Tamarin prover

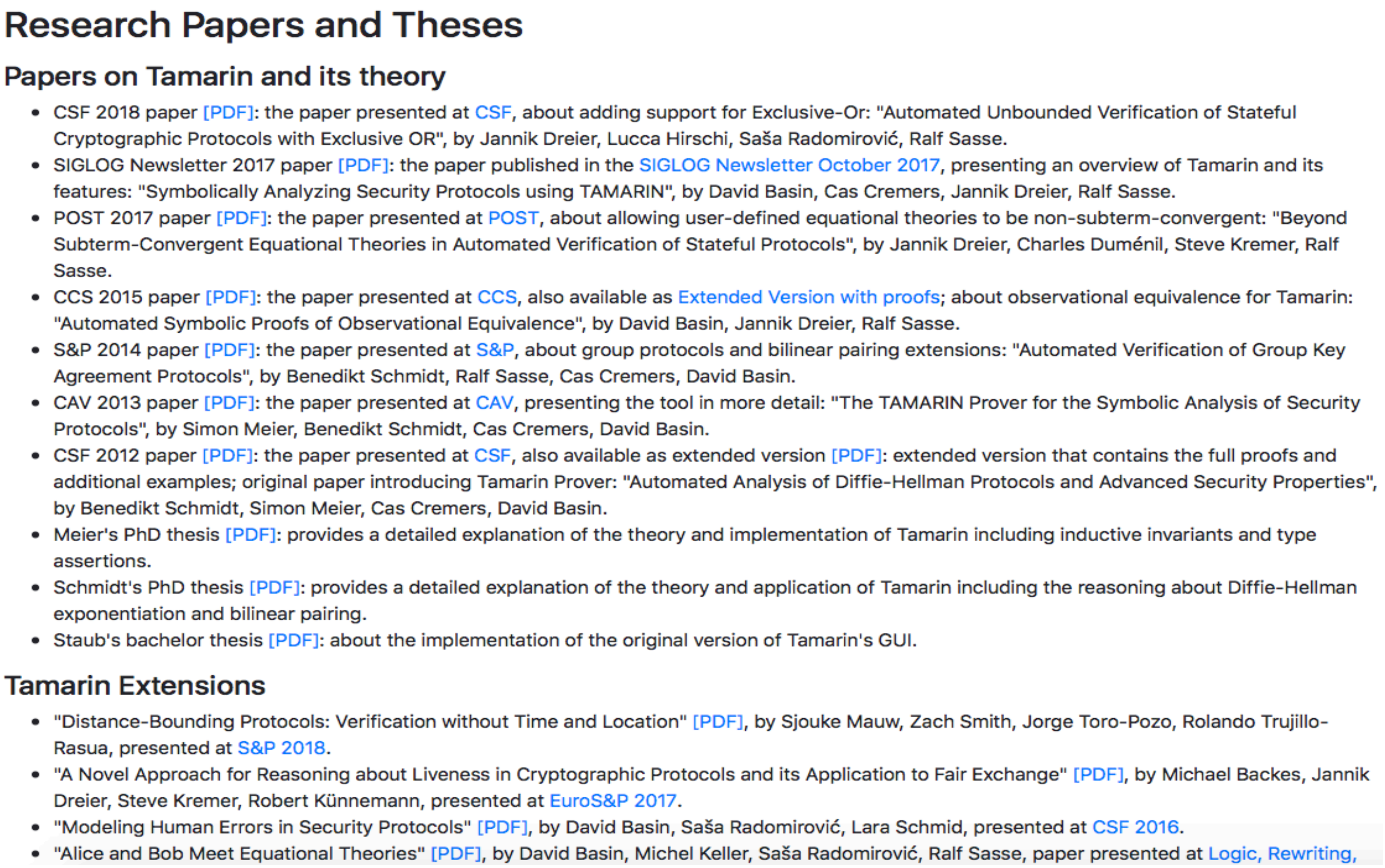
Farzane Karami

November 2019

Tamarin

- A tool for modeling and analysis of security protocols
- Core team:
  - David Basin, Cas Cremers, Jannik Dreier, Simon Meier, Ralf Sasse, Benedikt Schmidt
- https://tamarin-prover.github.io/manual/tex/tamarin-manual.pdf

Tamarin



Research Papers and Theses

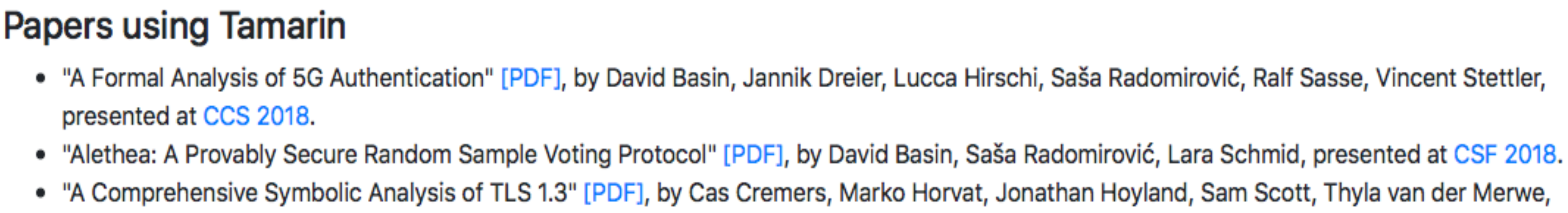
Papers on Tamarin and its theory

- CSF 2018 paper [PDF]: the paper presented at CSF, about adding support for Exclusive-Or: "Automated Unbounded Verification of Stateful Cryptographic Protocols with Exclusive OR", by Jannik Dreier, Lucca Hirschi, Saša Radomirović, Ralf Sasse.
- SIGLOG Newsletter 2017 paper [PDF]: the paper published in the SIGLOG Newsletter October 2017, presenting an overview of Tamarin and its features: "Symbolically Analyzing Security Protocols using TAMARIN", by David Basin, Cas Cremers, Jannik Dreier, Ralf Sasse.
- POST 2017 paper [PDF]: the paper presented at POST, about allowing user-defined equational theories to be non-subterm-convergent: "Beyond Subterm-Convergent Equational Theories in Automated Verification of Stateful Protocols", by Jannik Dreier, Charles Duménil, Steve Kremer, Ralf Sasse.
- CCS 2015 paper [PDF]: the paper presented at CCS, also available as Extended Version with proofs; about observational equivalence for Tamarin: "Automated Symbolic Proofs of Observational Equivalence", by David Basin, Jannik Dreier, Ralf Sasse.
- S&P 2014 paper [PDF]: the paper presented at S&P, about group protocols and bilinear pairing extensions: "Automated Verification of Group Key Agreement Protocols", by Benedikt Schmidt, Ralf Sasse, Cas Cremers, David Basin.
- CAV 2013 paper [PDF]: the paper presented at CAV, presenting the tool in more detail: "The TAMARIN Prover for the Symbolic Analysis of Security Protocols", by Simon Meier, Benedikt Schmidt, Cas Cremers, David Basin.
- CSF 2012 paper [PDF]: the paper presented at CSF, also available as extended version [PDF]: extended version that contains the full proofs and additional examples; original paper introducing Tamarin Prover: "Automated Analysis of Diffie-Hellman Protocols and Advanced Security Properties", by Benedikt Schmidt, Simon Meier, Cas Cremers, David Basin.
- Meier's PhD thesis [PDF]: provides a detailed explanation of the theory and implementation of Tamarin including inductive invariants and type assertions.
- Schmidt's PhD thesis [PDF]: provides a detailed explanation of the theory and application of Tamarin including the reasoning about Diffie-Hellman exponentiation and bilinear pairing.
- Staub's bachelor thesis [PDF]: about the implementation of the original version of Tamarin's GUI.

Tamarin Extensions

- "Distance-Bounding Protocols: Verification without Time and Location" [PDF], by Sjouke Mauw, Zach Smith, Jorge Toro-Pozo, Rolando Trujillo-Rasua, presented at S&P 2018.
- "A Novel Approach for Reasoning about Liveness in Cryptographic Protocols and its Application to Fair Exchange" [PDF], by Michael Backes, Jannik Dreier, Steve Kremer, Robert Künnemann, presented at EuroS&P 2017.
- "Modeling Human Errors in Security Protocols" [PDF], by David Basin, Saša Radomirović, Lara Schmid, presented at CSF 2016.
- "Alice and Bob Meet Equational Theories" [PDF], by David Basin, Michel Keller, Saša Radomirović, Ralf Sasse, paper presented at Logic, Rewriting,

Tamarin



Papers using Tamarin

- "A Formal Analysis of 5G Authentication" [PDF], by David Basin, Jannik Dreier, Lucca Hirschi, Saša Radomirović, Ralf Sasse, Vincent Stettler, presented at CCS 2018.
- "Alethea: A Provably Secure Random Sample Voting Protocol" [PDF], by David Basin, Saša Radomirović, Lara Schmid, presented at CSF 2018.
- "A Comprehensive Symbolic Analysis of TLS 1.3" [PDF], by Cas Cremers, Marko Horvat, Jonathan Hoyland, Sam Scott, Thyla van der Merwe,

- Security protocols are specified as rewriting logic systems
  - Security protocols
  - Rewriting logic systems

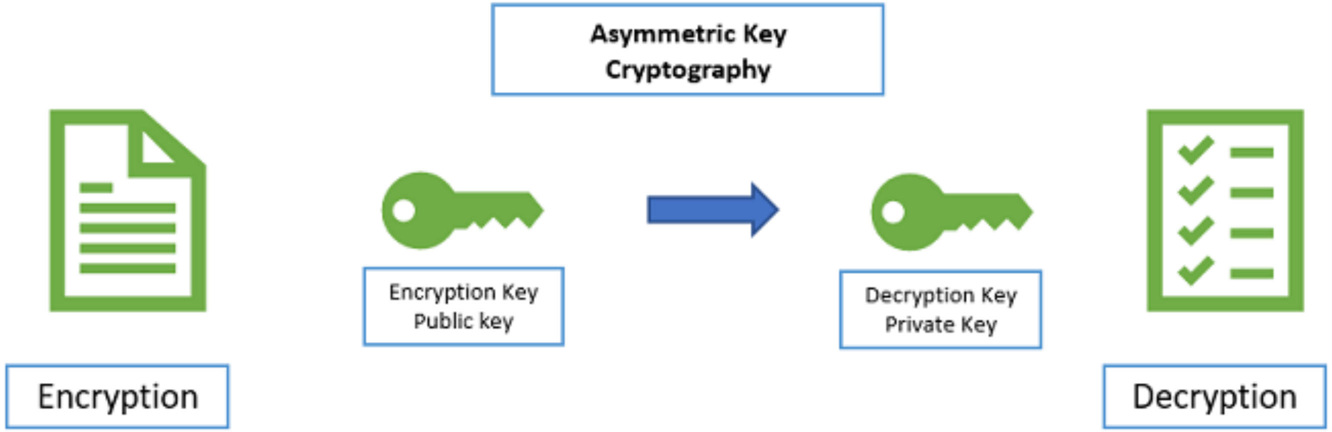
Security protocols

- Securing communication between agents
  - Transport Layer Security (TLS) to secure communication over the Internet
  - Authentication
  - Money transfer (HTTPS)
  - Voting

- Cryptography

A bit of cryptography

- Asymmetric encryption: (public key and private key) [1]



- Symmetric encryption:
  - The agents in a communication agree on a shared secret key
  - Diffie Hellman (DH) key exchange algorithm

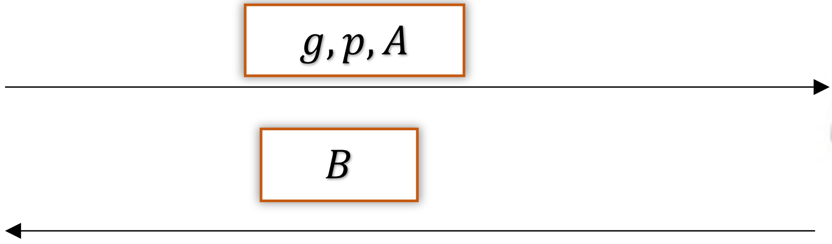
A bit of cryptography (DH)

a, g, p

$A = g^a \bmod p$

$K = B^a \bmod p$

$K = g^{ab} \bmod p$



g, p, A

B

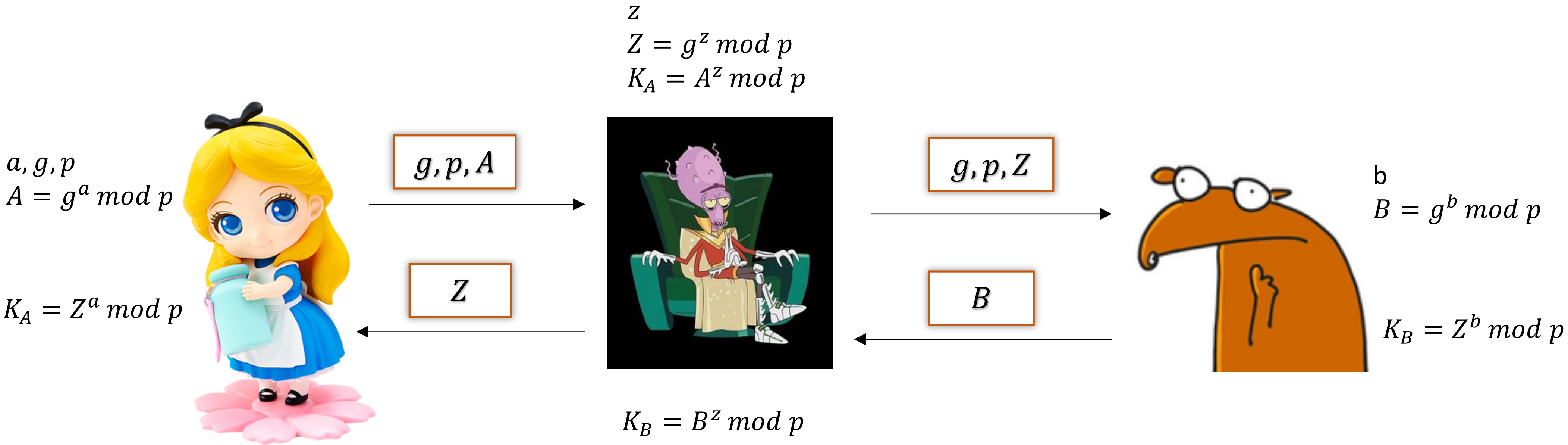
b

$B = g^b \bmod p$

$K = A^b \bmod p$

$K = g^{ab} \bmod p$

Man-in-the-middle attack



a, g, p
$A = g^a \bmod p$

$K_A = Z^a \bmod p$

$\boxed{g, p, A}$ →

← $\boxed{Z}$

z
$Z = g^z \bmod p$
$K_A = A^z \bmod p$

$K_B = B^z \bmod p$

$\boxed{g, p, Z}$ →

← $\boxed{B}$

b
$B = g^b \bmod p$

$K_B = Z^b \bmod p$

Replay attack

- The attacker sends to the victim the same previous message which was used before in the victim’s communication

- The victim thinks that it is a valid message and reacts to this message accordingly

Security protocols

- Security protocols must be robust and work in hostile environments where an attacker can:
    - eavesdrop messages
    - intercept messages
    - impersonate any agent
    - encrypt or decrypts massages with the keys he has got
    - repeat fake messages

- A model checker is required to check the correctness of protocols

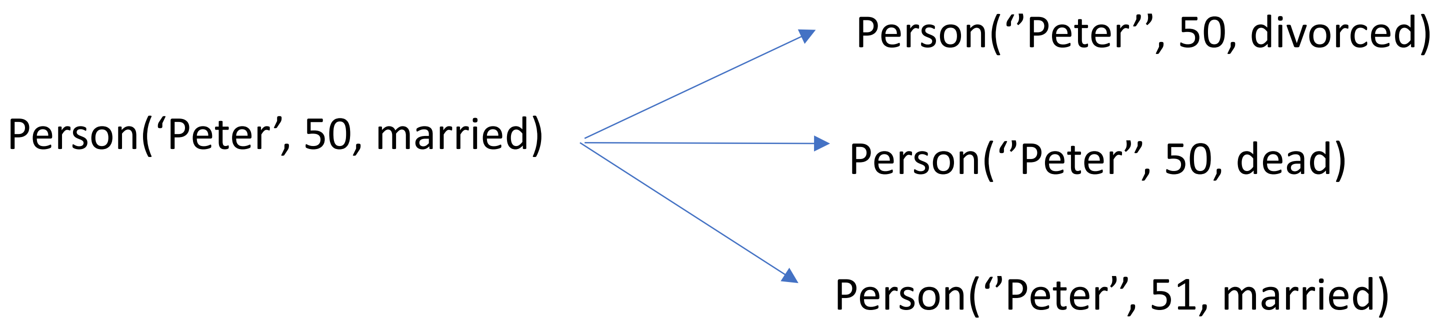
Tamarin [2]

- A method based on operational semantics
- Protocols and adversaries are specified in multiset rewriting rules
- Security properties are defined as trace properties, checked against the traces of the transition system

- Rewrite rules specify:
  - the protocol initiator, responder, and trusted key server
  - the attacker’s knowledge
  - the messages on the network
  - the state of a protocol changes by interacting messages

Rewriting Logic

- Modelling behavior of a dynamic system, which defines how the system state evolves
- What is a dynamic system?
  - For example, modelling how a person ages [4]



Person(‘Peter’, 50, married)

Person(‘’Peter’’, 50, divorced)

Person(‘’Peter’’, 50, dead)

Person(‘’Peter’’, 51, married)

  - One step of execution: →

Rewriting logic

- Equations define the deterministic features and rewrite rules define the non-deterministic features
- Rules are labeled:
  - $birthday$: $Person(X, N, S) \longrightarrow Person(X, N+1, S)$
  - $divorce$: $Person(X, N, S) \longrightarrow Person(X, N, divorced)$
    $\text{if} \quad N > 40 \ \wedge \ S == married$
  - $marriage$: ….
  - …

Rewriting logic

- A rewriting logic specification is a tuple $\mathcal{R} = (\Sigma, E, L, R)$, where Σ is a signature, E is a set of equations, L is a set of labels, and R is a set of unconditional and conditional labeled rewrite rules [5].
  - l: $t \longrightarrow t'$
- Rules are non-deterministically applied

- Rules are applied to the subterms of term t (or t itself), until it is not reducible anymore

Modelling security protocols [6]

- Rewriting logic model for formalizing and reasoning about security protocols
- Rewrite logic for specification of a protocol:
  - Protocol roles
  - Messages are represented as terms communicated between agents
  - Protocol agents states evolve by getting messages
  - Based on different roles each agent reacts to a message and generates events

Formalizing a protocol[6]

- Basic terms: *Agent, Role, Fresh, Var, Func, TID, AdvConst, …*
    - agent names $\{Alice, Bob\} \epsilon \ Agent$
    - Protocol roles $\{Init, Recp\} \epsilon \ Role$
    - Freshly generated terms like nonce, session keys
    - Variables
    - Function names
    - Thread identifiers (the protocol role instance) $tid \ \epsilon \ TID$
    - The set of fresh values generated by the adversary.
    - A term t is local to a thread: t#tid

Terms and events[6]

- Term ::= BasicTerm | (Term,Term)| pk(Term) | sk(Term) | k(Term,Term) | {| Term |}aTerm | {| Term |}sTerm | Func(Term$*$)
    - sk(Alice) : private key of agent Alice
    - pk(Alice) : public key
    - k(Alice, Bob) : shared symmetric key
    - $\{|t_1|\}_{t_2}^{a}$: asymmetric encryption of the term t1 with the key t2

- *Event ::= create(Role, Sub) | send(Term) | recv(Term)*

A protocol Exm. [6]

- A protocol (P) is a mapping from roles to event sequences
    - $Role \rightarrow event^*$

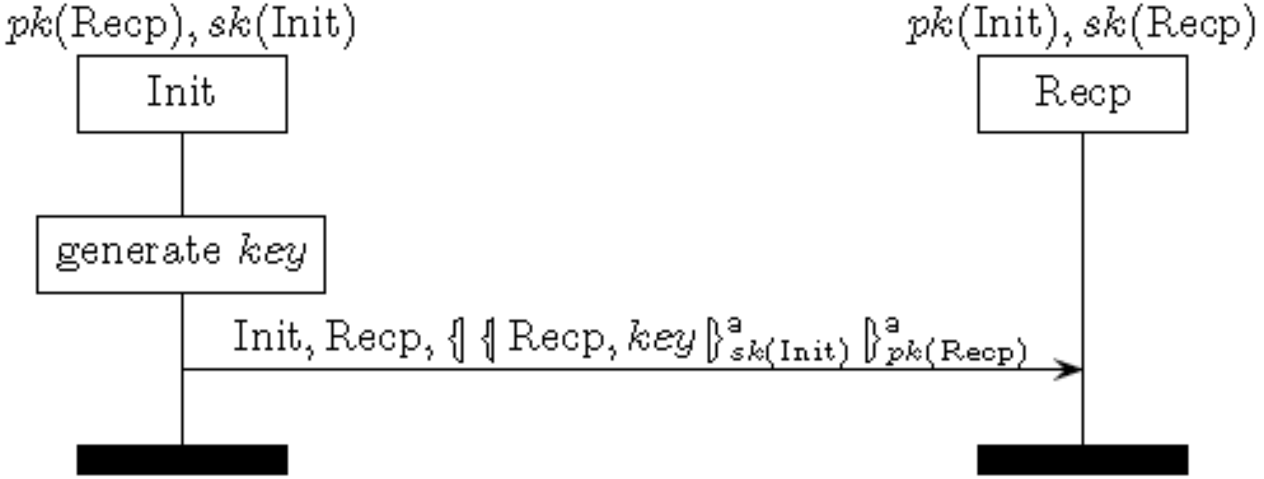


Fig. 24.3. Simple protocol

$$P(\text{Init}) = \langle \text{send}(\text{Init}, \text{Recp}, \{\!|\, \{\!|\, \text{Recp}, key \,|\!\}^{\mathsf{a}}_{sk(\text{Init})} \,|\!\}^{\mathsf{a}}_{pk(\text{Recp})}) \rangle$$

$$P(\text{Recp}) = \langle \text{recv}(\text{Init}, \text{Recp}, \{\!|\, \{\!|\, \text{Recp}, x \,|\!\}^{\mathsf{a}}_{sk(\text{Init})} \,|\!\}^{\mathsf{a}}_{pk(\text{Recp})}) \rangle$$

Adversary power

- Dolev-Yao model:
    - all communicated messages between agents are intercepted by the adversary
    - all received messages are sent by the adversary

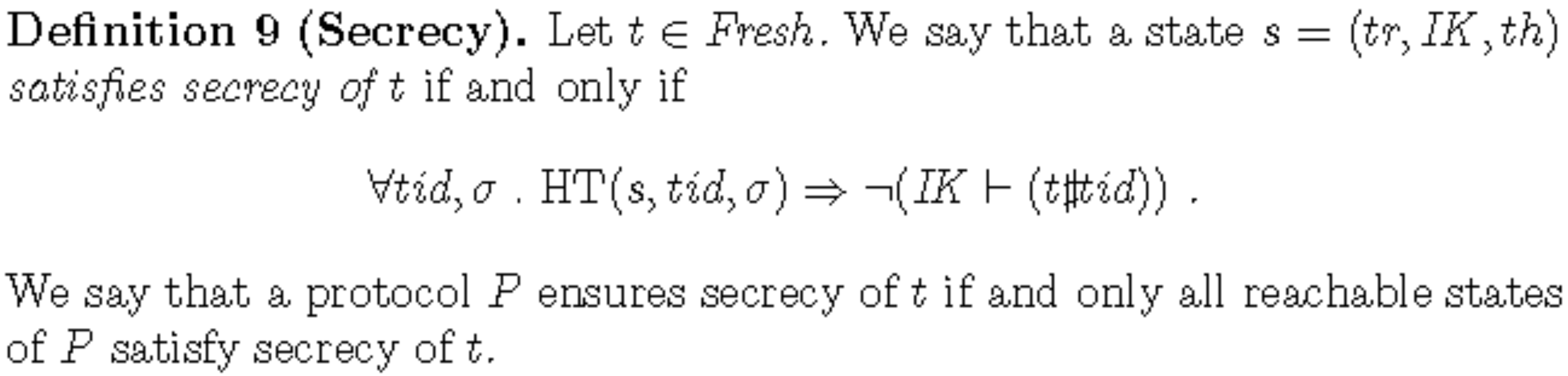
- The adversary knows agent names and their public key
- It can generate constants (AdvConst)
- It has compromised some of the private keys of agents
- $M \vdash t$, The adversary can infer t, from M (a set of terms)

Execution model[6]

- The semantics of a protocol $P\epsilon\ Protocol$ is defined by rewrite rules
- The rewrite rules define a transition system
- Each rule describes how each event causes a state transition
- State configuration: $< trace, Adersary\ knowledge, event >$

$$\frac{th(tid) = \langle \mathsf{send}(m) \rangle \hat{}\, l}{(tr, IK, th) \longrightarrow (tr \hat{}\, \langle (tid, \mathsf{send}(m)) \rangle, IK \cup \{m\}, th[tid \mapsto l])}\,[\mathsf{send}]$$

Security properties [6]



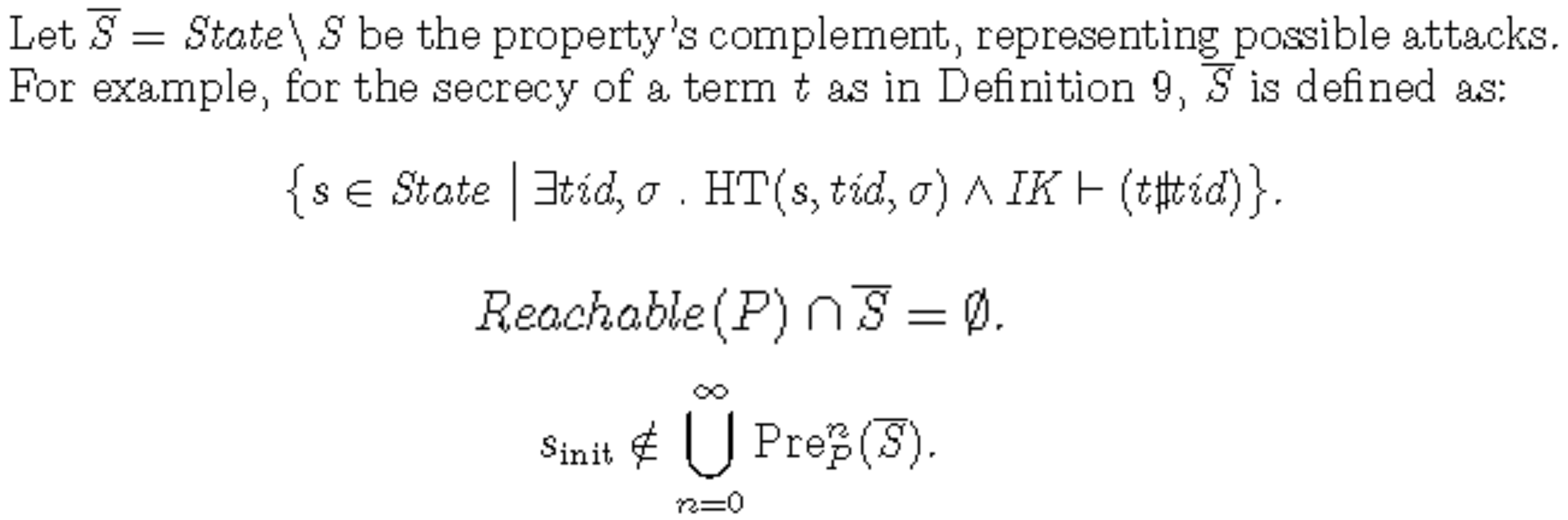
Definition 9 (Secrecy). Let $t \in \mathit{Fresh}$. We say that a state $s = (tr, IK, th)$ *satisfies secrecy of* t if and only if

$$\forall tid, \sigma \ . \ \mathrm{HT}(s, tid, \sigma) \Rightarrow \neg(IK \vdash (t\sharp tid)) \ .$$

We say that a protocol P ensures secrecy of t if and only all reachable states of P satisfy secrecy of t.

HT: honest agents which are not compromised by the attacker

Model checking of security protocols [6]



Let $\overline{S} = State \setminus S$ be the property's complement, representing possible attacks. For example, for the secrecy of a term t as in Definition 9, $\overline{S}$ is defined as:

$$\left\{ s \in State \;\middle|\; \exists tid, \sigma \;.\; \mathrm{HT}(s, tid, \sigma) \wedge IK \vdash (t \sharp tid) \right\}.$$

$$Reachable(P) \cap \overline{S} = \emptyset.$$

$$s_{\mathrm{init}} \notin \bigcup_{n=0}^{\infty} \mathrm{Pre}_P^n(\overline{S}).$$

The set of reachable states is infinite,
limiting the number of threads or sessions that can be created to make it finite

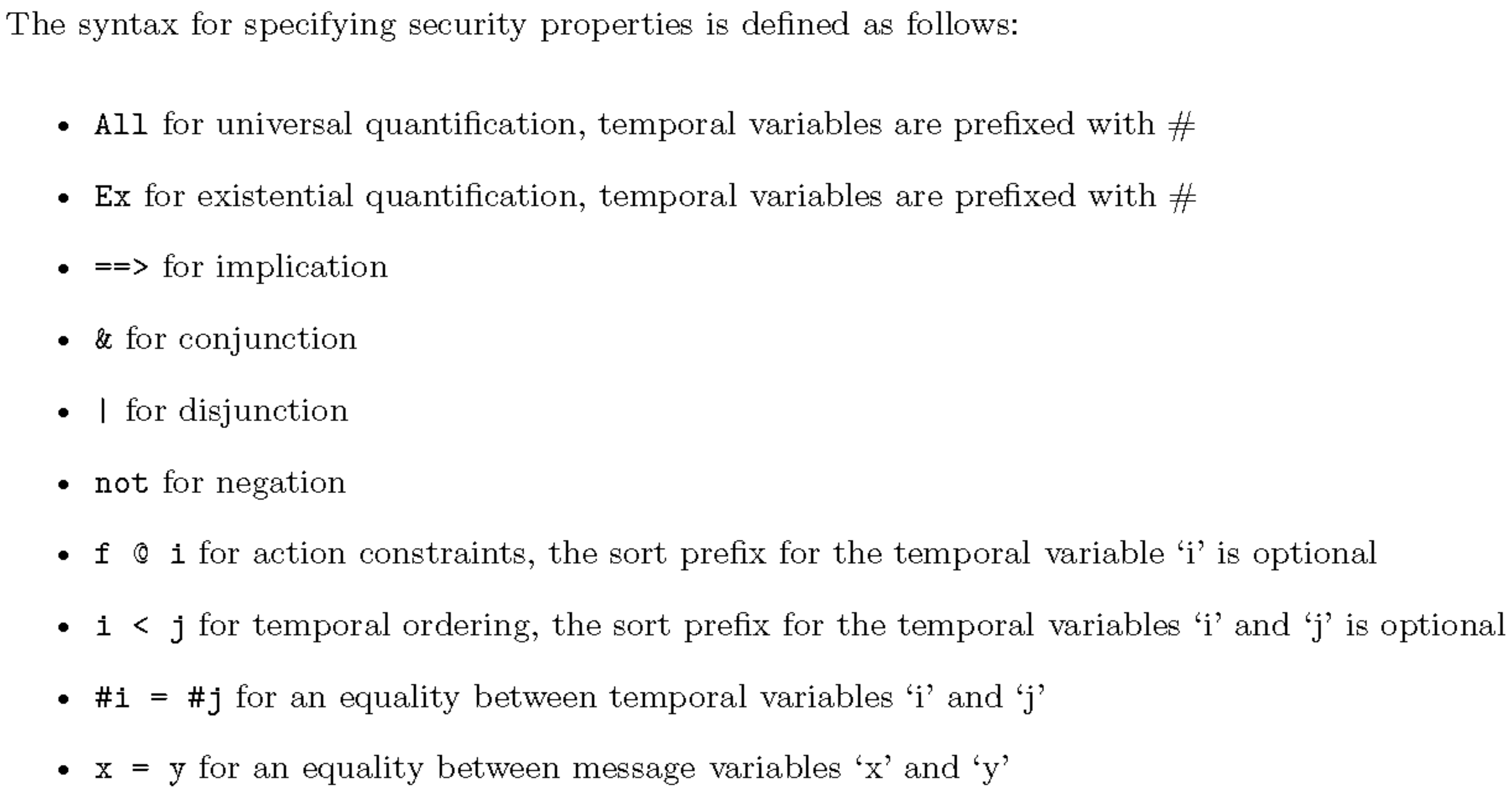
Tamarin [2]

- $\mathcal{R} = (\Sigma, E, L, R)$
- E defining cryptographic operators
- R defining a protocol
- a formula φ defining a trace property
- Tamarin can either check the validity or the satisfiability of φ for the traces of executions

Tamarin [2]

- The Tamarin multiset rewriting rules define a labeled transition system.
- Each rule defines how the system state evolves to a new state
- If the current state of a system has a subterm, where its pattern maches the left-hand-side of a rule, then this rule can be applied
- This subterm is replaced by an instance of the right-hand-side
- A term is reduced and rewritten by rules until it is not reducable

Tamarin [2]



The syntax for specifying security properties is defined as follows:

- `All` for universal quantification, temporal variables are prefixed with #
- `Ex` for existential quantification, temporal variables are prefixed with #
- `==>` for implication
- `&` for conjunction
- `|` for disjunction
- `not` for negation
- `f @ i` for action constraints, the sort prefix for the temporal variable ‘i’ is optional
- `i < j` for temporal ordering, the sort prefix for the temporal variables ‘i’ and ‘j’ is optional
- `#i = #j` for an equality between temporal variables ‘i’ and ‘j’
- `x = y` for an equality between message variables ‘x’ and ‘y’

References

- [1] https://cheapsslsecurity.com/blog/what-is-asymmetric-encryption-understand-with-simple-examples/
- [2] https://tamarin-prover.github.io/manual/tex/tamarin-manual.pdf
- [3] https://www.virusbulletin.com/blog/2015/05/weak-keys-and- prime-reuse-make-diffie-hellman-implementations-vulnerable
- [4] *Designing Reliable Distributed Systems: A Formal Methods Approach Based on Executable Modeling in Maude*, Peter Csaba Olveczky, 2018, Springer.
- [5] *A logical theory of concurrent objects and its realization in the Maude language*, Jose Meseguer, Research Directions in Concurrent Object-oriented Programming, 1993, MIT Press.
- [6] *Model checking security protocols*, David Basin, Cas Cremers, and Catherine Meadows, Handbook of Model Checking, 2011, Citeseer.
- [7] https://cheapsslsecurity.com/blog/what-is-asymmetric-encryption-understand-with-simple-examples/