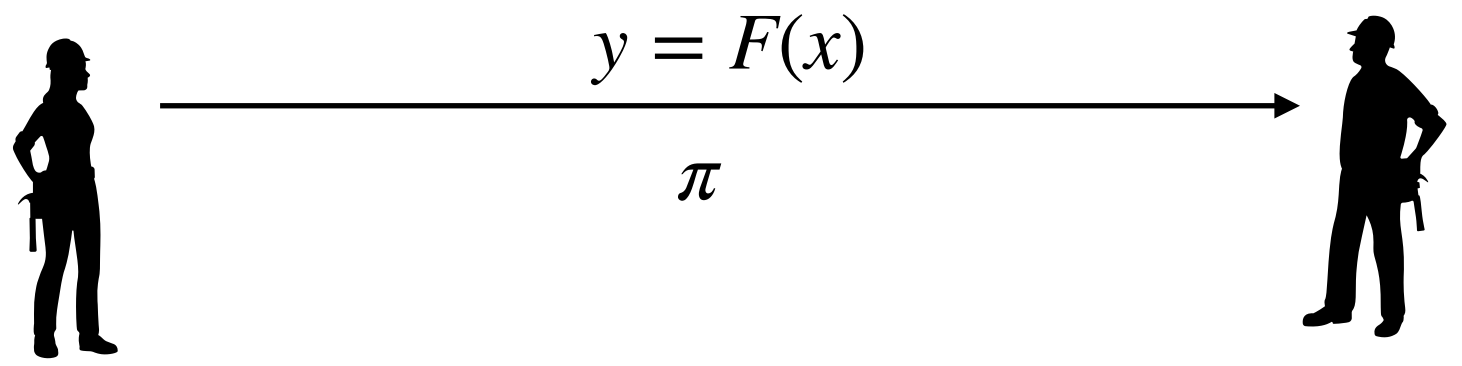
How to Bulid SNARK

Shuangjun Zhang

SNARK

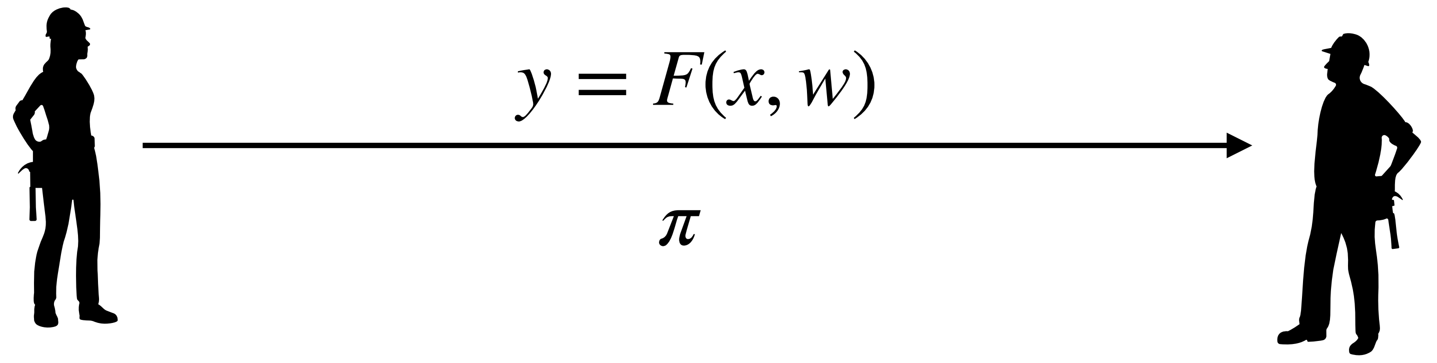
Succinct Non-Interactive ARguments of Knowledge

Cryptographic proofs for computation integrity
that are super short and are super fast to verify.



π : prove y is indeed the result of $F(x)$

Prover may use some private data w, the protocol should be zero-knowledge.



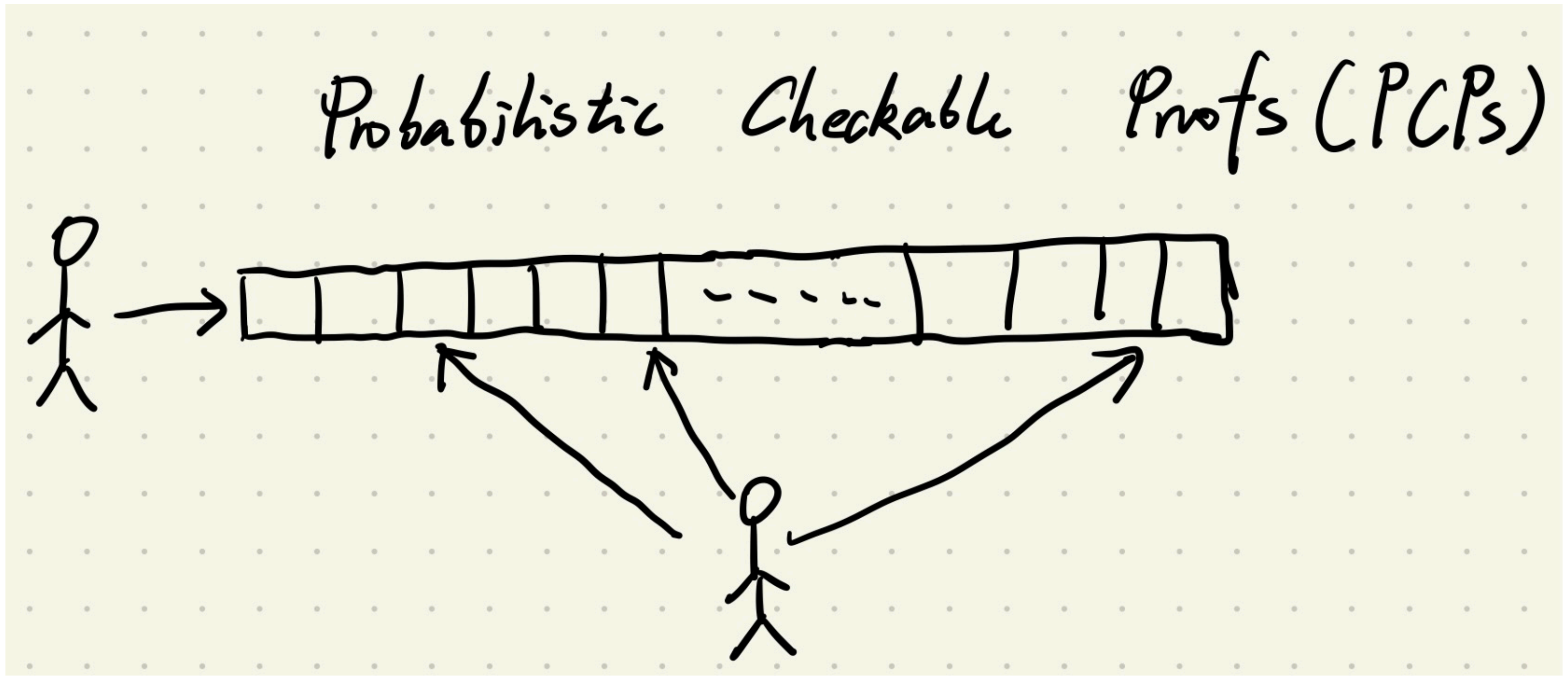
π: prove y is the correct answer of $F(x, w)$ without reveal any information of w

zkSNARK: SNARK with zero-knowledge

SNARK from PCP

The First SNARK

The first SNARK comes from PCP(Probabilistic Checkable Proof)



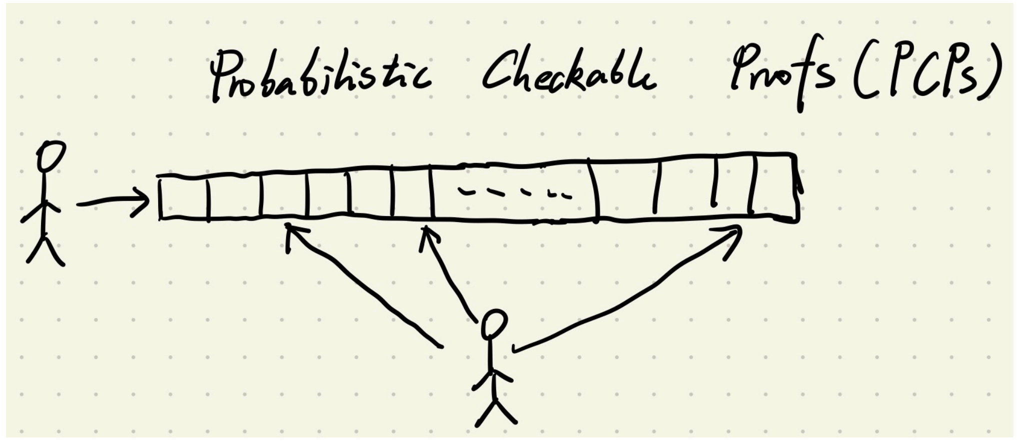
Check a few locations of the proof, but the proof is large.

PCP theorem[ALMSS98]: any NP problem can be verified
by just querying few bits (3 is enough) from the proof.

SNARK from PCP

Attempt 1: Send full proof

Not succinct, the proof is too large.

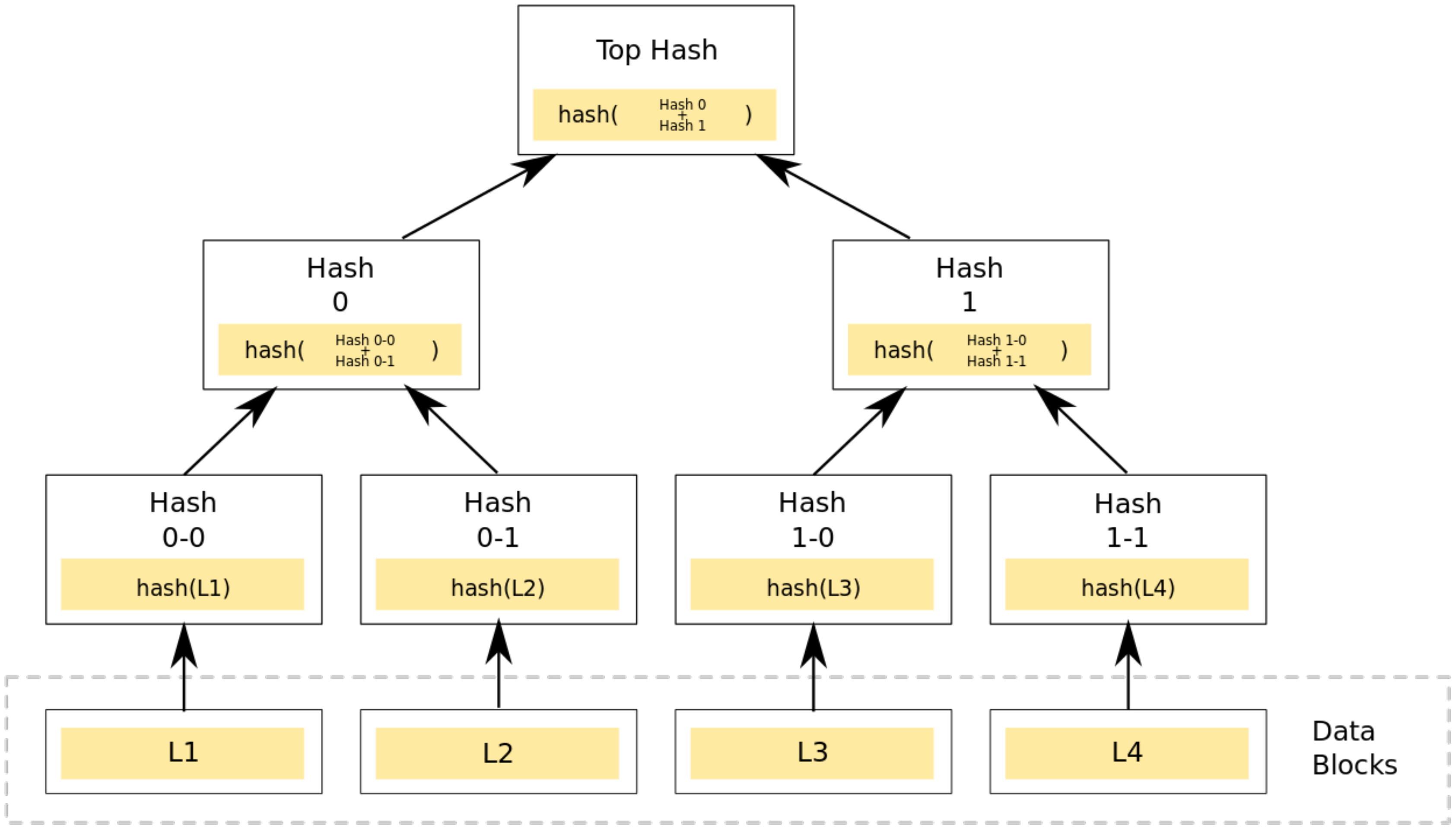


Attempt 2: Send q_1, q_2, q_3 to Prover. P replies with a_1, a_2, a_3.

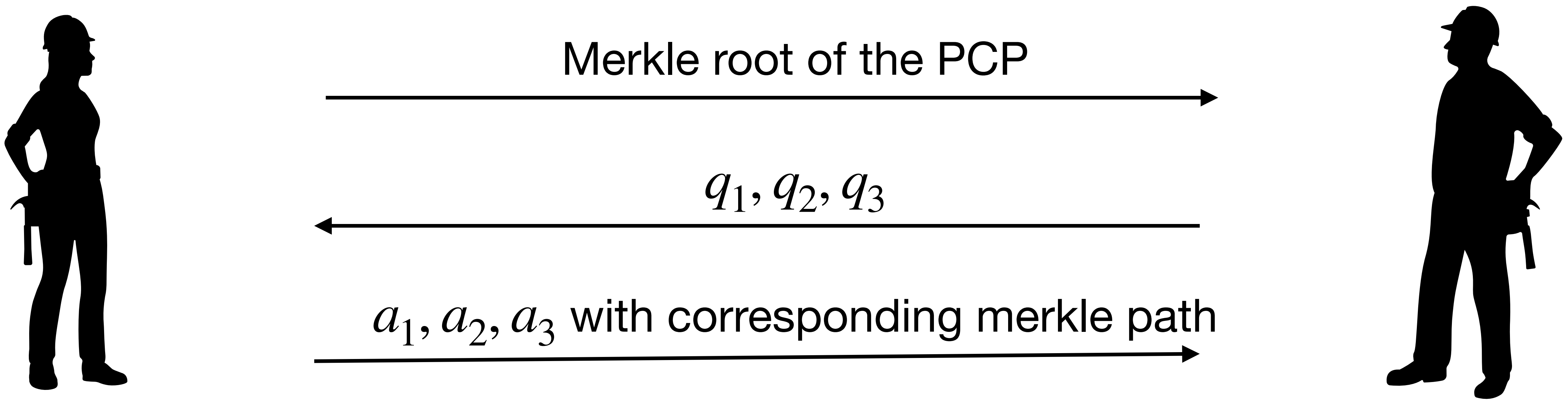
The prover may cheat. For example, q_1 is the query for 7-th location of the proof, but a_1 is the 9-th location.

Any Idea? (Do not consider non-interactive now)

Merkle Tree


Top Hash
hash(Hash 0 + Hash 1)
Hash 0
hash(Hash 0-0 + Hash 0-1)
Hash 1
hash(Hash 1-0 + Hash 1-1)
Hash 0-0
hash(L1)
Hash 0-1
hash(L2)
Hash 1-0
hash(L3)
Hash 1-1
hash(L4)
L1
L2
L3
L4
Data Blocks

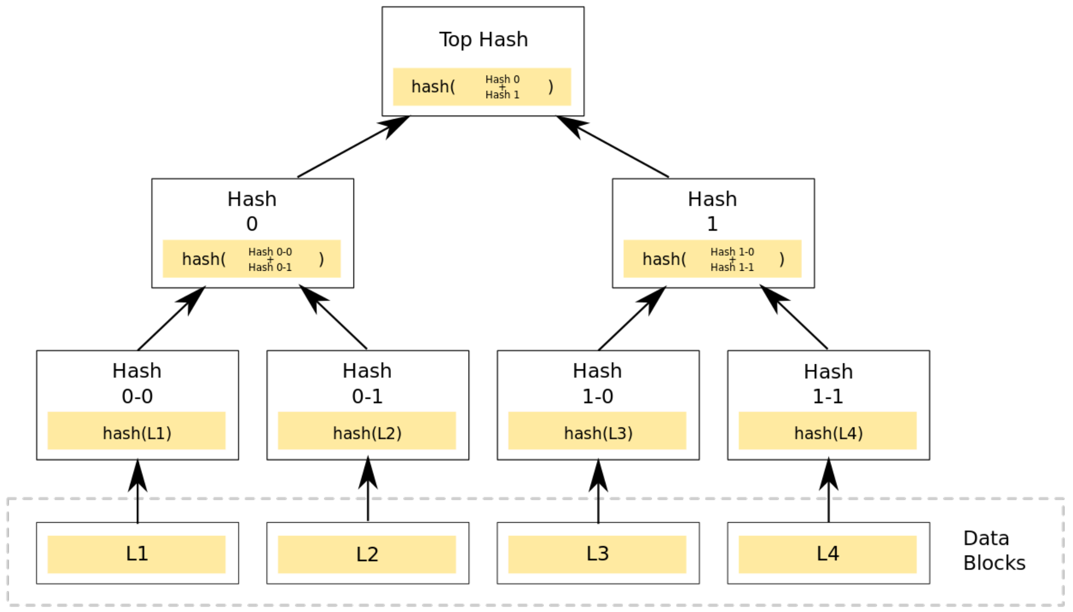
SNARK from PCP



Suppose the length of PCP is l and we use SHA256.

The Merkle root has length 256 bit.

The length of merkle path is $O(\log l)$

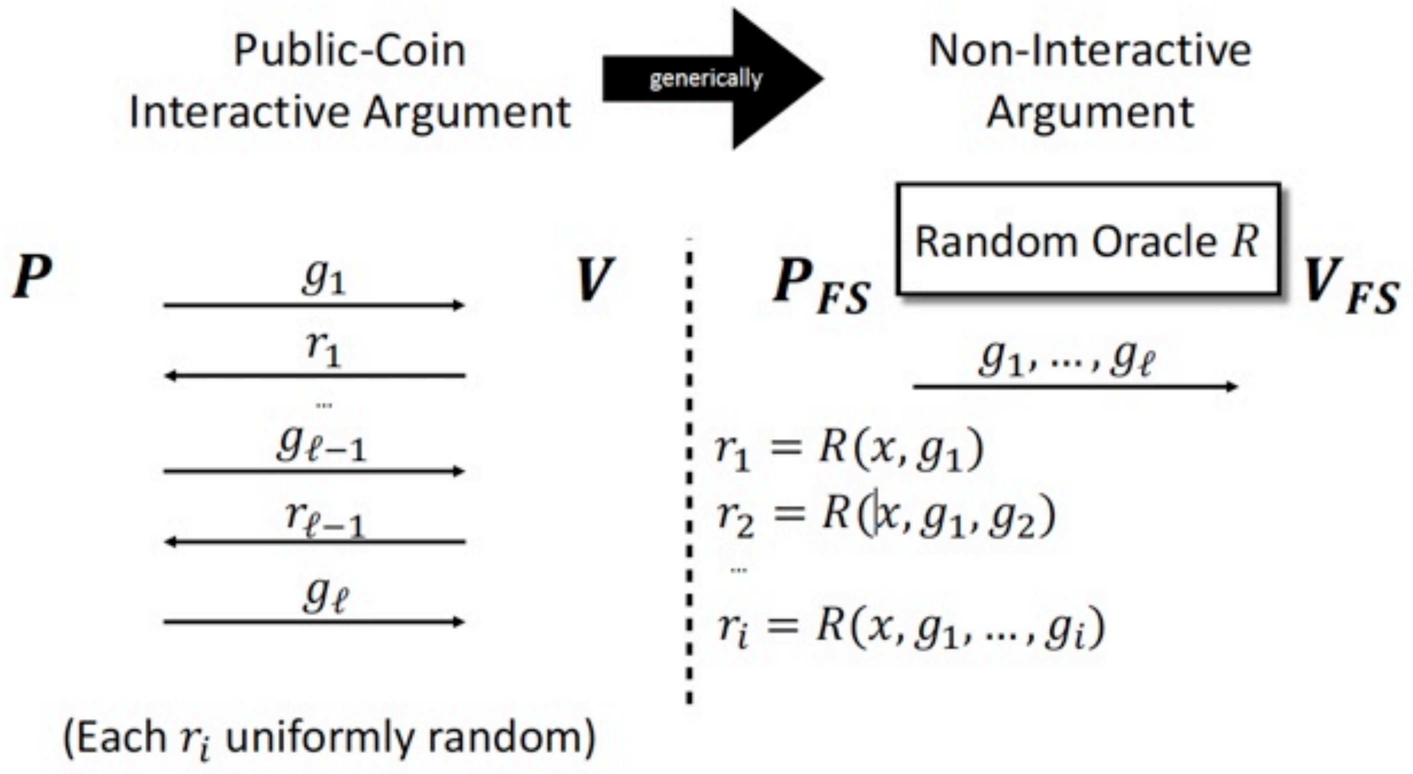


Non-Interactive

Any public-coin protocol can be made non-interactive by Fiat-Shamir Transformation

Public-coin protocol: All the messages sent by verifier are uniformly random and public.

For example, Graph Non-Isomorphism protocol is not public-coin, Sum-Check protocol is public-coin.



We can view R as an ideal hash function.

Example: Schnorr Signature

G is a cyclic group, with $|G| = p$, g is a generator.

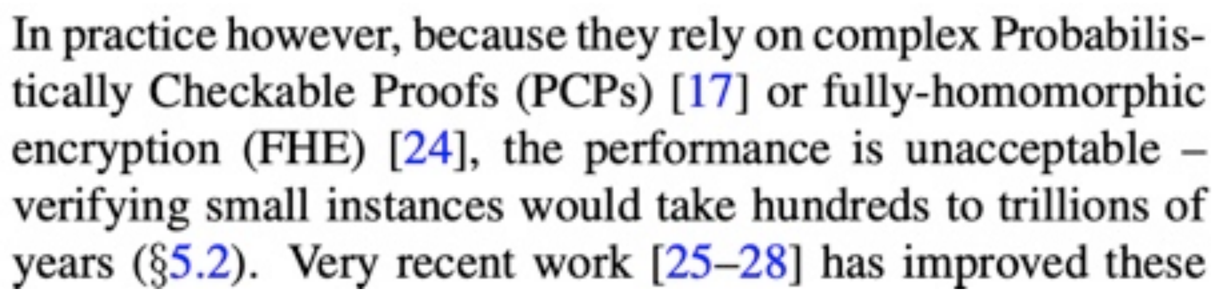
$h = g^x$ is the common input, x is Prover P's secret.
P proves he know the secret x in zero knowledge.

$$\begin{tikzcd}
r \leftarrow \{0,1,2,\ldots,p-1\} & & \\
a = g^r \arrow[rr, "a"] & & e \leftarrow \{0,1,2,\ldots,p-1\} \\
{} & & {} \arrow[ll, "e"'] \\
y = x \cdot e + r \arrow[rr, "y"] & & a \cdot h^e = g^y
\end{tikzcd}$$

$$\begin{tikzcd}
r \leftarrow \{0,1,2,\ldots,p-1\} & & \\
a = g^r & & \\
e = H(x,a) & & \\
y = x \cdot e + r \arrow[rr, "{a, e, y}"] & & \text{Check that } e = H(x,a) \\
& & a \cdot h^e = g^y
\end{tikzcd}$$

PCP is bad

We do not know how to construct PCP efficiently!



In practice however, because they rely on complex Probabilistically Checkable Proofs (PCPs) [17] or fully-homomorphic encryption (FHE) [24], the performance is unacceptable – verifying small instances would take hundreds to trillions of years (§5.2). Very recent work [25–28] has improved these

Despite all this, PCP theorem is a gem in Theoretical Computer Science.

Main application of PCP is Hardness of Approximation.

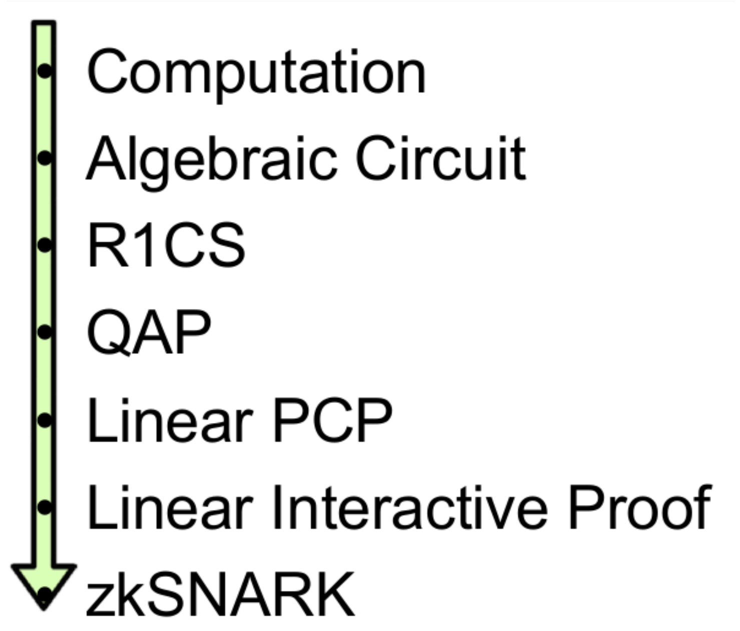
Linear PCP

Variations of PCP: Linear PCP

The most popular SNARK (used in Zcash) comes from Linear PCP

The proof is $\pi \in \mathbb{F}^l$

The query of verifier is also a vector $q \in \mathbb{F}^l$, the answer is $a = \langle \pi, q \rangle$.



Quadratic Arithmetic Program (QAP)

Input: $\{L_i(x)\}_{i=1}^n, \{R_i(x)\}_{i=1}^n, \{O_i(x)\}_{i=1}^n$,with degree $n-1$

$$x = (z_1, z_2, \ldots, z_m)(m < n)$$

$$t(x) = (x - r_1)(x - r_2)\cdots(x - r_n)$$

Output: Yes if there exists $w = (z_{m+1}, z_{m+2}, \cdots, z_n)$ such that

$$\left(\sum_{i=1}^{n} z_i L_i(x)\right) \cdot \left(\sum_{i=1}^{n} z_i R_i(x)\right) \equiv \left(\sum_{i=1}^{n} z_i O_i(x)\right) \mod t(x)$$

Linear PCP for QAP

$$\left(\sum_{i=1}^{n} z_i L_i(x)\right) \cdot \left(\sum_{i=1}^{n} z_i R_i(x)\right) \equiv \left(\sum_{i=1}^{n} z_i O_i(x)\right) \mod t(x)$$

If and only if $\exists$ a polynomial $h(x)$ with degree less than $n-2$ such that

$$\left(\left(\sum_{i=1}^{n} z_i L_i(x)\right) \cdot \left(\sum_{i=1}^{n} z_i R_i(x)\right) - \left(\sum_{i=1}^{n} z_i O_i(x)\right)\right) = t(x) \cdot h(x)$$

Let $h(x) = h_0 + h_1 x + h_2 x^2 + \cdots + h_{n-2} x^{n-2}$

The proof $\pi = (z_1, z_2, \cdots, z_n, h_0, h_1, \cdots, h_{n-2}) \in \mathbb{F}^{2n-1}$

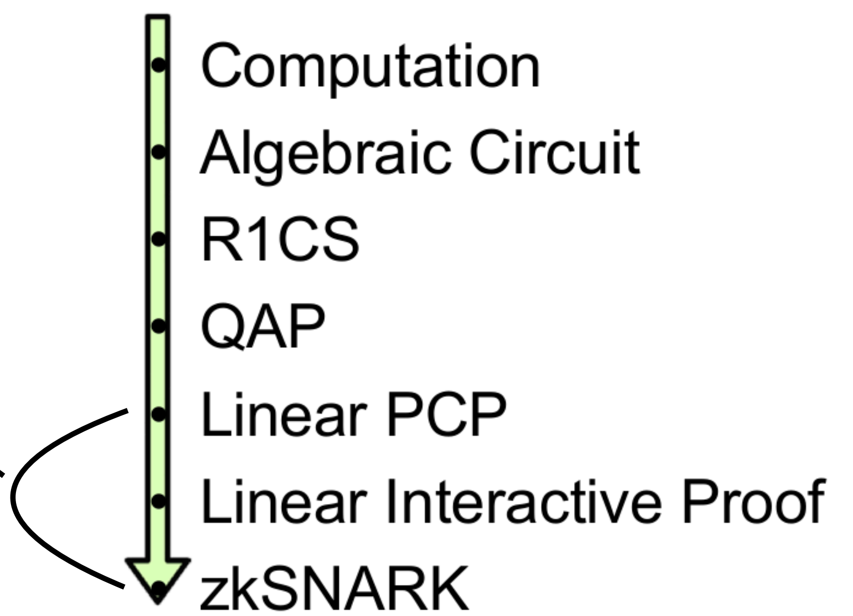
The Verifier:

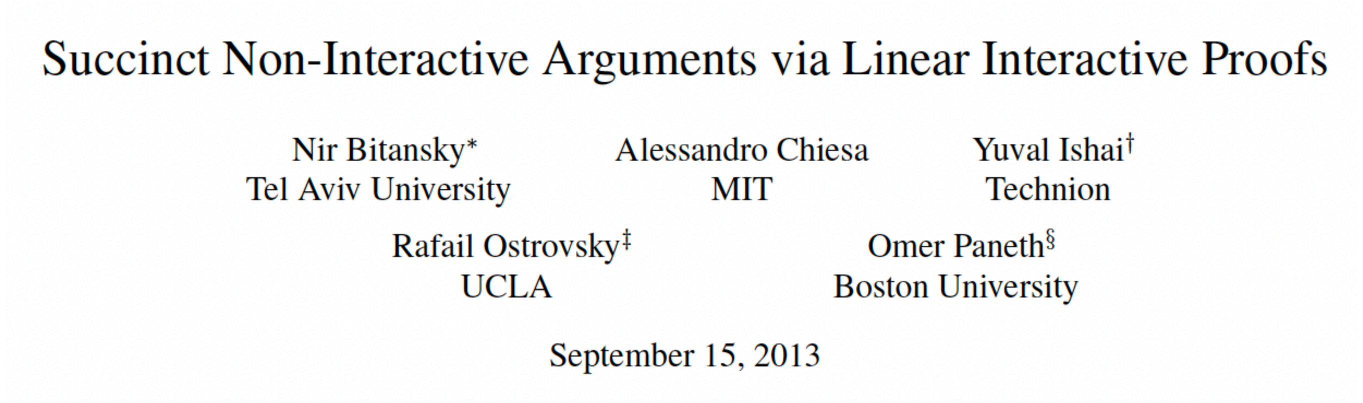
1. Sample $s \leftarrow \mathbb{F}$.
2. Query π at $q_1 = (L_1(s), L_2(s), \cdots, L_n(s), 0, \cdots, 0)$, suppose the answer is A.
3. Query π at $q_2 = (R_1(s), R_2(s), \cdots, R_n(s), 0, \cdots, 0)$, suppose the answer is B.
4. Query π at $q_3 = (O_1(s), O_2(s), \cdots, O_n(s), 0, \cdots, 0)$, suppose the answer is C.
5. Query π at $q_4 = (0, \cdots, 0, 1, s, s^2, \cdots s^{n-2})$, suppose the answer is D
6. Check that $A \cdot B - C = D \cdot t(s)$

Query Complexity: 4

SNARK from Linear PCP

The most difficult part, many many many many many many Cryptography required (Pairing Based Cryptography) 😰😰



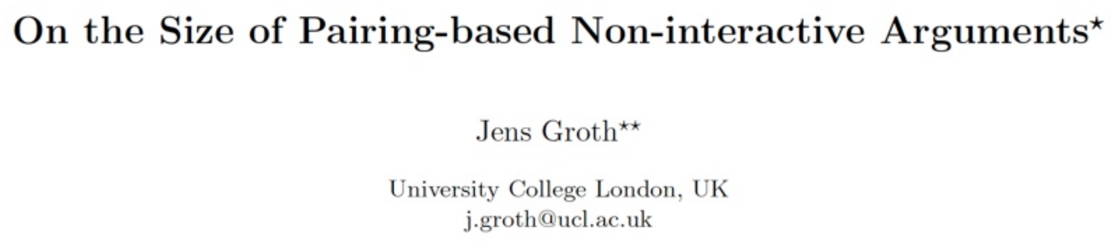

Succinct Non-Interactive Arguments via Linear Interactive Proofs

Nir Bitansky* Tel Aviv University
Alessandro Chiesa MIT
Yuval Ishai† Technion
Rafail Ostrovsky‡ UCLA
Omer Paneth§ Boston University

September 15, 2013

SNARK = Linear PCP + Pairing based cryptography

Trusted Setup phase required but very efficient[Groth16]


On the Size of Pairing-based Non-interactive Arguments*

Jens Groth**
University College London, UK
j.groth@ucl.ac.uk

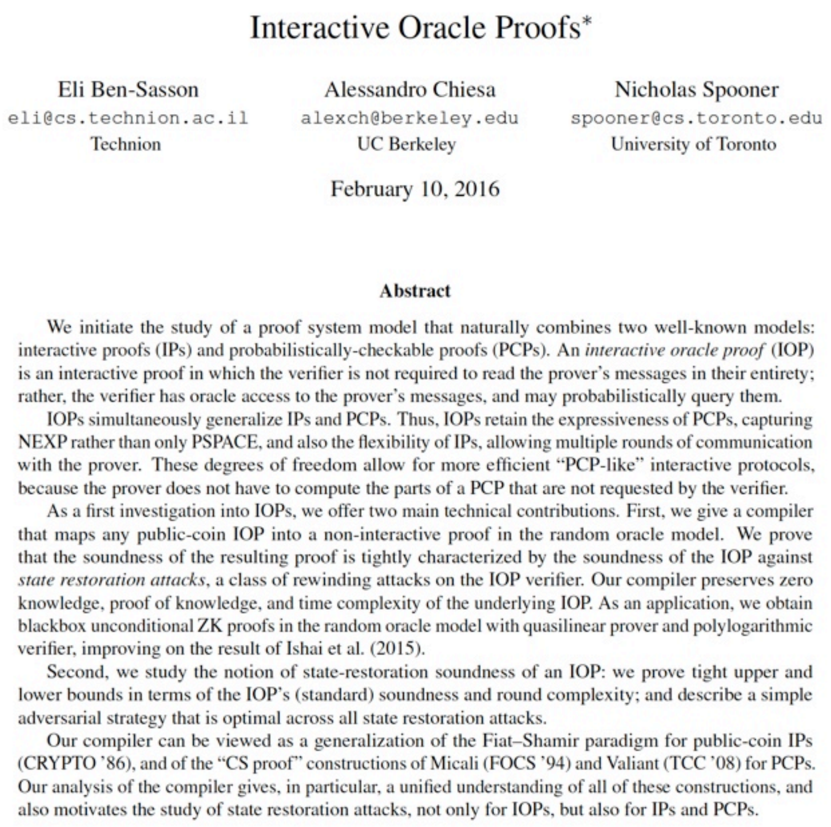
SNARK = PCP + Merkle tree

No Trusted Setup but not efficient

IOP (Interactive Oracle Proof)

Interactive Oracle Proof

IOP = IP + PCP



Interactive Oracle Proofs*

Eli Ben-Sasson
eli@cs.technion.ac.il
Technion

Alessandro Chiesa
alexch@berkeley.edu
UC Berkeley

Nicholas Spooner
spooner@cs.toronto.edu
University of Toronto

February 10, 2016

Abstract

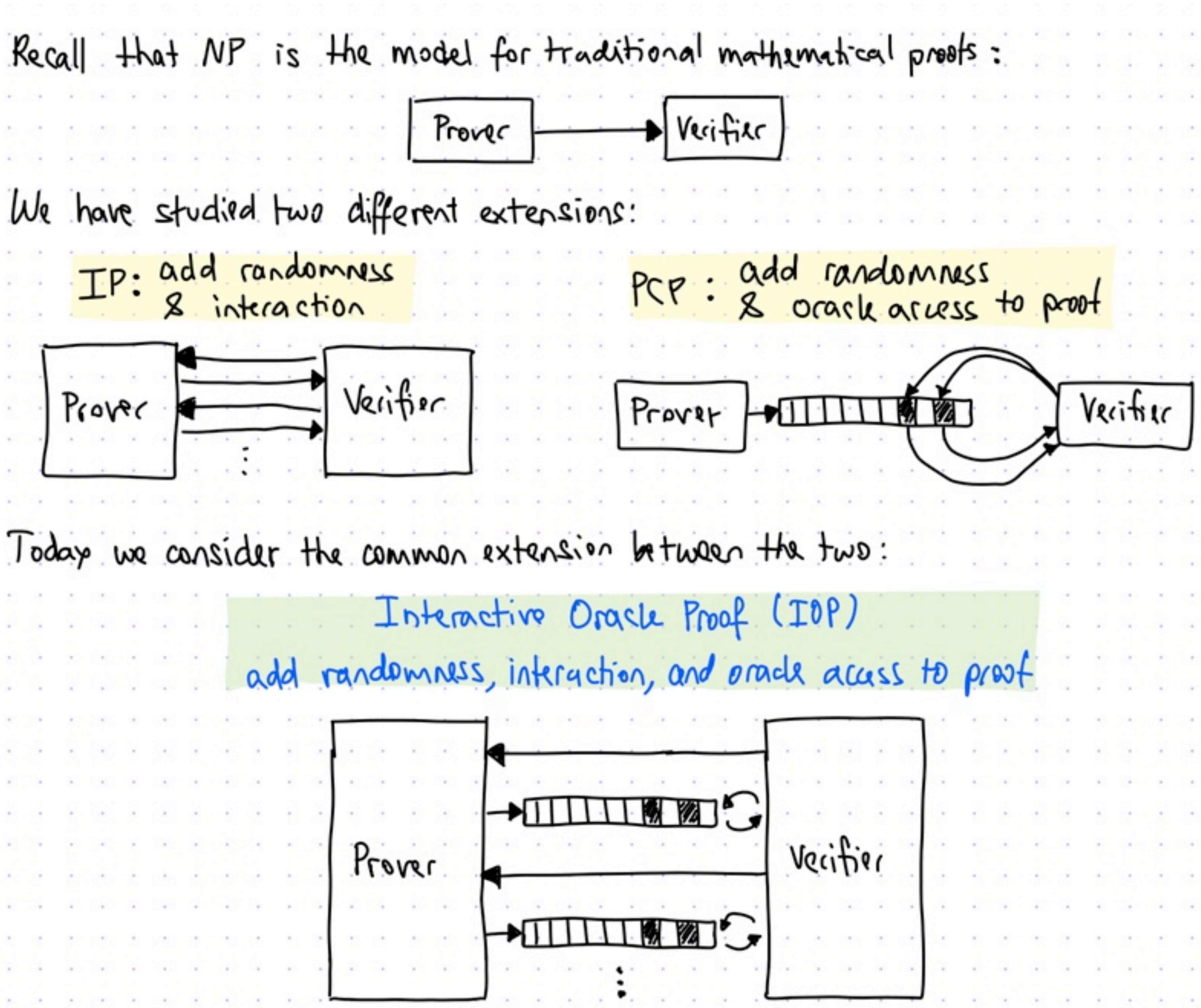
We initiate the study of a proof system model that naturally combines two well-known models: interactive proofs (IPs) and probabilistically-checkable proofs (PCPs). An *interactive oracle proof* (IOP) is an interactive proof in which the verifier is not required to read the prover's messages in their entirety; rather, the verifier has oracle access to the prover's messages, and may probabilistically query them.

IOPs simultaneously generalize IPs and PCPs. Thus, IOPs retain the expressiveness of PCPs, capturing NEXP rather than only PSPACE, and also the flexibility of IPs, allowing multiple rounds of communication with the prover. These degrees of freedom allow for more efficient "PCP-like" interactive protocols, because the prover does not have to compute the parts of a PCP that are not requested by the verifier.

As a first investigation into IOPs, we offer two main technical contributions. First, we give a compiler that maps any public-coin IOP into a non-interactive proof in the random oracle model. We prove that the soundness of the resulting proof is tightly characterized by the soundness of the IOP against *state restoration attacks*, a class of rewinding attacks on the IOP verifier. Our compiler preserves zero knowledge, proof of knowledge, and time complexity of the underlying IOP. As an application, we obtain blackbox unconditional ZK proofs in the random oracle model with quasilinear prover and polylogarithmic verifier, improving on the result of Ishai et al. (2015).

Second, we study the notion of state-restoration soundness of an IOP: we prove tight upper and lower bounds in terms of the IOP's (standard) soundness and round complexity; and describe a simple adversarial strategy that is optimal across all state restoration attacks.

Our compiler can be viewed as a generalization of the Fiat–Shamir paradigm for public-coin IPs (CRYPTO '86), and of the "CS proof" constructions of Micali (FOCS '94) and Valiant (TCC '08) for PCPs. Our analysis of the compiler gives, in particular, a unified understanding of all of these constructions, and also motivates the study of state restoration attacks, not only for IOPs, but also for IPs and PCPs.



Recall that NP is the model for traditional mathematical proofs:

We have studied two different extensions:

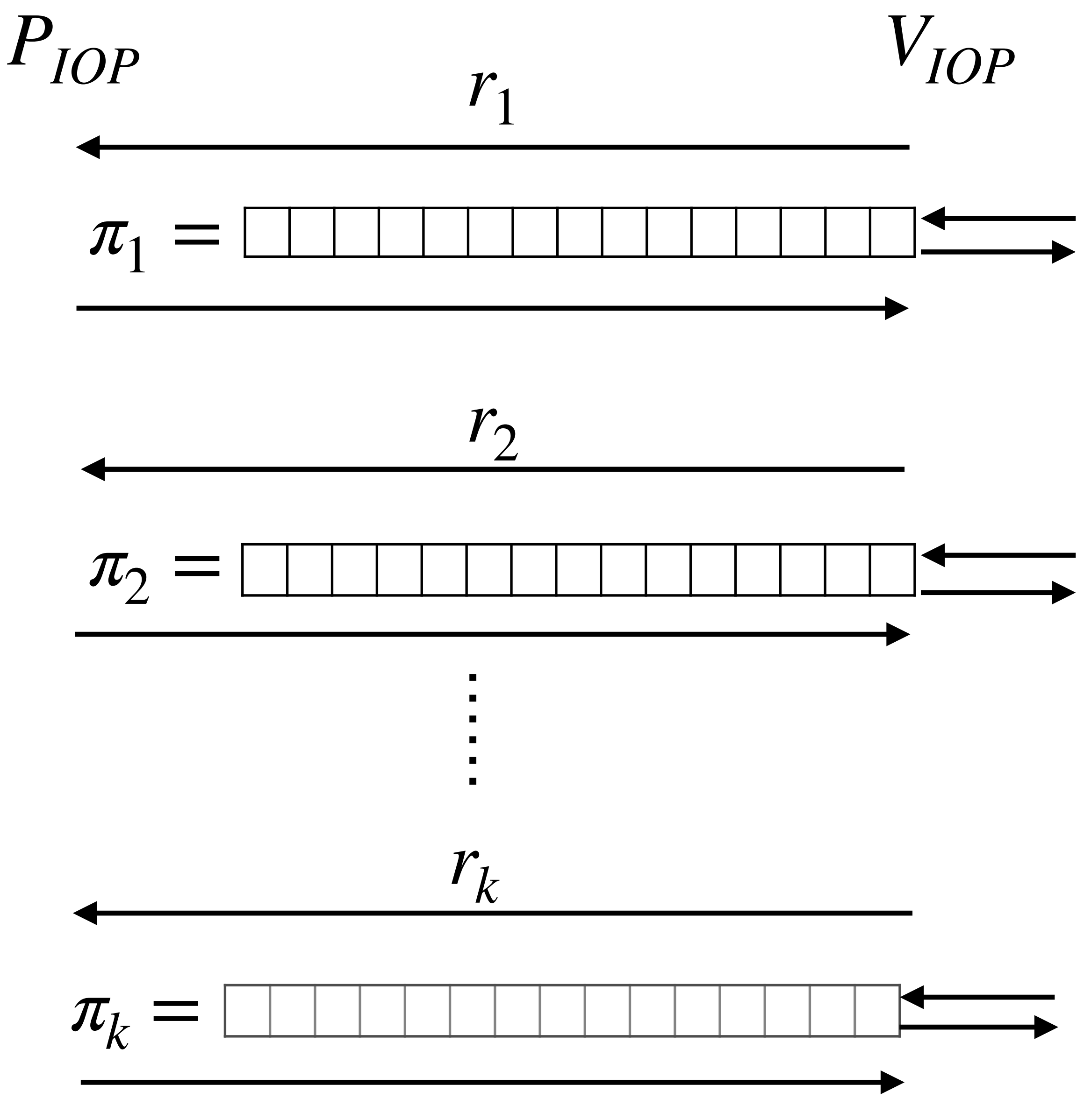
IP: add randomness & interaction

PCP: add randomness & oracle access to proof

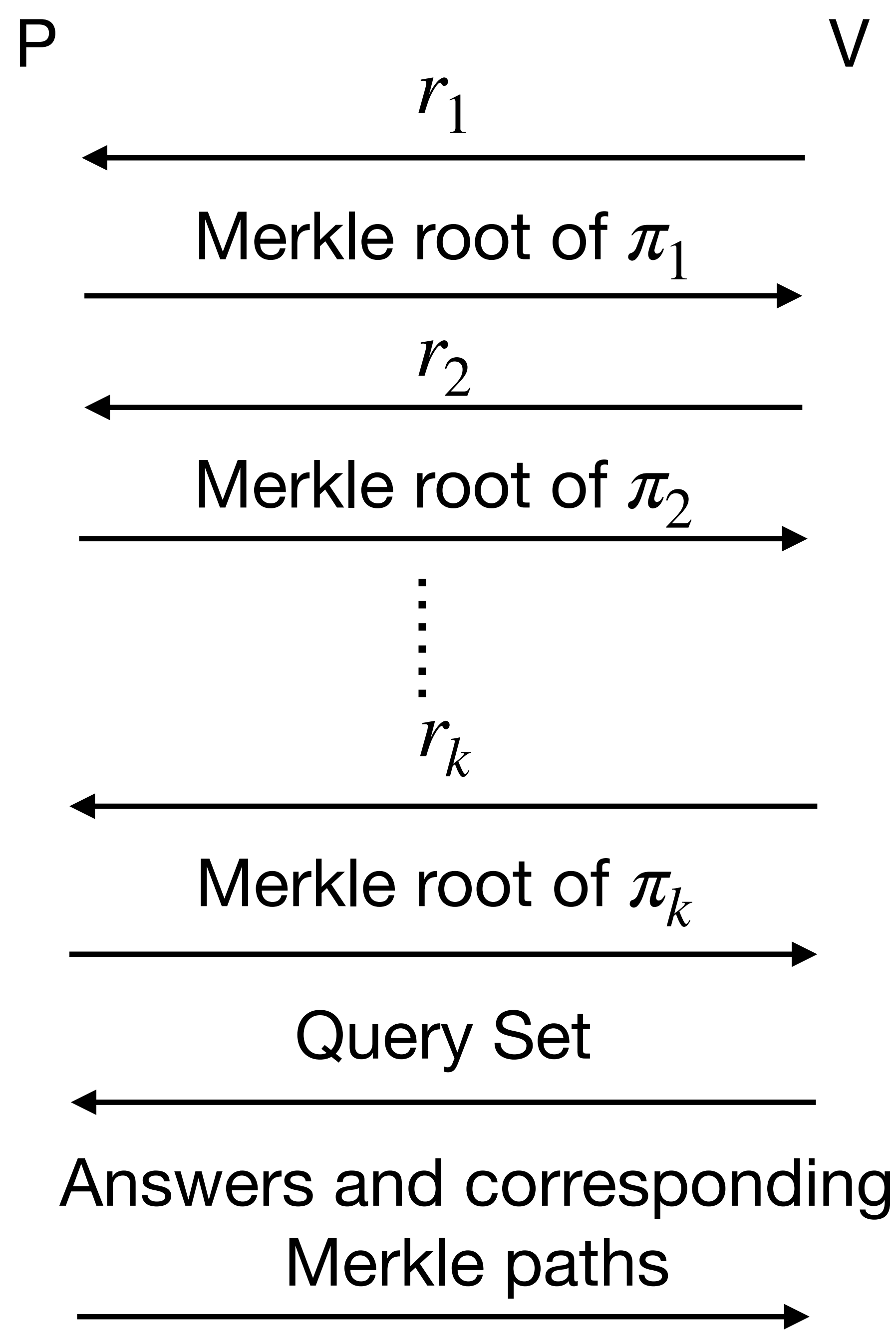
Today we consider the common extension between the two:

Interactive Oracle Proof (IOP)
add randomness, interaction, and oracle access to proof

SNARK from IOP

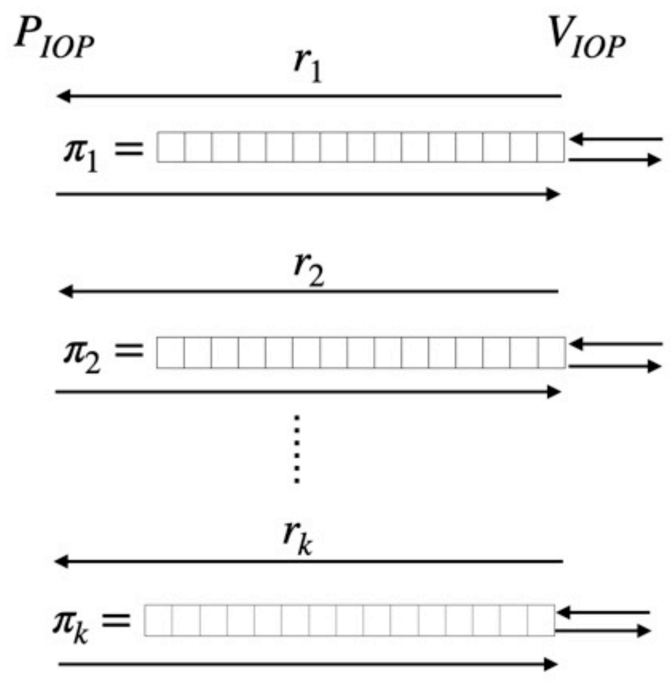


Non-Interactive comes from
Fiat-Shamir Transformation



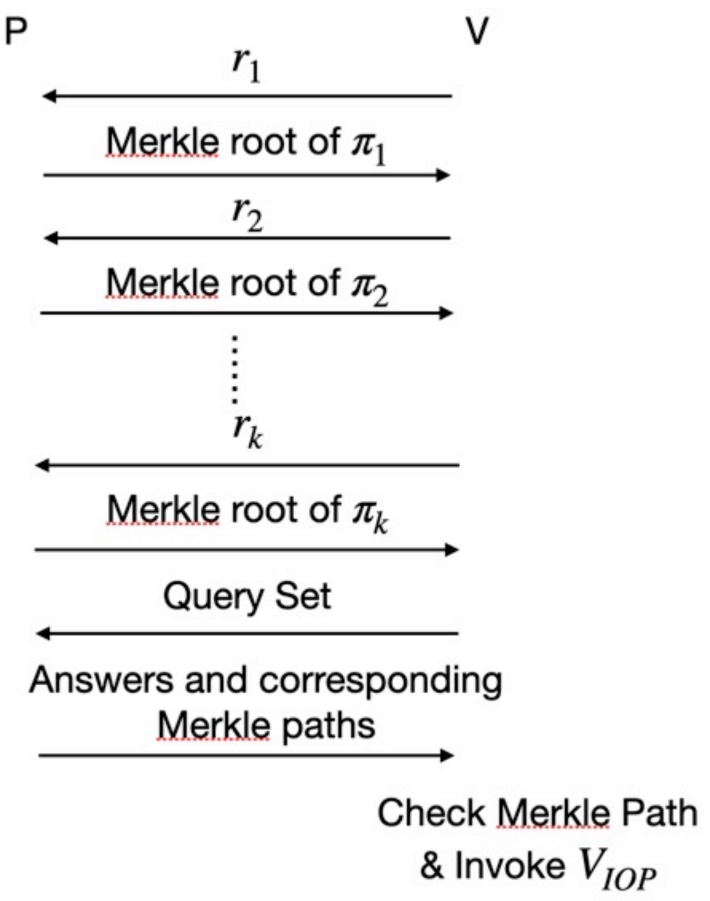
Check Merkle Path
& Invoke V_{IOP}

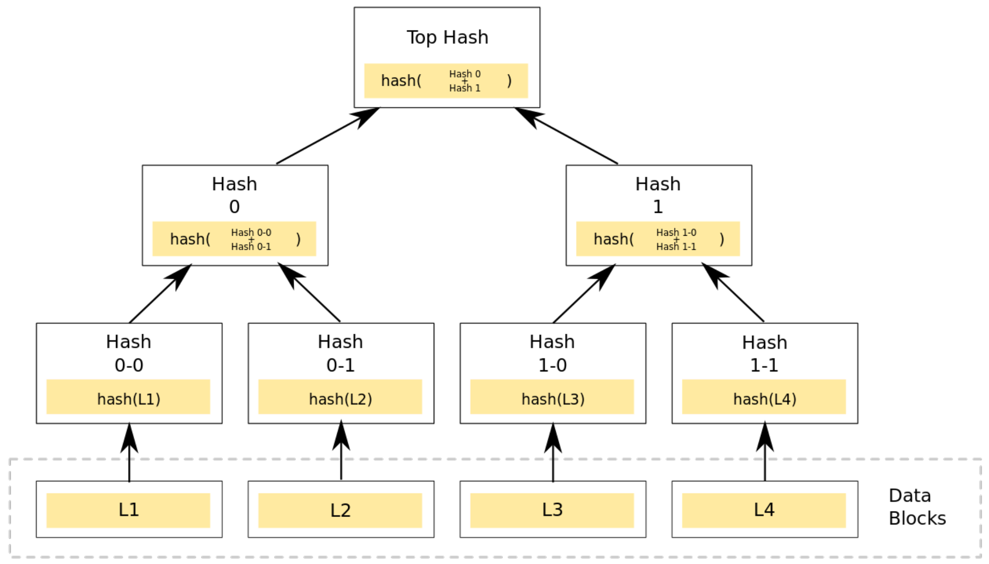
Properties of the Transformation



Non-Interactive comes from
Fiat-Shamir Transformation

文本





Suppose the IOP protocol has

Prover time: Time(P_{IOP})

Verifier time: Time(V_{IOP})

Proof length: $|\pi_1| = |\pi_2| = \cdots = |\pi_k| = l$

Query complexity: q

Round Complexity: k

Suppose we use SHA256 in Merkle Tree

After the transformation The SNARK has

Prover time: Time(P_{IOP})+$O(kl)$

Verifier time: Time(V_{IOP})+$O(q \cdot \log l)$

Proof length: $O(k + q \cdot \log l)$

In sum, design efficient IOP leads to efficient SNARK.

IOP for Univariate Sum-Check

F is a finite field, H, L are subsets of F and $H \cap L = \phi, f \in F[x]$ with degree less than d. $d, |H|, |L| = O(n)$ and $d < |L|$

$\tilde{f} : L \to \mathbb{F}$ is a function table for all $x \in L$. i.e. $\tilde{f} = (f(x_1), f(x_2), \cdots, f(x_{|L|}))$

The verifier do not know the polynomial f, but can query $\tilde{f} : L \to \mathbb{F}$ at any point. V want to compute $\sum_{a \in H} f(a)$.

Attempt: query $\tilde{f}$ at $d+1$ points, using interpolation reconstruct $f(x)$, compute $f(a)$ for all $a \in H$.

Query complexity $d + 1 = O(n)$ and not efficient.

Delegating this task to an untrusted party, we can design an IOP for it.

The idea is very different than multivariate Sumcheck.

IOP for Univariate Sum-Check

Step 1: we can reduce the problem to the case $d < |H|$

- Let $V_H(x) = \Pi_{i \in H}(x - i)$ be the vanishing polynomial of the set H.
- Divide $f(x)$ by $V_H(x)$: $f(x) = h(x) \cdot V_H(x) + g(x)$ with $\deg(g) < |H|$ and $\deg(h) = \deg(f) - |H|$
- $\sum_{a \in H} f(a) = \sum_{a \in H} \big(h(a) \cdot V_H(a) + g(a)\big) = \sum_{a \in H} g(a)$

IOP for Univariate Sum-Check

Step 2: Assume H has nice algebraic structure.

Lemma: suppose H is a multiplicative subgroup of $\mathbb{F}^\star$, then

$$\sum_{a \in H} g(a) = |H| \cdot g(0)$$

Proof: Consider a monomial $\sum_{a \in H} a^i = \sum_{j=0}^{|H|-1} (w^j)^i = \sum_{j=0}^{|H|-1} (w^i)^j$

$$= |H| \text{ if } i = 0$$

$$= \frac{1-(w^i)^{|H|}}{1-w^i} = 0 \text{ else}$$

$$\sum_{a \in H} g(a) = \sum_{a \in H} \left(g_0 + g_1 a + \cdots + g_d a^d\right) = \sum_{a \in H} g_0 = |H| \cdot g(0)$$

Notice if H is a multiplicative subgroup, then $V_H(x) = x^{|H|} - 1$, which can be computed in time $O(\log n)$

IOP for Univariate Sum-Check

From Step 1 and Step 2, we have

$$\sum_{a \in H} f(a) = \sum_{a \in H} \big(h(a) \cdot V_H(a) + g(a)\big) = \sum_{a \in H} g(a) = |H| \cdot g(0)$$

Suppose the Prover Claim that $\sum_{a \in H} f(a) = \gamma$, it is suffice to prove that $g(0) = \frac{\gamma}{|H|}$

$$f(x) = V_H(x) \cdot h(x) + g(x) = V_H(x) \cdot h(x) + \left(x \cdot p(x) + \frac{\gamma}{|H|}\right)$$

Suppose $\deg(f) \leq d$, we have deg$(h) \leq d - |H|$, deg$(p) < |H| - 1$

Prover
- Compute $h(x)$ and $p(x)$
- Compute the function table $\tilde{h}, \tilde{p} : L \to \mathbb{F}$ of $h(x), p(x)$

$$\xrightarrow{\tilde{h} : L \to \mathbb{F} \quad \tilde{p} : L \to \mathbb{F}}$$

Verifier

- test that deg$(h) \leq d - |H|$
- test that deg$(p) < |H| - 1$
- Sample $s \leftarrow L$, query $\tilde{f}, \tilde{h}, \tilde{p}$ at s

Check that $f(s) = V_H(s) \cdot h(s) + \left(s \cdot p(s) + \frac{\gamma}{|H|}\right)$

Ignore the low degree testing phase, the query complexity is 3; Verifier's time is $O(\log n)$. Proof length $O(n)$

Low Degree Testing

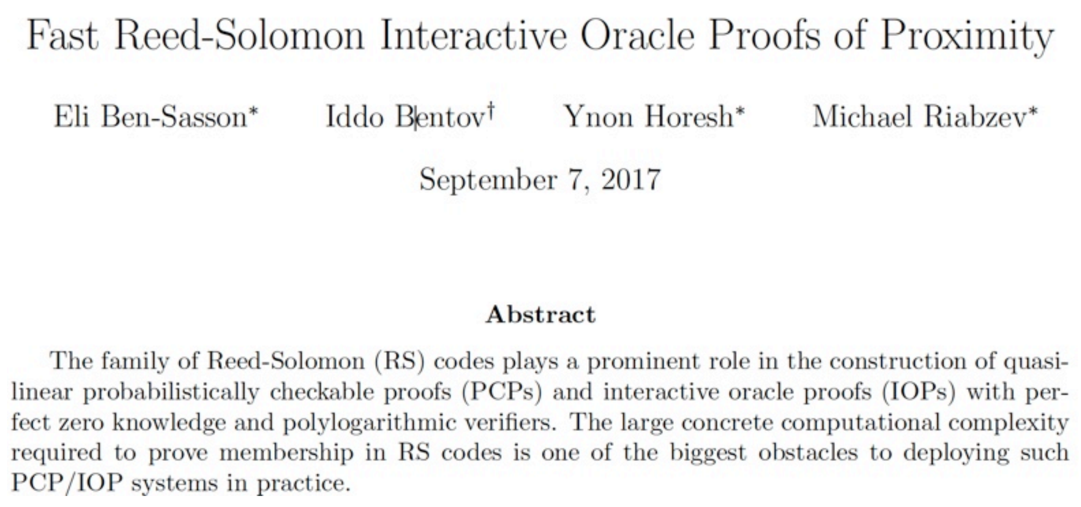
L is a subset of $\mathbb{F}$, $\tilde{p} : L \to \mathbb{F}$ is the function table for a polynomial $p \in \mathbb{F}[x]$, we want to test that $\deg(p) \leq d$ (Suppose $d < |L|$).

Attempt: Interpolation. Query Complexity: $d + 1 = O(n)$ 😰😰

Test $\tilde{p}$ is a Reed-Solomon Codeword with rate $\leq \dfrac{p+1}{|L|}$.

Using FRI protocol, we can done LDT with query complexity $O(\log n)$, round complexity $O(\log n)$, verifier time $O(\log n)$, proof length $O(n)$

A tight soundness analysis remain an open problem.


Fast Reed-Solomon Interactive Oracle Proofs of Proximity

Eli Ben-Sasson* Iddo Bentov† Ynon Horesh* Michael Riabzev*

September 7, 2017

Abstract

The family of Reed-Solomon (RS) codes plays a prominent role in the construction of quasi-linear probabilistically checkable proofs (PCPs) and interactive oracle proofs (IOPs) with perfect zero knowledge and polylogarithmic verifiers. The large concrete computational complexity required to prove membership in RS codes is one of the biggest obstacles to deploying such PCP/IOP systems in practice.

Univariate SumCheck for Arbitrary Set

We have showed how to do univariate sum check when H is a multiplicative subgroup

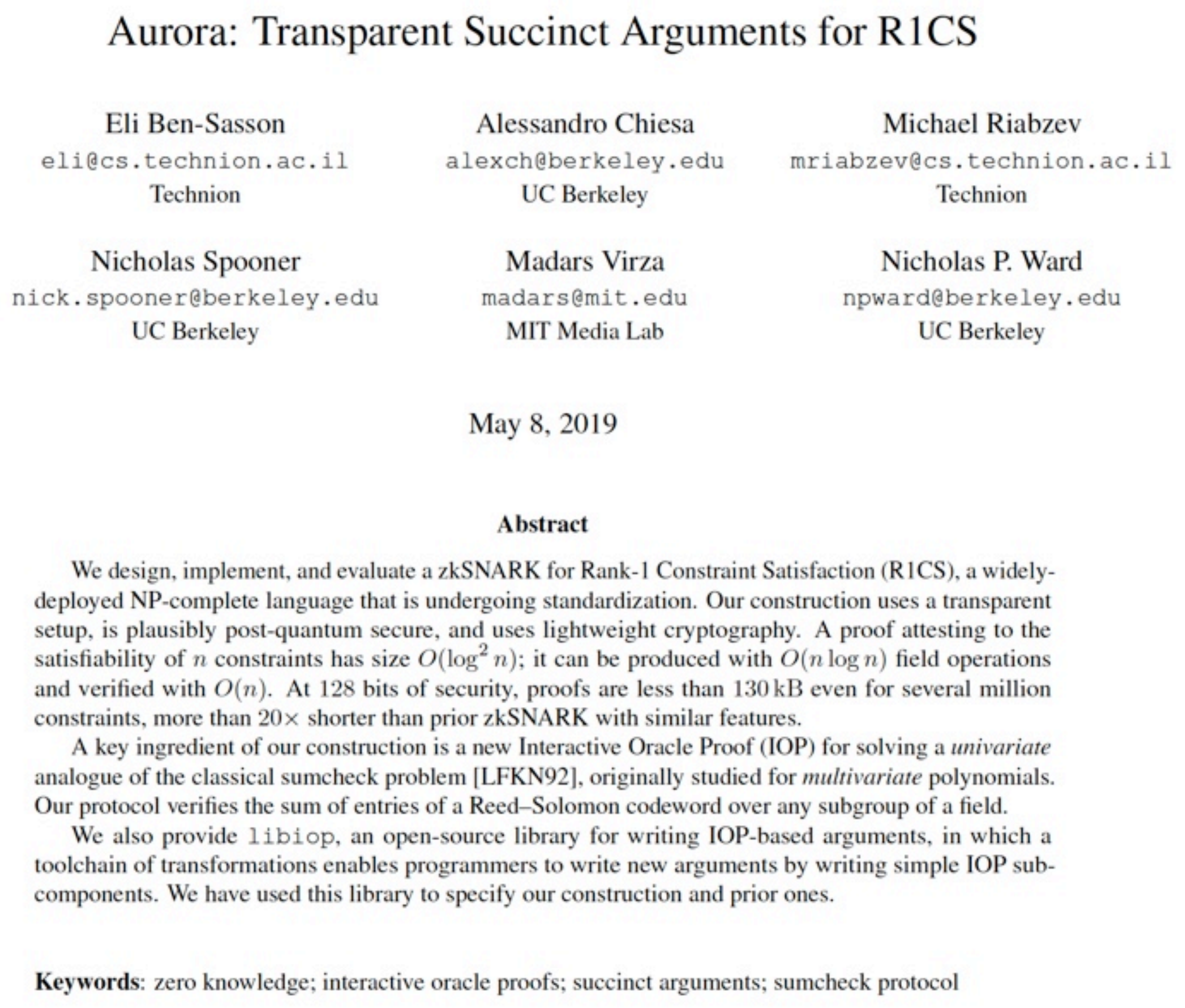
We can also design a protocol when H is a additive subgroup or coset.

How about for arbitrary set?

An open problem🥲🥲

IOP for R1CS from Univariate SumCheck

No time to explain🤪🤪



Aurora: Transparent Succinct Arguments for R1CS

Eli Ben-Sasson
eli@cs.technion.ac.il
Technion

Alessandro Chiesa
alexch@berkeley.edu
UC Berkeley

Michael Riabzev
mriabzev@cs.technion.ac.il
Technion

Nicholas Spooner
nick.spooner@berkeley.edu
UC Berkeley

Madars Virza
madars@mit.edu
MIT Media Lab

Nicholas P. Ward
npward@berkeley.edu
UC Berkeley

May 8, 2019

Abstract

We design, implement, and evaluate a zkSNARK for Rank-1 Constraint Satisfaction (R1CS), a widely-deployed NP-complete language that is undergoing standardization. Our construction uses a transparent setup, is plausibly post-quantum secure, and uses lightweight cryptography. A proof attesting to the satisfiability of n constraints has size $O(\log^2 n)$; it can be produced with $O(n \log n)$ field operations and verified with $O(n)$. At 128 bits of security, proofs are less than 130 kB even for several million constraints, more than 20× shorter than prior zkSNARK with similar features.

A key ingredient of our construction is a new Interactive Oracle Proof (IOP) for solving a *univariate* analogue of the classical sumcheck problem [LFKN92], originally studied for *multivariate* polynomials. Our protocol verifies the sum of entries of a Reed–Solomon codeword over any subgroup of a field.

We also provide `libiop`, an open-source library for writing IOP-based arguments, in which a toolchain of transformations enables programmers to write new arguments by writing simple IOP sub-components. We have used this library to specify our construction and prior ones.

Keywords: zero knowledge; interactive oracle proofs; succinct arguments; sumcheck protocol

libiop: a C++ library for IOP-based zkSNARKs

https://github.com/scipr-lab/libiop

Summary

SNARK =	Information-theoretic Proof System	+	Cryptograph Tool
	PCP	+	Merkle Tree
	Linear PCP	+	Pairing based Cryptography
	IP	+	Polynomial Commitment
	IOP	+	Merkle Tree

⋮

Cryptograph Tool: There are many excellent Cryptography open courses and textbooks.

Information-theoretic Proof System: [CS294: Foundations of Probabilistic Proofs (F2020)]