Security protocol analysis using the Tamarin Prover

Jannik Dreier, Lucca Hirshi

Based on slides by Cas Cremers
Overview & Structure

• **Overview**
  1) Advanced Features
     • Heuristics
     • Equational Theories, Observational Equivalence
     • Recent developments
  2) Related work
     • Other verification tools
     • Symbolic vs Computation verification
  3) Tamarin’s Algorithm – some intuitions
  4) Some recent research:
     • Partial deconstructions & auto-sources
     • Using Tamarin for real: analyzing 5G AKA
Overview & Structure

• **Mode of operation**
  - No need to use tools during the talk.
  - There will be time for questions during the talk and at the end.
  - Please *raise your hand in zoom* or ask questions offline in the *discord channel* #security-protocols.
Outline

Part 1:
Advanced Features
Heuristics?

• If Tamarin terminates, one of two options:
  – **Proof**, or
  – **counterexample** (in this context: attack)

• At each stage in proof, multiple constraint solving rules might be applicable
  – Similar to “how shall I try to prove this?”
  – Choice influences speed & termination, but not the outcome after termination

• Complex **heuristics choose rule**
  – user can give hints, choose heuristic or override
Lemmas

- When it doesn’t terminate…
- Guide the proof manually; export
- Write (intermediate) lemmas
  - “Hints” for the prover
    - They don't change the proof obligation, only help finding a proof
  - Specify lemma that can be used to prune proof trees at multiple points
Complexity and termination

• Basic examples
  – Key exchange protocols
  – Signature-based protocols

• More complex often needs hints (lemmas)
  – XOR
  – Protocols with complex loops/state machines
  – Diff-equivalence
Other Tamarin features....

• Advanced equational theory support
  – Diffie-Hellman, XOR, multisets, subterm-convergent and more...

• Construct your own proof interactively, and export it so others can verify

• Program your own heuristic

• Diff-equivalence (observational equivalence)

• Restrictions & conditional rules

• Applied-Pi input (through SAPIC integration)
Some recent results

- More accurate modeling of cryptography
  - *Seems Legit: Automated Analysis of Subtle Attacks on Protocols that Use Signatures*
    - CCS’19
    - Jackson, Cremers, Cohn-Gordon, Sasse – ia.cr/2019/779
  - *Prime, Order Please! Revisiting Small Subgroup and Invalid Curve Attacks on Protocols using Diffie-Hellman*
    - CSF’19
    - Cremers, Jackson – ia.cr/2019/526

- Improving automation
  - *Automatic Generation of Sources Lemmas in Tamarin: Towards Automatic Proofs of Security Protocols*
    - ESORICS’20
    - Cortier, Delaune, Dreier – Springer/HAL report

- EMV Chip and pin → attack to circumvent PIN requirement for VISA contactless
  - *The EMV Standard: Break, Fix, Verify*
    - SP’21
    - Basin, Sasse, Toro – emvrace.github.io

- Spectral analysis of Noise → security hierarchy of protocols from the Noise framework supported by formal analyses
    - Usenix’20
    - Girol, Hirschi, Sasse, Jackson, Cremers, Basin – Usenix Paper
Tamarin: conclusions

• **Tamarin offers many unique features**
  - State machine modeling, flexible properties, equational theories, global state, …
  - Enables automated analysis in areas previously unexplored
  - Many case studies available, from small protocols to large real-world protocols
  - Tamarin *found many new attacks*, impacting several real-world deployments

• Tool and sources are **free**; development on Github tamarin-prover.github.io
  - A real team effort!
Next up: Related Work

- What we saw:
  - Advanced Features
  - Heuristics, Lemmas
  - Equational Theories, Observational Equivalence
  - Recent developments
Outline

Part 2:
Related Work
Other **Symbolic Verification Tools**

- **Two main categories:**
  1) **Semi-decision procedures for unbounded # sessions**
     *Tamarin, ProVerif, etc.* → **Differences:** language, automation, modeling features
  2) **Decision procedures for bounded # sessions**
     *DeepSec, Akiss, Spec, etc.*

- **All tools can check** trace properties (reachability)

- **Some of them can check** equivalence properties
  (often used to check privacy):
  - Unbounded # sessions: stricter notion of equivalence (diff-equivalence) → spurious attacks (sound but incomplete)
  - Bounded # sessions → decision (sound and complete)
Computational Model vs. Symbolic Model

- **Computational model:**
  - Messages → bitstrings
  - Primitives → probabilistic algorithms
  - Protocol roles → probabilistic polynomial-time Turing machines acting as oracles giving access to data
  - Attacker → any probabilistic polynomial-time Turing machine
  - Security goals often are quantitative and probabilistic (expressed as a probabilistic experiment)

+ much more precise model
- less automation and scalability
Modeling real-world objects

Reality

Symbolic
Modeling real-world objects

Reality

Computational

Symbolic
Modeling real-world objects

Reality

Computational

Symbolic
Survey

• More on the topic, with a comprehensive survey and comparison of the state-of-the-art tools:

SoK: Computer-Aided Cryptography

Manuel Barbosa*, Gilles Barthe†‡, Karthik Bhargavan§, Bruno Blanchet§, Cas Cremers¶, Kevin Liao†||, Bryan Parno**
*University of Porto (FCUP) and INESC TEC, †Max Planck Institute for Security & Privacy, §IMDEA Software Institute,
§INRIA Paris, ¶CISPA Helmholtz Center for Information Security, ||MIT, **Carnegie Mellon University

Abstract—Computer-aided cryptography is an active area of research that develops and applies formal, machine-checkable approaches to the design, analysis, and implementation of cryptography. We present a cross-cutting systematization of the computer-aided cryptography literature, focusing on three main areas: (i) design-level security (both symbolic security and computational security), (ii) functional correctness and efficiency, and (iii) implementation-level security (with a focus on digital side-channel resistance). In each area, we first clarify the role of computer-aided cryptography—how it can help and what the caveats are—in addressing current challenges. We next present a taxonomy of state-of-the-art tools, comparing their accuracy, completeness, and coverage. This sheds light on the features that can help solve critical challenges in the field, which are difficult to catch by code testing or auditing; ad-hoc constant-time coding recipes for mitigating side-channel attacks are tricky to implement, and yet may not cover the whole gamut of leakage channels exposed in deployment. Unfortunately, the current modus operandi—relying on a select few cryptography experts armed with rudimentary tooling to vouch for security and correctness—simply cannot keep pace with the rate of innovation and development in the field.

Computer-aided cryptography, or CAC for short, is an active area of research that aims to address these challenges. It en-
Which tool should I be using?

- **If you are starting out in the domain:**
  - Try to find **existing protocol models** that are close(ish) to your problem for each tool
    - Pick the tool with the closest existing model, start by adapting that model
  - Also aim for a tool with a formal language that you are most familiar with
    - Example: ProVerif & DeepSec → process calculus (applid-pi)
    - Tamarin → multiset rewriting

- **More advanced:**
  - Choice can be driven by the **security property** and threat model that you are interested in
  - Most approaches give incomparable guarantees; as a consequence they cover different attacks
• What we saw:
  – Other symbolic verification tools
  – Symbolic vs Computational Analysis

Next up: Tamarin’s Algorithm
Outline

Part 3:

Tamarin’s Algorithm
Tamarin: high-level

- **Modeling** protocol & adversary done using multiset rewriting
  - Specifies transition system; induces set of traces
- **Property** specification using fragment of first-order logic
  - Specifies “good” traces
- Tamarin tries to
  - provide proof that all system traces are good, or
  - construct a counterexample trace of the system (attack)
Basic principles

• Backwards search using constraint reduction rules (20+)
• Turn negation of formula into set of constraints
• Case distinctions
  – E.g.: Possible sources of a message or fact
• Try to establish:
  – no solutions exist for constraint system, or
  – there exists a „realizable“ execution (trace)
• If multiple rules can be applied: use heuristics
Tamarin workflow

Property P

System S

Solution exists: ATTACK

No solution exists: PROOF

Provide **hints** for the prover (e.g. invariants)

Tamarin prover

Tamarin’s algorithm

Interactive mode

Inspect partial proof

Run out of time or memory
Tamarin workflow

- **Property P**
- **System S**
- **constraint from (not P)**
- **constraints from S**
- **Dedicated constraint solver**

**Solution exists:** ATTACK

**No solution exists:** PROOF

**Provide hints** for the prover (e.g. invariants)

**Interactive mode**

- Inspect partial proof
- Run out of time or memory
Tamarin prover

Constraint solver
Tamarin prover

Theorem Prover

Constraint solver
Demo
• What we saw:
  – Some intuition on Tamarin’s Algorithm

Next up: two recent pieces of research on Tamarin

1) Improving Tamarin with sources lemmas generation (ESORICS’20)
2) Using Tamarin to conduct a security analysis of 5G authentication (CCS’18)
Intermission

Time for some questions