Lecture 10: zk-SNARK Primer I

Lecturer: Shumo Chu *Scribes: Gwyneth Allwright, Lianke Qin*

10.1 Recap of Previous Discussions

- Layer 1:
  - Consensus.
  - Smart contracts.
- Layer 2:
  - Oracles.
- Universal verifiability: all data is public.
- Assets are controlled by signatures. Secret keys can be thought of as specialized zero-knowledge proof systems.

10.2 Introduction to zk-SNARKs

- The main benefit of zk-SNARKs is privacy. zk-SNARKs allow us to encrypt public data on blockchains.
- Toy example: we move from state `i` to state `i+1` on the blockchain using the transaction T.
  - We encrypt `i`, `i+1` and T.
  - We wish to verify that T is a sound transaction for state `i` and that it produces state `i+1`.
  - This verification should take the form of a zero-knowledge proof Π.

10.3 Introduction to Circuits

- Boolean circuits are composed using the binary operators AND, OR and NOT.
  - $\text{AND}(x, y) = x \cdot y$
  - $\text{OR}(x, y) = x + y - x \cdot y$
  - $\text{NOT}(x) = 1 - x$
  - We can represent any boolean function as a circuit $C : \{0,1\}^N \longrightarrow \{0,1\}$.
- Arithmetic circuits are mappings $C : F_p^N \longrightarrow F_p$, where F_p is a finite field.
  - Intuitively, finite fields are finite sets of objects with certain operations defined on them.
  - For example, there is an addition operation. A property of finite fields is that

$$a, b \in F_p \implies a + b \in F_p. \tag{10.1}$$

  - Arithmetic circuits can be represented using directed acyclic graphs (DAGs).
  - You can express boolean circuits as arithmetic circuits (but it may be inefficient to do so).

- The link between circuits and hash functions is as follows. Imagine that we have a circuit

$$C_{\text{hash}} : (h, m) \longrightarrow \{0, 1\}, \tag{10.2}$$

which is zero if $h = H(m)$ and unity otherwise, where H is a hash function (e.g. SHA256).

10.4 Milestones in the History of Proof Systems

- In 1992, it was proven that the complexity class IP is equivalent to the class PSPACE (Fortnow, Karloff, Nisan, Shamir).
- The PCP theorem was proven (Arora, Lund, Safra, Sudan). Idea: any NP statement (anything expressible in a circuit) has a probabilistically checkable proof such that this proof can be verified in polynomial time relative to the size of the classic proof.
- Developments in non-interactive proof systems (Kilian, Micali, Groth).

10.5 Illustration of a Scheme for Zero-Knowledge Proofs

- In this scenario, we have a prover (Alice) and a verifier (Bob).
- Imagine that there are two finite field elements $x \in F_p^n$ and $w \in F_p^m$.
  - x is public, while w (the witness) is private (known only to Alice).
- Alice uses x and w to compute the arithmetic circuit $C(x, w) \in F_p$.
  - The circuit C is public.
  - Note that C could be a representation of a boolean circuit. We will assume that it evaluates to `true`.
- Alice wishes to prove the following to Bob:
  - Soundness: there exists a w such that $C(x, w)$ evaluates to `true`.
  - Knowledge: Alice knows the w in question.
- Why can't Alice just send w to Bob? Three reasons:
  - w needs to be private.
  - w may be too long to send.
  - $C(x, w)$ may be inefficient to compute.
- Instead of sending w, Alice will send a zero-knowledge proof Π. This will be constructed as follows:
  - The key elements of the construction are mappings S, P and V. S is used to generate keys, while P and V are used for proving and verifying respectively.
  - First, we act S on the circuit C to obtain a pair of keys, p_k (proving key) and v_k (verification key).

- Alice computes $P(p_k, x, w)$ to obtain a proof Π.
- Bob computes $V(v_k, x, \Pi)$ to obtain either `true` (if the proof was correct) or `false`.
- In order for this scheme to be successful, we require that

$$V(v_k, x, \Pi) = \texttt{true} \implies V(v_k, x, P(p_k, x, w)) = \texttt{true}, \tag{10.3}$$

  as well as the fact that $V(v_k, x, \Pi) = \texttt{true} \implies$ Alice knows w such that $C(x, w)$ is `true`.
- This proof is zero-knowledge: Π and x reveal nothing about w.

10.6 Proof of Knowledge

In cryptography, a proof of knowledge is an interactive proof in which the prover succeeds in "convincing" a verifier that the prover knows something. What it means for a machine to "know something" is defined in terms of computation. Knowledge extractor is introduced to capture this idea.

- π : its size could be 2KB.
- P is used for proving
- V is used for verifying
- S is used to generate keys.
- Extractor

Then we say $(S,\ P,\ V)$ is a proof of knowledge that satisfies:

- $\forall$ c, s.t. $\forall$ unbounded adversary. $A = (A_0, A_1)$
  - $(pk, vk) := S(C)$
  - $(x, st) := A_0(pk)$ is to generate a fake w
  - $\pi' := A_1(pk, x, st)$ is to generate a fake π
- s.t. $Pr[V(vk, x, \pi'] = true] > negl(\cdot)$
- There is an efficient Extractor(use A as black box),
  - $(pk, vk) := S(C)$, $(x, st) := A_0(pk)$
  - $w := E(pk, x, st)$
  - $Pr[C(x, w) = 0] > negl(\cdot)$
- Argument of knowledge : A is $poly(\cdot)$

10.7 Zero Knowledge

By zero knowledge we want to ensure :

- $(x,\ \pi)$ did not reveal w.

- $(S,\ P,\ V)$ is a zero knowledge proof for circuit C.
- There is an efficient simulator s.t.:
  - $\forall\ x \in F_p^n,\ \exists\ w : C(x, w) = 0$, we have :
  - $(pk,\ vk,\ x,\ \pi)$ where $(pk, vk) := S(C)$ and $\pi := P(pk, x, w)$ is indistinguishable with :
  - $(pk,\ vk,\ x,\ \pi)$ where $(pk, vk, \pi) := Sim(x)$
  - This means that $Sim(x)$ can simulate π without w

10.8 zk-SNARK protocols

The acronym zk-SNARK stands for "Zero-Knowledge Succinct Non-Interactive Argument of Knowledge," and refers to a proof construction where one can prove possession of certain information without revealing that information, and without any interaction between the prover and verifier.

- Zero-Knowledge has the meaning explained above: nothing is revealed beyond truth of statement to the verifier.
- Succinct means that the proof is tiny compared to the computation.
  - The proof size is constant $O(1)$.
  - Verification time is $O(1)$ and does not depend on the running time of f.
- Non-interactive means that we can write and store a proof, without the need to have question/answer cycles. So a proof can be computed and published and everyone can verify it.
- ARgument of Knowledge means that soundness is guaranteed only against a computationally bounded server and the proof cannot be constructed without access to a witness.

There is a simple comparison among several popular zk-SNARK protocols below:

	Size of π	Size of pk	verification time	setup
Groth16 [1]	O(1)	O(log$\lvert C\rvert$)	O(1)	Yes/Per circuit
PLONK [2]	O(1)	O(log$\lvert C\rvert$)	O(1)	Yes/Update
STARK [3]	O(log$\lvert C\rvert$)	O(1)	O(log$\lvert C\rvert$)	No

Table 10.1: Mostly recently used zk-SNARK protocols

References

[1] Groth, Jens. " the size of pairing-based non-interactive arguments." Annual international conference on the theory and applications of cryptographic techniques. Springer, Berlin, Heidelberg, 2016.

[2] Gabizon, Ariel, Zachary J. Williamson, and Oana Ciobotaru. "PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge." IACR Cryptol. ePrint Arch. 2019 (2019): 953.

[3] Ben-Sasson, Eli, et al. "Scalable, transparent, and post-quantum secure computational integrity." IACR Cryptol. ePrint Arch. 2018 (2018): 46.