Succinct Zero Knowledge Proofs and Arguments

Overview and techniques

Alexandros Zacharakis

20 May, 2021

Department of Information and Communication Technologies, University Pompeu Fabra

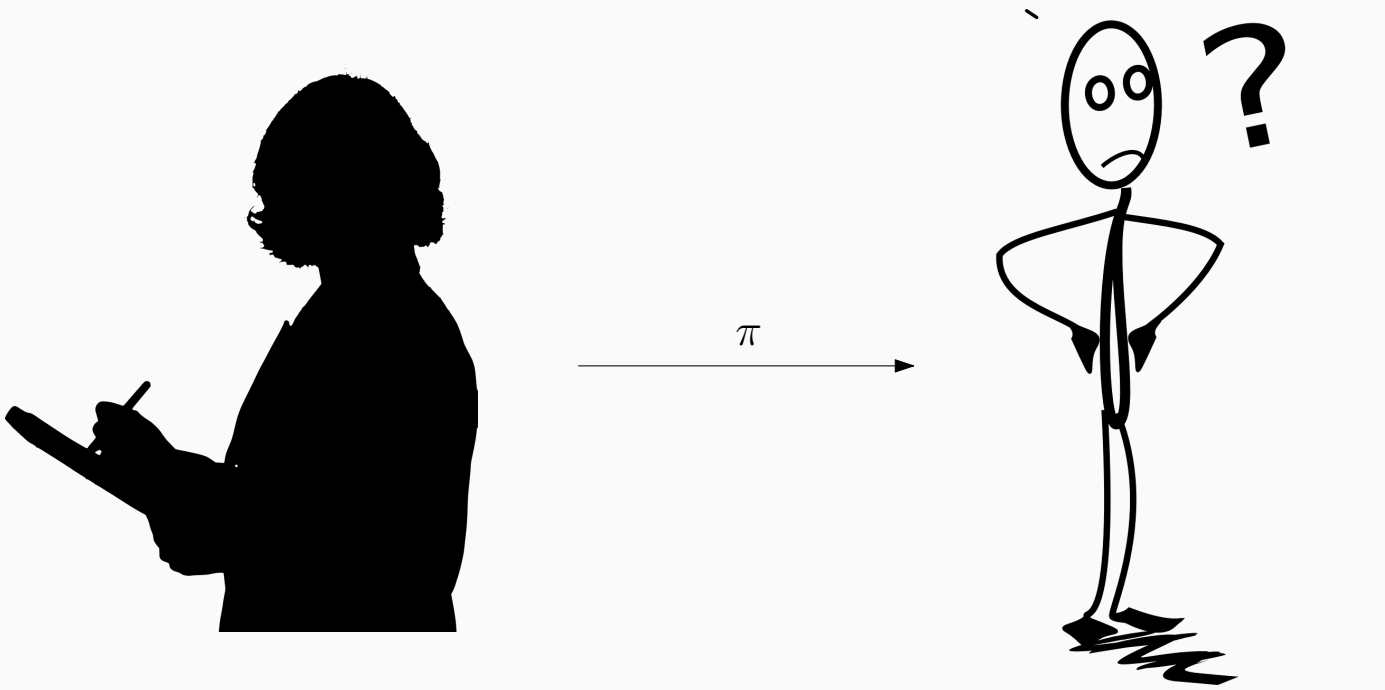
Part 1: High Level Overview

Modeling of proofs

- Central question in TCS: Model theorems and proofs
- Still not a good understanding
- Theorems as set membership ($\mathcal{L} = \{p \mid p \text{ encodes a prime number } \}$, $x \in_? \mathcal{L}$)
- Asysmptotic efficiency
- **P**: efficient (poly time) proving
- **NP**: efficient (poly time) verification: $\exists w$ such that $\mathcal{V}$ accepts x in poly time

Revolutionizing the notion of proofs

Classical proofs

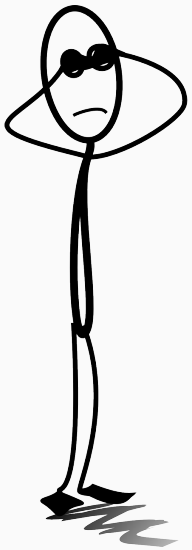


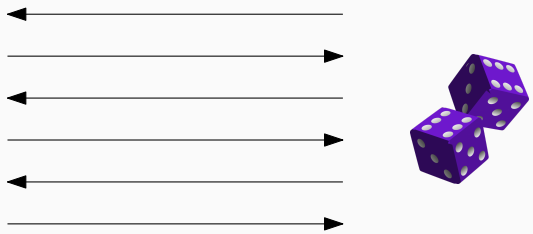
$f(x) = y$!

Hmmm, I accept!

Revolutionizing the notion of proofs

Interactive proofs





$f(x) = y$!

Hmmm, I accept!

Properties

- *Randomness* and *Interaction* $\rightarrow$ adaptive verifier
- Two requirements: Completeness and Soundness
  - The verifier *always* accepts valid statements
  - The verifier rejects invalid statements *except with negligible probability*
- Arguments: Soundness only against polynomial adversaries
- Knowledge Soundness: $\mathcal{P}$ *knows* a witness

$$\mathcal{L} = \{g, h \mid \exists x \text{ s.t. } h = g^x\}$$

- Zero Knowledge
  - Fascinating and counterintuitive concept
  - $\mathcal{V}$ learns *nothing but that the statement is true*
  - $\mathcal{V}^*$ interact with $\mathcal{P}$ and outputs some knowledge.
  - ZK if there exists $\mathcal{S}$ whose outputs is "identical"

(Almost) removing interaction ...again

Interaction has cons:

- $\mathcal{V}$ needs to be online
- Proofs are not publicly verifiable (ZK)

But... NIZK do not exist in the standard model.

Ways to remove it:

- Using idealized models $\rightarrow$ (Fiat-Shamir).
- 2 round IPs (setup & prove).

Extra property: *succinctness*

- Succinctness
  - λ sec. param., x input, w witness.
  - Proof size $\text{poly}(\lambda)o(|w|)$
  - Verification time $\text{poly}(\lambda)(|x| + o(|w|))$.
  - E.g. $\mathcal{O}(\lambda)$ communication and $\mathcal{O}(\lambda \cdot |x|)$ verification (!).

zk-SNARGs/zk-SNARKs

- zk: zero knowledge
- N: non-interactive
- S: succinct
- ARG (ARK): arguements (of knowledge)

But...

SNARGs do not exist under *falsifiable* assumptions!

Some notation first

Implicit notation for groups:

- Let $\mathbb{G}$ be a group and g a fixed generator.
- $[x]$ is the element g^x.

Example:

$$[1], [a], [b], [ab] \in \mathrm{DDH}$$

$$\equiv$$

$$g, g^a, g^b, g^{ab} \in \mathrm{DDH}$$

This emphasizes the underlying linear algebra

Efficient falsifiable assumptions

- Well defined game between (PPT) challenger $\mathcal{C}$ and adversary $\mathcal{A}$.
- After the game we can efficiently decide if $\mathcal{A}$ wins or loses.

Example: DLOG

1. $\mathcal{C}$ samples a group $\mathsf{gk} = (\mathbb{G}, p, g) \leftarrow \mathsf{GroupGen}(1^\lambda)$.
   and uniform $h = [x] \leftarrow \mathbb{G}$.
2. $x' \leftarrow \mathcal{A}(\mathsf{gk}, h)$.
3. $\mathcal{C}$ compares with $[x']$ with h.
   $\mathcal{A}$ wins if $h = [x'](= [x])$ and loses otherwise.

Non-Falsifiable assumptions: knowledge of exponents

- Consider a tuple $g = [1], g_1 = [a]$.
- An adversary $\mathcal{A}$ computes a DDH tuple $g = [1], g_1 = [a], g_2 = [b], g_3 = [ab]$
- This is easy!
  1. Sample $b \in \mathbb{Z}_p$
  2. Compute $g_2 = b[1] = [b]$ and $g_3 = b[a] = ab$

KEA states this is *the only way* this can be done:

$$\forall \text{ PPT } \mathcal{A} \; \exists \text{ PPT } \mathcal{E} \text{ s.t.}$$

$$\Pr\left[[b], [ab] \leftarrow \mathcal{A}([1], [a])\right] \approx \Pr\left[b \leftarrow \mathcal{E}([1], [a])\right]$$

Falsifiable vs non-falsifiable assumptions

- Black box vs non-black box or *what* vs *how*:
  - Falsifiable: assume no PPT $\mathcal{A}$ can do computation X
  - Non-falsifiable: assume *how* every PPT $\mathcal{A}$ does computation X
- Less understanding of non-falsifiable assumptions.
- Concrete meaning? (and so efficiency?)
  - Falsifiable:
    1. consider the best known attack
    2. consider the resources needed to perform it
    3. set λ accordingly
  - Non-falsifiable: there is no similar notion. How big λ?
- We *have to*[1] resort to (1) non-falsifiable assumptions or (2) heuristic methods!

[1] What about subclasses of **NP**? E.g. **P**?

The general framework

1. Start from a computation model (in this talk *arithmetic circuits*)
2. Translate it to an algebraic statement
3. Construct a proof for the statement:
   - Using probabilistic methods
   - Restricting the Prover capabilities (i.e. can only compute affine functions)
4. Compile it with cryptographic tools to enforce prover behavior

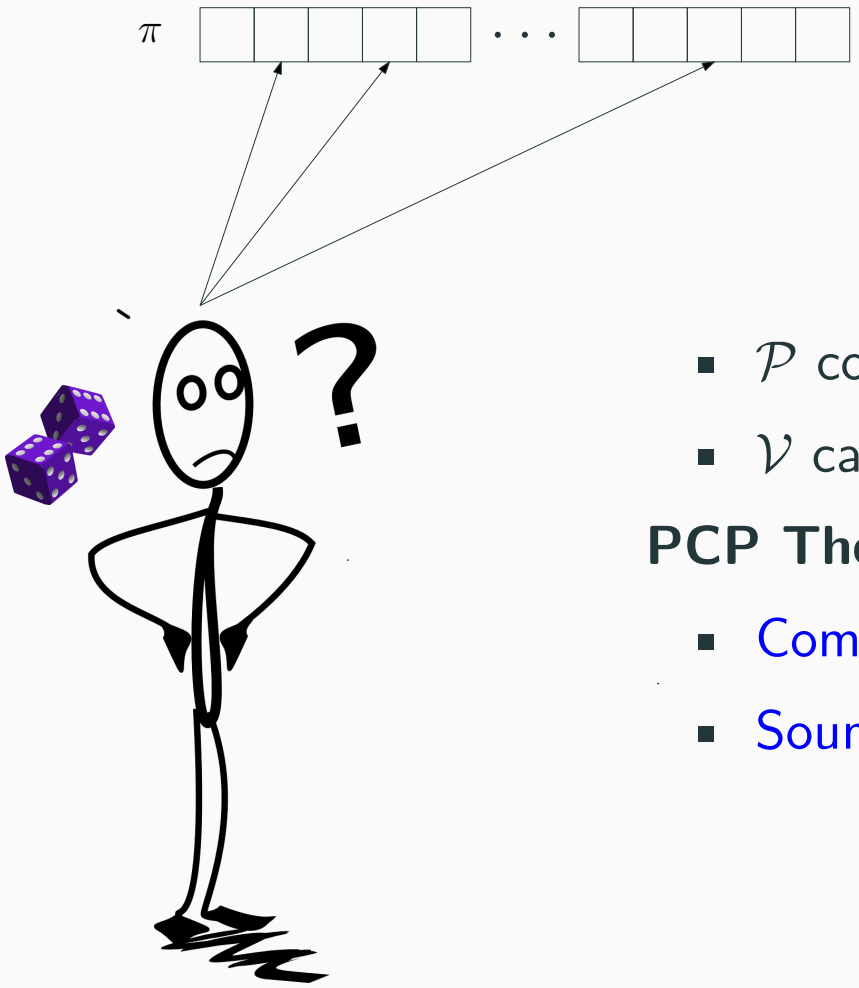
+: Separate combinatorial, probabilistic and cryptographic work

+: Easier understanding and analysis

+: Modular approach

- Not always clear how.
- Important to understand.

Information theoretic part: PCPs model



- $\mathcal{P}$ constructs PCP string π
- $\mathcal{V}$ can see random parts of π and decides

PCP Theorem: $\mathbf{PCP}[\log n, 1] = \mathbf{NP}$

- Completeness: $\forall x \in \mathcal{L}$ $\mathcal{V}$ accepts.
- Soundness: $\forall x \notin \mathcal{L}$ $\mathcal{V}$ rejects w.o.p.

Cryptographic compilation: Vector Commitments

Vector commitment

- $\mathcal{P}$ commits to $c = \mathsf{Com}(\mathsf{ck}, \vec{x} := (x_1, \ldots, x_n)^\top)$
- $\mathcal{P}$ gives openings $x_{i_1}, \ldots, x_{i_k}$ and PoD π
- $\mathcal{V}$ asserts opening: $0/1 \leftarrow \mathsf{Verify}(\mathsf{ck}, c, x_{i_1}, \ldots, x_{i_k}, \pi)$.

Security guarantee: Position binding:

- No (PPT) $\mathcal{P}$ can produce $x_{i_1}, \ldots x_{i_k} \neq y_{i_1}, \ldots y_{i_k}$ and π_x, π_y s.t.

$$\mathsf{Verify}(\mathsf{ck}, c, x_{i_1}, \ldots, x_{i_k}, \pi_x) = \mathsf{Verify}(\mathsf{ck}, c, y_{i_1}, \ldots, y_{i_k}, \pi_y) = 1.$$

Construction: CRHF & Merkle trees

Cryptographic compiler

Natural approach to "compile" PCPs to SNARGs:

- $\mathcal{P}$ commits to PCP string $c = \mathsf{Com}(\vec{\pi})$.
- $\mathcal{V}$ queries PCP by asking $\pi_{i_1}, \ldots, \pi_{i_k}$.
- $\mathcal{P}$ sends them with proof of correct opening w.r.t. c

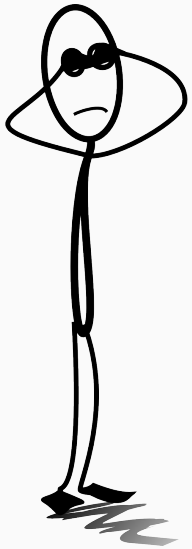
Binding: Infeasible to decommit $(\pi_{i_1}, \ldots, \pi_{i_k}) \neq (\pi'_{i_1}, \ldots, \pi'_{i_k})$. Effective as a PCP sting!

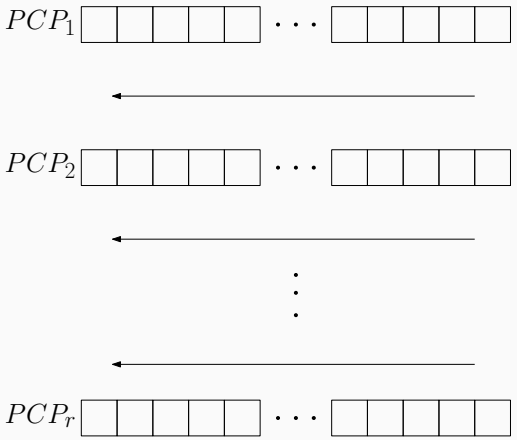
- If c and PoD are sublinear the proof is sublinear!
- Use FS to make it non-interactive.
- First succinct arguments.
- *Problem*: too complex and concretely inefficient.

+: transparent setup, (plausibly) PQ -: Bad concrete efficient

Extention: interactive oracle proofs

Interaction is powerful!





$f(x) = y$!

Hmmm, I accept!

1. Construct an interactive protocol.
2. Replace $\mathcal{P} \to \mathcal{V}$ messages with PCPs.
3. $\mathcal{V}$ queries PCPs in each round.

- Public coin $\mathcal{V} \to$ non-interactive.
- Compiled using vector commitments as well!

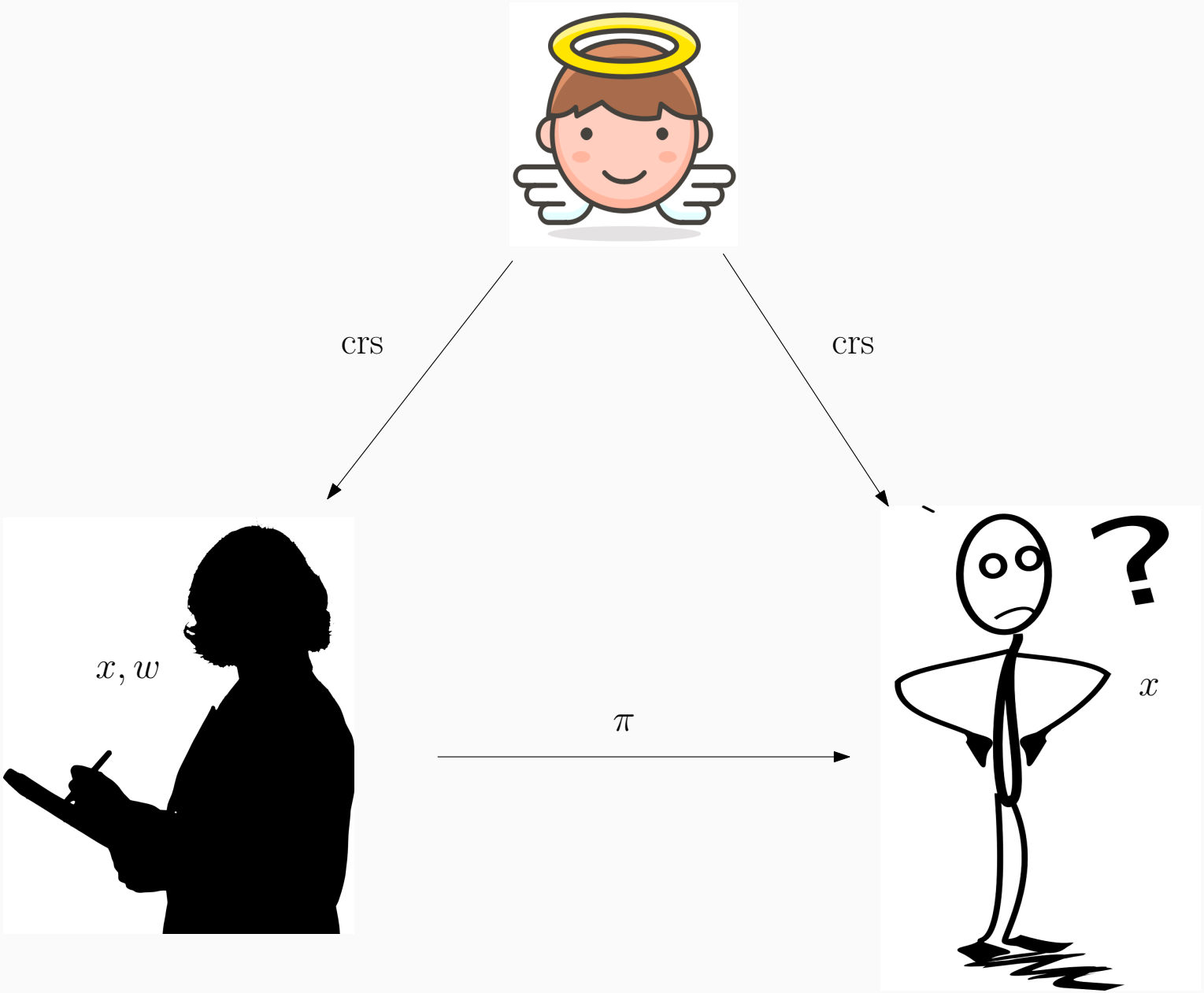
Preprocessing

- Let C be an arithmetic circuit.
- $\mathcal{L} = \{C, x \mid \exists w \text{ s.t. } C(x, w) = 1\}$.
- Minimum requirement: $\mathcal{V}$ should know statement!
- In general $|C| = \text{poly}(|w|)$!
- To overcome:
  1. Uniformity assumption on circuits.
  2. Preprocessing.

NIZK with preprocessing

- Not completely non-interactive.
- Two round protocols:
  1. First message depends on C, independent of x
  2. Second depends on x
- First message reusable! Reference string.
- Can use for many $x_1, x_2, \ldots$ "non-interactively".
- Different parties, same reference string.

NIZK with preprocessing



Reference string trust?

1.
   - Best trust is no trust!
   - Transparent: uniformly distributed.
2.
   - But... structure $\Rightarrow$ efficiency.
   - Structured reference string.
   - Produced with trapdoor (toxic waste).
   - Trapdoor allows faking proofs!

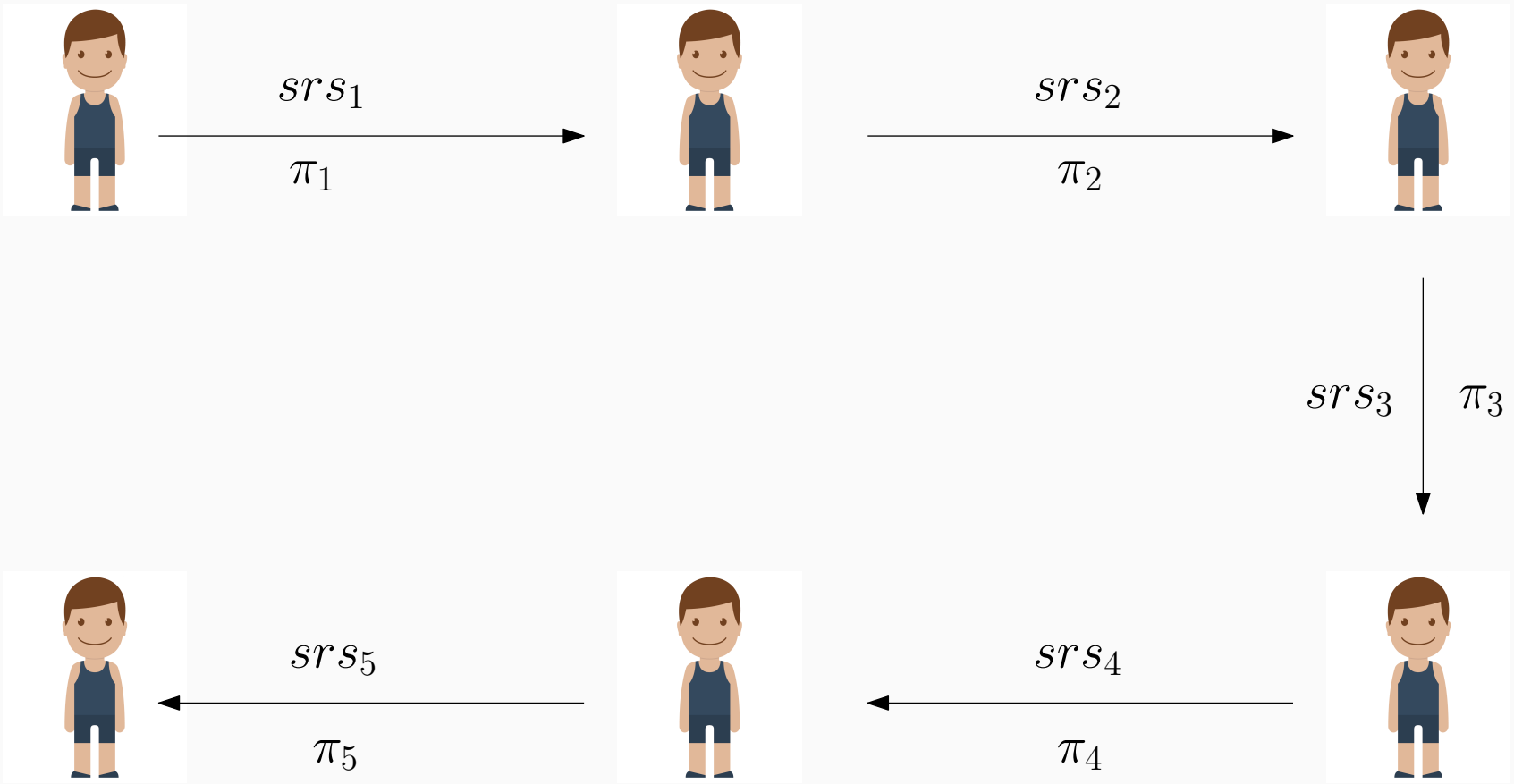
- Use MPC to create SRS.
- Assumption: one honest party.

Toxic waste

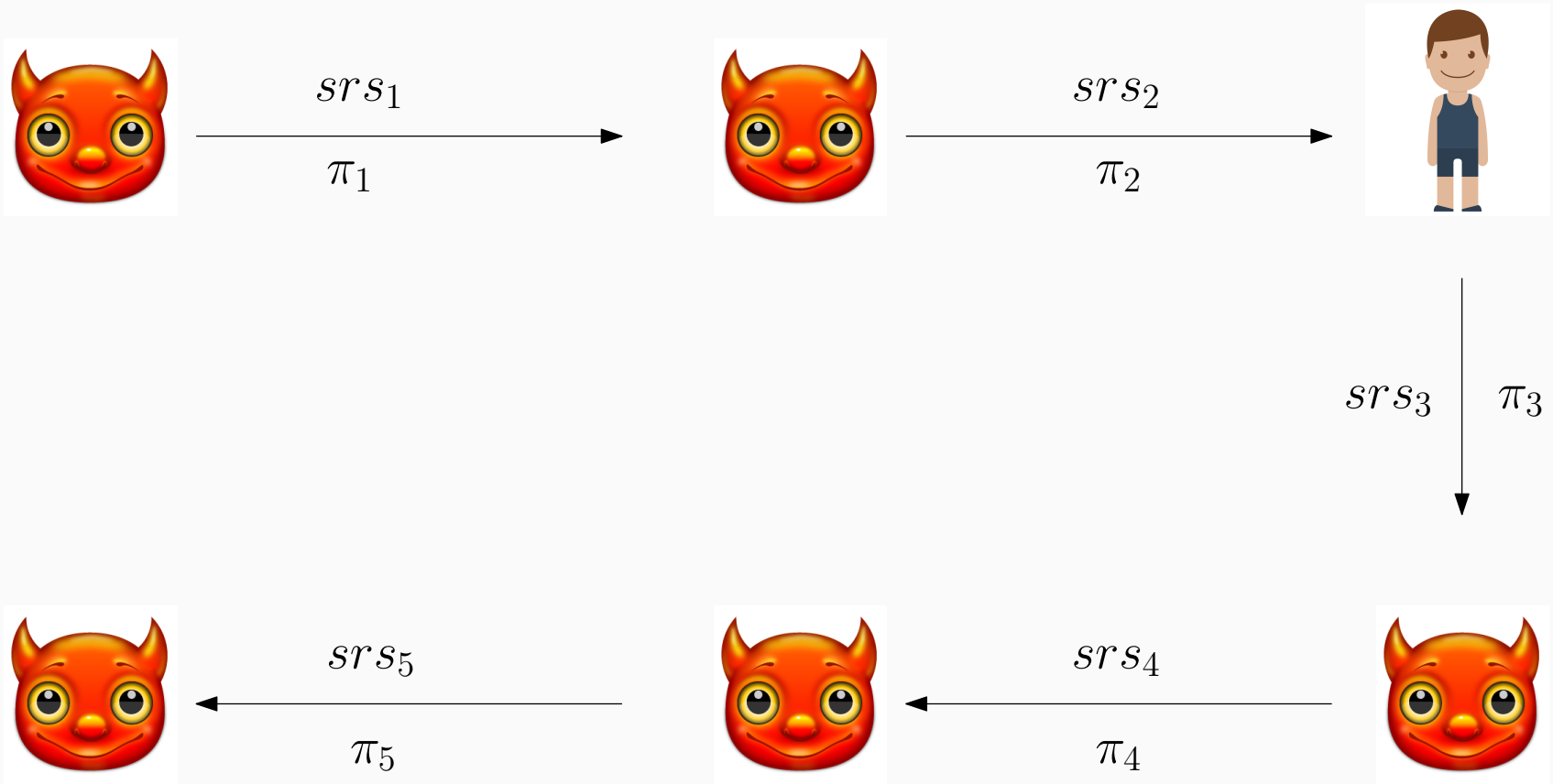
Minimizing trust: Updateable s.r.s

- Use MPC that allows participation of many parties.
- Non-interactive style MPC:
  - Party i gets srs_{i-1}.
  - Computes and announces srs_i.
  - Proves correct computation.
- One honest update guarantees trust.
- Thousands can participate!
- Impossibility: allows only limited structure...

Minimizing trust: Updateable s.r.s



Minimizing trust: Updateable s.r.s



Universal s.r.s

- Updateable or not, costly process.
- Ideally, should be run once.
- Universal: for any* circuit.
- Bugs, updates etc will not require recreation.

Universal s.r.s

- Create universal parameters srs_U. (needs trust!)
- For any C ($|C| \leq n$) *derive* srs_C from srs_U.
  - No trust.
  - E.g. deterministically from C, srs_U.
- Prove and verify statements using srs_C.

A good compromize of trust vs efficiency:

Universal and Updateable srs.

- Notion of *holographic proofs*.

Post-processing?

- Preprocessing: one costly round for future cheap uses
- Why not the other way as well? Postprocessing?

Let $\pi_1, \pi_2, \ldots$ be (expensive to check) proofs

- Check part of π_i.
- *Defer* rest part assuming it correct.
- Much later *aggregate* deferred parts.

Very powerful technique!

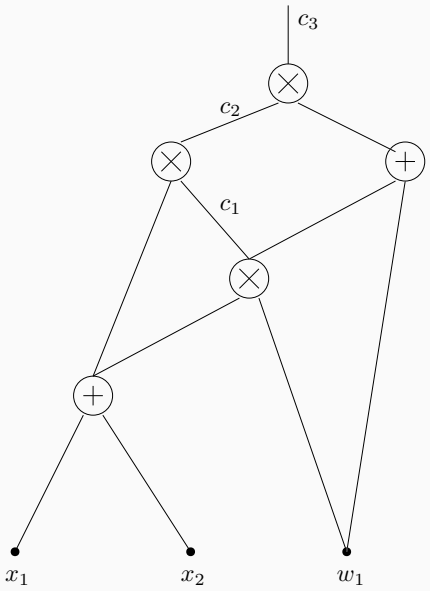
- Accumulated proofs and proof composition.
- Prove that a proof verifies.
- (Specific) NIZK with linear verification $\Rightarrow$ NIZK with (aggregated) sublinear verification!
- Hot new topic!

Some techniques for constructing SNARKs

It's all about aggregation

- Implicitly or explicitly every snark aggregates equations.
  1. Linear: $a_i = \sum w_i c_i$ for known w_i.
  2. Quadratic: $c_i = a_i b_i$.
  3. Combinations.
- Every arithmetic circuit can be written as constraints.
- We will review some techniques for it.

Algebraic reduction



$$c_1 = \ell_1 \cdot r_1 \qquad \ell_1 = x_1 + x_2 \qquad r_1 = w_1 + 0$$
$$c_2 = \ell_2 \cdot r_2 \qquad \ell_2 = x_1 + x_2 \qquad r_2 = c_1 + 0$$
$$c_3 = \ell_3 \cdot r_3 \qquad \ell_3 = 0 + c_2 \qquad r_3 = c_1 + w_3$$

$$c_1 = (x_1 + x_2)w_1$$
$$c_2 = (x_1 + x_2)c_1$$
$$c_3 = c_2(c_1 + w_3)$$

$$\vec{c} = \vec{\ell} \circ \vec{r}$$

$$\vec{\ell} = \mathbf{L} \cdot \begin{pmatrix} \vec{x} \\ \vec{w} \\ \vec{c} \end{pmatrix} \qquad \vec{r} = \mathbf{R} \cdot \begin{pmatrix} \vec{x} \\ \vec{w} \\ \vec{c} \end{pmatrix}$$

Algebraic reduction

More generally:

1. quadratic: $c = \ell \circ r$,
2. linear: $\ell = Lc$, $r = Rc$

or equivalently

$$c = (L \cdot c) \circ (R \cdot c)$$

Heavy machinery: Schwartz-Zippel lemma

Lemma
Let $p \in \mathbb{F}[X_1, X_2, \ldots, X_k]$ be a non-zero polynomial with individual degree d. Then

$$\Pr_{s_1,\ldots,s_k \leftarrow \mathbb{F}} [p(s_1, s_2, \ldots, s_k) = 0] \leq \frac{dk}{|\mathbb{F}|}$$

For univariate polynomials: non-zero $p \in \mathbb{F}[X]$ can have at most d roots.

Allows polynomial testing:

$$\text{for } \vec{s} \leftarrow \mathbb{F}\ p(s_1, \ldots, s_k) = 0 \quad \Rightarrow_{\mathbb{F}} \quad p(X_1, \ldots, X_k) \equiv 0$$

Detour: pairing groups

- $(\mathbb{G}, \mathbb{G}_T, p, e) \leftarrow \mathsf{GroupGen}(1^\lambda)$.
- Fix $\mathcal{P}$ generator of $\mathbb{G}$, $e(\mathcal{P}, \mathcal{P})$ generator of $\mathbb{G}_T$.
- Implicit notation.
- e is a bilinear operation $\mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$

$$e([a]_1, [b]_2) = e([1]_1, [b]_2)^a = e([a]_1, [1]_2)^b = e([1]_1, [1]_2)^{ab}$$

Detour: pairing groups

Checking Diffie-Hellman relations:

- Claim: $A \in \mathbb{G}_1, B \in \mathbb{G}_2, C \in \mathbb{G}_1$:

$$A = [a]_1 \qquad B = [b]_2 \qquad C = [ab]_1$$

- We can efficiently check this in pairing groups!
- Effectively used as DDH oracle.
- Let $C = [c]$ for some c. Then

$$T_1 = e(A, B) = e([a]_1, [b]_2) = e([1]_1, [b]_2)^a = e([1]_1, [1]_2)^{ab}$$

$$T_2 = e(C, [1]_2) = e([c]_1, [1]_2) = e([1]_1, [1]_2)^c$$

- $T_1 = T_2$ iff $ab = c$!

Detour: pairing groups

- In DLOG setting:
  - linear operations in the exponent.
  - CDH, DDH hard.
  - DLOG setting $\equiv$ linear algebra in exponent
- In pairing setting:
  - Additionally check quadratic relations!

Example: Given a_1, a_2, a_3, a_4, a_5 in the exponent test

$$3a_1a_3 + 8a_2 = 5a_1a_4 + 3a_2a_5$$

- in DLOG setting imposible... DDH
- in pairing setting, if a_1, a_2 encoded in $\mathbb{G}_1$, and a_3, a_4, a_5 in $\mathbb{G}_2$

$$e(3[a_1]_1, [a_3]_2)e(8[a_2]_1, [1]_2) = e(5[a_1]_1, [a_4]_2) + 3e([a_2]_1, [a_5]_2)$$

Lagrange basis

Basis for $\mathbb{F}[X]$ with degree less than n.

- Monomial basis: $\{1, X, \ldots, X^{n-1}\}$
- Lagrange basis w.r.t. set S of size n: $\{\lambda_{s_1}(X), \lambda_{s_2}(X), \ldots, \lambda_{s_n}(X)\}$

The main property of Lagrange basis is it is orthogonal:

$$\forall s_i, s_j \in S: \ \lambda_{s_i}(s_j) = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases}$$

Concretely (but not importantly)

$$\lambda_{s_i}(X) = \prod_{s_j \in S \setminus s_i} \frac{X - s_j}{s_i - s_j}$$

Lagrange basis

- Let $\vec{\lambda}(X) = (\lambda_1(X), \ldots, \lambda_n(X))^\top$.
- Let $\vec{c} = (c_1, \ldots, c_n)^\top$.

The polynomial

$$p(X) = \vec{c}^\top \vec{\lambda}(X) = c_1\lambda_1(X) + \ldots + c_n\lambda_n(X)$$

is the *unique* polynomial with:

- degree less than n
- for all $s_i \in S$: $p(s_i) = c_i$

Let $t(X) = \prod_{s_i \in S}(X - s_i)$ be the *vanishing* polynomial in S.

Lagrange basis

Lemma
Let $\vec{a}, \vec{b}, \vec{c} \in \mathbb{F}^n$ *and* $a(X) = \vec{a}^\top \vec{\lambda}(X)$, $b(X) = \vec{b}^\top \vec{\lambda}(X)$ $c(X) = \vec{c}^\top \vec{\lambda}(X)$

Then, $\vec{c} = \vec{a} \circ \vec{b}$ iff $\exists h(X)$ s.t. $a(X)b(X) = c(X) + h(X)t(X)$.

Proof.
$(\Rightarrow)$

- Consider $p(X) = a(X)b(X) - c(X)$.
- For all $s_i \in S$: $p(s_i) = a(s_i)b(s_i) - c(s_i) = a_i b_i - c_i = 0$.
- Each s_i is a root. Equivalently:

  $p(X) = (X - s_1) \dots (X - s_n)h(X) = t(X)h(X)$

$(\Leftarrow)$

- for all s_i $t(s_i) = 0$.
- Thus $a(s_i)b(s_i) = c(s_i) \iff a_i b_i = c_i$.

□

Proving quadratic constraints using Lagrange basis

- Let $\vec{a}, \vec{b}, \vec{c}$
- Claim: $\forall i \; a_i b_i = c_i$.
- let

$$a(X) = \vec{\lambda}(X)^\top \vec{a}, \quad b(X) = \vec{\lambda}(X)^\top \vec{b}, \quad c(X) = \vec{\lambda}(X)^\top \vec{c},$$

Constraints hold $\iff$ $p(X) = a(X)b(X) - c(X) - h(X)t(X) \equiv 0$
Enough to sample s and check $p(s) = 0$.

Note: this is a trivial statement. It becomes interesting when $\vec{c}, \vec{a}$ and $\vec{c}, \vec{b}$ are linearly related. The next is only for educational purposes.

Proving quadratic constraints using Lagrange basis

- Sample $s \leftarrow \mathbb{F}$ and compute

$$[1]_{1,2}, [s]_{1,2} \ldots, [s^{n-1}]_{1,2}, [T]_2 = [t(s)]_2$$

- Note: $[\lambda_i(s)]$ computable by $[1], \ldots, [s]^{n-1}$.
- $\mathcal{P}$ computes

$$[A]_1 = [\lambda(s)]_1^\top \vec{a}, \quad [B]_2 = [\lambda(s)]_2^\top \vec{b}, \quad [C]_1 = [\lambda(s)]_1^\top \vec{c},$$

$$h(X) \text{ s.t. } a(X)b(X) = c(X) + h(X)t(X)$$

$$[H]_1 = [h(s)]_1$$

- Verifier checks

$$e([A]_1, [B]_2) = e([C]_2, [1]_2)e([H]_1, [T]_2)$$

$$a(s)b(s) - c(s) + h(s)t(s) = 0$$

Proving quadratic constraints using Lagrange basis

- Assume we could extract $\vec{a}, \vec{b}, \vec{c}, \vec{h}$ from verifying proof.
- We have computed $h(s)$ s.t.:

$$h(s) = p(s)/t(s) \text{ where } p(s) = a(s)b(s) - c(s)$$

- If additionally $\vec{a} \circ \vec{b} \neq \vec{c}$ then $t(X)$ does not divide $p(X)$.

One can reduce this to a (falsifiable) assumption.

- How to extract $\vec{a}, \vec{b}, \vec{c}$?
- Note that proof contains much less information that $\vec{a}, \vec{b}, \vec{c}$.
- We resort to non-falsifiable assumptions.

Extracting the witness

- Include in the srs: $([\alpha\lambda_i(s)]_1, [\alpha]_2)$.
- $\mathcal{P}$ gives everything twice

$$V = [v]_1, \quad V' = [\alpha v]_1$$

- Consider $([V]_1, [V']_1)$ an encoding of v.
- $\mathcal{V}$ checks an encoding is valid by testing

$$e([V]_1, [\alpha]_2) = e([V']_1, [1]_2)$$

$$e([v]_1, [\alpha]_2) = e([v']_1, [1]_2) \iff e([1]_1, [1]_2)^{\alpha v} = e([1]_1, [1]_2)^{v'}$$

n-power Knowledge of Exponent:

For all PPT $\mathcal{A}$ with input $([1]_1, [s]_1, \ldots, [s]_q, [\alpha]_1, [\alpha s]_1, \ldots, [\alpha s^n]_1, [\alpha]_2$ that outputs $[A]_1$, $[\alpha A]_1$, there exists a PPT extractor $\mathcal{E}$ that outputs a_i such that $A = \sum a_i s^i$.

Only way to compute encodings is to "know" linear combinations

Adding zero knowledge

- Usually easy task.
- General strategy: have $\mathcal{P}$ give uniform elements that make the proof verify
- Include in srs $[T]_1 = [t(s)]_1$ as well.
- $\mathcal{P}$ computes

$$[A]_1 = [\lambda(s)]_1^\top \vec{a} + r_a[T]_1, \quad [B]_2 = [\lambda(s)]_2^\top \vec{b} + r_b[T]_2,$$

$$[C]_1 = [\lambda(s)]_1^\top \vec{c} + r_c[T]_1,$$

$$h(X) \text{ s.t. } a(X)b(X) = c(X) + h(X)t(X)$$

$$[H]_1 = [h(s)]_1 + (r_a r_b - r_c)[T]_1$$

- Verifier checks

$$e([A]_1, [B]_2) = e([C]_2, [1]_2)e([H]_1, [T]_2)$$

- All elements are uniform *conditioned* on the verification equation.

Aggregating linear equations

- Let $\vec{w}_i$ be m public vectors with dimension n.
- The claim is $a_i = \vec{w}_i^\top \vec{c}$.
- Concatenating $\vec{w}_i$ to a matrix we can write $\vec{a} = \mathbf{W}\vec{b}$.
- *Strategy*:
  1. Collapse all m equations to 1. Aggregation!
  2. Prove the *aggregated* equation holds.

Collapsing the equations

- First rewrite $a_i - \vec{w}_i^\top \vec{c} = 0$
- Embed each equation as a coefficient of a monomial.
- The i-th equation is $Y^{i-1}(a_i - \vec{w}_i^\top \vec{c}) = 0$
- Add all the equations:

$$\sum Y^{i-1}(a_i - \vec{w}_i \vec{c}) = 0 \tag{1}$$

- Consider the above as a polynomial $P(Y)$.
- Equations hold $\iff$ $P(Y) \equiv 0$.
- Polynomial identity testing! sample $y \leftarrow \mathbb{F}$ and check $P(y) = 0$.
- By Schwartz-Zippel lemma equations are satisfied w.o.p.
- *Note*: denoting $\vec{y} = (1, y, \ldots, y^{n-1})^\top$, the check can be thought as

$$\vec{y}^\top (\vec{W}\vec{c} - \vec{a}) = 0$$

Tool: Pedersen commitment

- Commitment scheme with $\mathcal{M} = \mathbb{F}^n$, $\mathcal{C} = \mathbb{G}$.
- $\mathsf{ck} := [\vec{r}] \leftarrow \mathbb{G}^n$, $[\rho] \leftarrow \mathbb{G}$.
- $[c] = [\vec{r}]^\top \vec{m} + [\rho] z$
- To open $\mathcal{P}$ just gives $\vec{m}, z$.

Security:

- Computationally binding under DLOG.
- Perfectly hiding: $[\rho] z$ perfectly randomized commitment.

Inner product argument

Consider the non-hiding version ($\rho = 0$)

$$\mathcal{L}_{\vec{r},\vec{s}} = \left\{ [c], [d], z \mid \exists \vec{a}, \vec{b} \text{ s.t. } [c] = [\vec{r}]^\top \vec{a} \wedge [d] = [\vec{s}]^\top \vec{b} \wedge z = \vec{a}^\top \vec{b} \right\}$$

- Let n be the vector size.
- Call n the size of the statement.
- Strategy: reduce it to $x' \in \mathcal{L}_{\vec{r}',\vec{s}'}$ with size n'.
- Apply recursively!

IPA with the folding technique

- The commitment key is $\vec{r}$
- Denote $\vec{r}_L, \vec{r}_R$ the leftmost and rightmost $n/2$ elements.
- Similarly $\vec{a}_L, \vec{a}_R$ for the committed values.
- $\mathcal{V}$ has $[c], [d], z$
- $\mathcal{P}$ has also witness $\vec{a}, \vec{b}$

IPA with the folding technique

- $\mathcal{P}$ commits to the "cross terms"

$$[c_{-1}] = [\vec{r}_L]\vec{a}_R, \qquad [c_1] = [\vec{r}_R]\vec{a}_L$$
$$[d_{-1}] = [\vec{s}_L]\vec{b}_R, \qquad [d_1] = [\vec{s}_R]\vec{b}_L$$
$$z_{-1} = \vec{a}_L\vec{b}_R, \qquad z_1 = \vec{a}_R\vec{b}_L$$

- $\mathcal{V}$ sends a random challenge $x \leftarrow \mathbb{F}$.
- $\mathcal{P}$ and $\mathcal{V}$ compute the new statement:

$$[\vec{r}'] = [\vec{r}_L] + x[\vec{r}_R], \qquad [\vec{s}'] = [\vec{s}_L] + x^{-1}[\vec{s}_R]$$
$$[c'] = x^{-1}[c_{-1}]_1 + [c] + x[c_1]_1, \qquad [d'] = x[c_{-1}]_1 + [c] + x^{-1}[c_1]_1$$
$$z' = z_{-1}x + z + z_1x^{-1}$$

- $\mathcal{P}$ computes the new witness

$$\vec{a}' = \vec{a}_L + \vec{a}_Rx^{-1}, \qquad \vec{b}' = \vec{b}_L + \vec{b}_Rx$$

- The "folded" statement is of half size!
- When the statement is small (e.g. $\mathcal{O}(1)$) $\mathcal{P}$ simply sends the witness.

Completeness of recursion step

1. $[c']$ is a commitment to $\vec{a'}$ w.r.t. $\vec{r'}$:

$$\begin{aligned}\vec{r'}^{\top}\vec{a'} &= (\vec{r}_L + x\vec{r}_R)^{\top}(\vec{a}_L + \vec{a}_R x^{-1}) \\ &= \vec{r}_L^{\top}\vec{a}_L + \vec{r}_L^{\top}\vec{a}_R x^{-1} + \vec{r}_R^{\top}\vec{a}_L x + \vec{r}_R^{\top}\vec{a}_R \\ &= c_{-1}x^{-1} + c + c_1 x = c'\end{aligned}$$

2. $[d']$ is a commitment to $\vec{b'}$ w.r.t. $\vec{s'}$ (identically)
3. $z' = \vec{a'}^{\top}\vec{b'}$:

$$\begin{aligned}\vec{a'}^{\top}\vec{b'} &= (\vec{a}_L + \vec{a}_R x^{-1})^{\top}(\vec{b}_L + \vec{b}_R x) \\ &= \vec{a}_L^{\top}\vec{b}_L + \vec{a}_L^{\top}\vec{b}_R x + \vec{a}_R^{\top}\vec{b}_L x^{-1} + \vec{a}_R^{\top}\vec{b}_R) \\ &= z_{-1}x + z + z_1 x^{-1} = z'\end{aligned}$$

Soundness of the reduction step

Strategy: induction!

- Assume we can extract a witness for the $(i+1)$-th iteration.
- We will extract a witness for the i-th iteration.
- Since in the end $\mathcal{P}$ sends witness, base case trivially holds.
- Basic tool: Rewind the prover.
- Get enough transcripts to extract i-th level witness.

Soundness of the reduction step

Consider we extract at level i.

- We get the cross term commitments.
- By I.H. for any uniform x, we get $\vec{a'}, \vec{b'}$ valid witness for next iteration $i+1$.
- Rewind for 3 challenges.
- By linear algebra calculations we can get valid witness $\vec{a}$, $\vec{b}$
- Intuition:
  - The prover gives the cross terms *before* seeing the challenge.
  - The messages "behave" similarly to degree 3 polynomials.
  - We can "interpolate" them in the exponent.

Only strategy for adversary: use different representations for the elements w.r.t. commitment keys

Infeasible by binding property (soundness under DLOG)!

Protocol Properties:

- Transparent!
- $\log n$ rounds.
- $\mathcal{O}_\lambda(\log n)$ communication: 6 elements per round
- $\mathcal{O}_\lambda(n)$ prover (no expensive interpolation)
- $\mathcal{O}_\lambda(n)$ verifier:

  in each round compute the new key...
- Public coin: can turn to non-interactive using FS!

Exercise:

Consider a commitment key $[\vec{r}]$ with $n = 2^\nu$ elements. Assume $\mathcal{P}$ and $\mathcal{V}$ execute an iteration of the IP protocol and verifier uses randomness $x_1, x_2, \ldots, x_\nu$. Find an expression for the final (one element) key at the end of the protocol.

Back to aggregating m linear equations

Prove commitments to $\vec{a}$ and $\vec{b}$ satisfy $\vec{a} = \mathbf{W}\vec{b}$.

- srs $= [\vec{r}]_1$
- $\mathcal{P}$ claims:
  1. $[\gamma]$ is commitment to $\vec{c}$: $[\gamma] = [\vec{r}]^\top \vec{c}$.
  2. $[\alpha]$ is commitment to $\vec{a}$: $[\alpha] = [\vec{r}]^\top \vec{a}$.
  3. $\vec{a} - \vec{W}\vec{c} = \vec{0}$ for public $\vec{W}$.
- $\mathcal{V}$ sends challenge y. Let $\vec{y} = (1, y, \ldots y^{n-1})^\top$.
- $\mathcal{P}$ will convince it that $\vec{y}^\top(\vec{a} - \vec{W}\vec{c}) = \vec{y}^\top\vec{a} - (\vec{y}^\top\vec{W})\vec{c} = 0$.
- This reduces easily to two inner products!
- Let

$$[\upsilon] = [\vec{r}]^\top \vec{y}, \qquad [\omega] = [\vec{r}]^\top(\vec{y}^\top \vec{W}),$$

- Verifier can compute these commitments on its own!
- The two claims are

$$([\upsilon], [\vec{a}], k) \in \mathcal{L}_{\vec{r}}, \qquad ([\omega], [\vec{c}], k) \in \mathcal{L}_{\vec{r}},$$

What about Zero knowledge?

- Many ways to achieve it.
- Should use hiding commitments.
- Can randomize $\mathcal{P}$ messages.
- We demonstrate a technique using Σ protocols.

What about Zero knowledge?

Simple HVZK protocol for

$$\mathcal{L}_{\vec{r},\rho} = \left\{([\gamma], \vec{a}, k) \mid \exists \vec{c}, z \text{ s.t. } [\gamma] = [\vec{r}]^\top \vec{c} + [\rho] z \wedge \vec{a}^\top \vec{c} = k\right\}$$

- $\mathcal{V}$ has $[\vec{r}], [\rho], [\gamma], \vec{a}, k$. $\mathcal{P}$ has witness $\vec{c}, z$.
- The protocol goes as follows:
  1. $\mathcal{P}$ samples $\vec{d} \leftarrow \mathbb{F}^n$, $\sigma \leftarrow \mathbb{F}$ and sends
  $$[\delta] = [\vec{r}]^\top \vec{d} + [\rho]\sigma, \qquad \ell = \vec{a}^\top \vec{d}$$
  2. $\mathcal{V}$ sends $x \leftarrow \mathbb{F}$
  3. $\mathcal{P}$ replies with
  $$\vec{z} = x\vec{c} + \vec{d}, \qquad \phi = c\rho + \sigma$$
  4. $\mathcal{V}$ checks
  $$\vec{r}^\top \vec{z} + \rho\phi = [\delta] + x[\gamma] : \ \vec{z} \text{ is a commitment to } [\delta] + c[\gamma]$$
  $$\vec{a}^\top \vec{z} = k + x\ell := \vec{a}^\top (c + x\vec{d})$$

What about Zero knowledge?

- Simple to verify it is
  1. Complete
  2. Special sound
  3. HVZK
- Hint: modify Σ protocol proofs.
- We can compress the last step!
  - Prover gives a trivial (non-ZK) prove that

$$\vec{r}^{\top}\vec{z} + \rho\phi = [\delta] + c[\gamma]$$

$$\vec{a}^{\top}\vec{z} = k + x\ell$$

  - Equivalently,
    1. $(\vec{z}^{\top}, \phi)$ is an opening of $[\delta] + c[\gamma]$
    2. $(\vec{a}^{\top}, 0)$ is an opening of some commitment $[\epsilon]$
    3. The inner product of the openings is $k + c\ell$
  - Instead of doing the last step do logarithmic IP.
  - Need *not* be ZK!

Exercise:

A polynomial commitment allows a prover to (succinctly) commit to a polynomial $p(X)$ of degree less than n and later reveal one (or many) openings $k = p(x)$. It should be

- Binding: $\mathcal{P}$ cannot produce (1) commitment c, (2) two different openings $y_1 \neq y_2$ for some point x and (3) a verifying proof.
- Hiding: the opening/proof reveals nothing more than the fact the $p(x) = y$.

1. Use Pedersen commitment + IPA to construct a (non-hiding) P.C. with commitment size $\mathcal{O}_\lambda(1)$ and opening proof $\mathcal{O}_\lambda(\log n)$.
2. Use the sigma protocol technique to make it hiding (under FS transform).

Multivariate sumcheck protocol

- Celebrated result! (IP=PSPACE)
- Let $p \in \mathbb{F}[X_1, \ldots, X_n]$ be a multivariate polynomial with individual degree d.
- $\mathcal{P}$ claims $\sum_{\vec{b} \in \{0,1\}^n} p(\vec{b}) = k$.
- Inductive IP.
- Use structure of the polynomial.

Multivariate sumcheck protocol

- $\mathcal{P}$ sends

$$p_1(X) = \sum_{(b_2,\ldots b_n)\in\{0,1\}^{n-1}} p(X, b_2, \ldots, b_n)$$

- $\mathcal{V}$ asserts $p_1(0) + p_1(1) = k$. It sends $s_1 \leftarrow \mathbb{F}$ to $\mathcal{P}$.
- $\mathcal{P}$ sends

$$p_2(X) = \sum_{(b_3,\ldots b_n)\in\{0,1\}^{n-2}} p(s_1, X, b_2, \ldots, b_n)$$

- $\mathcal{V}$ asserts $p_2(0) + p_2(1) = p_1(s_1)$. It sends $s_2 \leftarrow \mathbb{F}$ to $\mathcal{P}$.

$$\vdots$$

- $\mathcal{V}$ asserts $p_n(0) + p_n(1) = p_{n-1}(s_{n-1})$.
- $\mathcal{V}$ samples s_n and checks $p_n(s_n) = p(s_1, \ldots, s_n)$.

Multivariate sumcheck protocol: soundness

- Consider round 1.
- Assume $\sum p(X_1, \ldots, X_n) \neq k$
- Since $p_1(0) + p_1(1) \neq k$ the prover has to send:

$$p_1^*(X) \neq p_1(X) \text{ such that } p_1(0) + p_1(1) \neq k$$

- Two cases:
  1. $p_1^*(s_1) = p_1(s_1)$: SZ lemma $\Rightarrow \frac{d}{\mathbb{F}}$ probability
  2. $\mathcal{P}$ has to lie at next round for p_2:
     otherwise $p_2(0) + p_2(1) = p_1(s_1) \neq p_1(s_1)$
- Continue inductively.
- In the last round $\mathcal{P}$ *cannot* cheat! $\mathcal{V}$ computes last evaluation on its own!
- Soundness error $\leq \frac{n \cdot d}{\mathbb{F}}$

Multivariate sumcheck protocol: soundness

- Consider round 1.
- Assume $\sum p(X_1, \ldots, X_n) \neq k$
- Since $p_1(0) + p_1(1) \neq k$ the prover has to send:

$$p_1^*(X) \neq p_1(X) \text{ such that } p_1(0) + p_1(1) \neq k$$

- Two cases:
  1. $p_1^*(s_1) = p_1(s_1)$: SZ lemma $\Rightarrow \frac{d}{\mathbb{F}}$ probability
  2. $\mathcal{P}$ has to lie at next round for p_2:
     otherwise $p_2(0) + p_2(1) = p_1(s_1) \neq p_1(s_1)$
- Continue inductively.
- In the last round $\mathcal{P}$ *cannot* cheat! $\mathcal{V}$ computes last evaluation on its own!
- Soundness error $\leq \frac{n \cdot d}{\mathbb{F}}$

Multivariate sumcheck protocol: soundness

- n rounds.
- $\mathcal{O}_{\mathbb{F}}(d)$ communication.
- $\mathcal{O}_{\mathbb{F}}(d)$ verifier computation per round and 1 evaluation of p.
- Public coin.

Univariate sumcheck protocol

- We want a similar approach for univariate polynomials.
- $p \in \mathbb{F}[X]$ of degree less than n
- For $S \subseteq \mathbb{F}$: $\sum_{s \in S} p(s) = 0$.
- Useful tool for universal SNARKs.
- We will consider S with structure.

Lagrange basis for multiplicative subgroups

- Consider a subgroup $\mathbb{H} \subseteq \mathbb{F}$ of size n.
- Let $\vec{\lambda}(X)$ be the Lagrange basis associated to $\mathbb{H}$.
- Then:

$$\lambda_i(X) = \frac{h_i}{n} \frac{(X^n - 1)}{(X - h_i)} \qquad t(X) = X^n - 1 \qquad \lambda_i(0) = \frac{1}{n}$$

- Easy to verify properties!
- Efficiency: $\lambda_i(X), t(X)$ computable in $\mathcal{O}_{\mathbb{F}}(\log n)$.

Univariate sumcheck protocol

Theorem
Let $S \subseteq \mathbb{H}$. *Then* $\sum_{s \in S} p(s) = \sigma$ *iff* $\exists h(X), r(X)$ *with* $\deg(r) \leq n - 1$ *s.t.*

$$p(X)n - \sigma = Xr(X) + t(X)h(X)$$

Derieved protocol from polynomial commitments (High level):

- $\mathcal{P}$ commits to $p(X)$. Claims $\sum_{s \in S} P(X) = \sigma$.

  Also sends commitments to $h(X), r(X)$.
- $\mathcal{V}$ sends $x \leftarrow \mathbb{F}$.
- $\mathcal{P}$ opens $p_x = p(x), h_x = h(x), r_x = r(x)$ and proofs of opening.
- $\mathcal{V}$ asserts

$$p_x n - \sigma = x r_x + t(x) h_x$$


THANK
YOU