Interactive Proofs:

Sum-Check Protocol and Arithmetization

Talking is cheap, show me the proof

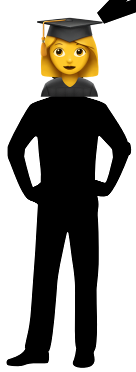
Shuangjun Zhang

November 2021

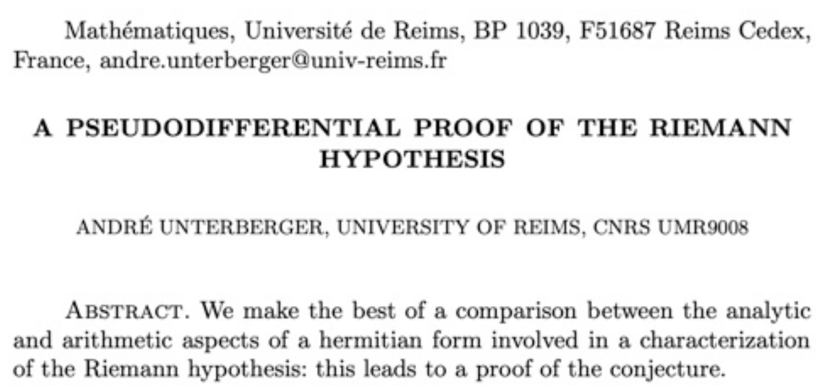
Traditional Proofs

I know how to prove Riemann Hypothesis

Show me your proof!



Prover (P)

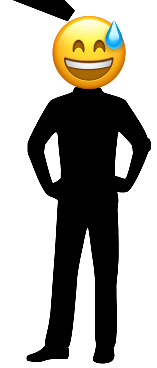


Mathématiques, Université de Reims, BP 1039, F51687 Reims Cedex, France, andre.unterberger@univ-reims.fr

A PSEUDODIFFERENTIAL PROOF OF THE RIEMANN HYPOTHESIS

ANDRÉ UNTERBERGER, UNIVERSITY OF REIMS, CNRS UMR9008

ABSTRACT. We make the best of a comparison between the analytic and arithmetic aspects of a hermitian form involved in a characterization of the Riemann hypothesis: this leads to a proof of the conjecture.



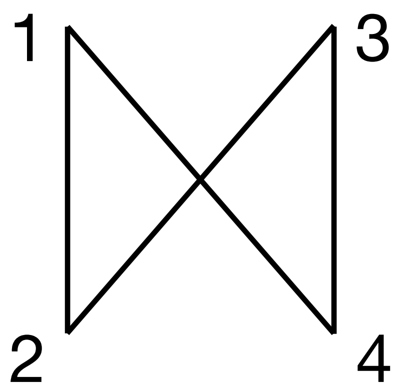
Verifier (V)

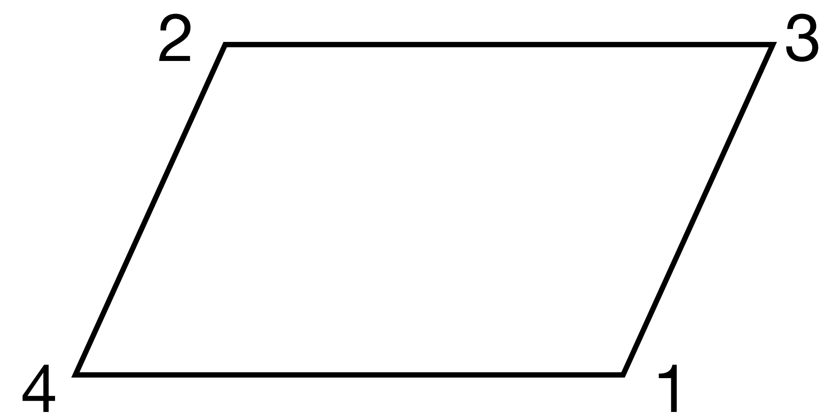
- I will check line by line.

Traditional Proof = NP in complexity theory

NP: Problems can be verified in polynomial time.

Examples in NP: Graph Isomorphism



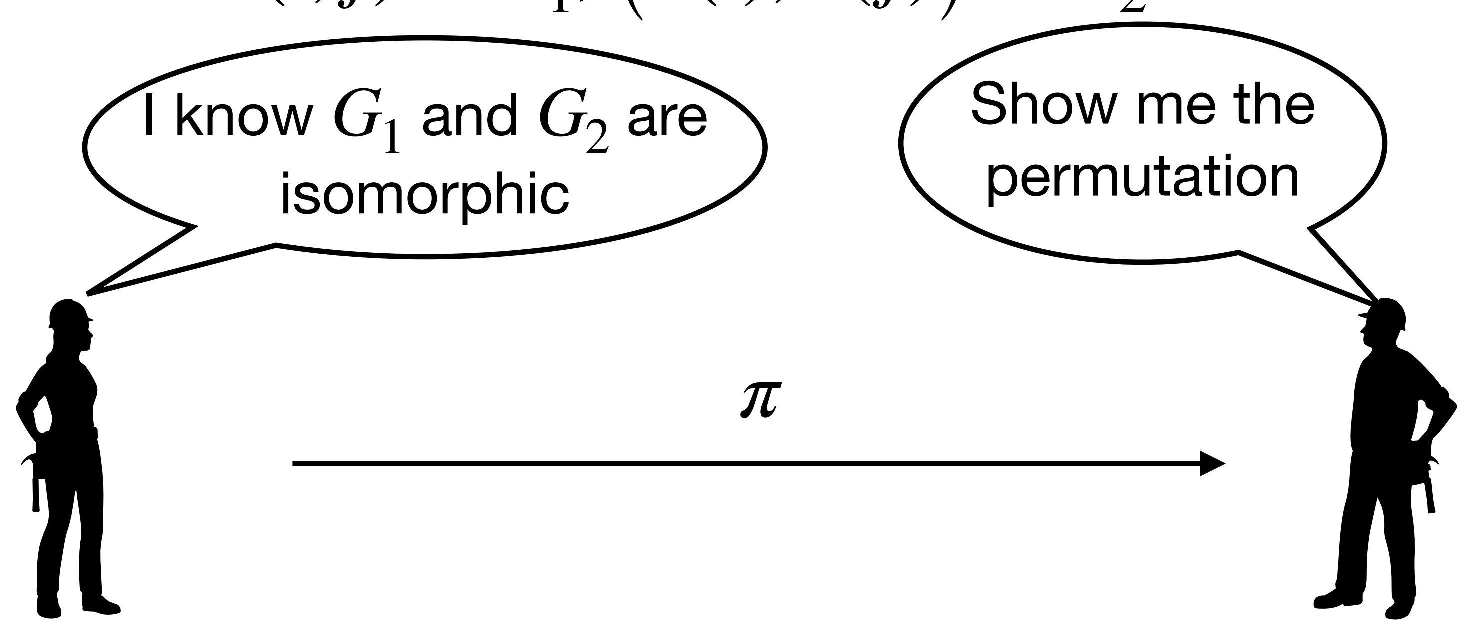


$\pi(1) = 2; \pi(2) = 4; \pi(3) = 1; \pi(4) = 3$

- Input two graphs G_1 and G_2 with n nodes, decides whether $G_1 \cong G_2$
- π is a permutation of $\{1,2,\ldots,n\}$

$$G_1 \cong G_2 \text{ if and only if } \exists \pi \text{ such that } \forall (i, j) \in E_1, \big(\pi(i), \pi(j)\big) \in E_2$$

- GI is an NP problem
- No polynomial time algorithm known (2021)



A New Proof Model: Interactive Proofs

SIAM J. COMPUT.
Vol. 18, No. 1, pp. 186-208, February 1989

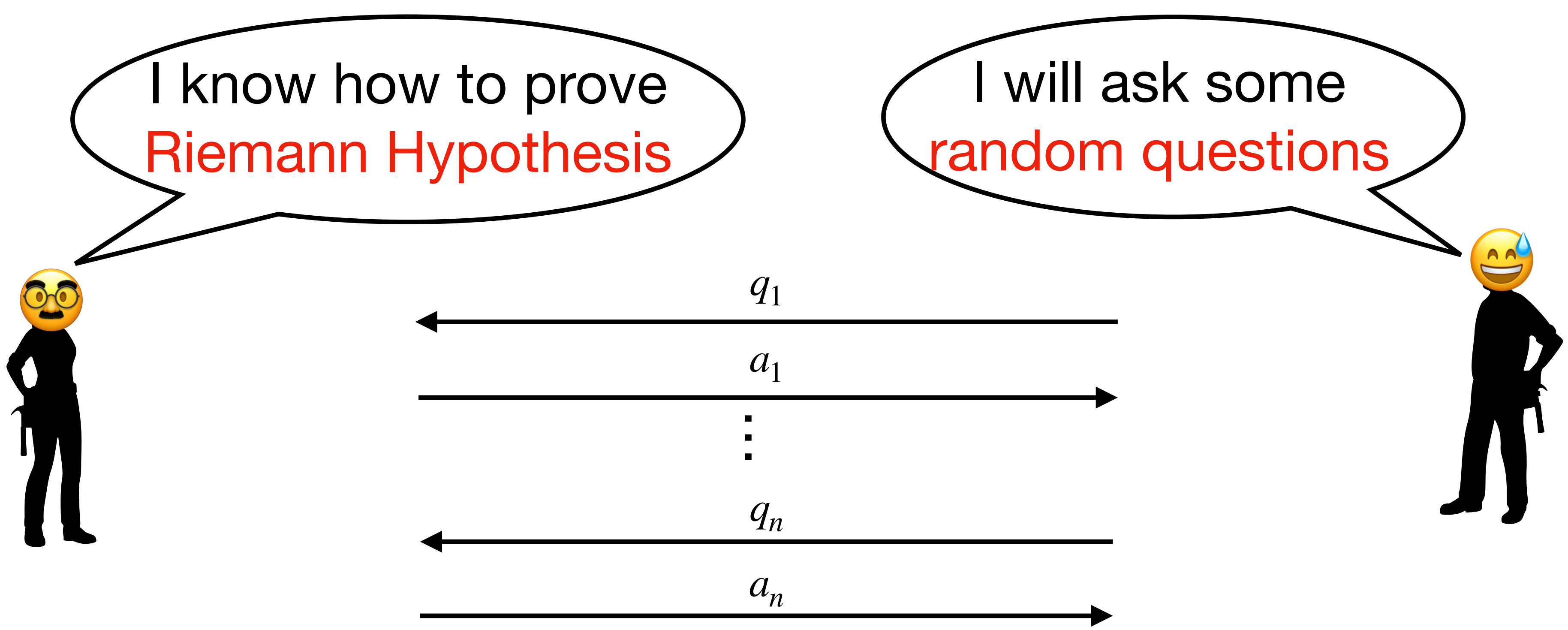
THE KNOWLEDGE COMPLEXITY OF
INTERACTIVE PROOF SYSTEMS*

SHAFI GOLDWASSER†, SILVIO MICALI†, AND CHARLES RACKOFF‡

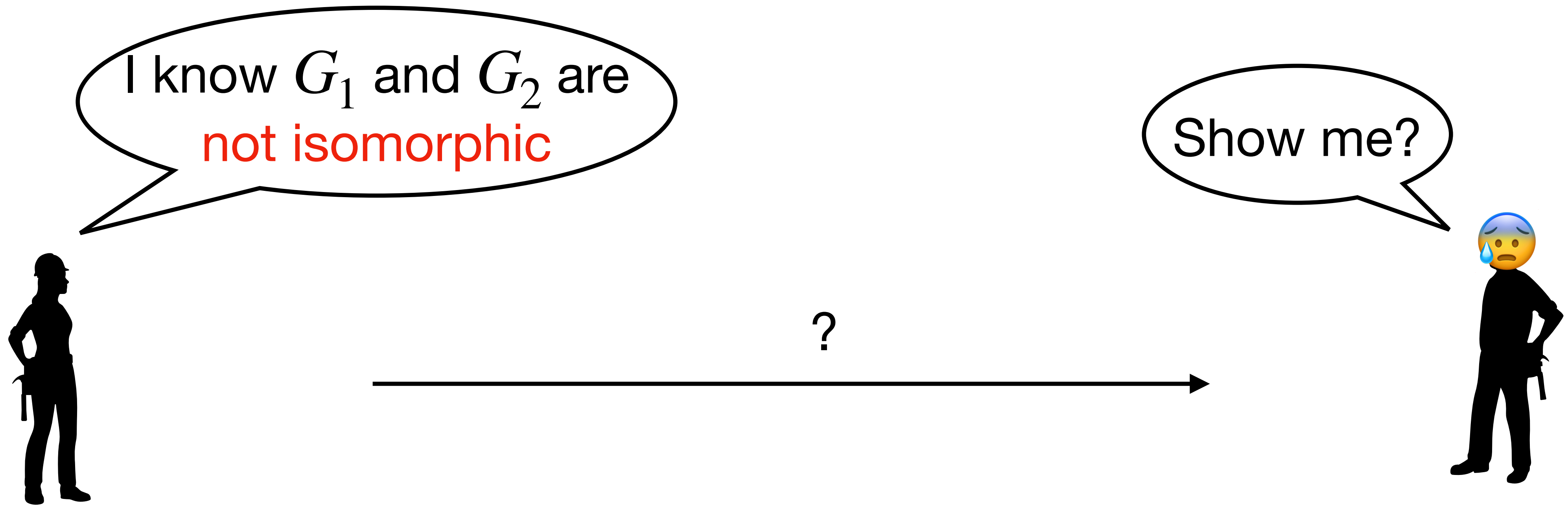
Abstract. Usually, a proof of a theorem contains more knowledge than the mere fact that the theorem is true. For instance, to prove that a graph is Hamiltonian it suffices to exhibit a Hamiltonian tour in it; however, this seems to contain more knowledge than the single bit Hamiltonian/non-Hamiltonian.

In this paper a computational complexity theory of the "knowledge" contained in a proof is developed. Zero-knowledge proofs are defined as those proofs that convey no additional knowledge other than the correctness of the proposition in question. Examples of zero-knowledge proof systems are given for the languages of quadratic residuosity and quadratic nonresiduosity. These are the first examples of zero-knowledge proofs for languages not known to be efficiently recognizable.

The verifier may (1). Use randomness; (2). Interact with Prover



Example: Graph Non-Isomorphism



- Verifier:
  - $\cdot\ b \leftarrow \{0,1\}$.
  - $\cdot\ h \leftarrow S_n$ (S_n is all permutations in $\{1,2,\ldots,n\}$).
  - $\cdot$ Compute $H = h(G_b)$.
  - $\cdot$ Ask P a question, which graph is isomorphic to H.
- Prover:
  - $\cdot$ Replay a bit b'
- Verifier:
  - $\cdot$ If $b = b'$, accept; else reject.

- If G_1 and G_2 are not isomorphic, V will accept with probability 1.

- If G_1 and G_2 are isomorphic, V will accept with probability at most $\frac{1}{2}$

Sum-Check Protocol

Sum-Check Problem

Suppose $\mathbb{F}$ is a finite field, $p(x_1, ..., x_n) \in \mathbb{F}[x_1, ..., x_n]$, how to compute

$$\sum_{b_1, b_2, ..., b_n \in \{0,1\}} p(b_1, b_2, \ldots b_n)$$

· Naive idea takes $O(2^n \cdot T)$ time, where T is the time for evaluating p at a single point.

· Delegating this task to a powerful but untrusted party, we can design a protocol to verify the result is correct or not in time $O(n + T)$. Such a protocol is called "Sum-Check" Protocol.

Sum-Check Protocol plays an important role not only in theory but also in practice.

· Theory (80s, 90s): IP=PSPACE, MIP=NEXP, PCP Theorem.

· Practice (now): Many zkSNARK systems use Sum-Check protocol.

(Hyrax, zk-vSQL, Libra, Virgo, Spartan, Aurora, Marlin, Fractal, et.al.)

Sum-Check Protocol

The untrusted party, say a Prover (P), claim that

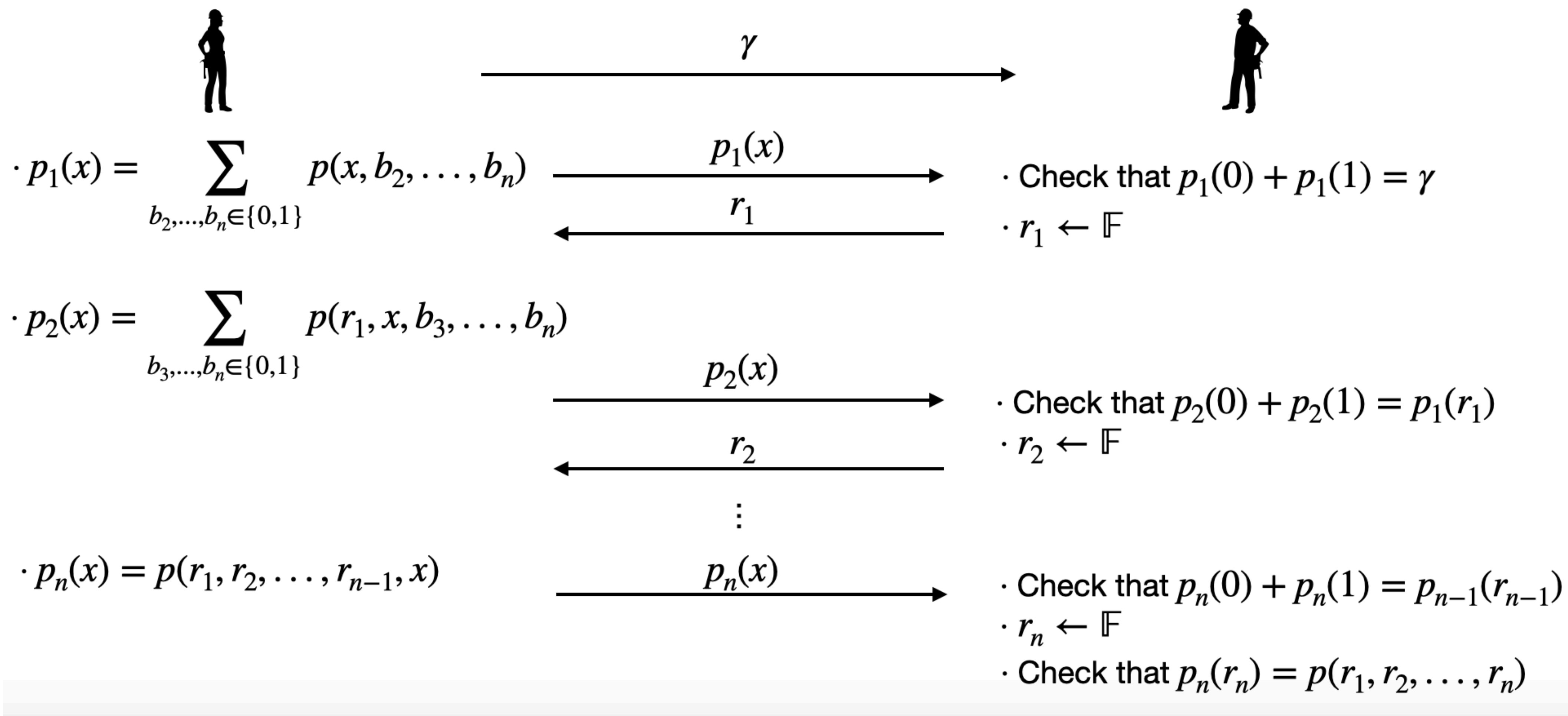
$$\sum_{b_1,b_2,\ldots,b_n\in\{0,1\}} p(b_1, b_2, \ldots b_n) = \gamma \qquad (p \in \mathbb{F}[x_1, x_2, \ldots, x_n], \gamma \in \mathbb{F})$$

The verifier (V) should verify that γ is a correct answer.

$$\xrightarrow{\quad \gamma \quad}$$

$\cdot\ p_1(x) = \sum_{b_2,\ldots,b_n\in\{0,1\}} p(x, b_2, \ldots, b_n)$	$\xrightarrow{p_1(x)}$	$\cdot$ Check that $p_1(0) + p_1(1) = \gamma$
	$\xleftarrow{r_1}$	$\cdot\ r_1 \leftarrow \mathbb{F}$
$\cdot\ p_2(x) = \sum_{b_3,\ldots,b_n\in\{0,1\}} p(r_1, x, b_3, \ldots, b_n)$	$\xrightarrow{p_2(x)}$	$\cdot$ Check that $p_2(0) + p_2(1) = p_1(r_1)$
	$\xleftarrow{r_2}$	$\cdot\ r_2 \leftarrow \mathbb{F}$
	$\vdots$	
$\cdot\ p_n(x) = p(r_1, r_2, \ldots, r_{n-1}, x)$	$\xrightarrow{p_n(x)}$	$\cdot$ Check that $p_n(0) + p_n(1) = p_{n-1}(r_{n-1})$
		$\cdot\ r_n \leftarrow \mathbb{F}$
		$\cdot$ Check that $p_n(r_n) = p(r_1, r_2, \ldots, r_n)$

Properties of Sum-Check Protocol



- Round Complexity: $O(n)$
- Verifier's Time $O(n \cdot \text{deg}_{\text{ind}}(p) + T)$ (T is the time for evaluating p at a single point; $\text{deg}_{\text{ind}}(p)$ is the max Individual degree of p).
- If the results is correct, V will accept with probability 1.

Soundness Analysis

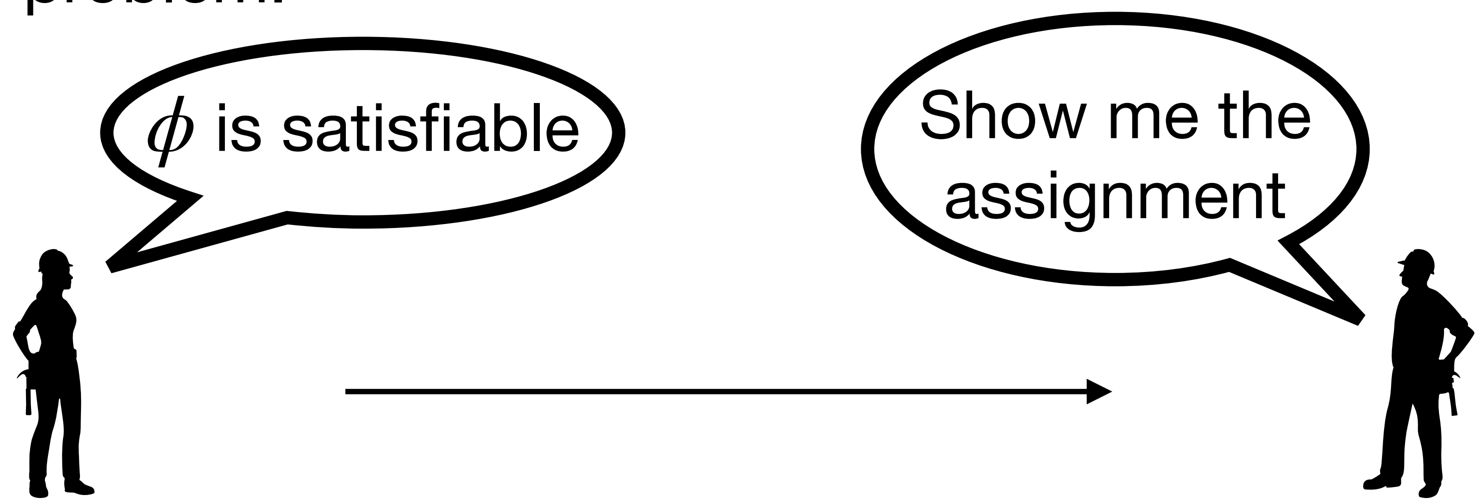
If $\sum_{b_1,b_2,\ldots,b_n\in\{0,1\}} p(b_1, b_2, \ldots b_n) \neq \gamma$, For any prover P*, Pr $\left[\text{V accepts}\right] \leq \frac{n \cdot \text{deg}_{\text{ind}}(p)}{|\mathbb{F}|}$

Proof: Omit～～ 🥲🥲🥲🥲

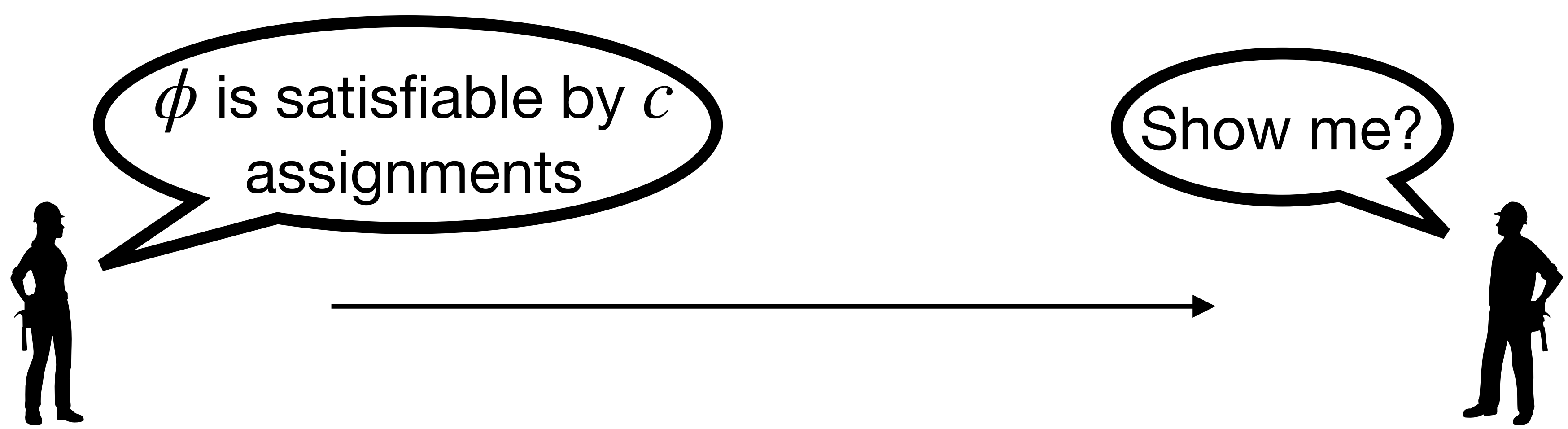
An Interactive Proof for #SAT

- SAT: Input a boolean formula ϕ, decide whether ϕ can be satisfied.
- Example: $\phi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_2 \vee x_3 \vee \bar{x}_4)$

  ϕ can be satisfied by $x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0$.
- SAT is an NP problem.



- #SAT: Input a boolean formula ϕ, compute how many satisfiable assignments?



Arithmetization for #SAT

- Suppose ϕ has n variables:

The number of satisfiable assignments $= \displaystyle\sum_{b_1,b_2,\ldots,b_n\in\{0,1\}} \phi(b_1, b_2, \ldots, b_n)$

- We can't run sum check directly, since ϕ is not a polynomial.
- Arithmetization for a boolean formula:

$$\bar{x} \to 1 - x \qquad x \wedge y \to x \cdot y \qquad x \vee y \to x + y - x \cdot y$$

$$\phi \to p$$

Claim: - $\forall (b_1, b_2, \ldots, b_n) \in \{0,1\}^n, p(b_1, b_2, \ldots, b_n) = \phi(b_1, b_2, \ldots, b_n)$

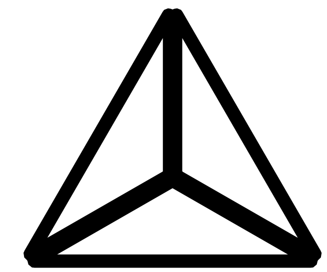
- The number of satisfiable assignments of ϕ is $\displaystyle\sum_{b_1,b_2,\ldots,b_n\in\{0,1\}} p(b_1, b_2, \ldots, b_n)$

Now, we can run sumcheck.

An Interactive Proof for Counting Triangles

Input: a graph with n vertices.

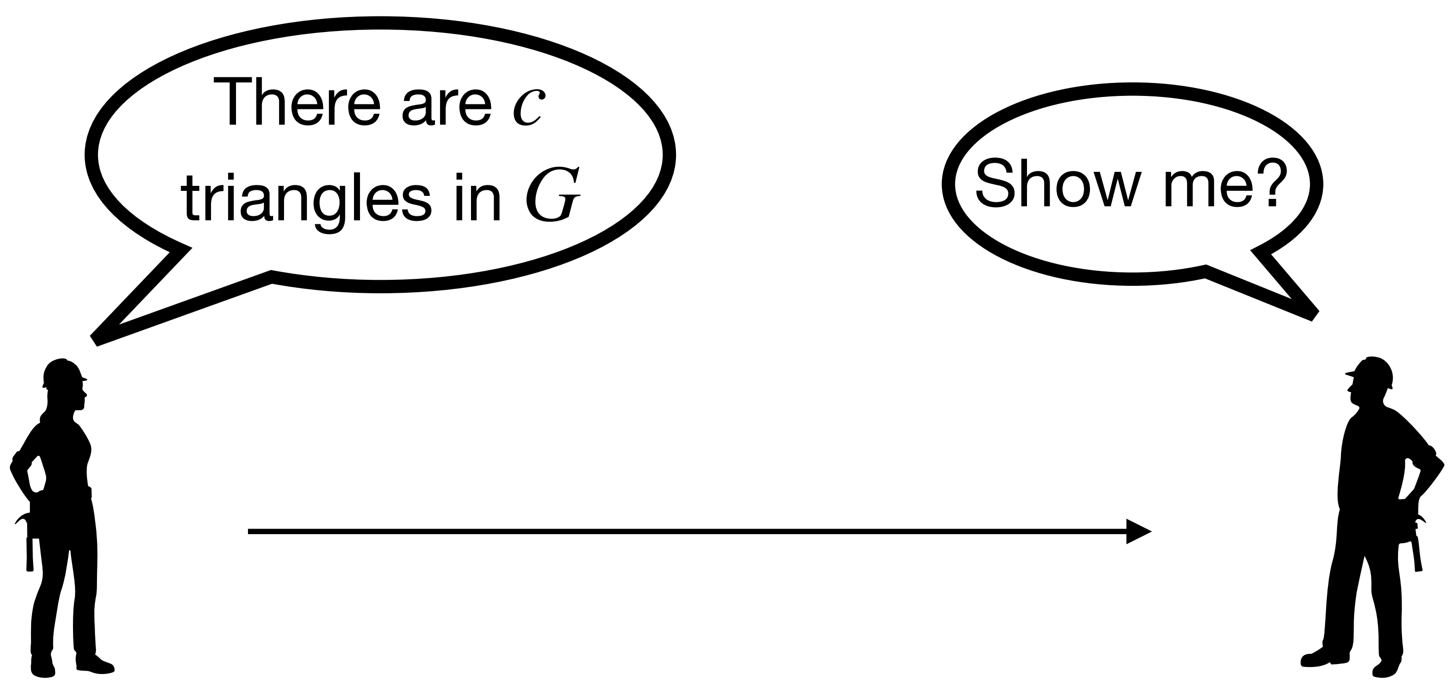
Output: the number of triangles in the graph.



- Naive Idea: for every 3 vertices, check whether they form a triangle.

Time Complexity: $O(n^3)$

Best Algorithm: Same as Matrix Multiplication $O(n^{2.3728596})$ [Alman, Williams 2020]



Arithmetization for Counting triangles

- G is a graph with n vertices.
- $A \in \{0,1\}^{n\times n}$ is the adjacency matrix ($A_{i,j} = 1$ if and only if $(i,j) \in E$)
- $g : [n] \times [n] \to \{0,1\}, g(i,j) = A_{i,j}$
- $f : \{0,1\}^{\log n} \times \{0,1\}^{\log n} \to \{0,1\}, f(\overrightarrow{x}, \overrightarrow{y}) = A_{i,j}$.

$\overrightarrow{x}, \overrightarrow{y}$ is the binary representation of i, j.

- (i,j,k) form a triangle if and only if $(i,j) \in E, (j,k) \in E, (k,i) \in E$.

If and only if $A_{i,j} = 1, A_{j,k} = 1, A_{i,k} = 1$.

If and only if $A_{i,j} \cdot A_{j,k} \cdot A_{i,k} = 1$.

If and only if $f(\overrightarrow{x}, \overrightarrow{y}) \cdot f(\overrightarrow{x}, \vec{z}) \cdot f(\overrightarrow{y}, \vec{z}) = 1$.

- Number of triangles $= \dfrac{1}{6} \displaystyle\sum_{i,j,k \in [n]} A_{i,j} \cdot A_{j,k} \cdot A_{i,k} = \dfrac{1}{6} \sum_{\overrightarrow{x}, \overrightarrow{y}, \vec{z} \in \{0,1\}^{\log n}} f(\overrightarrow{x}, \overrightarrow{y}) \cdot f(\overrightarrow{y}, \vec{z}) \cdot f(\overrightarrow{x}, \vec{z})$.
- However, f is not a polynomial. We can transform f into a polynomial by interpolation.
- Now, we can run sum-check protocol, by some careful analysis, time complexity of V is $O(n^2)$

More about Arithmetization

- Polynomials = Error Correcting Codes (Reed-Solomon Code, Reed-Muller Codes)
- Suppose I want to test a long binary string s is all zero ($|s| = n$).
  - choose a random point i, check whether $s[i] = 0$.
    - If s is all 0 except 1 position, then V will accept with probability $1 - \frac{1}{n}$.
  - Before testing, encoding s using some error correcting code with relative distance $\geq \frac{1}{2}$ (Suppose Enc: $\{0,1\}^n \to \{0,1\}^m$).
    - If $s = 0^n, Enc(s) = 0^m$
    - If s has at least an 1, $Enc(s)$ has at least $\frac{m}{2}$ 1, V will accept with probability at most $\frac{1}{2}$.

In zkSNARK, Any Computation $\xrightarrow{\text{Arithmetization}}$ Quadratic Arithmetic Program (QAP)

Arithmetization in zkSNARK

Arithmetization in zkSNARK

Arithmetization steps in zkSNARK

- Computation
- Algebraic Circuit
- R1CS
- QAP
- Linear PCP
- Linear Interactive Proof
- zkSNARK

Computation → Algebraic Circuit:

Fact: Every computation can be converted into an equivalent circuit, which can be made algebraic by standard arithmetization technique.

Rank-1 Constraint System (R1CS)

$A, B, C \in \mathbb{F}^{n\times n}$, $x \in \mathbb{F}^m$ $(m < n)$, Is there a vector $w \in \mathbb{F}^{n-m}$ such that $Az \circ Bz = Cz$? where $z = (x, w) \in \mathbb{F}^n$

- Computation
- Algebraic Circuit
- R1CS
- QAP
- Linear PCP
- Linear Interactive Proof
- zkSNARK

Algebraic Circuit to R1CS

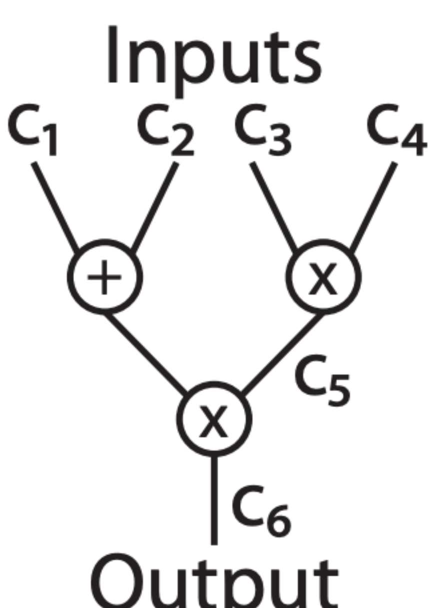
$x = (z_1, \ldots, z_{io})$ is the input/output wires of the circuit.

$w = (z_{io+1}, \ldots, z_n)$ is the other wires.

For a multiplicative gate, $z_i \cdot z_j = z_k$

For a additive gate, compress it to a multiplicative gate

$$z_i + z_j = z_k; z_k \cdot z_l = z_p \Rightarrow (z_i + z_j) \cdot z_l = z_p$$



$$(C_1 + C_2) \times C_5 = C_6$$

$$C_3 \times C_4 = C_5$$

$$\begin{pmatrix}1&1&0&0&0&0\\0&0&1&0&0&0\end{pmatrix}\begin{pmatrix}C_1\\C_2\\C_3\\C_4\\C_5\\C_6\end{pmatrix} \circ \begin{pmatrix}0&0&0&0&1&0\\0&0&0&1&0&0\end{pmatrix}\begin{pmatrix}C_1\\C_2\\C_3\\C_4\\C_5\\C_6\end{pmatrix}$$

$$= \begin{pmatrix}0&0&0&0&0&1\\0&0&0&0&1&0\end{pmatrix}\begin{pmatrix}C_1\\C_2\\C_3\\C_4\\C_5\\C_6\end{pmatrix}$$

Have a Try !

CRYPTO 2020:

Spartan = R1CS + Sum-Check + Polynomial Commitment.

Spartan: Efficient and general-purpose zkSNARKs without trusted setup

Srinath Setty

Microsoft Research

Abstract

This paper introduces Spartan, a new family of zero-knowledge succinct non-interactive arguments of knowledge (zkSNARKs) for the rank-1 constraint satisfiability (R1CS), an NP-complete language that generalizes arithmetic circuit satisfiability. A distinctive feature of Spartan is that it offers the first zkSNARKs without trusted setup (i.e., transparent zkSNARKs) for NP where verifying a proof incurs sub-linear costs—without requiring uniformity in the NP statement's structure. Furthermore, Spartan offers zkSNARKs with a time-optimal prover, a property that has remained elusive for nearly all zkSNARKs in the literature.

Quadratic Arithmetic Program (QAP)

$A, B, C \in \mathbb{F}^{n \times n}$, $x \in \mathbb{F}^m$ $(m < n)$, Is there a vector $w \in \mathbb{F}^{n-m}$ such that $Az \circ Bz = Cz$? where $z = (x, w) \in \mathbb{F}^n$

- Computation
- Algebraic Circuit
- R1CS
- QAP
- Linear PCP
- Linear Interactive Proof
- zkSNARK

R1CS to QAP

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{12} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{nn} \end{pmatrix} \quad B = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{12} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & & \vdots \\ b_{1n} & b_{2n} & \cdots & b_{nn} \end{pmatrix} \quad C = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{12} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & & \\ c_{1n} & c_{2n} & \cdots & c_{nn} \end{pmatrix}$$

Let $r_1, r_2, \ldots, r_n \leftarrow \mathbb{F}$,

For $i \in \{1,2,\ldots,n\}$, $L_i(x)$ be a polynomial with degree less than $n-1$ such that $L_i(r_j) = a_{ij}$ for any $j \in \{1,2\ldots,n\}$.

For $i \in \{1,2,\ldots,n\}$, $R_i(x)$ be a polynomial with degree less than $n-1$ such that $R_i(r_j) = b_{ij}$ for any $j \in \{1,2\ldots,n\}$.

For $i \in \{1,2,\ldots,n\}$, $O_i(x)$ be a polynomial with degree less than $n-1$ such that $O_i(r_j) = c_{ij}$ for any $j \in \{1,2\ldots,n\}$.

If the R1CS can be satisfied by $z = (z_1, z_2, \ldots, z_n)$.

$$\left(\sum_{i=1}^{n} z_i L_i(x)\right) \cdot \left(\sum_{i=1}^{n} z_i R_i(x)\right) = \left(\sum_{i=1}^{n} z_i O_i(x)\right) \text{ for every } x = r_1, r_2, \ldots, r_n.$$

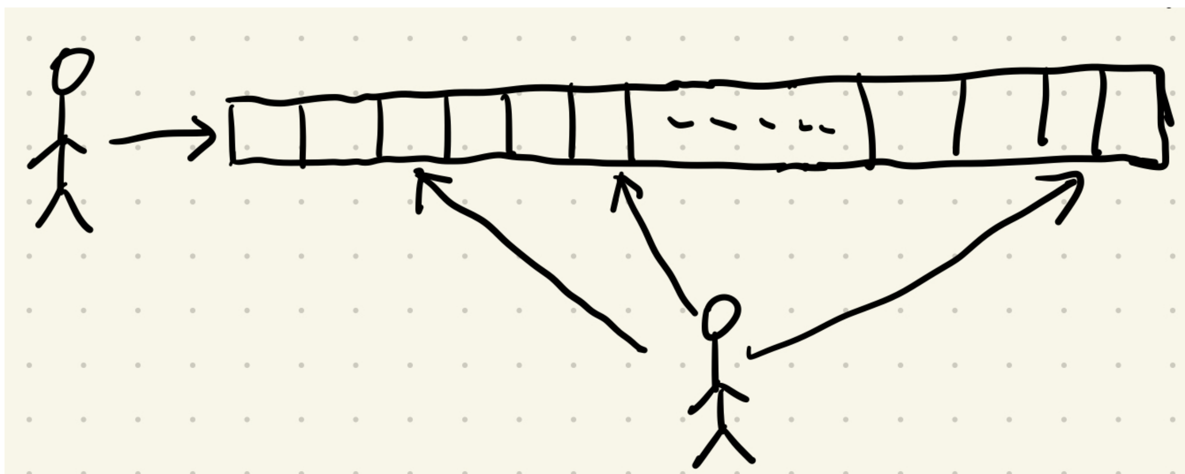
Let $t(x) = (x - r_1)(x - r_2)\cdots(x - r_n)$

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

Linear PCP for QAP

We need a new proof model:

Probabilistic Checkable Proof (PCP)



- Computation
- Algebraic Circuit
- R1CS
- QAP
- Linear PCP
- Linear Interactive Proof
- zkSNARK

May introduce next time ………_(:з」∠)_ or reading [BCIOP2013]

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