The Alan Turing Institute

Formalising Σ-Protocols and Commitment Schemes using CryptHOL

David Butler, Andreas Lochbihler, David Aspinall, Adria Gascon

Motivation: Do we have a problem with cryptographic proof?

"... Yes we do. The problem is that as a community, we generate more proofs than we carefully verify."

S. Halevi. 2005

“In our opinion, many proofs in cryptography have become essentially unverifiable. Our field may be approaching a crisis of rigor."

Bellare and Rogaway. 2004

“Security proof for even simple cryptographic systems are dangerous and ugly beasts. Luckily, they are only rarely seen: they are usually safely kept in the confines of ‘future full-versions’ of papers, or only appear in cartoon-ish form, generically labeled as ... ‘proof sketch’.”

Bristol Crypto Group. 2017

One method to alleviate this ‘crisis of rigor’: Machine Proofs

Machine Checked Proofs

Higher guarantees of correctness of proofs.

Assumptions and definitions are explicitly defined.

These need to be checked by a human ‘verifier’.

All parts of a proof must be proven for a proof to be valid.

Help increase the level of rigour in proofs.

Tools have been developed to overcome this crisis of rigor.

CertiCrypt

CryptHOL ⟷ **Σ-protocols**
(and commitment schemes)

Our work

EasyCrypt

FCF

ProVerif

This talk:

1. Hope to show how formal methods can be beneficial.
2. Insight into how such a formalisation works.
3. Promote discussion about the future direction of formalisation efforts in this area.

Preliminaries: One slide on Σ-protocols

- Run between a Prover (P) and a Verifier (V) over relation R.
  - P proves to V they know w for h such that (h,w) is in R.

Prover		Verifier
$(h,w) \in R$		h
$(a,rd) \leftarrow \mathrm{init}(h,w)$	$\xrightarrow{\quad a \quad}$	
	$\xleftarrow{\quad e \quad}$	$e \leftarrow \mathrm{challenge}$
$r \leftarrow \mathrm{response}(rd,w,e)$	$\xrightarrow{\quad r \quad}$	$\mathrm{check}(h,a,e,r)$ accept/reject

- (a,e,r) is the ‘conversation’.
- Properties: completeness, special soundness, honest verifier zero knowledge (HVZK)

Formal methods can be useful: Defining HVZK

A standard textbook definition looks as follows

- **Special honest verifier zero knowledge:** *There exists a probabilistic polynomial-time simulator M, which on input x and e outputs a transcript of the form (a, e, z) with the same probability distribution as transcripts between the honest P and V on common input x. Formally, for every x and w such that $\boxed{(x, w) \in R}$ and every $e \in \{0,1\}^t$ it holds that*

$$\left\{M(x,e)\right\} \equiv \left\{\langle P(x,w), V(x,e)\rangle\right\}$$

where $M(x,e)$ denotes the output of simulator M upon input x and e, and $\langle P(x,w), V(x,e)\rangle$ denotes the output transcript of an execution between P and V, where P has input (x,w), V has input x, and V's random tape (determining its query) equals e.

Fairly standard simulation based definition

Tells us what happens when (x,w) are in R

What happens when (x,w) are not in R? → **Causes issues in the OR construction**

OR construction for Σ-protocols

We want to consider the relation

$$R_{\mathrm{OR}} = \{((x_0, x_1), w) \mid (x_0, w) \in R \text{ or } (x_1, w) \in R\}$$

P proves they know a witness for x0 or x1

One of the public values is in the relation with w

No restriction on the other, could be anything

Does not have to be in the language

The HVZK definition does not tell us how to deal with this

The proof of completeness breaks down

The solution

$$R_{\mathrm{OR}} = \{((x_0, x_1), w) \mid (x_0, w) \in R \text{ or } (x_1, w) \in R\}$$

We need the simulator to always output a valid conversation.

With this condition the proof is valid → Change of definition

Have we just made this up? Surely this has been noticed before?

↓

Cramer's thesis, where Σ-protocols began

Seems like all Σ-protocols have this property

Defining HVZK - Cramer's PhD thesis

In the middle of page 27
we find what we want!

We now define a particular and weaker variant of zero knowledge. (A, B) is said to be *honest verifier (perfect) zero-knowledge* if it is easy to (perfectly) simulate conversations with an honest verifier. If, additionally, the simulator works by taking any uniformly chosen challenge c as input and outputs an accepting conversation where c is the challenge (an accepting conversation (x, a, c, r)), then (A, B) is said to be *special honest verifier (perfect) zero-knowledge*. In this case as well as in that of ordinary honest verifier zero knowledge, we need to define the behaviour of M when it is given $x \notin RX$. This is only for reasons of completeness of some of the protocols to follow. If $x \notin RX$, then we just assume that $M(x)$ (or $M(x, c)$ respectively) either outputs an accepting conversation (where c is the challenge), or outputs '?'. No conditions on the output distribution are required in this case.

(perfect) zero-knowledge. In this case as well as in that of ordinary honest verifier zero knowledge, we need to define the behaviour of M when it is given $x \notin RX$. This is only for reasons of completeness of some of the protocols to follow. If $\boxed{x \notin RX,}$ then we just assume that $M(x)$ (or $M(x, c)$ respectively) either outputs an accepting conversation (where c is the challenge), or outputs '?'. No conditions on the output distribution are required in this case.

The condition when not in the relation

Even mentions why it is needed (completeness)

This part of the definition does not appear to have filtered into the modern (textbook) literature.

Shows need for Halevi's call to be continued….

Hopefully this case study from our work has shown how formal methods can catch subtle points

Rest of this talk:

1. **Process of formalisation**
   a. **Defining security**
   b. **Instantiating definitions**
2. **Promote discussion**

Isabelle and CryptHOL

- CryptHOL is a framework for crypto inside Isabelle/HOL.
- Provides a probabilistic programming framework.
  - So far used for game-based and simulation-based security.

```
definition init :: "'grp pub_in ⇒ witness ⇒ (rand × 'grp msg) spmf"
  where "init h w = do {
    r ← sample_uniform (order 𝒢);
    return_spmf (r, g [^] r)}"
```

Probabilistic program describing the Initial message sent in Schnorr Σ-Protocol.

Defining Security in CryptHOL

- We use Isabelle's module system to fix the parameters for our definitions.

```
locale Σ_protocols_base =
  fixes init :: "'pub_input ⇒ 'witness ⇒ ('rand × 'msg) spmf"
    and response :: "'rand ⇒ 'witness ⇒ 'challenge  ⇒ 'response spmf"
    and check :: "'pub_input ⇒ 'msg ⇒ 'challenge ⇒ 'response ⇒ bool"
    and Rel :: "('pub_input × 'witness) set"
    and S :: "'pub_input ⇒ 'challenge ⇒ ('msg × 'challenge × 'response) spmf"
    and 𝒜ss :: "('pub_input, 'msg, 'challenge, 'response, 'witness) prover_adversary"
    and challenge_space :: "'challenge set"
    and valid_pub :: "'pub_input set"
  assumes domain_subset_valid_pub: "Domain Rel ⊆ valid_pub"
begin
```

- Using these parameters we can make our security definitions. For example:

```
definition "HVZK ≡ (∀e ∈ challenge_space.
    (∀(h, w)∈ Rel. R h w e = S h e)
        ∧ (∀h ∈ valid_pub. ∀(a, e', z) ∈ set_spmf (S h e). check h a e z))"
```

The extra requirement

The one technical slide: Instantiating our abstract definitions

We instantiate the abstract modules for protocols we want to consider

```
sublocale Schnorr_Σ: Σ_protocols_base initS responseS checkS R_DL SS 𝒜S challenge_space valid_pub
  unfolding Σ_protocols_base_def by(simp add: R_DL_def valid_pub_def; blast)
```

We then prove the properties of the instantiation

```
lemma Schnorr_HVZK: shows "Schnorr_Σ.HVZK"
```

Eventually showing the final result

```
theorem Schnorr_Σ_protocol: shows "Schnorr_Σ.Σ_protocol"
```

Modularisation comes naturally

As in paper proofs, we want to make assumptions on underlying protocols.

Isabelle's module system is well developed

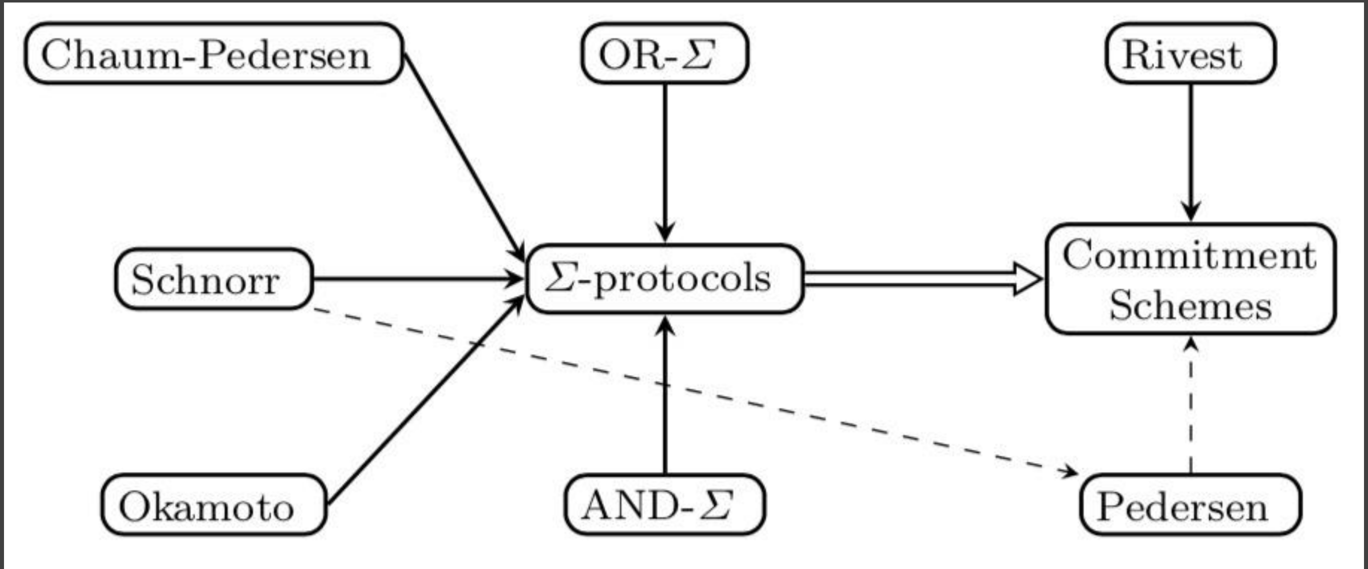
```
locale Σ_OR_base =
      Σ0: Σ_protocols_base init0 response0 check0 Rel0 S0 Ass0 "carrier L" valid_pub0
    + Σ1: Σ_protocols_base init1 response1 check1 Rel1 S1 Ass1 "carrier L" valid_pub1
  for init0 response0 check0 Rel0 S0 Ass0 challenge_space0 valid_pub0
  and init1 response1 check1 Rel1 S1 Ass1 challenge_space1 valid_pub1
  and L :: "'bool boolean_algebra" (structure)
    +
  assumes Σ_prot1: "Σ1.Σ_protocol"
    and Σ_prot0: "Σ0.Σ_protocol"
    and finite_L: "finite (carrier L)"
    and carrier_L_not_empty: "carrier L ≠ {}"
begin
```

We think it is very readable

Fix the two underlying protocols

Make the assumptions we need --- that they form Σ-protocols.

Our formalisation of Σ-Protocols and Commitment Schemes



Two abstract definitional ‘frameworks’

Numerous instantiations and constructions

Future directions of Formal Methods and this work

- **What is most beneficial from a formalisation effort?**
  - Definitions and basic case studies.
  - More complex constructions, modularisation.

Trade off with proof effort

Consider the HVZK definition we encountered.

1. We formalized the definitions from the literature (i.e. widely used textbooks/papers).
2. Everything was fine when we considered standalone Σ-protocols (e.g. Schnorr, Okamoto etc).
3. Noticed a problem however, with the OR compound constructions --- the proofs would not go through.
4. This led us to re-examine the definitions and paper proofs.

Future directions of Formal Methods and this work

Machine-paper hybrid proofs

- Do not have to have fully mechanised proofs
- Formal proof could focus on parts where intuition breaks down
- Good prototyping environment, do not need complete proofs
- The paper proof could consider the ‘safe’ parts of the proof.

Future directions of Formal Methods and this work

Influencing standardisation

Formal methods often used after standard is set

E.g. TLS was formally modelled

Resulted in change of standard

Integrating formal methods into process can help

Iterative approach will result in more robust standardisation

Future directions of Formal Methods and this work

- **What is most beneficial from a formalisation effort?**
  - Definitions and basic case studies.
  - More complex constructions, modularisation.

  Trade off with proof effort

- **Machine-paper hybrid proofs:**
  - Do not have to have fully mechanised proofs.
  - Focus on aspects where intuition is known to break down.

- **Influencing standardisation:**
  - Often formal methods is used after a standard is set.
  - Perhaps a more iterative method could result in more robust standardisation.