Computer-Aided Proofs of Security

May 4, 2025

Madrid

CAPS'25 program

9.00 - 9.30	Overview	
9.30 - 9.45	Example: KEM-DEM security	Computational model
9.45 - 10.30	Tool: ProofFrog	
Coffee break		
11.00 - 11.45	Tool: EasyCrypt	
11.45 - 12.15	Example: Signed DH	Symbolic model
12.15 - 13.00	Tool: ProVerif	
Lunch break		
14.15 - 15.00	Tool: Tamarin	
Coffee break		
15.45 - 17.15	Round Table with Tool Developers	

Links to proof implementations at https://caps-workshop.com/#program

General info about affiliated events

Lunches & Catering:

- All meals served on **floor –2**

Wifi:

- Network name: **UCM-CONGRESO**
- Password: **congresosUCM**

- Username: **Workshop@congreso.ucm.es**
- Password: **euro@474**

Mind the Eurocrypt **Code of Conduct :** https://eurocrypt.iacr.org/2025/conduct.php

Introduction to Computer-Aided Proofs of Security

Sabine Oechsner

I have a proof to formalize, what do I do?

Implicit question:

Which tool to choose for the job?

Answer:

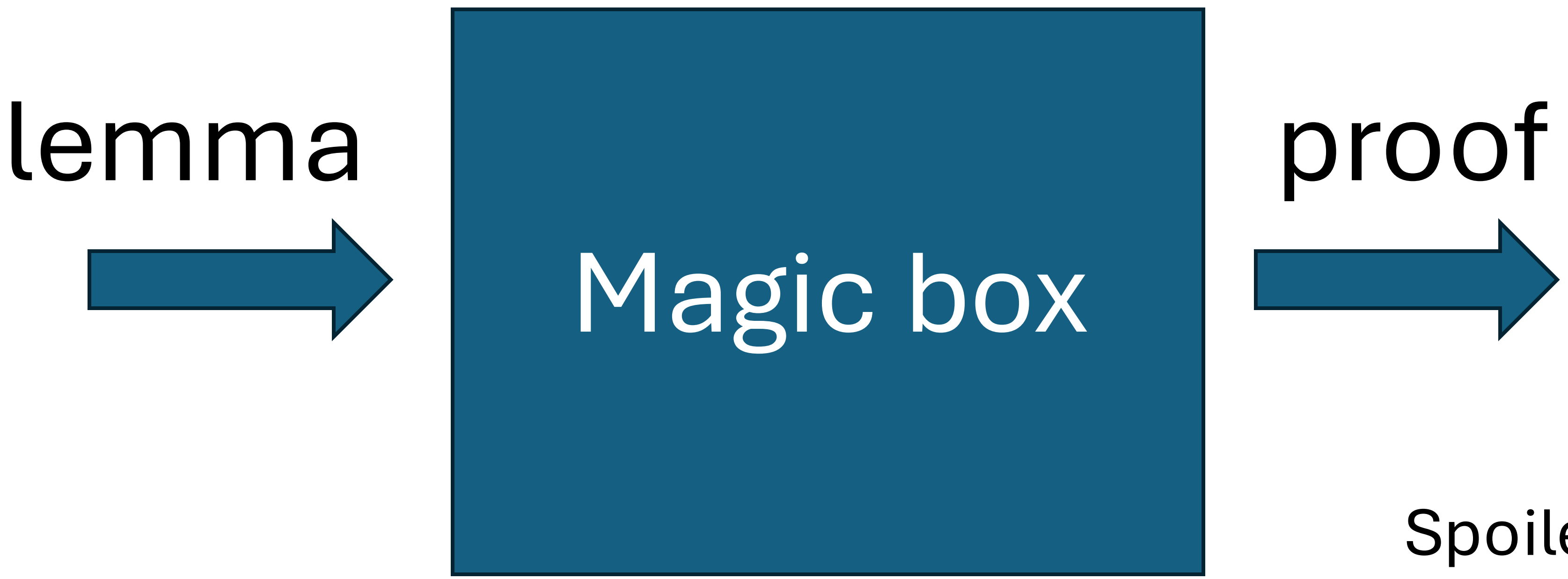
It depends. What are you trying to prove? And why?

- Questions are not independent!
- Each tool has constraints

Goal of today: Understand some of those constraints, and what goes into choosing a tool.

Mental model of the average cryptographer

How I imagine it, based on talking to many cryptographers:



Spoiler: not how it works…

Learning goals for today

How to work with "the box":

- What support can computers offer
- What goes into developing computer-aided security proofs
  - "inputs"
  - "outputs"
  - What happens when I "push the button"?
  - Human-intelligible proof?
  - Automation?
- How to work with tools
- How to get help

Not today

- Detailed tutorials
- How the tools work under the hood
- How to prove other interesting theorems

… but talk to our speakers if you want to learn more!

What can I get out of formal verification?

- Precise statement of your scheme, assumptions, and theorem,
- Independent check and trust,
- Counterexamples / attacks,
- Robust proofs that evolve with your definitions,
- Convenience: 'not working alone' but interactive

What can computers prove about crypto?

About "theory":

- Mathematical properties
- Design-level security
  = security of constructions

About software:

- Implementation correctness
- Implementation-level security

Mathematical properties

Examples:

- compute the smallest parameters that satisfy a security bound by exhaustive search
- "p is the first prime satisfying X"

Example tools:

- Lightweight: Computer Algebra Systems (e.g. Sage)
- Full proofs: Rocq (previously Coq), Isabelle, Lean, ...

Design-level security ⭐

Aka cryptographic security proofs

Examples:

- IND-CPA security of KEM-DEM
- Secrecy of a key exchange

Example tools:

- Cryptography: EasyCrypt, Tamarin, ProVerif, Cryptoverif, Squirrel, ...
- General purpose: Rocq, Isabelle, Lean, ...

Implementation correctness

Aka functional correctness

Examples:

- "This piece of C code implements a finite field"
- "This piece of Jasmin code implements ML-KEM"

Example tools:

- FiatCrypto, F*, Jasmin, …
- and general-purpose tools: Rocq, Isabelle, Lean, …

Implementation security

Implementation-specific security considerations

Examples:

- side channel resistance
- Panic / crash freedom
- secret-independent timing

Example tools:

- F*, VST, ….

What can computers prove about crypto?

About "theory":

- Mathematical properties
- Design-level security
  = security of constructions

Today at CAPS!

About software:

- Implementation correctness
- Implementation security

Modeling security

Classes of models: computational and symbolic

Security model = abstraction of the real world

- What to abstract away?
- Impact on attacker?

... and choices affect how proofs "look" like

Computational model (= "what we do on paper")

Construction: probabilistic algorithms that operate on bitstrings

- Expressed as TMs, pseudocode, ...
- Primitives or protocols

Encryption
- Algorithms (Kgen, Enc, Dec)
- Correctness: forall k in Kgen, forall m,
  Dec(k, Enc(k,m)) = m

Attacker: probabilistic algorithm

- Black box: we only assume what they cannot compute (e.g. break hard problems)

Security model: prescribed interaction between attacker and construction, e.g. through game

Symbolic model (or Dolev-Yao model)

Protocol: abstract machines that can exchange messages

- messages consist of symbols
- Primitives are abstract operators

Encryption

- symbols (Enc, Dec)
- rule: Dec(k, Enc(k,m)) = m

Attacker: describe what attacker <u>can</u> do

- read, intercept, modify, delete messages on the network
- inject their own messages, constructed from existing messages following certain rules

Security model:

- properties of protocol execution trace
- Network attacker that attacks protocol logic

Classes of tools today

Computational model

- ProofFrog
- EasyCrypt

Symbolic model

- ProVerif
- Tamarin

More approaches (not today)

For example: Use reasoning techniques from symbolic model to get computational guarantees

Examples: Cryptoverif, Squirrel

CAPS'25 program

9.00 - 9.30	Overview	
9.30 - 9.45	Example: KEM-DEM security	Computational model
9.45 - 10.30	Tool: ProofFrog	
Coffee break		
11.00 - 11.45	Tool: EasyCrypt	
11.45 - 12.15	Example: Signed DH	Symbolic model
12.15 - 13.00	Tool: ProVerif	
Lunch break		
14.15 - 15.00	Tool: Tamarin	
Coffee break		
15.45 - 17.15	Round Table with Tool Developers	

Links to proof implementations at https://caps-workshop.com/#program