Verification of security protocols: using SMT solvers in the Squirrel prover.

Stanislas Riou
David Baelde
Stéphanie Delaune

4/04/2024

IRISA - SPICY

This work received funding from the France 2030 program managed by the French National Research Agency under grant agreement No. ANR-22-PECY-0006.

Formal verification

Computational model

- Probabilistic model, $n \neq m$ means that it's unlikely for n and m to be equal.
- High security guarantees (used by cryptographers).

Formal verification

Computational model

- Probabilistic model, $n \neq m$ means that it's unlikely for n and m to be equal.
- High security guarantees (used by cryptographers).

Manual proofs are complex and tedious. Provers can automate some steps.

- ProVerif [Blanchet, 2001]
- Tamarin [Meier et al, 2013]
- CryptoVerif [Blanchet, 2005]
- Squirrel [Baelde et al, 2021]

Outline

1. Squirrel : an interactive prover
2. Automation using SMT solvers
3. Implementation
4. Evaluation

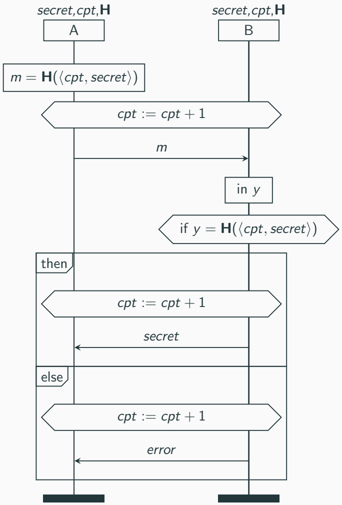
Squirrel: an interactive prover

Squirrel

Squirrel is an interactive prover in the computational model,
the user proves goals using *tactics*.

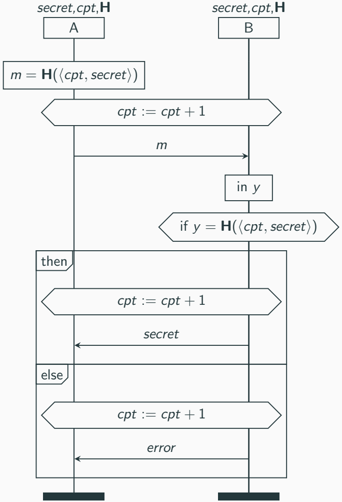
- Its syntax is a higher-order logic based on λ-calculus.
- Its semantics is based on indexed families of random variables.
- The model is probabilistic:
  two distinct names have a negligible probability of being equal.

Running example



A and B share access to a global counter cpt.

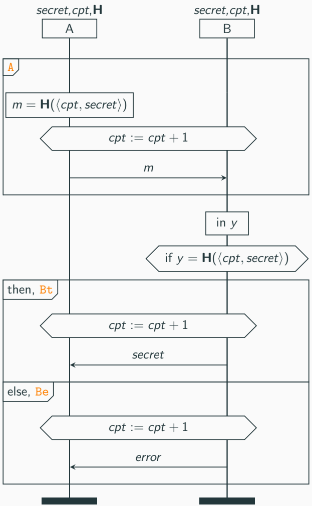
Running example



A and B share access to a global counter cpt.

Secrecy property

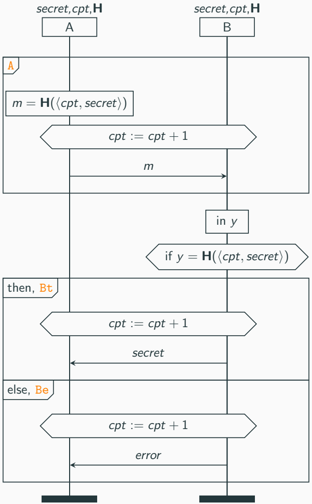
B never receives a hash of the current value of the counter, so the secret is never leaked.



Modelling protocols :

- protocols are represented step by step using *actions*;
- an execution trace of a protocol is represented by a sequence of actions;
- actions scheduled to be executed are represented using the predicate **happens**.

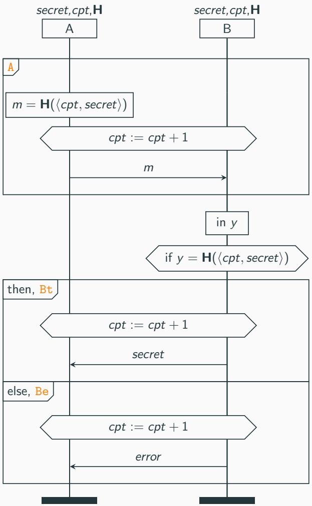
Traces



Examples of traces :

- A Be A Be A

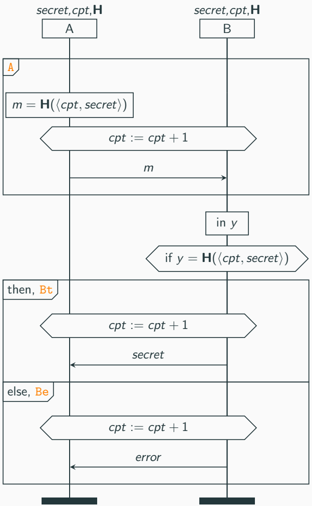
Traces



Examples of traces :

- A Be A Be A
- A A Be Bt A

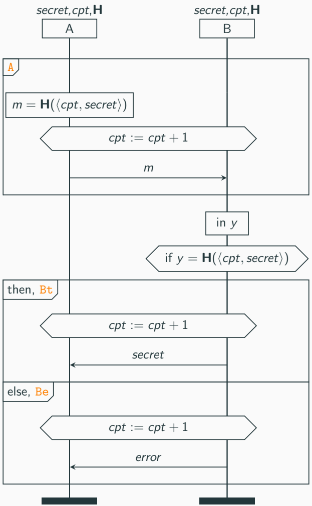
Traces



Examples of traces :

- A Be A Be A
- A A Be Bt A
- Bt A A Be

Traces



Examples of traces :

- A(1) Be(1) A(2) Be(2) A(3)
- A(1) A(2) Be(1) Bt(1) A(3)
- Bt(4) A(2) A(1) Be(8)

Logic

Secrecy property

B never receives a hash of the current value of the counter, so the secret is never leaked.

```
forall (j:index),
 happens(Bt(j)) ⇒
  cond@Bt(j) ⇒
   false
```

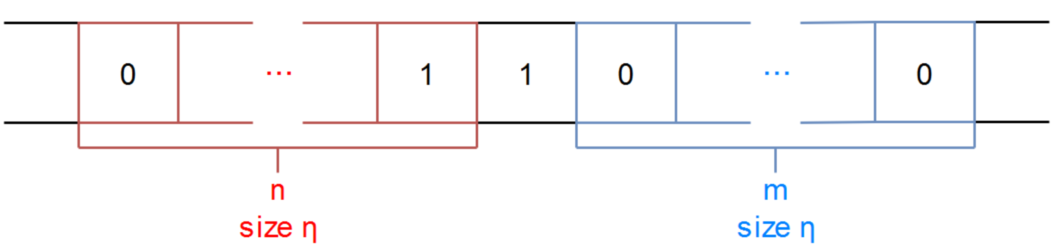
Listing 1: Secrecy property

Syntax :

- higher-order logic where terms are typed (e.g. messages, index, timestamps);
- timestamps are obtained from actions;
- macros give access to the content of actions or to mutable states.

Names interpretation

- Names are extracted from random tapes.
- Distinct name symbols come from distinct sections.
- Collisions are possible but unlikely.



Interpretation

A formula is interpreted according to :

- a term structure $\mathbb{M}$;
- a security parameter η;
- a set of random tapes ρ which depends on $\mathbb{M}$ and η.

We saw the interpretation of names.

The rest is standard λ-calculus assuming some built-in types (booleans, timestamps...) and functions (quantifiers, equality...)

Satisfiability and Validity

Satisfiability

A *local* formula ϕ is satisfiable if the following function is almost always true

$$\eta \mapsto \mathbf{Pr}(\rho : [\![\phi]\!]_{\mathbb{M}}^{\eta,\rho} = 1)$$

In that case, we note $\mathbb{M} \models \phi$

Validity

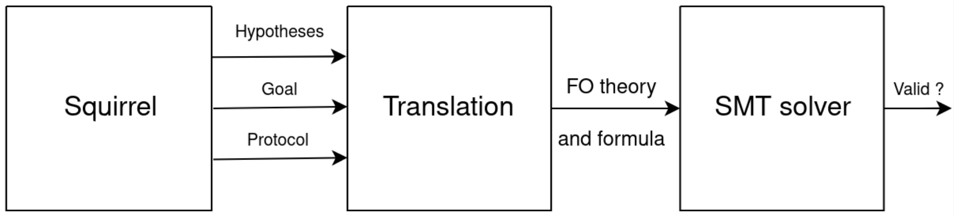
A *local* formula ϕ is valid if $\mathbb{M} \models \phi$ for any $\mathbb{M}$ that satisfies cryptographic hypotheses.

In that case, we note $\models \phi$

```
lemma counterIncrease (t,t':timestamp):
   t' < t => cpt@t' ~< cpt@t.
Proof.
  induction t => t Hind Ht.
  assert (t' < pred(t) || t' >= pred(t)) as H0 by case t.
  case H0.
    + apply Hind in H0 => //.
      use counterIncreasePred with t; 2: by constraints.
      by apply orderTrans _ (d@pred(t)).
    + assert t' = pred(t) as Ceq by constraints.
      use counterIncreasePred with t; 2: auto.
      by rewrite Ceq; auto.
Qed.
```

Listing 2: Proof in Squirrel

Contributions



Contributions :

- Translation
- Implementation
- Evaluation

Automation using SMT solvers

Soundness

We want to be sure that the translation never returns Valid when the goal is invalid.

Theorem : Soundness

If the translation of ϕ is SMT-valid, then ϕ is valid in Squirrel.

One model is probabilistic, the other is not $\longrightarrow$ the notion of validity changes.

Ex : $n \neq m$ valid means that n and m are never equal in SMT.
However in Squirrel they can be equal for a negligible amount of tapes.

Soundness

General idea

1. Take an invalid Squirrel formula.
2. Transform a Squirrel interpretation into an SMT interpretation.
3. Show that the SMT formula is invalid.

Proof overview : general case

Natural translation :

- Symbols (functions, variables, macros) $\longrightarrow$ abstract symbols
- Types $\longrightarrow$ abstract types

Ex : $f(n(i)) \neq \text{input@}A(i) \longrightarrow f\ (n\ i) \neq \text{input}\ (A\ i)$

Squirrel model $\mathbb{M} \longrightarrow$ SMT interpretation :

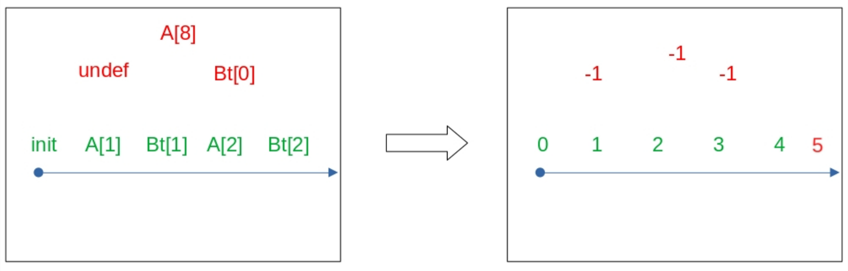
- We pick a tuple (η, ρ) that evaluates our formula to 0.
- We interpret everything according to $(\mathbb{M}, \eta, \rho)$.

Proof overview : timestamps

Translation :

- Timestamps $\longrightarrow$ integers.
- Happens checks if the integer is in [0,max_ts] (max_ts is an arbitrary constant).
- Equality and the order relation are not natural (they depend on happens).

Squirrel interpretation $\longrightarrow$ SMT interpretation :



Implementation

Implementation

Scope covered :

- macros and their axioms;
- timestamps, their axioms and dependencies;
- custom types.

Usage in Squirrel :

- relies on Why3 $\longrightarrow$ supported solvers can be used;
- callable with `smt ~pure ~style ~prover ~slow`

Evaluation

Comparison with an existing tactic

SMT was also compared to an ad-hoc tactic, `constraints` :

- tested on the entire directory of Squirrel examples;
- only pure trace formulas were translated;
- 1994 cases where `smt` concludes but not `constraints`
- 34 cases where `constraints` concludes but not `smt`

	Valid	Unknown	Failure
`smt`	45975	958	8578
`constraints`	51933	3578	0

Case study : running example

```
lemma counterIncrease (t,t':timestamp):
  t' < t ⇒ cpt@t' ~< cpt@t.
Proof.
  induction t ⇒ t Hind Ht.
  assert (t' < pred(t) || t' >= pred(t)) as H0 by case t.
  case H0.
    + apply Hind in H0 ⇒ //.
      use counterIncreasePred with t; 2: by constraints.
      by apply orderTrans _ (d@pred(t)).
    + assert t' = pred(t) as Ceq by constraints.
      use counterIncreasePred with t; 2: auto.
      by rewrite Ceq; auto.
Qed.
```

Listing 3: Proof without SMT

Case study : running example

```
lemma counterIncrease (t,t':timestamp):
   t' < t ⇒  cpt@t' ∼< cpt@t.
Proof.
  induction t. smt.
Qed.
```

Listing 4: Proof with SMT

Gains with SMT (in lines of code, without the protocol) :

- Toy counter : from 33 lines to 22
- Canauth [Van Herrewege et al, 2011] : from 448 lines to 215

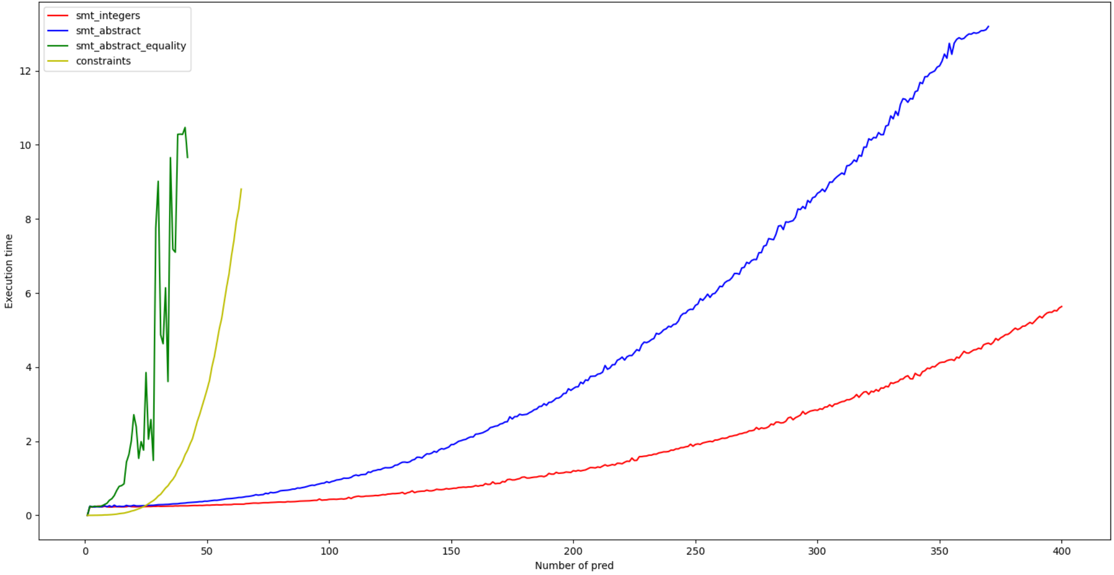
Comparing theories

Evaluation of the tactic on a simple family of lemmas for three different theories.

```
lemma predpred (t,t':timestamp) :
    pred(pred...(pred(t)))≤t'⇒
        (pred(t)=t'||pred(pred(t))=t'||...||t≤t').
```

Listing 5: Predecessors

Comparing theories



Conclusion

Conclusion and future works

- Automated tactic for Squirrel.
- Proof of soundness.
- Tactic implemented and tested.

Conclusion and future works

- Automated tactic for Squirrel.
- Proof of soundness.
- Tactic implemented and tested.

Ongoing works

- Support more elements (try find, diff, polymorphism).
- Reduce the number of smt failures.

Future works

- Work on completeness.
- Conduct bigger case studies.

Questions?