x\} \notin_{AC} \Gamma$ for every term w of sort s, and $y{:}s$ fresh
$\mathbf{S}_{\forall}$:	$\Gamma \rightsquigarrow_P (\psi\sigma, \Gamma)$	if $(\forall \vec{x}.\ \neg(f@i) \vee \psi) \in \Gamma$, $dom(\sigma) = set(\vec{x})$, $(f@i)\sigma \in_{AC} as(\Gamma)$, and $\psi\sigma \notin_{AC} \Gamma$

Graph Constraint Reduction Rules

U$_{lbl}$: $\Gamma \rightsquigarrow_P (ri \approx ri', \Gamma)$ if $\{i : ri, i : ri'\} \subseteq \Gamma$ and $ri \neq_{AC} ri'$

DG1$_1$: $\Gamma \rightsquigarrow_P \bot$ if $i \lessdot_\Gamma i$

DG1$_2$: $\Gamma \rightsquigarrow_P (f \approx f', \Gamma)$ if $c \rightarrowtail p \in \Gamma$, $(c, f) \in cs(\Gamma)$, $(p, f') \in ps(\Gamma)$, and $f \neq_{AC} f'$

DG2$_1$: $\Gamma \rightsquigarrow_P (\text{if } u = v \text{ then } \Gamma\{i/j\} \text{ else } \bot)$ if $\{(i, v) \rightarrowtail p, (j, u) \rightarrowtail p\} \subseteq \Gamma$ and $i \neq j$

DG2$_{2,P}$: $\Gamma \rightsquigarrow_P \|_{ri \in \lceil P \rceil^{DH} \cup \{\textsc{Isend}, \textsc{Fresh}\}} \|_{u \in idx(concs(ri))} (i : ri, (i, u) \rightarrowtail p, \Gamma)$
if p is an open f-premise in Γ, f is not a $\mathsf{K}^{\uparrow}$- or $\mathsf{K}^{\downarrow}$-fact, and i fresh

DG3 : $\Gamma \rightsquigarrow_P (\text{if } u = v \text{ then } \Gamma\{i/j\} \text{ else } \bot)$ if $\{c \rightarrowtail (i, v), c \rightarrowtail (j, u)\} \subseteq \Gamma$, c linear in Γ, and $i \neq j$,

DG4 : $\Gamma \rightsquigarrow_P \Gamma\{i/j\}$ if $\{i : \text{–}[]\!\rightarrow \mathsf{Fr}(m), j : \text{–}[]\!\rightarrow \mathsf{Fr}(m)\} \subseteq_{AC} \Gamma$ and $i \neq j$

N1 : $\Gamma \rightsquigarrow_P \bot$ if $(i : ri) \in \Gamma$ and ri not $\downarrow_{DH}$-normal

N5,6 : $\Gamma \rightsquigarrow_P \Gamma\{i/j\}$ if $\{((i,1), \mathsf{K}_e^d(t)), ((j,1), \mathsf{K}_{e'}^{d'}(t))\} \subseteq_{AC} cs(\Gamma)$, $i \neq j$, *and*
$d = d'$ or $\{i, j\} \cap \{k \mid \exists ri \in insts(\{\textsc{pair}\uparrow, \textsc{inv}\uparrow, \textsc{Coerce}\}). (k : ri) \in \Gamma\} = \emptyset$

N6 : $\Gamma \rightsquigarrow_P (i \lessdot j, \Gamma)$ if $((j, v), \mathsf{K}_{e'}^{\uparrow}(t)) \in ps(\Gamma)$, $m \in_{AC} inp(t)$, $((i, u), \mathsf{K}_e^{\downarrow}(m)) \in cs(\Gamma)$, and not $i \lessdot_\Gamma j$

N7 : $\Gamma \rightsquigarrow_P \bot$ if $(i : \mathsf{K}_{\mathsf{exp}}^{\downarrow}(s_1), \mathsf{K}_e^{\uparrow}(t_1) \text{–}[]\!\rightarrow \mathsf{K}_{\mathsf{noexp}}^{\downarrow}(s_2 \,\hat{}\, t_2)) \in \Gamma$, s_2 is of sort *pub*, and $inp(t_2) \subseteq inp(t_1)$

Message Deduction Constraint Reduction Rules

$\mathbf{DG2}_{2,\uparrow i}$: $\Gamma \rightsquigarrow_P \|_{(l \text{-}[]\rightarrow \mathsf{K}_e^{\uparrow}(t)) \in ND^{c\text{-}expl}} (i : (l \text{-}[]\rightarrow \mathsf{K}_e^{\uparrow}(t)),\ t \approx m,\ (i,1) \twoheadrightarrow p,\ \Gamma)$
if p is an open implicit m-construction in Γ, m non-trivial, and i fresh

$\mathbf{DG2}_{2,\uparrow e}$: $\Gamma \rightsquigarrow_P \|_{ri \in ND^{c\text{-}expl}} (i : ri,\ (i,1) \rightarrowtail p,\ \Gamma)$
if p is an open $\mathsf{K}_e^{\uparrow}(m)$-premise in Γ, $\{m\} = inp(m)$, m non-trivial, and i fresh

$\mathbf{DG2}_{2,\downarrow}$: $\Gamma \rightsquigarrow_P (i : \mathsf{Out}(y) \text{-}[]\rightarrow \mathsf{K}_{\mathsf{exp}}^{\downarrow}(y),\ (i,1) \dashrightarrow p,\ \Gamma)$ if p is an open $\mathsf{K}_e^{\downarrow}(m)$-premise in Γ and y, i fresh

$\mathbf{DG2}_{\dashrightarrow}$: $(c \dashrightarrow p, \Gamma) \rightsquigarrow_P (c \rightarrowtail p, \Gamma) \parallel \|_{ri \in ND^{destr}} (i : ri,\ c \rightarrowtail (i,1),\ (i,1) \dashrightarrow p,\ \Gamma)$
if $(c, \mathsf{K}_e^{\downarrow}(m)) \in cs(\Gamma)$, $m \notin \mathcal{V}_{msg}$, and i fresh

Properties of $\rightsquigarrow_P$

Theorem. The constraint-reduction relation $\rightsquigarrow_P$ is sound and complete; i.e., for every $\Gamma \rightsquigarrow_P \{\Gamma_1, ..., \Gamma_n\}$, the set of P-solutions of Γ is equal to the union of the sets of P-solutions of all Γ_i

Theorem. A P-solution can be constructed from every solved system in the state Ω

Intuition for Backward Reachability (1)

$$\frac{\mathbf{Fr}(x), \mathbf{Fr}(k)}{\mathsf{St}(x,k), \mathbf{Out}(\mathsf{enc}(x,k)), \mathsf{Key}(k)}[\,] \quad \frac{\mathsf{St}(x,k), \mathbf{In}(\langle x, x\rangle)}{\emptyset^\#}[\,\mathsf{Fin}(x,k)\,] \quad \frac{\mathsf{Key}(k)}{\mathbf{Out}(k)}[\,\mathsf{Rev}(k)\,]$$

- We want to prove the unreachability of a Rev-action
- Formally: $\varphi = \forall k \forall i.\, \neg \mathsf{Rev}(k)@i$
- We do so by solving a constraint system $\Gamma_0 = \{\exists k \exists i.\, \mathsf{Rev}(k)@i\}$
- To solve the constraint system, we apply some transformations
- First, note that Γ_0 has the same solutions as $\{\mathsf{Rev}(k)@i\}$, because the free variables of a constraint system are existentially quantified

Intuition for Backward Reachability (2)

$$\frac{\mathbf{Fr}(x), \mathbf{Fr}(k)}{\mathsf{St}(x,k), \mathbf{Out}(\mathsf{enc}(x,k)), \mathsf{Key}(k)}[\,] \quad \frac{\mathsf{St}(x,k), \mathbf{In}(\langle x, x\rangle)}{\emptyset^{\#}}[\,\mathsf{Fin}(x,k)\,] \quad \frac{\mathsf{Key}(k)}{\mathbf{Out}(k)}[\,\mathsf{Rev}(k)\,]$$

- As there is only one rule in whose instances have a Rev-action, the solutions of $\{\mathsf{Rev}(k)@i\}$ are therefore equal to the solutions of

  $$\Gamma_1 = \left\{ i : \frac{\mathsf{Key}(k)}{\mathbf{Out}(k)}[\,\mathsf{Rev}(k)\,] \right\}$$

  I.e., the dependency graph must contain the above node
- In all solutions of Γ_1, the Key-premise must have an incoming edge from a Key-conclusion

Intuition for Backward Reachability (3)

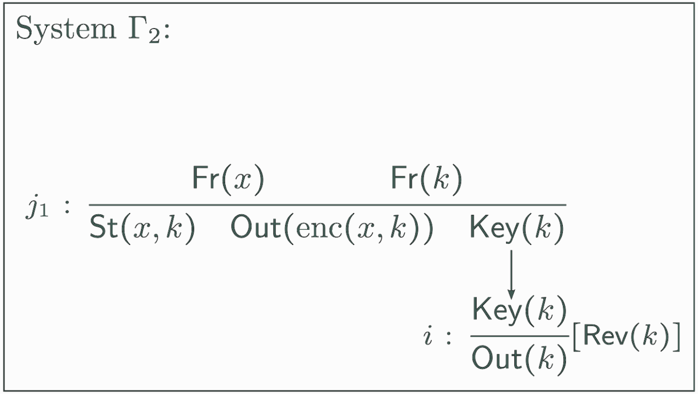
$$\frac{\mathbf{Fr}(x), \mathbf{Fr}(k)}{\mathsf{St}(x,k), \mathbf{Out}(\mathsf{enc}(x,k)), \mathsf{Key}(k)} \quad \frac{\mathsf{St}(x,k), \mathbf{In}(\langle x,x\rangle)}{\emptyset^\#}[\,\mathsf{Fin}(x,k)\,] \quad \frac{\mathsf{Key}(k)}{\mathbf{Out}(k)}[\,\mathsf{Rev}(k)\,]$$

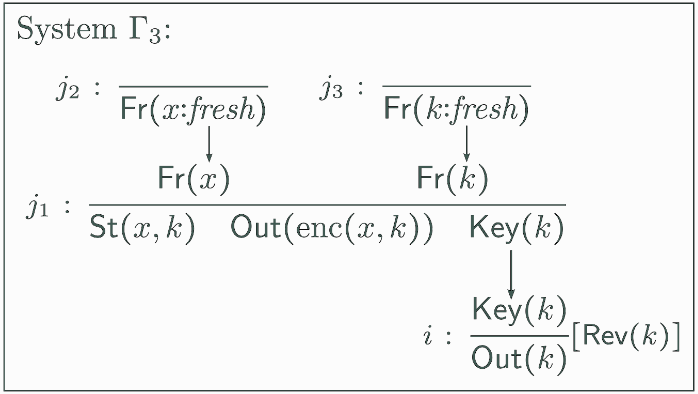
- As there is only one rule in whose instances have a Key-conclusion, the solutions of Γ_1 are therefore equal to the solutions of

$$\Gamma_2 = \left\{ i : \frac{\mathsf{Key}(k)}{\mathbf{Out}(k)}[\,\mathsf{Rev}(k)\,],\ j_1 : \frac{\mathbf{Fr}(x), \mathbf{Fr}(k)}{\mathsf{St}(x,k), \mathbf{Out}(\mathsf{enc}(x,k)), \mathsf{Key}(k)},\ (j_1, 3) \rightarrow (i, 1) \right\}$$

  I.e., the dependency graph must contain the two nodes above, connected by the specified edge

Intuition for Backward Reachability (4)





Γ_3 is the solved constraint system, and a counterexample to φ

Constraint Solving: Example (1) (Meier, 2013)

Same protocol as before:

$$\frac{\mathbf{Fr}(x), \mathbf{Fr}(k)}{\mathsf{St}(x,k), \mathbf{Out}(\mathsf{enc}(x,k)), \mathsf{Key}(k)}[\,] \quad \frac{\mathsf{St}(x,k), \mathbf{In}(\langle x, x\rangle)}{\emptyset^{\#}}[\,\mathsf{Fin}(x,k)\,] \quad \frac{\mathsf{Key}(k)}{\mathbf{Out}(k)}[\,\mathsf{Rev}(k)\,]$$

We want to prove:

$$\varphi = \forall x_1\, x_2\, k\, i_1\, i_2 \mathsf{Fin}(x_1, k)@i_1 \wedge \mathsf{Fin}(x_2, k)@i_2 \rightarrow (i_1 \doteq i_2) \wedge (x_1 \simeq x_2)$$

Constraint Solving: Example (2)

φ holds iff $\{\hat{\varphi}\}$ has no solutions, where:

$\hat{\varphi} = \exists x_1\, k\, i_1. \mathsf{Fin}(x_1, k)@i_1 \wedge (\exists x_2\, i_2. \mathsf{Fin}(x_2, k)@i_2 \wedge (\neg(i_1 \doteq i_2) \vee \neg(x_1 \simeq x_2)))$

We start by applying $\mathsf{S}_\exists$, $\mathsf{S}_\wedge$, $\mathsf{S}_\exists$, $\mathsf{S}_\wedge$ to $\{\hat{\varphi}\}$, in this order, which results in a new constraint system:

$$\begin{aligned}
\Gamma := \{\, & \exists x_1\, k\, i_1.\, \mathsf{Fin}(x_1, k)@i_1 \wedge (\exists x_2\, i_2.\, \mathsf{Fin}(x_2, k)@i_2 \wedge (\neg(i_1 \doteq i_2) \vee \neg(x_1 \approx x_2))) \\
&, \mathsf{Fin}(x_1, k)@i_1 \wedge (\exists x_2\, i_2.\, \mathsf{Fin}(x_2, k)@i_2 \wedge (\neg(i_1 \doteq i_2) \vee \neg(x_1 \approx x_2))) \\
&, \boxed{\mathsf{Fin}(x_1, k)@i_1}, (\exists x_2\, i_2.\, \mathsf{Fin}(x_2, k)@i_2 \wedge (\neg(i_1 \doteq i_2) \vee \neg(x_1 \approx x_2))) \\
&, \mathsf{Fin}(x_2, k)@i_2 \wedge (\neg(i_1 \doteq i_2) \vee \neg(x_1 \approx x_2)) \\
&, \boxed{\mathsf{Fin}(x_2, k)@i_2}, \boxed{\neg(i_1 \doteq i_2) \vee \neg(x_1 \approx x_2)} \,\}\,.
\end{aligned}$$

Constraint Solving: Example (3)

All constraints in Γ except for the greyed ones are solved (no other rule applies to them)

We continue by solving $\mathsf{Fin}(x_1, k)@i_1$ using rule $\mathsf{S}_@$, which produces:

$$\Gamma_1 = \Gamma \cup \left\{ i_1 : \frac{\mathsf{St}(x', k'), \mathbf{In}(\langle x', x' \rangle)}{\emptyset^\#}[\,\mathsf{Fin}(x', k')\,], \mathsf{Fin}(x_1, k) \simeq \mathsf{Fin}(x', k') \right\}$$

and

$$\Gamma_2 = \Gamma \cup \left\{ i_1 : \frac{\mathsf{Key}(k')}{\mathbf{Out}(k')}[\,\mathsf{Rev}(k')\,], \mathsf{Fin}(x_1, k) \simeq \mathsf{Rev}(k') \right\}$$

Γ_2 reduces to $\perp$ because the terms in the equality cannot be unified

Constraint Solving: Example (4)

We proceed by solving $\mathsf{Fin}(x_1, k) \simeq \mathsf{Fin}(x', k')$ with rule $\mathsf{S}_{\simeq}$, which results in:

$$\Gamma_{12} = \Gamma_1 \cup \left\{ i_1 : \frac{\mathsf{St}(x_1, k), \mathbf{In}(\langle x_1, x_1 \rangle)}{\emptyset^{\#}} [\,\mathsf{Fin}(x_1, k)\,], \mathsf{Fin}(x_1, k) \simeq \mathsf{Fin}(x_1, k) \right\}$$

Below, the gray element is the one chosen for the next reduction, and only the new formulas at each step are shown

System Γ_{12}:

$$i_1 : \frac{\mathsf{St}(x_1, k) \quad \mathsf{In}(\langle x_1, x_1 \rangle)}{}[\mathsf{Fin}(x_1, k)] \qquad \mathsf{Fin}(x_2, k) @ i_2$$

System Γ_{13}:

$$j_1 : \frac{\mathsf{Fr}(x_1) \quad \mathsf{Fr}(k)}{\mathsf{St}(x_1, k) \quad \mathsf{Out}(\mathrm{enc}(x_1, k)) \quad \mathsf{Key}(k)}$$

$$i_1 : \frac{\mathsf{St}(x_1, k) \quad \mathsf{In}(\langle x_1, x_1 \rangle)}{}[\mathsf{Fin}(x_1, k)] \qquad \mathsf{Fin}(x_2, k) @ i_2$$

Constraint Solving: Example (5)

System Γ_{13}:

$$j_1 : \frac{\mathsf{Fr}(x_1) \qquad \mathsf{Fr}(k)}{\mathsf{St}(x_1, k) \quad \mathsf{Out}(\mathrm{enc}(x_1, k)) \quad \mathsf{Key}(k)}$$

$$\downarrow$$

$$i_1 : \frac{\mathsf{St}(x_1, k) \quad \mathsf{In}(\langle x_1, x_1 \rangle)}{}[\mathsf{Fin}(x_1, k)]$$

$\mathsf{Fin}(x_2, k) @ i_2$

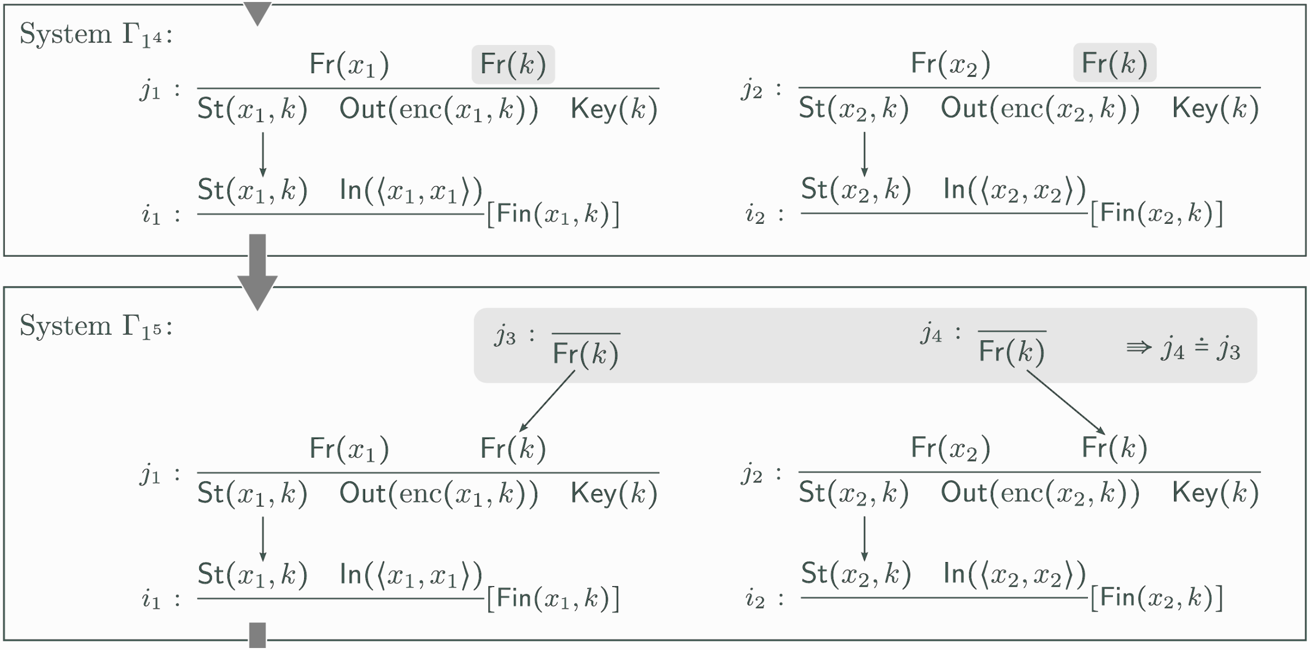
System Γ_{14}:

$$j_1 : \frac{\mathsf{Fr}(x_1) \qquad \mathsf{Fr}(k)}{\mathsf{St}(x_1, k) \quad \mathsf{Out}(\mathrm{enc}(x_1, k)) \quad \mathsf{Key}(k)} \qquad j_2 : \frac{\mathsf{Fr}(x_2) \qquad \mathsf{Fr}(k)}{\mathsf{St}(x_2, k) \quad \mathsf{Out}(\mathrm{enc}(x_2, k)) \quad \mathsf{Key}(k)}$$

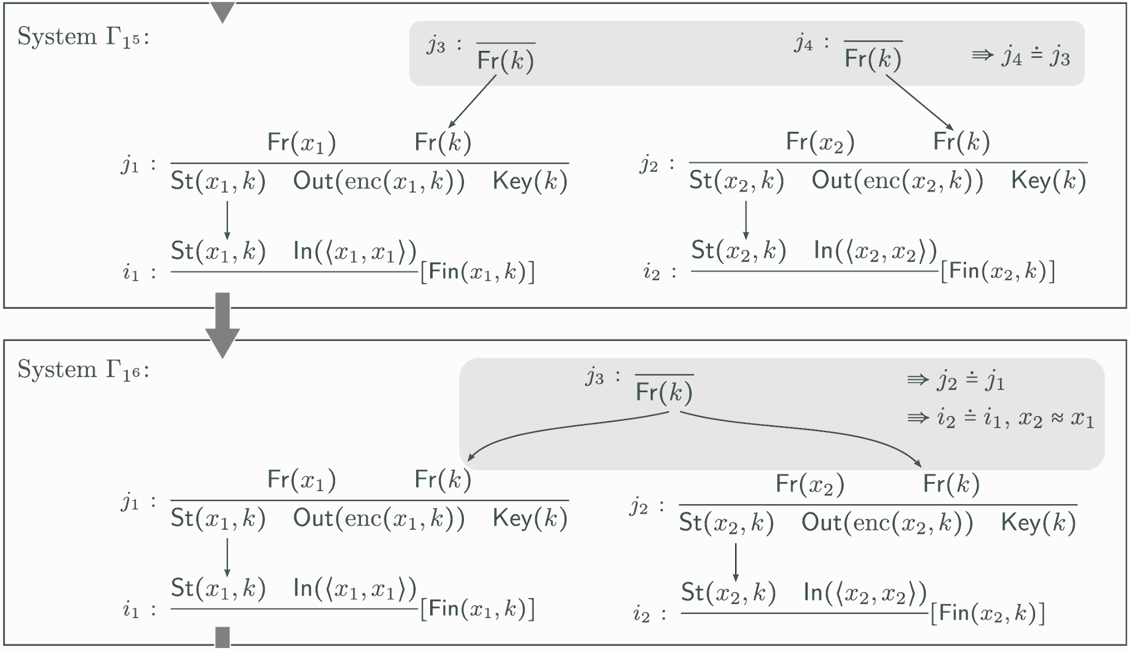
$$\downarrow \qquad\qquad\qquad\qquad \downarrow$$

$$i_1 : \frac{\mathsf{St}(x_1, k) \quad \mathsf{In}(\langle x_1, x_1 \rangle)}{}[\mathsf{Fin}(x_1, k)] \qquad i_2 : \frac{\mathsf{St}(x_2, k) \quad \mathsf{In}(\langle x_2, x_2 \rangle)}{}[\mathsf{Fin}(x_2, k)]$$

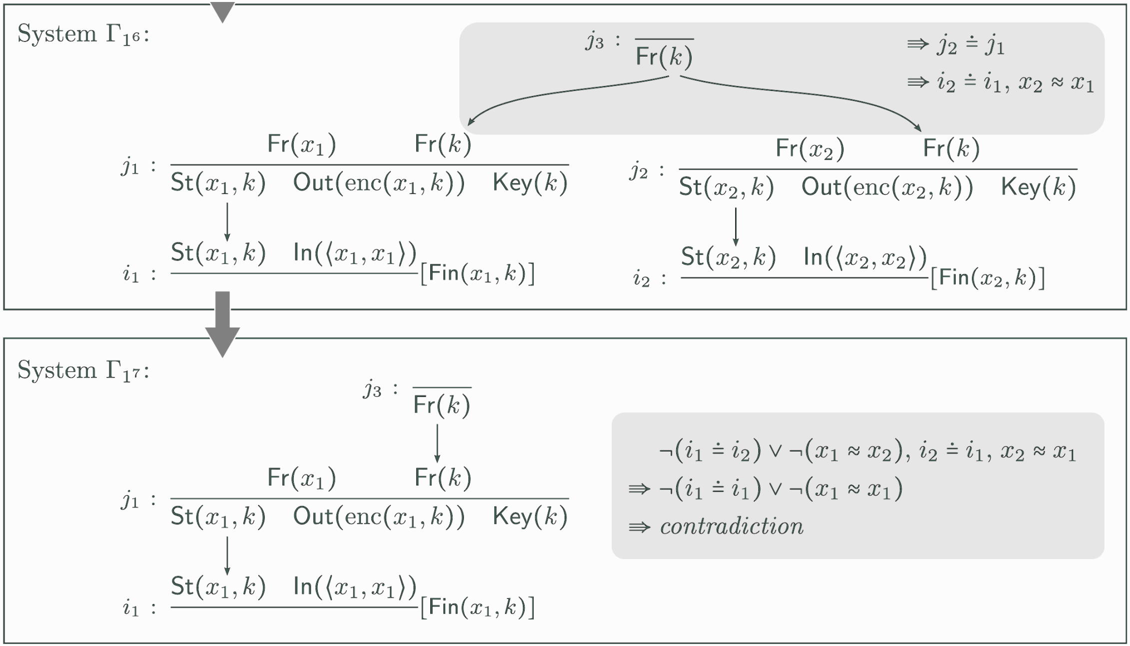
Constraint Solving: Example (6)



Constraint Solving: Example (7)



Constraint Solving: Example (8)



References

Abadi, M. & Needham, R. (1996). Prudent Engineering Practice for Cryptographic Protocols. *IEEE Transactions on Software Engineering*, *22*(1), 6–15.

Dolev, D. & Yao, A. (1983). On the Security of Public Key Protocols. *IEEE Transactions on Information Theory*, *29*(2), 198–208.

Even, S. & Goldreich, O. (1983). On the Security of Multi-Party Ping-Pong Protocols. In On the Security of Multi-Party Ping-Pong Protocols., *24th Annual Symposium on Foundations of Computer Science (SFCS 1983)*. IEEE, Author.

Meier, S. (2013). *Advancing automated security protocol verification.* (PhD thesis). ETH Zurich.

Schmidt, B., … (2012a). Automated Analysis of Diffie-Hellman Protocols and Advanced Security Properties. In Automated Analysis of Diffie-Hellman Protocols and Advanced Security Properties., *2012 IEEE 25th Computer Security Foundations Symposium*. Author.

Schmidt, B. (2012b). *Formal analysis of key exchange protocols and physical protocols.* (PhD thesis). ETH Zurich.