Succinct Zero Knowledge Proofs and Arguments

Overview and techniques

Alexandros Zacharakis alexandros.zacharakis@upf.edu 20 May, 2021

Department of Information and Communication Technologies, University Pompeu Fabra

Part 1: High Level Overview

- Central question in TCS: Model theorems and proofs
- Still not a good understanding
- Theorems as set membership (L = {p | p encodes a prime number }, x ∈_? L)
- Asysmptotic efficiency
- P: efficient (poly time) proving
- NP: efficient (poly time) verification: ∃w such that V accepts x in poly time

Classical proofs



Interactive proofs



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: *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 S whose outputs is "identical"

Interaction has cons:

- $\ensuremath{\mathcal{V}}$ 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).

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

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

But...

SNARGs do not exist under falsifiable assumptions! [GW11]

Implicit notation for groups:

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

Example:

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

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

This emphasizes the underlying linear algebra

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

Example: DLOG

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

A wins if h = [x'](= [x]) and loses otherwise.

- Consider a tuple $g = [1], g_1 = [a]$.
- An adversary A computes a DDH tuple g = [1], g₁ = [a], g₂ = [b], g₃ = [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:

 $\forall \mathsf{PPT} \ \mathcal{A} \exists \mathsf{PPT} \ \mathcal{E} \mathsf{ s.t.}$ $\mathsf{Pr}\left[[b], [ab] \leftarrow \mathcal{A}([1], [a])\right] \approx \mathsf{Pr}\left[b \leftarrow \mathcal{E}([1], [a])\right]$

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!

¹What 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: $PCP[\log n, 1] = NP$

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

Vector commitment

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

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.

$$\mathsf{Verify}(\mathsf{ck}, \mathit{c}, \mathit{x}_{i_1}, \ldots, \mathit{x}_{i_k}, \pi_{\mathsf{x}}) = \mathsf{Verify}(\mathsf{ck}, \mathit{c}, \mathit{y}_{i_1}, \ldots, \mathit{y}_{i_k}, \pi_{\mathsf{y}}) = 1.$$

Construction: CRHF & Merkle trees

Natural approach to "compile" PCPs to SNARGs:

- \mathcal{P} commits to PCP string $c = \text{Com}(\vec{\pi})$.
- \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 $\mathcal{P} \rightarrow \mathcal{V}$ messages with PCPs.
- 3. $\ensuremath{\mathcal{V}}$ queries PCPs in each round.
- Public coin $\mathcal{V} \to$ non-interactive.
- Compiled using vector commitments as well!

[BCS16; Ben+18; Ame+17; Ben+19]

- 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| = poly(|w|)!
- To overcome:
 - 1. Uniformity assumption on circuits.
 - 2. 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.

[Gro10; Gen+13; Par+13; Dan+14; Bit+13; Gro16]

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.



- 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



- 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 $C(|C| \le n)$ 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*.

[Gro+18; Mal+19; DRZ20; Chi+20; Cam+20; RZ21]

Post-processing?

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

Let π_1, π_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 ⇒ NIZK with (aggregated) sublinear verification!
- Hot new topic!

[BGH19; COS20; Bün+20b; Bün+20a]

Some techniques for constructing SNARKs

- 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



$c_1 = \ell_1 \cdot r_1$	ℓ_1	$= x_1 + x_2$	$r_1=w_1+0$
$c_2 = \ell_2 \cdot r_2$	ℓ_2	$= x_1 + x_2$	$r_2=c_1+0$
$c_2 = l_2 \cdot r_2$	l2	$= 0 + c_{2}$	$p = \alpha + w_{2}$

$c_1 = (x_1 + x_2)w_1$	$\vec{c} = \vec{\ell} \circ \vec{r}$	$\vec{c} = \vec{\ell} \circ \vec{r}$		
- (/ -	(\vec{x})	(\vec{x})		
$c_2 = (x_1 + x_2)c_1$	$\vec{l} = \mathbf{L} \cdot \left(\vec{w} \right) \vec{r} =$	R · ₩		
$c_3 = c_2(c_1 + w_3)$	$\left(\vec{c}\right)$	$\langle \vec{c} \rangle$		

More generally:

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

or equivalently

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

Lemma

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

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

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

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

- $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e) \leftarrow \mathsf{GroupGen}(1^{\lambda}).$
- Fix $\mathcal{P}_1, \mathcal{P}_2$ generators of $\mathbb{G}_1, \mathbb{G}_2$, $e(\mathcal{P}_1, \mathcal{P}_2)$ generator of \mathbb{G}_T .
- Implicit notation.
- *e* is a bilinear operation $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$

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

Checking Diffie-Hellman relations:

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

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

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

 $T_1 = e(A, B) = e([a]_1, [b]_2) = e([1]_1, [b]_2)^a = e([1]_1, [1]_2)^{ab}$ $T_2 = e(C, [1]_2) = e([c]_1, [1]_2) = e([1]_1, [1]_2)^c$ $T_1 = T_2 \text{ iff } ab = c!$
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

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

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

- Monomial basis: $\{1, X, \dots, X^{n-1}\}$
- Lagrange basis w.r.t. set S of size n: {λ_{s1}(X), λ_{s2}(X),..., λ_{sn}(X)}

The main property of Lagrange basis is it is orthogonal:

$$orall s_i, s_j \in S: \ \lambda_{s_i}(s_j) = egin{cases} 1, & ext{if } i=j \ 0, & ext{otherwise} \end{cases}$$

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 $\vec{\lambda}(X) = (\lambda_1(X), \dots, \lambda_n(X))^\top$.
- 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
$$s_i \in S$$
: $p(s_i) = a(s_i)b(s_i) - c(s_i) = a_ib_i - c_i = 0$.

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

 (\Leftarrow)

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

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

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

- Note: $[\lambda_i(s)]$ computable by $[1], \ldots, [s]^{n-1}$.
- *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) 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.
- $\ensuremath{\mathcal{V}}$ checks an encoding is valid by testing

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

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

n-power Knowledge of Exponent:

For all PPT \mathcal{A} with input $([1]_1, [s]_1, \ldots, [s^n]_1, [\alpha]_1, [\alpha s]_1, \ldots, [\alpha s^n]_1, [\alpha]_2)$ 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 $\ensuremath{\mathcal{P}}$ give uniform elements that make the proof verify
- Include in srs $[T]_1 = [t(s)]_1$ as well.
- \mathcal{P} computes

 $[A]_{1} = [\lambda(s)]_{1}^{\top} \vec{a} + r_{a}[T]_{1}, \quad [B]_{2} = [\lambda(s)]_{2}^{\top} \vec{b} + r_{b}[T]_{2},$ $[C]_{1} = [\lambda(s)]_{1}^{\top} \vec{c} + r_{c}[T]_{1},$ $h'(X) \text{ s.t. } (a(X) + r_{a}t(X))(b(X) + r_{b}t(X)) = c(X) + r_{c}t(X) + h'(X)t(X)$ $[H]_{1} = [h'(s)]_{1}$

Verifier checks

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

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

- 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:

$$\sum Y^{i-1}(a_i - \vec{w}_i \vec{c}) = 0$$
 (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 $\vec{y} = (1, y, \dots, y^{n-1})^{\top}$, the check can be thought as

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

- 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}, \mathbf{z} .

Security:

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

Consider the non-hiding version ($\rho = 0$) $\mathcal{L}_{\vec{r},\vec{s}} = \left\{ [c], [d], z \mid \exists \vec{a}, \vec{b} \text{ s.t. } [c] = [\vec{r}]^\top \vec{a} \land [d] = [\vec{s}]^\top \vec{b} \land z = \vec{a}^\top \vec{b} \right\}$

- 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!

- 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

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

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

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

$$\vec{r}^{\top} \vec{a'} = (\vec{r}_L + x\vec{r}_R)^{\top} (\vec{a}_L + \vec{a}_R x^{-1})$$

= $\vec{r}_L^{\top} \vec{a}_L + \vec{r}_L^{\top} \vec{a}_R x^{-1} + \vec{r}_R^{\top} \vec{a}_L x + \vec{r}_R^{\top} \vec{a}_R$
= $c_{-1} x^{-1} + c + c_1 x = c'$

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

$$\vec{a}^{\prime \top} \vec{b}^{\prime} = (\vec{a}_L + \vec{a}_R x^{-1})^{\top} (\vec{b}_L + \vec{b}_R x)$$
$$= \vec{a}_L^{\top} \vec{b}_L + \vec{a}_L^{\top} \vec{b}_R x + \vec{a}_R^{\top} \vec{b}_L x^{-1} + \vec{a}_R^{\top} \vec{b}_R)$$
$$= \mathbf{z}_{-1} \mathbf{x} + \mathbf{z} + \mathbf{z}_1 \mathbf{x}^{-1} = \mathbf{z}^{\prime}$$

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 $\ensuremath{\mathcal{P}}$ sends witness, base case trivially holds.
- Basic tool: Rewind the prover.
- Get enough transcripts to extract *i*-th level witness.

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)!

- Transparent!
- log n rounds.
- *O*_λ(log *n*) communication: 6 elements per round
- *O*_λ(*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!

Used in many works: [Bün+18], [Wah+18], ...

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 = $[\vec{r}]_1$
- *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 $\vec{y} = (1, y, \dots y^{n-1})^{\top}$.
- \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

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

- Many ways to achieve it.
- Should use hiding commitments.
- Can randomize \mathcal{P} messages.
- We demonstrate a technique using Σ protocols [AC20].

What about Zero knowledge?

Simple HVZK protocol for

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

- \mathcal{V} has $[\vec{r}], [\rho], [\gamma], \vec{a}, k. \mathcal{P}$ has witness \vec{c}, \vec{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

 $\vec{r}^{\top}\vec{z} + \rho\phi = [\delta] + x[\gamma]: \vec{z} \text{ is a commitment to } [\delta] + c[\gamma]$ $\vec{a}^{\top}\vec{z} = k + x\ell := \vec{a}^{\top}(c + x\vec{d})$

- 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

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

- Equivalently,
 - 1. (\vec{z}^{\top}, ϕ) 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: *P* cannot produce (1) commitment *c*, (2) two different openings y₁ ≠ y₂ 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).

- Celebrated result! (IP=PSPACE)
- Let p ∈ 𝔅[X₁,...,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

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

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

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

- 1

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

•
$$\mathcal{V}$$
 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 p₁(0) + p₁(1) ≠ k the prover has to send:

 $p_1^*(X)
eq p_1(X)$ such that $p_1^*(0) + p_1^*(1) = k$

- 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 *P* cannot cheat! *V* computes last evaluation on its own!
- Soundness error $\leq \frac{n \cdot d}{\mathbb{F}}$

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

Both explicitly used for SNARGS (e.g. [Wah+18; Xie+19]) and inspiration for other techniques.

- 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.

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

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

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

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

$$p_{x}n-\sigma=xr_{x}+t(x)h_{x}$$



References

[AC20]

- Thomas Attema and Ronald Cramer. "Compressed Σ-Protocol Theory and Practical Application to Plug & Play Secure Algorithmics". In: *CRYPTO 2020, Part III.* Ed. by Daniele Micciancio and Thomas Ristenpart.
 Vol. 12172. LNCS. Springer, Heidelberg, Aug. 2020, pp. 513–543. DOI: 10.1007/978–3–030–56877–1_18.
- [Ame+17] Scott Ames et al. "Ligero: Lightweight Sublinear Arguments Without a Trusted Setup". In: ACM CCS 2017. Ed. by Bhavani M. Thuraisingham et al. ACM Press, Oct. 2017, pp. 2087–2104. DOI: 10.1145/3133956.3134104.

Bibliography ii

- [BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. "Interactive Oracle Proofs". In: TCC 2016-B, Part II. Ed. by Martin Hirt and Adam D. Smith. Vol. 9986. LNCS. Springer, Heidelberg, Oct. 2016, pp. 31–60. DOI: 10.1007/978-3-662-53644-5_2.
- [Ben+18] Eli Ben-Sasson et al. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046. https://eprint.iacr.org/2018/046. 2018.
- [Ben+19] Eli Ben-Sasson et al. "Aurora: Transparent Succinct Arguments for R1CS". In: EUROCRYPT 2019, Part I. Ed. by Yuval Ishai and Vincent Rijmen. Vol. 11476. LNCS. Springer, Heidelberg, May 2019, pp. 103–128. DOI: 10.1007/978-3-030-17653-2_4.

Bibliography iii

- [BGH19] Sean Bowe, Jack Grigg, and Daira Hopwood. Halo: Recursive Proof Composition without a Trusted Setup. Cryptology ePrint Archive, Report 2019/1021. https://eprint.iacr.org/2019/1021. 2019.
- [Bit+13] Nir Bitansky et al. "Succinct Non-interactive Arguments via Linear Interactive Proofs". In: TCC 2013. Ed. by Amit Sahai. Vol. 7785. LNCS. Springer, Heidelberg, Mar. 2013, pp. 315–333. DOI: 10.1007/978-3-642-36594-2_18.
- [Boo+16] Jonathan Bootle et al. "Efficient Zero-Knowledge Arguments for Arithmetic Circuits in the Discrete Log Setting". In: EUROCRYPT 2016, Part II. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9666. LNCS. Springer, Heidelberg, May 2016, pp. 327–357. DOI: 10.1007/978-3-662-49896-5_12.
Bibliography iv

- [Bün+18] Benedikt Bünz et al. "Bulletproofs: Short Proofs for Confidential Transactions and More". In: 2018 IEEE Symposium on Security and Privacy. IEEE Computer Society Press, May 2018, pp. 315–334. DOI: 10.1109/SP.2018.00020.
- [Bün+20a] Benedikt Bünz et al. Proof-Carrying Data without Succinct Arguments. Cryptology ePrint Archive, Report 2020/1618. https://eprint.iacr.org/2020/1618. 2020.
- [Bün+20b] Benedikt Bünz et al. "Recursive Proof Composition from Accumulation Schemes". In: TCC 2020, Part II. Ed. by Rafael Pass and Krzysztof Pietrzak. Vol. 12551. LNCS. Springer, Heidelberg, Nov. 2020, pp. 1–18. DOI: 10.1007/978-3-030-64378-2_1.

[Cam+20] Matteo Campanelli et al. Lunar: a Toolbox for More Efficient Universal and Updatable zkSNARKs and Commit-and-Prove Extensions. Cryptology ePrint Archive, Report 2020/1069. https://eprint.iacr.org/2020/1069. 2020.

[Chi+20] Alessandro Chiesa et al. "Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS". In: EUROCRYPT 2020, Part I. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12105. LNCS. Springer, Heidelberg, May 2020, pp. 738–768. DOI: 10.1007/978–3–030–45721–1_26.

Bibliography vi

[COS20] Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. "Fractal: Post-quantum and Transparent Recursive Proofs from Holography". In: EUROCRYPT 2020, Part I. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12105. LNCS. Springer, Heidelberg, May 2020, pp. 769–793. DOI: 10.1007/978-3-030-45721-1_27.

[Dan+14] George Danezis et al. "Square Span Programs with Applications to Succinct NIZK Arguments". In: ASIACRYPT 2014, Part I. Ed. by Palash Sarkar and Tetsu Iwata. Vol. 8873. LNCS. Springer, Heidelberg, Dec. 2014, pp. 532–550. DOI: 10. 1007/070. 2, 662, 45611, 2, 28

10.1007/978-3-662-45611-8_28.

Bibliography vii

[DRZ20]

Vanesa Daza, Carla Ràfols, and Alexandros Zacharakis.
"Updateable Inner Product Argument with Logarithmic Verifier and Applications". In: *PKC 2020, Part I.* Ed. by Aggelos Kiayias et al. Vol. 12110. LNCS. Springer, Heidelberg, May 2020, pp. 527–557. DOI: 10.1007/978–3–030–45374–9_18.

 [Gen+13] Rosario Gennaro et al. "Quadratic Span Programs and Succinct NIZKs without PCPs". In: EUROCRYPT 2013.
 Ed. by Thomas Johansson and Phong Q. Nguyen.
 Vol. 7881. LNCS. Springer, Heidelberg, May 2013, pp. 626–645. DOI: 10.1007/978-3-642-38348-9_37.

Bibliography viii

- [Gro+18] Jens Groth et al. Updatable and Universal Common Reference Strings with Applications to zk-SNARKs. Cryptology ePrint Archive, Report 2018/280. https://eprint.iacr.org/2018/280. 2018.
- [Gro10] Jens Groth. "Short Pairing-Based Non-interactive Zero-Knowledge Arguments". In: ASIACRYPT 2010. Ed. by Masayuki Abe. Vol. 6477. LNCS. Springer, Heidelberg, Dec. 2010, pp. 321–340. DOI: 10.1007/978-3-642-17373-8 19.
- [Gro16] Jens Groth. "On the Size of Pairing-Based Non-interactive Arguments". In: EUROCRYPT 2016, Part II. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9666. LNCS. Springer, Heidelberg, May 2016, pp. 305–326. DOI: 10.1007/978-3-662-49896-5_11.

Bibliography ix

- [GW11] Craig Gentry and Daniel Wichs. "Separating succinct non-interactive arguments from all falsifiable assumptions".
 In: 43rd ACM STOC. Ed. by Lance Fortnow and Salil P. Vadhan. ACM Press, June 2011, pp. 99–108. DOI: 10.1145/1993636.1993651.
- [Mal+19] Mary Maller et al. "Sonic: Zero-Knowledge SNARKs from Linear-Size Universal and Updatable Structured Reference Strings". In: ACM CCS 2019. Ed. by Lorenzo Cavallaro et al. ACM Press, Nov. 2019, pp. 2111–2128. DOI: 10.1145/3319535.3339817.
- [Par+13] Bryan Parno et al. "Pinocchio: Nearly Practical Verifiable Computation". In: 2013 IEEE Symposium on Security and Privacy. IEEE Computer Society Press, May 2013, pp. 238–252. DOI: 10.1109/SP.2013.47.

Bibliography x

- [RZ21] Carla Ràfols and Arantxa Zapico. An Algebraic Framework for Universal and Updatable SNARKs. Cryptology ePrint Archive, Report 2021/590. https://eprint.iacr.org/2021/590. 2021.
- [Wah+18] Riad S. Wahby et al. "Doubly-Efficient zkSNARKs Without Trusted Setup". In: 2018 IEEE Symposium on Security and Privacy. IEEE Computer Society Press, May 2018, pp. 926–943. DOI: 10.1109/SP.2018.00060.
- [Xie+19] Tiancheng Xie et al. "Libra: Succinct Zero-Knowledge Proofs with Optimal Prover Computation". In: *CRYPTO 2019, Part III*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Vol. 11694. LNCS. Springer, Heidelberg, Aug. 2019, pp. 733–764. DOI: 10.1007/978–3–030–26954–8 24.