Formally Verified Cryptographic Proof Systems in Lean

Quang Dao Devon Tuma Gregor Mitscha-Baude

November 11, 2024

Chapter 1

Introduction

The goal of this project is to formalize Succinct Non-Interactive Arguments of Knowledge (SNARKs) in Lean. Our focus is on SNARKs based on Interactive Oracle Proofs (IOPs). We plan to build a general framework for IOP-based SNARKs that can state specifications of the protocols and prove their security properties in a clean and modular way.

Chapter 2

Oracle Reductions

2.1 Definitions

Definition 1 (Type Signature of an Interactive Protocol)**.** An *n-message interactive protocol* between two parties P and V consists of a sequence of messages $m_0, \ldots, m_n$, where each message m_i is associated with a *direction* (to P or to V), and a *type*. The specification of such a protocol is given by an object of type `ProtocolSpec (n : Nat) := Fin n → (Direction × Type)`.

This definition generalizes similar ones in the literature in that we do not fix a particular protocol flow (e.g. alternating between prover's and verifier's messages). This is meant to capture all protocols in the most generality, and have some benefits in composition.

In the interactive protocols we consider, both parties P and V may have access to a shared oracle O. An interactive protocol becomes an *interactive (oracle) reduction* if its execution reduces an input relation R_{in} to an output relation R_{out}. Here a relation is just a function $\mathsf{IsValid} : \mathsf{Statement} \times \mathsf{Witness} \to \mathsf{Bool}$, for some types `Statement` and `Witness`. We do not concern ourselves with the running time of $\mathsf{IsValid}$ in this project (though future extensions may prove that relations can be decided in polynomial time, for a suitable model of computation).

Definition 2 (Type Signature of a Prover)**.** A prover P in an interactive protocol, with access to a shared oracle O with the verifier, is a stateful oracle computation that at each step of the protocol, either takes in a new message from the verifier, or sends a new message to the verifier.

Our modeling of interactive protocols only consider *public-coin* verifiers; that is, verifiers who only outputs uniformly random challenges drawn from the (finite) types, and uses no other randomness. Because of this fixed functionality, we can bake the verifier's behavior in the interaction phase directly into the protocol execution semantics. For the rest of the documentation, we will omit the term public-coin.

After the interaction phase, the verifier may then run some verification procedure to check the validity of the prover's responses. This procedure differs depending on whether the verifier has full access, or only oracle access, to the prover's messages. Note that there is no difference on the prover side whether the protocol is an *interactive oracle reduction (IOR)* or simply an *interactive reduction (IR).*

Definition 3 (Type Signature of a Verifier)**.**

Definition 4 (Type Signature of an Oracle Verifier)**.**

Finally, we come to the definition of an interactive (oracle) protocol, which is a combination of its type signature, a prover for that type signature, and an (oracle) verifier for that type signature.

Definition 5 (Interactive Reduction)**.** An interactive reduction is a combination of a type signature `ProtocolSpec`, a prover for `ProtocolSpec`, and a verifier for `ProtocolSpec`.

Definition 6 (Interactive Oracle Reduction)**.** An interactive oracle reduction is a combination of a type signature `ProtocolSpec`, a prover for `ProtocolSpec`, and an oracle verifier for `ProtocolSpec`.

We can now define properties of interactive reductions. The two main properties we consider in this project are completeness and various notions of soundness. We will cover zero-knowledge at a later stage.

Definition 7 (Completeness)**.**

For soundness, we need to consider different notions, such as (plain) soundness, knowledge soundness, *round-by-round* soundness, and round-by-round knowledge soundness. The last notion is the strongest and is necessary for security of the SNARK obtained from the interactive protocol after composing with a commitment scheme and applying the Fiat-Shamir transform.

Definition 8 (Soundness)**.**

Definition 9 (Knowledge Soundness)**.**

To define round-by-round (knowledge) soundness, we need to define the notion of a *state function*. This is a (possibly inefficient) function `StateF` that, for every challenge sent by the verifier, takes in the transcript of the protocol so far and outputs whether the state is doomed or not. Roughly speaking, the requirement of round-by-round soundness is that, for any (possibly malicious) prover P, if the state function outputs that the state is doomed on some partial transcript of the protocol, then the verifier will reject with high probability.

Definition 10 (State Function)**.**

Definition 11 (Round-by-Round Soundness)**.**

Definition 12 (Round-by-Round Knowledge Soundness)**.**

2.2 Composition

The reason why we consider interactive (oracle) reductions at the core of our formalism is that we can *compose* these reductions to form larger reductions. Equivalently, we can take a complex *interactive (oracle) proof* (which differs only in that it reduces a relation to the *trivial* relation that always outputs true) and break it down into a series of smaller reductions. The advantage of this approach is that we can prove security properties (completeness and soundness) for each of the smaller reductions, and these properties will automatically transfer to the larger reductions.

This section is devoted to the composition of interactive (oracle) reductions, and proofs that the resulting reductions inherit the security properties of the two (or more) constituent reductions.

Chapter 3

Commitment Schemes

3.1 Definitions

3.2 Composition

Chapter 4

Proof Systems

4.1 The Sum-Check Protocol

4.2 The Spartan Protocol

4.3 The Ligero Polynomial Commitment Scheme

Chapter 5

Supporting Results

5.1 Polynomials

Definition 13 (Multilinear Extension)**.**

Theorem 14 (Multilinear Extension is Unique)**.**

5.2 Coding Theory

Definition 15 (Code Distance)**.**

Definition 16 (Distance from a Code)**.**

Definition 17 (Generator Matrix)**.**

Definition 18 (Parity Check Matrix)**.**

Definition 19 (Interleaved Code)**.**

Definition 20 (Reed-Solomon Code)**.**

Definition 21 (Proximity Measure)**.**

Definition 22 (Proximity Gap)**.**

Chapter 6

References