# Lecture 7: The sumcheck and GKR protocols

Zero-knowledge proofs

263-4665-00L

Lecturer: Jonathan Bootle

# Last time

Sigma protocols from DLOG

- Intro level: Schnorr and homomorphisms
- Medium level: multiplicative relations
- Advanced level: low-degree circuit proofs

New topic: ZK arguments with short proofs

- The sumcheck protocol

# Plan for the next few lectures

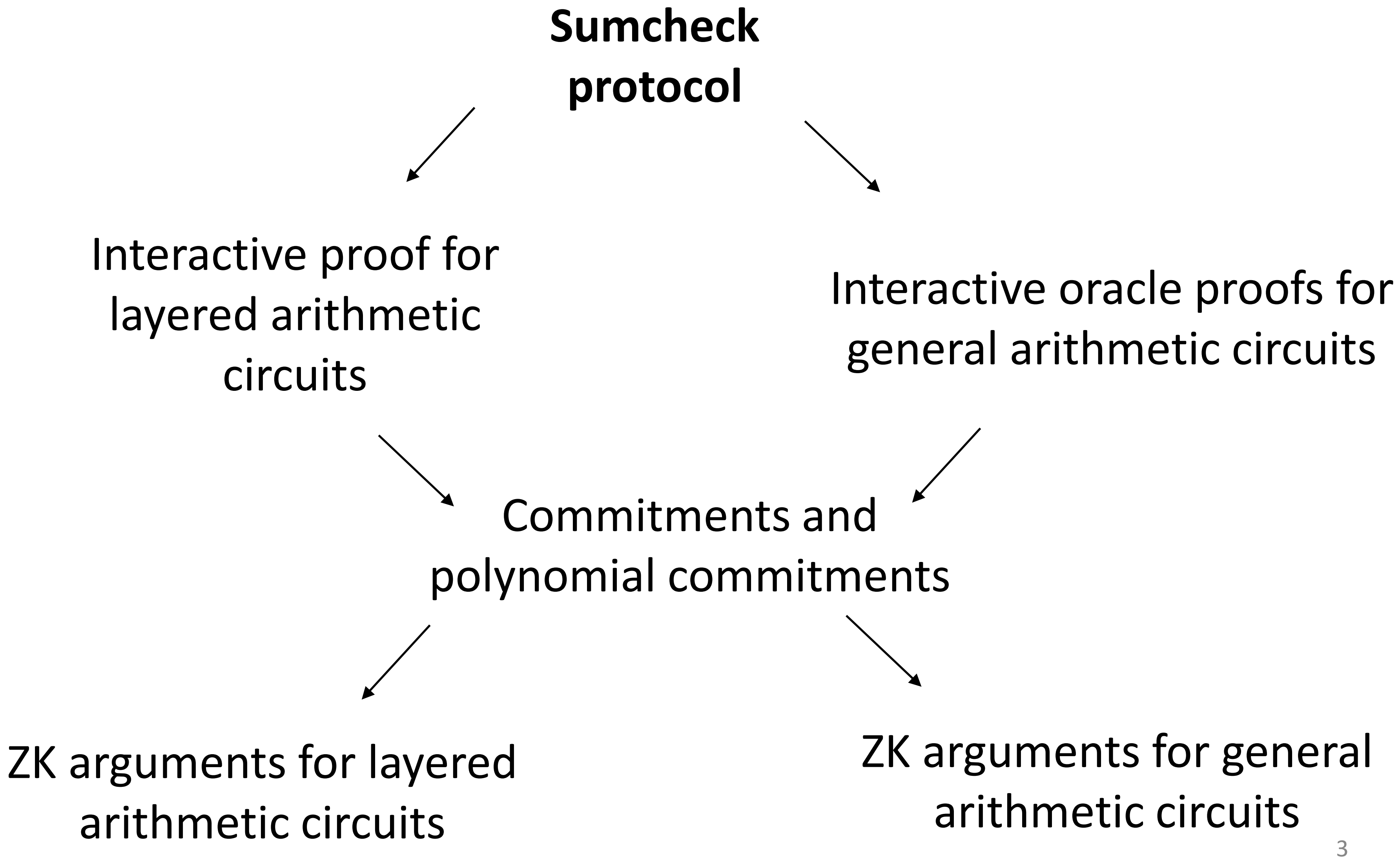
Information theoretic proof systems

+

Cryptography

=

ZK arguments with short proofs



**Sumcheck protocol**

Interactive proof for layered arithmetic circuits

Interactive oracle proofs for general arithmetic circuits

Commitments and polynomial commitments

ZK arguments for layered arithmetic circuits

ZK arguments for general arithmetic circuits

# The sumcheck protocol [LFKN92] (recursively)

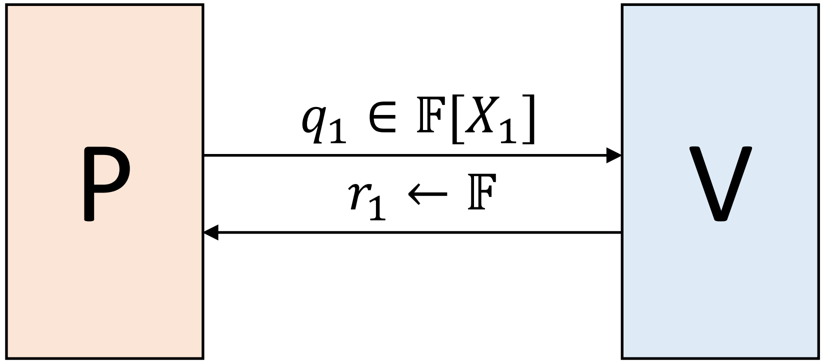
**Instance:**

- polynomial $p(X_1, \dots, X_\ell)$ over field $\mathbb{F}$
- value $u \in \mathbb{F}$, subset $H \subset \mathbb{F}$

**Language:**

- satisfies $\sum_{\vec{\omega} \in H^\ell} p(\omega_1, \dots, \omega_\ell) = u$

Computes polynomial
$q_1(X_1) := \sum_{\vec{\omega} \in H^{\ell-1}} p(X_1, \omega_2, .., \omega_\ell)$

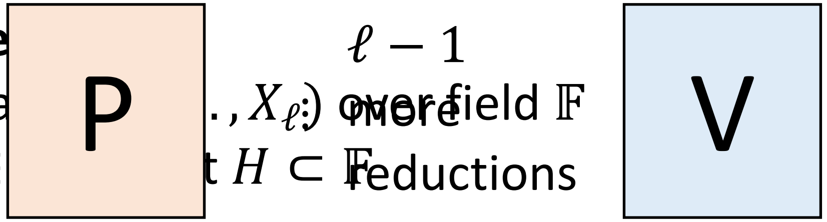


Checks $\sum_{\omega_1 \in H} q_1(\omega_1) = u$

$$p'(X_2, \dots, X_\ell) := p(r_1, X_2, \dots, X_\ell)\,, \quad u' := q_1(r_1)$$

**New instance**

- polynomia ., $X_\ell$) over field $\mathbb{F}$
- value $u' \in$ t $H \subset \mathbb{F}$



$\ell - 1$
more
reductions

**Language:**

- satisfies $\sum_{\vec{\omega} \in H^{\ell-1}} p'(\omega_2, \dots, \omega_\ell) = u'$

$$p^{(\ell)} := p(r_1, r_2, \dots, r_\ell)\,, \quad u^{(\ell)} := q_\ell(r_\ell)$$

**Language:** $p^{(\ell)} = u^{(\ell)}$
$p(r_1, r_2, \dots, r_\ell) = u^{(\ell)}$

**Final instance:** $p^{(\ell)}, u^{(\ell)} \in \mathbb{F}$

# The sumcheck protocol [LFKN92] (unrolled)

**Instance:**

- polynomial $p(X_1, \dots, X_\ell)$ over field $\mathbb{F}$
- value $u \in \mathbb{F}$, subset $H \subset \mathbb{F}$

**Language:**

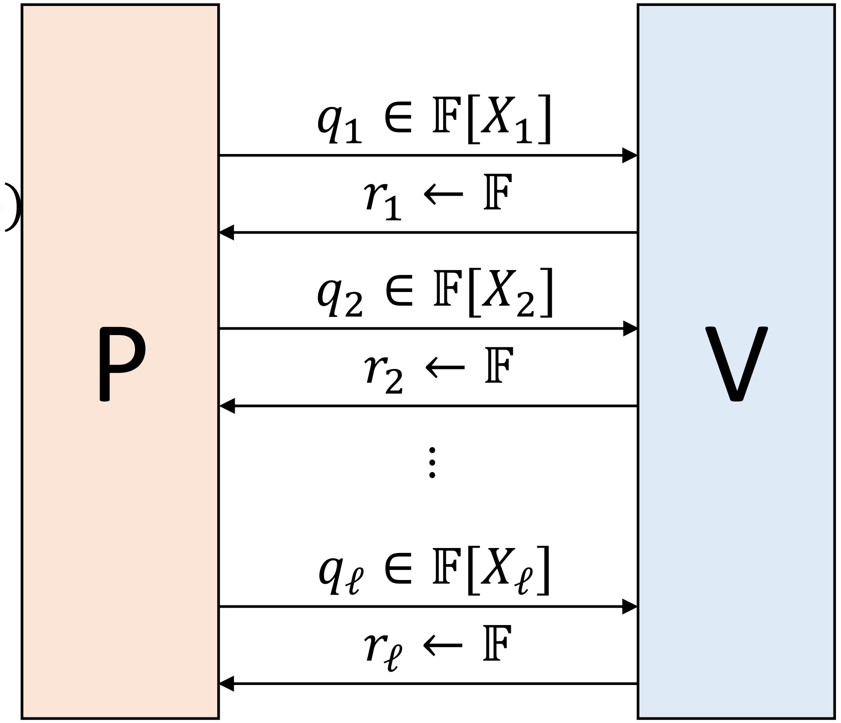
- satisfies $\sum_{\vec{\omega} \in H^\ell} p(\omega_1, \dots, \omega_\ell) = u$

Computes polynomials $q_i(X_i) := \sum_{\vec{\omega} \in H^{\ell-i}} p(r_1, .., r_{i-1}, X_i, \omega_{i+1}, .., \omega_\ell)$

Send degree $d = \max_i \deg_{X_i} p$ polynomial each round $\{q_i(\omega)\}_{\omega \in H}$

Total communication $O(\ell \cdot d)$

Original instance size: $p$ has $(d+1)^\ell$ coefficients



P

V

$q_1 \in \mathbb{F}[X_1]$

$r_1 \leftarrow \mathbb{F}$

$q_2 \in \mathbb{F}[X_2]$

$r_2 \leftarrow \mathbb{F}$

⋮

$q_\ell \in \mathbb{F}[X_\ell]$

$r_\ell \leftarrow \mathbb{F}$

Checks cost $O(\ell \cdot d)$ $\mathbb{F}$-ops

Checks $\sum_{\omega_1 \in H} q_1(\omega_1) = u$

Checks $\sum_{\omega_2 \in H} q_2(\omega_2) = q_1(r_1)$

⋮

Checks $\sum_{\omega_\ell \in H} q_\ell(\omega_\ell) = q_{\ell-1}(r_{\ell-1})$

**Instance:**

- polynomial $p(X_1, \dots, X_\ell)$ over field $\mathbb{F}$
- values $r_1, \dots, r_\ell \in \mathbb{F}$

**Language:**

- satisfies $p(r_1, \dots, r_\ell) = u^{(\ell)}$

Evaluate $p$ once, not $|H|^\ell$ times

# Prover complexity ($d + 1 \leq |H|$)

$q_1(X_1) := \sum_{\vec{\omega} \in H^{\ell-1}} p(X_1, \omega_2, .., \omega_\ell)$

- To compute $q_1(X_1) = \sum_{\omega_2,\dots,\omega_\ell \in H} p(X_1, \omega_2, \dots, \omega_\ell)$:
- Get table of $|H|^\ell$ evaluations $\{p(\vec{\omega})\}_{\vec{\omega} \in H^\ell}$. $|H|^\ell$ $p$-evaluations $O(d|H|^{\ell-1})$ ops
- For each $\vec{\omega} \in H^{\ell-1}$, interpolate $\{p(\omega_1, \vec{\omega})\}_{\omega_1 \in H}$ in $X_1$ to get $p(X_1, \vec{\omega})$.
- For each $\vec{\omega} \in H^{\ell-1}$, evaluate $p(X_1, \vec{\omega})$ at $r_1$ to get $\{p(r_1, \vec{\omega})\}_{\vec{\omega} \in H^{\ell-1}}$.
- Recurse for $O(|H|^\ell)$ ops and $p$-evaluations in total. $O(d|H|^{\ell-1})$ ops

- Better algorithms if e.g. $|H|, p$ have special structure.
- Sometimes the coefficient view of $p$ is better.

# Completeness of the sumcheck protocol

Instance:

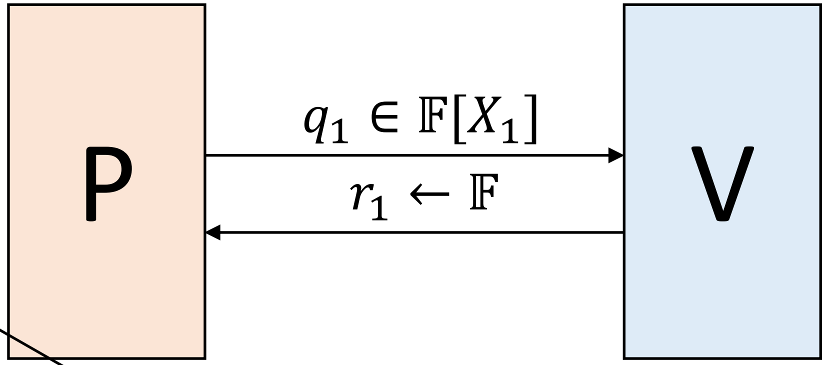
- polynomial $p(X_1, \ldots, X_\ell)$ over field $\mathbb{F}$
- value $u \in \mathbb{F}$, subset $H \subset \mathbb{F}$

Language:

- satisfies $\sum_{\vec{\omega} \in H^\ell} p(\omega_1, \ldots, \omega_\ell) = u$

Computes polynomial

$q_1(X_1) := \sum_{\vec{\omega} \in H^{\ell-1}} p(X_1, \omega_2, .., \omega_\ell)$



Checks $\sum_{\omega_1 \in H} q_1(\omega_1) = u$

By induction (full protocol):

- Each check is satisfied.
- Final instance $\left(\mathbb{F}, H, p^{(\ell)}, u^{(\ell)}\right) \in \mathcal{L}_{SC}$.

Evaluate at $r_1$

$$\boxed{p'(X_2, \ldots, X_\ell) := p(r_1, X_2, \ldots, X_\ell), \quad u' := q_1(r_1)}$$

$\sum_{\omega_1 \in H}$

$u' = q_1(r_1) = \sum_{\omega_2, \ldots, \omega_\ell \in H} p(r_1, \omega_2, \ldots, \omega_\ell)$

$= \sum_{\omega_2, \ldots, \omega_\ell \in H} p'(\omega_2, \ldots, \omega_\ell)$.

Therefore, $(\mathbb{F}, H, p', u') \in \mathcal{L}_{SC}$.

$\sum_{\omega_1 \in H} q(\omega_1) = \sum_{\omega_1, \ldots, \omega_\ell \in H} p(\omega_1, \ldots, \omega_\ell) = u$.

$V$'s first check passes.

True claims reduce to true claims and checks pass

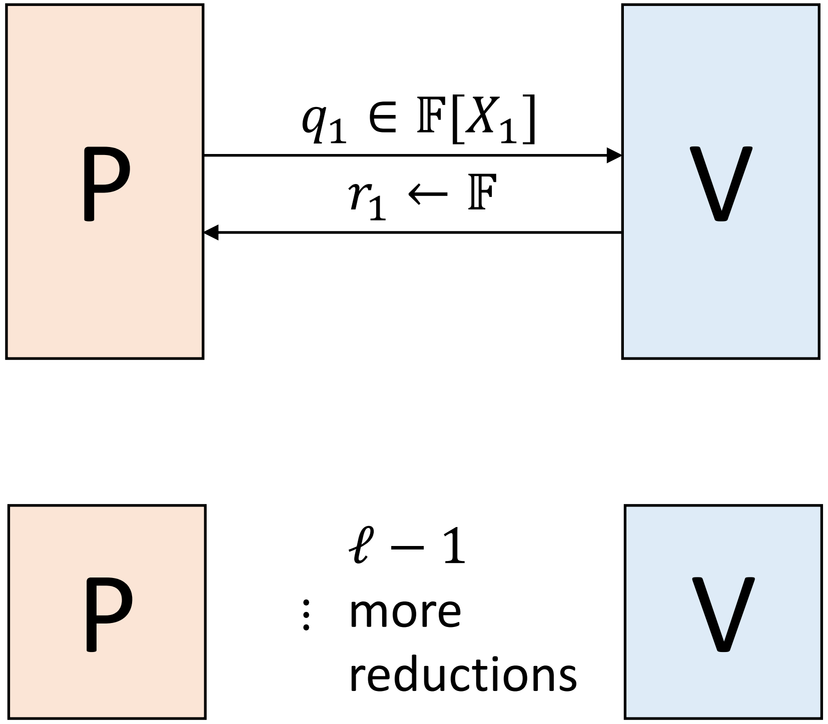
# Soundness analysis strategy

**Instance:**

- polynomial $p(X_1, \dots, X_\ell)$ over field $\mathbb{F}$
- value $u \in \mathbb{F}$, subset $H \subset \mathbb{F}$

**Not in language:**

- $\sum_{\vec{\omega} \in H^\ell} p(\omega_1, \dots, \omega_\ell) \neq u$



P

$q_1 \in \mathbb{F}[X_1]$

$r_1 \leftarrow \mathbb{F}$

V

P

$\vdots$ $\ell - 1$ more reductions

V

$\Pr[\text{False}\rightarrow\text{True} \wedge \text{checks pass}] \leq d/|\mathbb{F}|$

**Language:**

- satisfies $\sum_{\vec{\omega} \in H^{\ell-1}} p'(\omega_2, \dots, \omega_\ell) = u'$

Final instance is false w.h.p.
Total soundness error $\ell \cdot d/\ |\mathbb{F}|$

False claims reduce to false claims, or checks fail, w.h.p.

Round by round soundness

Make non-interactive via Fiat-Shamir with ~~RO~~ LWE!
[CCHLRR18], [PS19]

# Soundness analysis (one round)

**Instance:**

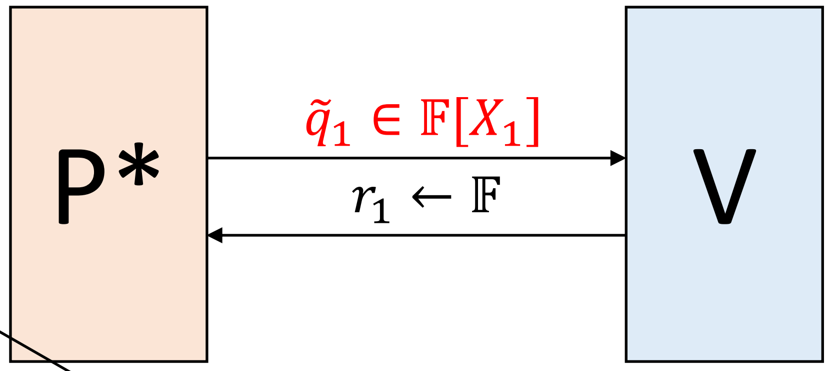
- polynomial $p(X_1, \dots, X_\ell)$ over field $\mathbb{F}$
- value $u \in \mathbb{F}$, subset $H \subset \mathbb{F}$

**Not in language:**

- $\sum_{\vec{\omega} \in H^\ell} p(\omega_1, \dots, \omega_\ell) \neq u$

May not send 'honest' polynomial

$q_1(X_1) := \sum_{\vec{\omega} \in H^{\ell-1}} p(X_1, \omega_2, .., \omega_\ell)$



P*

$\tilde{q}_1 \in \mathbb{F}[X_1]$

$r_1 \leftarrow \mathbb{F}$

V

Checks $\sum_{\omega_1 \in H} q_1(\omega_1) = u$

Evaluate at $r_1$

$p'(X_2, \dots, X_\ell) := p(r_1, X_2, \dots, X_\ell)$ , $u' := \tilde{q}_1(r_1)$

Roots property:
Degree $d$ polynomials
have $\leq d$ roots

$\sum_{\omega_1 \in H}$

If $P^*$ sends bad $\tilde{q}_1(X_1) \neq q_1(X_1)$:

$u' := \tilde{q}_1(r_1) \neq q_1(r_1)$ except w.p. $\leq d / |\mathbb{F}|$

$q_1(r_1) = \sum_{\omega_2, \dots, \omega_\ell \in H} p(r_1, \omega_2, \dots, \omega_\ell)$

$= \sum_{\omega_2, \dots, \omega_\ell \in H} p'(\omega_2, \dots, \omega_\ell)$.

Therefore, $(\mathbb{F}, H, p', u') \notin \mathcal{L}_{SC}$ w.h.p.

If $P^*$ sends correct $\tilde{q}_1(X_1) = q_1(X_1)$:

$\sum_{\omega_1 \in H} q_1(\omega_1) = \sum_{\vec{\omega} \in H^\ell} p(\omega_1, \omega_2, .., \omega_\ell) \neq u$.

$V$'s first check fails.

Can improve analysis to $1-(1-d/|\mathbb{F}|)^\ell$

# Full soundness analysis by induction

- **Base case $\ell=0$:** $V$ immediately rejects false instance.
- **Induction hypothesis:**

$\forall(\mathbb{F},H,p',u')\notin\mathcal{L}_{SC}$, $\ell-1$ variables: $\Pr[V\ accepts]\leq(\ell-1)d/|\mathbb{F}|$

- **Induction step:**
- Let $BAD=Event\left(\sum_{\vec{\omega}\in H^{\ell-1}}p'(\vec{\omega})=v'\wedge\sum_{\omega_1\in H}q_1(\omega_1)=v\right)$.
- The previous slide shows $\Pr[BAD]\leq d/|\mathbb{F}|$.
- Generally,

By induction hypothesis

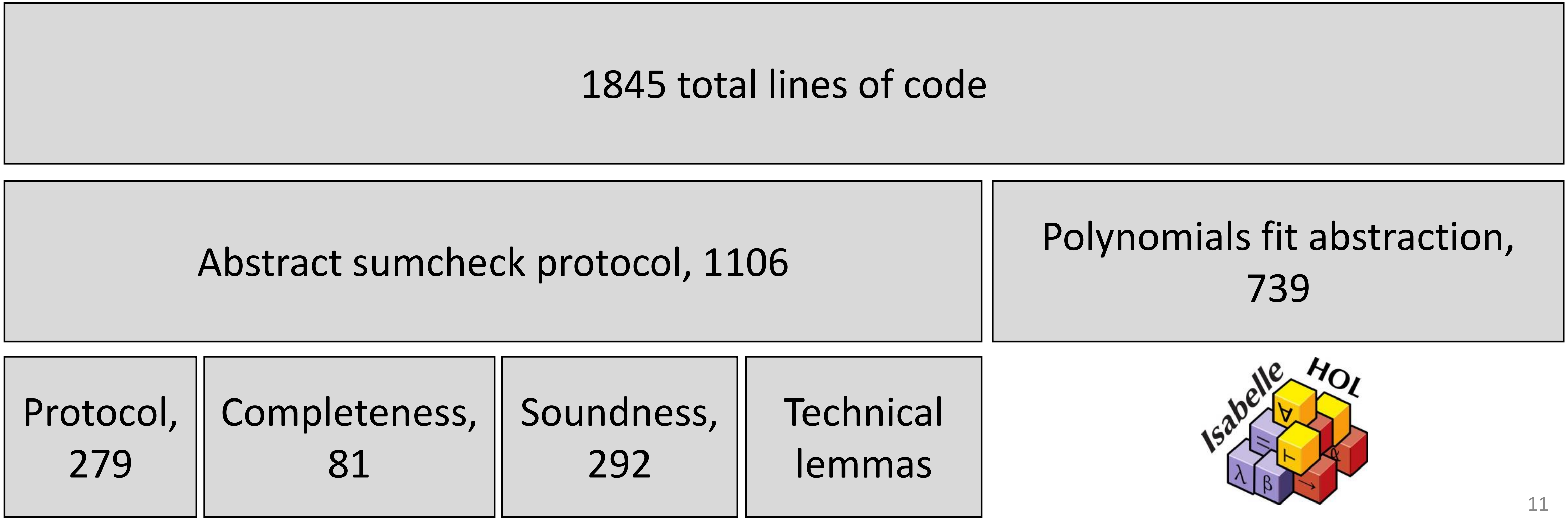
$$\Pr[V\ accepts]=\underbrace{\Pr[V\ accepts|BAD]}_{1}\underbrace{\Pr[BAD]}_{d/|\mathbb{F}|}+\underbrace{\Pr[V\ accepts|\neg BAD]}_{(\ell-1)d/|\mathbb{F}|}\underbrace{\Pr[\neg BAD]}_{1}$$

$\leq$

$$\leq\ell d/|\mathbb{F}|$$

# Formal verification

- Classic sumcheck machine formalised in Isabelle (ETHZ MSc Project)
- https://www.research-collection.ethz.ch/handle/20.500.11850/611002



1845 total lines of code

Abstract sumcheck protocol, 1106

Polynomials fit abstraction, 739

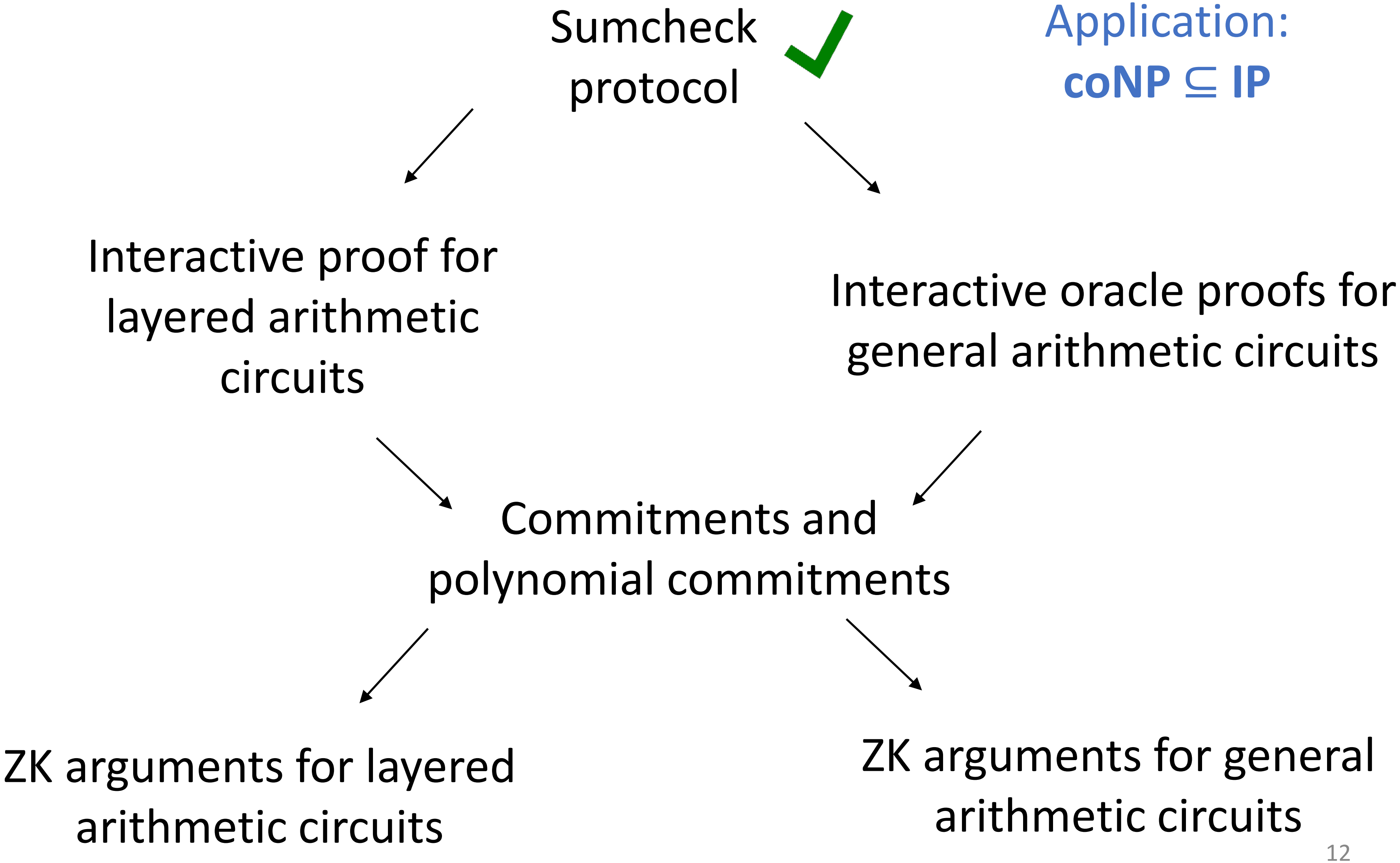
Protocol, 279

Completeness, 81

Soundness, 292

Technical lemmas

# Plan for the next few lectures



Sumcheck protocol ✓

Application: **coNP ⊆ IP**

Interactive proof for layered arithmetic circuits

Interactive oracle proofs for general arithmetic circuits

Commitments and polynomial commitments

ZK arguments for layered arithmetic circuits

ZK arguments for general arithmetic circuits

# The complexity class **coNP**

**Definition:**

**coNP** is the set of languages $\mathcal{L}$ for which

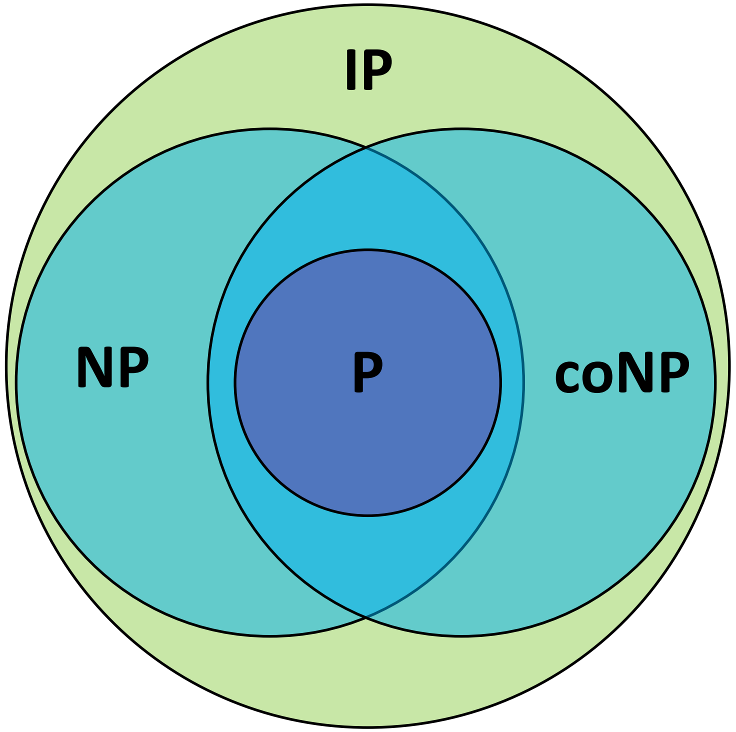
$\exists$ language $\mathcal{R}_{\bar{\mathcal{L}}}$ and polynomial $q$ such that

- $\forall x \notin \mathcal{L},\ \exists w$ with $|w| \leq q(|x|)$ and $(x, w) \in \mathcal{R}_{\bar{\mathcal{L}}}$;
- $\forall x \in \mathcal{L},\ \nexists w$ with $(x, w) \in \mathcal{R}_{\bar{\mathcal{L}}}$; and
- $\mathcal{R}_{\bar{\mathcal{L}}} \in$ **P**.

We call $w$ the *witness* to $x$.

a *refutation* of $x$

We will show this:



Problems whose NO solutions are easy to check ⟷ Efficient checks that $(x, w) \in \mathcal{R}_{\bar{\mathcal{L}}}$

**Example:** GI ∈ **NP** so GNI ∈ **coNP**.

W.L.O.G. $C$ is 3CNF i.e. $C = C_1 \wedge \cdots \wedge C_m$

$\forall i \in [m], C_i = a_i \vee b_i \vee c_i$

$a_i, b_i, c_i \in \{x_j, \neg x_j\}_{j\in[\ell]}$

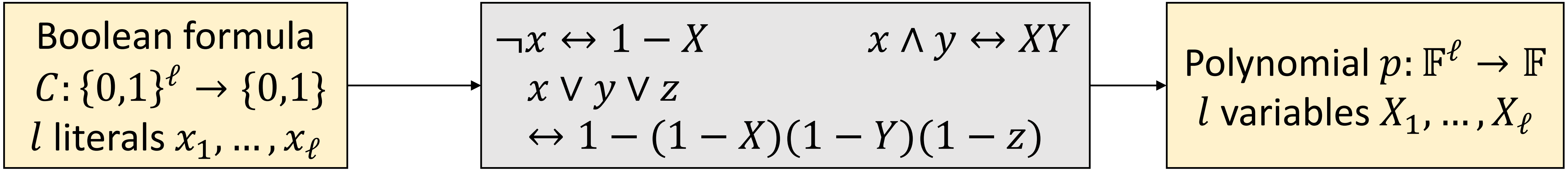
# UNSAT and arithmetisation

- Let $C : \{0,1\}^\ell \to \{0,1\}$ be a Boolean formula with literals $x_1, \dots, x_\ell$.
- $\mathcal{L}_{SAT} = \{C : \exists\, \vec{\omega} \in \{0,1\}^\ell, C(\vec{\omega}) = 1\}$. **NP**-complete
- $\mathcal{L}_{UNSAT} = \{C : \forall\, \vec{\omega} \in \{0,1\}^\ell, C(\vec{\omega}) \neq 1\}$. **coNP**-complete
- Goal: sumcheck-based IP for $\mathcal{L}_{UNSAT}$ to show **coNP** ⊆ **IP**.

3CNF with $m$ clauses

Values match on $\{0,1\}$ inputs.

$d \leq 3m$



Boolean formula

$C: \{0,1\}^\ell \to \{0,1\}$

$l$ literals $x_1, \dots, x_\ell$

$\neg x \leftrightarrow 1 - X$

$x \wedge y \leftrightarrow XY$

$x \vee y \vee z$

$\leftrightarrow 1 - (1 - X)(1 - Y)(1 - z)$

Polynomial $p: \mathbb{F}^\ell \to \mathbb{F}$

$l$ variables $X_1, \dots, X_\ell$

$p$ extends $C$ to $\mathbb{F}$

$\forall \vec{\omega} \in \{0,1\}^\ell, p(\vec{\omega}) = C(\vec{\omega})$

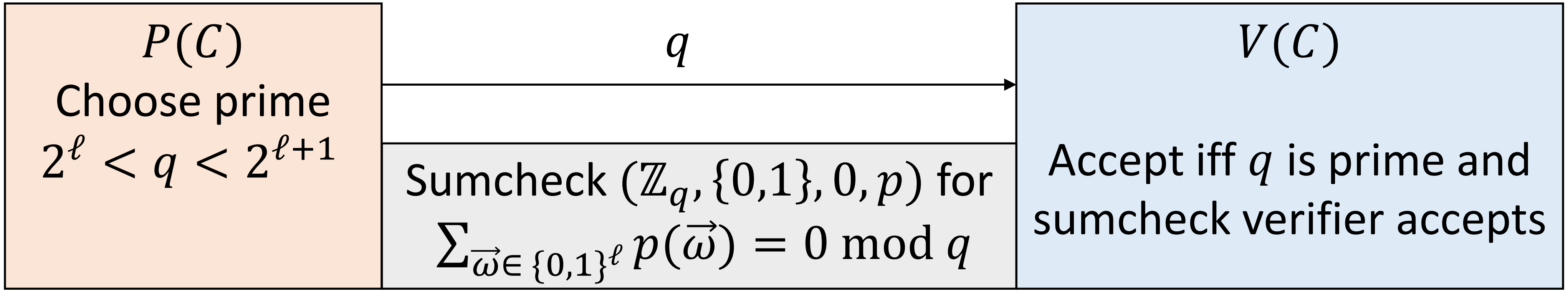
$p$ can be evaluated in $O(m)$ $\mathbb{F}$-ops

Need to evaluate outside $\{0,1\}$ for sumcheck

# IP for UNSAT

**Instance:** 3CNF $C$

AKS primality test
“PRIMES is in P”



Bertrand’s postulate

$P(C)$

Choose prime
$2^\ell < q < 2^{\ell+1}$

$q$

Sumcheck $(\mathbb{Z}_q, \{0,1\}, 0, p)$ for
$\sum_{\vec{\omega} \in \{0,1\}^\ell} p(\vec{\omega}) = 0 \bmod q$

$V(C)$

Accept iff $q$ is prime and sumcheck verifier accepts

**Completeness analysis:**

- $\forall \vec{\omega} \in \{0,1\}^\ell, C(\vec{\omega}) = 0.$
- $\sum_{\vec{\omega} \in \{0,1\}^\ell} C(\vec{\omega}) = 0.$
- $\sum_{\vec{\omega} \in \{0,1\}^\ell} C(\vec{\omega}) = 0 \bmod q.$
- $\sum_{\vec{\omega} \in \{0,1\}^\ell} p(\vec{\omega}) = 0 \bmod q.$

$p \equiv C$
on $\{0,1\}^\ell$

Sumcheck completeness
AKS primality test

$\Rightarrow V$ accepts

**Soundness analysis:**

Efficient by sumcheck,
AKS, $p$-eval efficiency

- $\exists \vec{\omega} \in \{0,1\}^\ell, C(\vec{\omega}) = 1.$
- $0 < \sum_{\vec{\omega} \in \{0,1\}^\ell} C(\vec{\omega}) \leq 2^\ell.$
- $\sum_{\vec{\omega} \in \{0,1\}^\ell} C(\vec{\omega}) \neq 0 \bmod q.$
- $\sum_{\vec{\omega} \in \{0,1\}^\ell} p(\vec{\omega}) \neq 0 \bmod q.$

$q > 2^\ell$

Sumcheck soundness
AKS primality test

$\Rightarrow \Pr[V \text{ accepts}] \leq \dfrac{3m\ell}{q}$

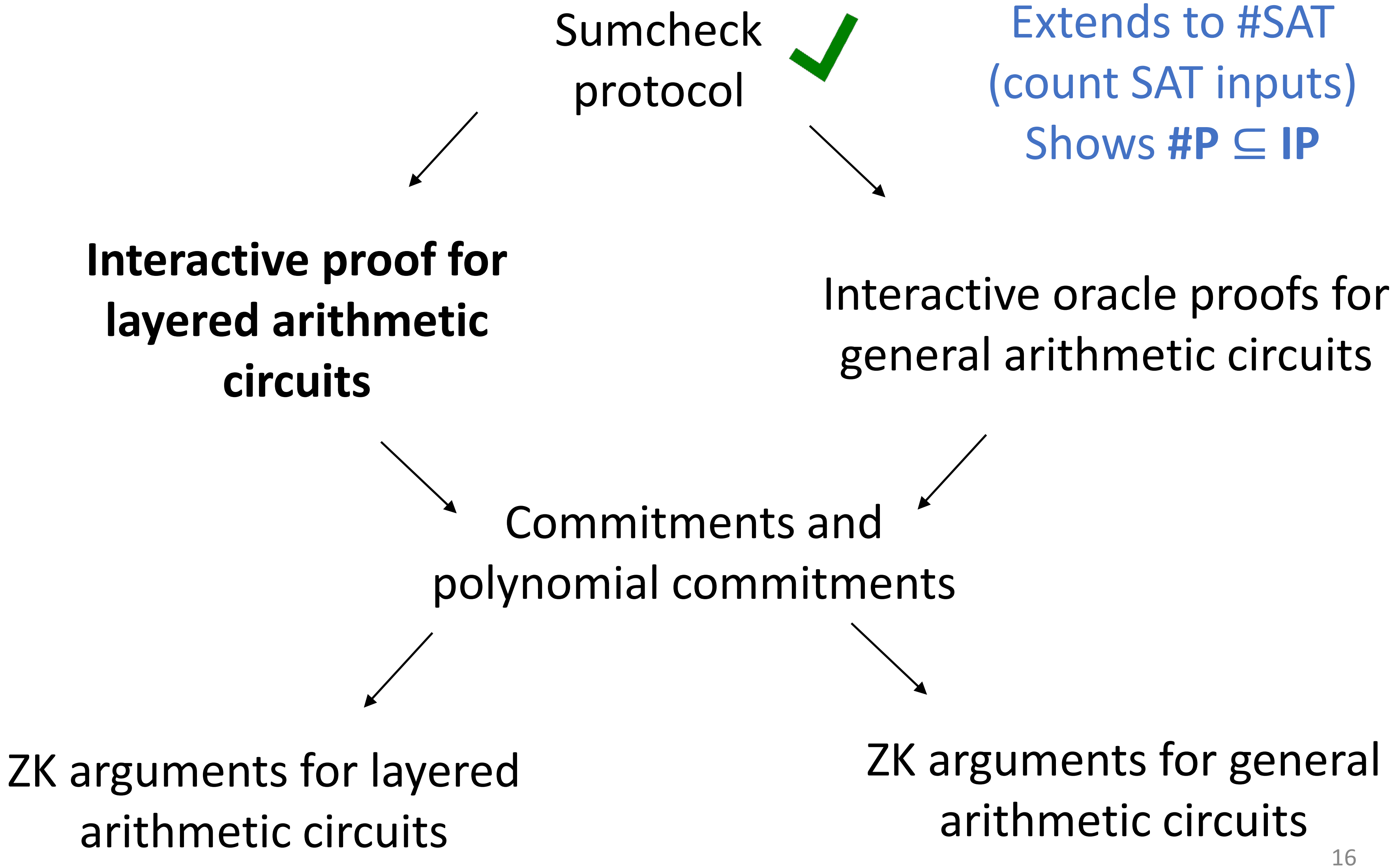
# Plan for the next few lectures

**coNP ⊆ IP** ✓

Extends to #SAT

(count SAT inputs)

Shows **#P ⊆ IP**



Sumcheck protocol ✓

**Interactive proof for layered arithmetic circuits**

Interactive oracle proofs for general arithmetic circuits

Commitments and polynomial commitments

ZK arguments for layered arithmetic circuits

ZK arguments for general arithmetic circuits

# Notes on succinct verification

- $V$ gets input $\mathbb{x}$.
- If $V$ uses all of $\mathbb{x}$, then $\mathrm{Time}_V(\mathbb{x}) \geq \mathrm{DescriptionSize}(\mathbb{x})$.
- Want $\mathrm{Time}_V(\mathbb{x}) \ll \mathrm{CheckingCost}_{\mathcal{R}}(\mathbb{x})$.
- Instance $\mathbb{x} =$ circuit $C$ with $N$ gates over $\mathbb{F}$.
- $\mathcal{R}_{SAT}$ costs $N$ $\mathbb{F}$-ops to check by evaluating $C$.
- Is $\mathrm{Time}_V(\mathbb{x}) \ll \mathrm{CheckingCost}_{\mathcal{R}}(\mathbb{x})$ possible?

$\mathrm{DescriptionSize}(C) = ??$

$\mathrm{CheckingCost}(C) = N$

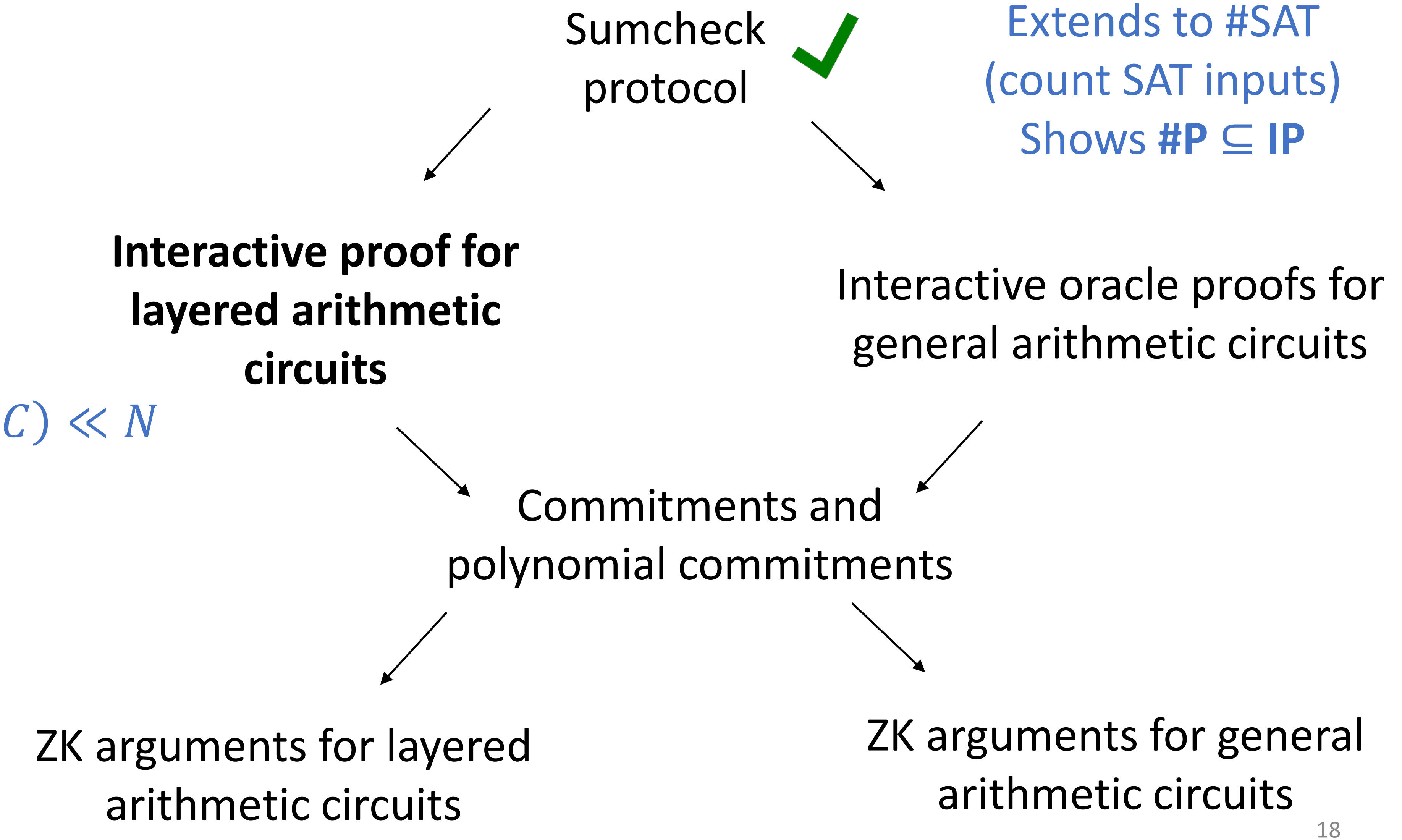
**Proofs for layered arithmetic circuits:** $\mathrm{DescriptionSize}(C) \ll N$

**Proofs for general arithmetic circuits:** $\mathrm{DescriptionSize}(C) = N$

Indirect access to instance

$V$ uses short digest or makes queries

# Plan for the next few lectures



Sumcheck protocol ✓

**coNP ⊆ IP** ✓

Extends to #SAT (count SAT inputs)

Shows **#P ⊆ IP**

**Interactive proof for layered arithmetic circuits**

$\text{DescriptionSize}(C) \ll N$

Interactive oracle proofs for general arithmetic circuits

Commitments and polynomial commitments

ZK arguments for layered arithmetic circuits

ZK arguments for general arithmetic circuits

# Layered circuits

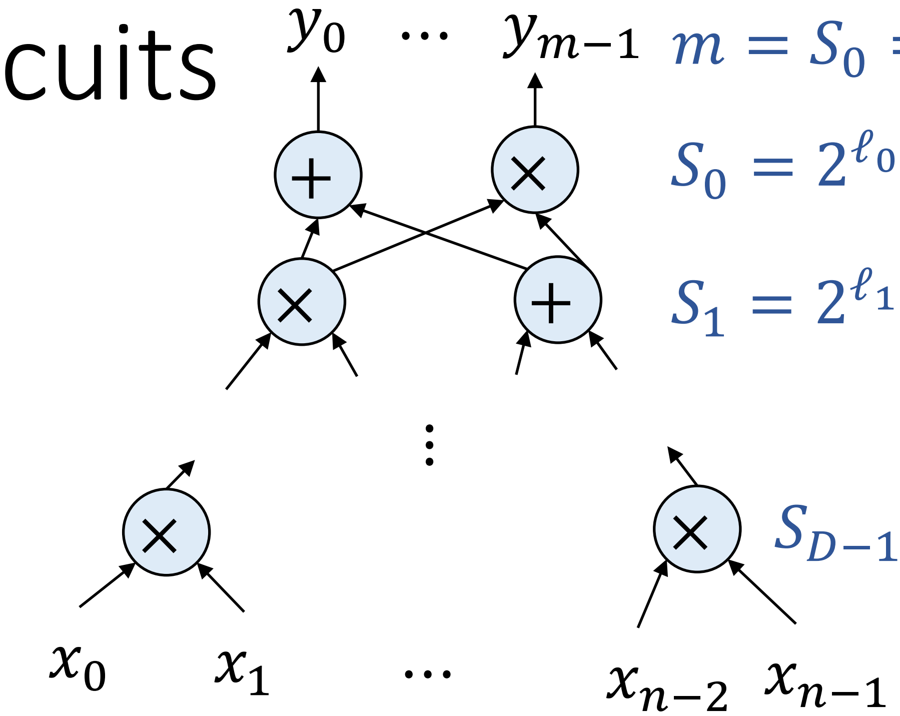
Output layer 0

Layer 1

⋮

Layer D − 1

Input layer D



$m = S_0 = 2^{\ell_0}$ outputs

$S_0 = 2^{\ell_0}$ gates

$S_1 = 2^{\ell_1}$ gates

$S_{D-1} = 2^{\ell_{D-1}}$ gates

$n = S_D = 2^{\ell_D}$ inputs

Each layer has at most $S$ gates

Label layer $i$ gates with strings $\in \{0,1\}^{\ell_i}$

gate functions

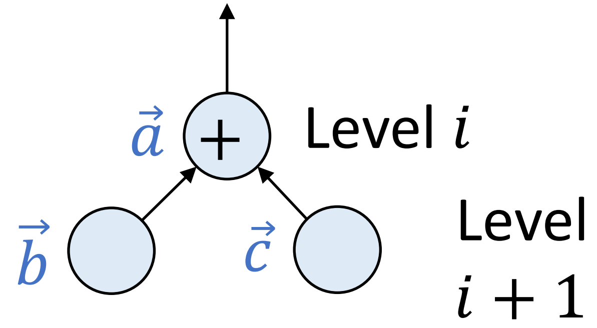
left input gate label, level $i+1$

Does this gate exist?

$$\text{mult}_i, \text{add}_i : \quad \{0,1\}^{\ell_i} \quad \times \quad \{0,1\}^{\ell_{i+1}} \times \{0,1\}^{\ell_{i+1}} \quad \to \quad \{0,1\}$$

output gate label level $i$

right input gate label level $i+1$



$\text{add}_i\left(\vec{a}, \vec{b}, \vec{c}\right) = 1$

$\text{add}_i(\vec{a}, \vec{b}, \vec{c}) \coloneqq (a_0 == b_0 == c_0)$ describes $2^{\ell_i}$ gates

# GKR protocol

- Goldwasser, Kalai, Rothblum, 2008, “Interactive Proofs for Muggles”.

- Efficient prover (polynomial time).

- $\mathcal{L}_{Eval} = \{(\{add_i, mult_i\}_{i=0}^{D-1}, \vec{x}, \vec{y}) : C(\vec{x}) = \vec{y}\}$.

- No witness or zero knowledge property.

- Superfast verification if $\mathrm{DescriptionSize}(C) \ll N$

**P** language because $\vec{x}$ is part of the instance

Will be added later when we introduce commitments

# Multilinear extensions

**Definition:** Given $f : \{0,1\}^\ell \rightarrow \mathbb{F}$, a polynomial $\tilde{f}(X_1, \dots, X_\ell) \in \mathbb{F}[X_1, \dots, X_\ell]$ satisfying

- $\deg_{X_i} \tilde{f} = 1, \forall i \in [\ell]$, and
- $f(\vec{\omega}) = \tilde{f}(\vec{\omega})\ \forall \vec{\omega} \in \{0,1\}^\ell$

is called the *multilinear extension* (MLE) of $f$.

**Fact:** $\tilde{f}$ exists for any such $f$ and is *unique*.

For vectors $\vec{a} \in \mathbb{F}^n$ ↔ $a : \{0,1\}^{\log n} \rightarrow \mathbb{F}$ ↔ $\tilde{a} \in \mathbb{F}[X_1, \dots, X_{\log n}]$

$$a(\vec{j}) \coloneqq (\vec{a})_j\ \forall j \in [n]$$

$$\vec{j} \coloneqq \text{Binary}(j)$$

# Wire value functions

**Definition:**

Given an instance $\left(\{\mathrm{add}_i, \mathrm{mult}_i\}_{i=0}^{D-1}, \vec{x}, \vec{y}\right)$, the *wire value functions*

$w_i : \{0,1\}^{\ell_i} \to \mathbb{F}$ are defined recursively as follows:

gate label gate output value

The $w_i$ are defined by the instance

Not a witness

- Input layer: $\forall j \in [n], w_D(\vec{j}) := (\vec{x})_j$
- Layers $i < D$: $\forall \vec{a}, \vec{b}, \vec{c}$ with $\mathrm{add}_i(\vec{a}, \vec{b}, \vec{c}) = 1, w_i(\vec{a}) := w_{i+1}(\vec{b}) + w_{i+1}(\vec{c})$
- Layers $i < D$: $\forall \vec{a}, \vec{b}, \vec{c}$ with $\mathrm{mult}_i(\vec{a}, \vec{b}, \vec{c}) = 1, w_i(\vec{a}) := w_{i+1}(\vec{b}) \cdot w_{i+1}(\vec{c})$

**Observation:** $\forall i < D, \vec{z} \in \{0,1\}^{\ell_i}$,

“Layer sum equation”

Also true replacing functions with MLEs

$$w_i(\vec{z}) = \sum_{\vec{b},\vec{c} \in \{0,1\}^{\ell_{i+1}}} \left(\mathrm{add}_i\left(\vec{z}, \vec{b}, \vec{c}\right)\left[w_{i+1}\left(\vec{b}\right) + w_{i+1}(\vec{c})\right] + \mathrm{mult}_i\left(\vec{z}, \vec{b}, \vec{c}\right)\left[w_{i+1}\left(\vec{b}\right) \cdot w_{i+1}(\vec{c})\right]\right)$$

# Schwartz-Zippel Lemma

**Lemma:**

- Let $\mathbb{F}$ be a field.
- Let $p \in \mathbb{F}[X_1, \ldots, X_\ell]$ be a non-zero polynomial.
- Let $\mathcal{S} \subseteq \mathbb{F}$.

Then $\Pr_{r_1,\ldots,r_\ell \leftarrow \mathcal{S}}[p(r_1, \ldots, r_\ell) = 0] \leq \frac{\deg p}{|\mathcal{S}|}$.

This can be seen as a generalization of the statement that a non-zero polynomial $p(X)$ has at most $\deg p$ roots.

# GKR protocol overview

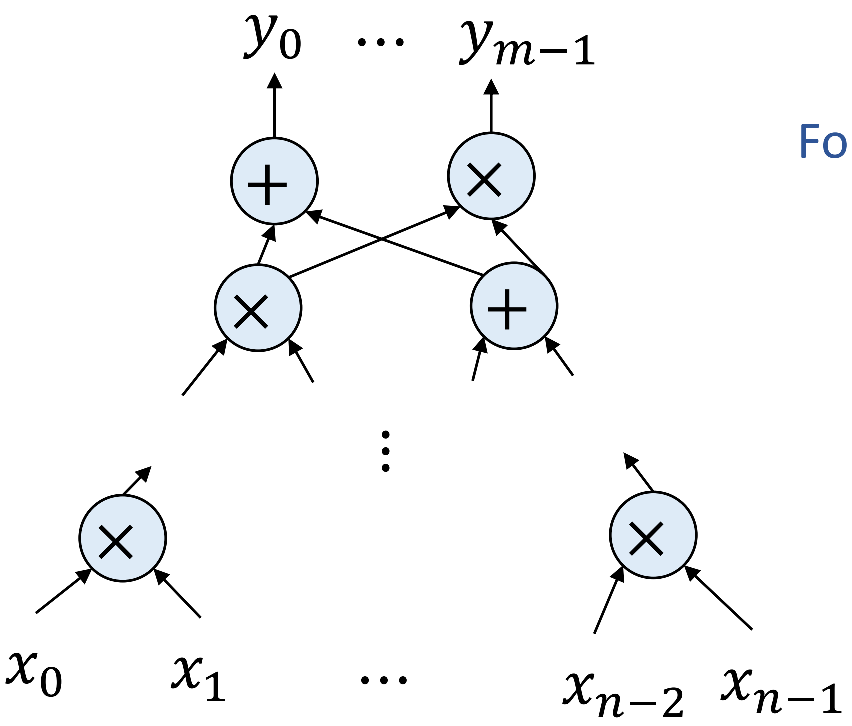
Idea: successively reduce claims about level $i$ to level $i+1$

**Instance:**

- $\left(\{\mathrm{add}_i, \mathrm{mult}_i\}_{i=0}^{D-1}, \vec{x}, \vec{y}\right)$

**Language:**

- satisfies $C(\vec{x}) = \vec{y}$



Initial claim: $\tilde{w}_0 \equiv \tilde{y}$

P Initial reduction V

Claim: $\tilde{w}_0(\vec{r}_0) = v_0$

For $i = 1, \ldots, D$:

Claim: $\tilde{w}_i(\vec{r}_i) = v_i$

P sumcheck V

Claims: $\tilde{w}_{i+1}(\vec{\beta}_{i+1}) = B_{i+1}$
$\tilde{w}_{i+1}(\vec{\gamma}_{i+1}) = C_{i+1}$

P 2-to-1 reduction V

Claim: $\tilde{w}_{i+1}(\vec{r}_{i+1}) = v_{i+1}$

Final claim: $\tilde{w}_D(\vec{r}_D) = v_D$

Check final claim using $\vec{x}$

ETH Zurich, Department of Computer Science	Dr. Jonathan Bootle
HS 2023	Karen Klein, Varun Maram, Antonio Merino Gallardo

# Zero-Knowledge Proofs
# Exercise 7

### 7.1 Arithmetic-Circuit Decomposition

The goal of this exercise is to analyse a concrete example of an MPC protocol and show that it satisfies suitable properties to be transformed into a ZK proof using a *variant* of the MPC-in-the-head method.

Consider a finite ring $(R, +, \cdot)$ and an arithmetic circuit $C$ (with $k$ input wires and 1 output wire) over $R$ built from addition and multiplication gates. Denote the total number of gates by $N$.

The protocol involves three parties $P_1$, $P_2$ and $P_3$ with respective private inputs (also referred to as shares) $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3 \in R^k$ and a common public input $y \in R$. The function the parties compute is whether $y = C(\mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3)$, i.e., the corresponding NP relation consists of the pairs $(y, \mathbf{x}) \in R \times R^k$ such that $y = C(\mathbf{x})$.

Each gate $C^j$ of the circuit for $j \in [\![N]\!]$[1] is decomposed into three "gate evaluation" functions $C_1^j, C_2^j, C_3^j$, where $C_i^j$ is to be interpreted as evaluating the gate $C^j$ from the point of view of party $P_i$.

$P_i$ maintains a vector $\mathbf{v}_i \in R^{k+N}$ containing their input share $x_i$ (in the first $k$ entries, starting at 0), as well as the intermediate values they compute (stored in the next $N$ entries) – i.e., $v_i[k+j-1]$ for $j \in [\![N]\!]$ contains the evaluation of $C_i^j$ corresponding to the $j$-th gate of the circuit $C^j$. More specifically, let $j_0, j_1 \in [\![k+j-1]\!]$ index the left and right input "gates" of $C^j$ respectively – i.e., if $1 \leq j_b \leq k$, then the "gate" just forwards the $j_b$-th input wire of the circuit, and if $k < j_b \leq k+j-1$, then we have the corresponding input gate to be the additional/multiplication gate $C^{j_b - k}$. Then if $C^j$ is

- a unary addition gate to a constant $\alpha \in R$ (with no right gate) , then for all $i \in [\![3]\!]$

$$\mathbf{v}_i[k+j-1] \leftarrow C_i^j(\mathbf{v}_i[j_0]) \coloneqq \begin{cases} \mathbf{v}_i[j_0] + \alpha \text{ if } i = 1 \\ \mathbf{v}_i[j_0] \text{ otherwise} \end{cases}$$

- a unary multiplication gate to a constant $\alpha \in R$ (with no right gate), then for all $i \in [\![3]\!]$

$$\mathbf{v}_i[k+j-1] \leftarrow C_i^j(\mathbf{v}_i[j_0]) \coloneqq \alpha \cdot \mathbf{v}_i[j_0]$$

- a binary addition gate, then for all $i \in [\![3]\!]$

$$\mathbf{v}_i[k+j-1] \leftarrow C_i^j(\mathbf{v}_i[j_0], \mathbf{v}_i[j_1]) \coloneqq \mathbf{v}_i[j_0] + \mathbf{v}_i[j_1]$$

[1] Note that $[\![N]\!] = \{1, 2, \ldots, N\}$.

- a binary multiplication gate, then for all $i \in [\![3]\!]$

$$\begin{aligned}\mathbf{v}_i[k+j-1] \leftarrow C_i^j\left(\mathbf{v}_i[j_0, j_1], \mathbf{v}_{i+1 \bmod 3}[j_0, j_1]\right) &\coloneqq \mathbf{v}_i[j_0] \cdot \mathbf{v}_i[j_1] + \mathbf{v}_{i+1 \bmod 3}[j_0] \cdot \mathbf{v}_i[j_1] \\ &+ \mathbf{v}_i[j_0] \cdot \mathbf{v}_{i+1 \bmod 3}[j_1] + \mathbf{r}_i[j-1] - \mathbf{r}_{i+1 \bmod 3}[j-1]\end{aligned}$$

  for uniformly random values $\mathbf{r}_i[j-1]$ and $\mathbf{r}_{i+1 \bmod 3}[j-1]$ maintained by parties $P_i$, $P_{i+1}$ in vectors $\mathbf{r}_i, \mathbf{r}_{i+1 \bmod 3} \in R^N$ respectively.
  In other words, the function $C_i^j$ takes as input values from $\mathbf{v}_i$ and $\mathbf{v}_{i+1 \bmod 3}$ and its output is stored in $\mathbf{v}_i$ at position $k+j-1$.

During the protocol, $P_i$ evaluates gate by gate the $i$-th decomposition of the circuit $C_i^j$ and sends $P_{i-1 \bmod 3}$ the value the latter needs for each multiplication gate. At the end of the protocol, each party broadcasts their share $y_i \leftarrow \mathbf{v}_i[k+N-1]$ of the circuit output, and each party returns 1 if $y = y_1 + y_2 + y_3$ and otherwise returns 0.

Show that the protocol is correct and that it satisfies a *weakened* version of 2-privacy, namely that there exists a probabilistic polynomial-time simulator $S$ such that for all $T \in \binom{[\![3]\!]}{2}$, the output of $S(T, y, (\mathbf{x}_i)_{i \in T})$ has the same distribution as the views $(\{\mathbf{r}_i, \mathbf{v}_i\}_{i \in T}, y_{i \notin T})$ *given* $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ satisfy $C(\mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3) = y$.[2]

## 7.2 From Honest-Verifier to Full Zero-Knowledge via Extractable Commitments

An extractable commitment scheme $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Verify}, \mathsf{ExtSetup}, \mathsf{Extract})$ is a commitment scheme (as defined in the lectures) that additionally features

- an extraction trapdoor setup algorithm $\mathsf{ExtSetup}$ that generates public parameters $pp$ and an extraction key $ek$ on input a security parameter $1^\lambda$; namely, we have $(pp, ek) \leftarrow \mathsf{ExtSetup}(1^\lambda)$. The distribution of the parameters $pp$ that it generates is indistinguishable from the output of the standard $\mathsf{Setup}$ algorithm.
- an extraction algorithm $\mathsf{Extract}$ that can compute a committed message from any valid commitment (the algorithm returns $\perp$ if the commitment is invalid) given an extraction key $ek$. More formally, given $(pp, ek) \leftarrow \mathsf{ExtSetup}(1^\lambda)$ and a valid commitment $C$, we have $(m, d) \leftarrow \mathsf{Extract}(pp, ek, C)$ such that $\mathsf{Verify}(pp, C, d, m) = 1$.

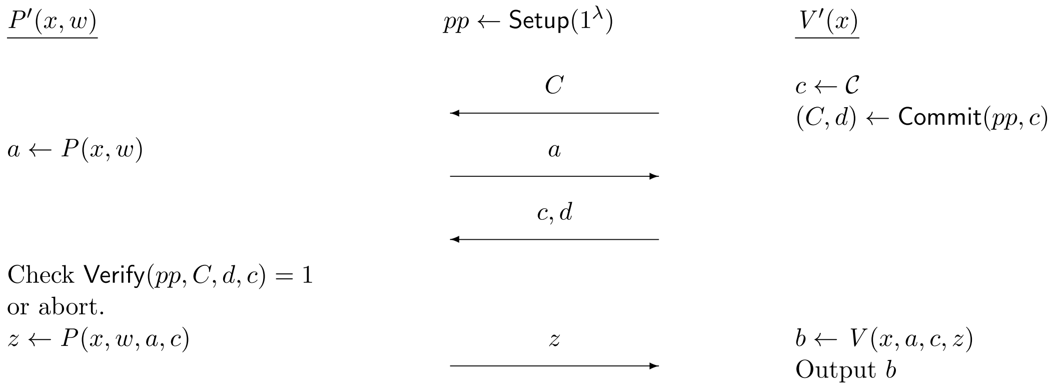
Let $(P, V)$ be a $\Sigma$-Protocol for a relation $R$. Consider the following protocol $(P', V')$ for the same $R$, with challenge space $\mathcal{C}$:

**a)** Show the completeness of protocol $(P', V')$ with respect to language $L(R)$.

[2] This definition of "weak" 2-privacy is weaker than the original definition of 2-privacy presented in the lectures on the following accounts:

- According to the original definition, the simulator $S$ is supposed to simulate views of the parties in $T$ for *any* final output of the MPC protocol; i.e., even when $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ satisfy $C(\mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3) \neq y$. However in the context of proving zero-knowledge of the "MPC-in-the-head" construction, recall from our SHVZK analysis in the lectures that we only need the MPC simulator $S$ to work when the parties in $T$ output 1, i.e., when $C(\mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3) = y$.
- Note that in an execution of the given MPC protocol, a party $P_i$ may obtain some values from $P_{i-1 \bmod 3}$ w.r.t. a multiplication gate. So technically, these values should also be a part of $P_i$'s view. However, it turns out that restricting our definition of $P_i$'s view to the vector $\mathbf{v}_i$ (along with $\mathbf{r}_i$) is sufficient to prove SHVZK property (and special-spundness) of the resulting $\Sigma$-protocol based on the "MPC-in-the-head" paradigm; see [GMO16] for a formal analysis.

The takeaway is that, in the context of constructing ZK proofs from MPC protocols via a *concrete* instantiation of the "MPC-in-the-head" paradigm, we should not consider the corresponding MPC properties (i.e., definitions of "views", "privacy", etc.) *in isolation* – and instead adapt the definitions in a way which allow us to establish the final properties (completeness, SHVZK, special-soundness) of our constructed "MPC-in-the-head" ZK proofs.


$P'(x, w)$
$pp \leftarrow \mathsf{Setup}(1^\lambda)$
$V'(x)$
$C$
$c \leftarrow \mathcal{C}$
$(C, d) \leftarrow \mathsf{Commit}(pp, c)$
$a \leftarrow P(x, w)$
$a$
$c, d$
Check $\mathsf{Verify}(pp, C, d, c) = 1$
or abort.
$z \leftarrow P(x, w, a, c)$
$z$
$b \leftarrow V(x, a, c, z)$
Output $b$

**b)** Show that $(P', V')$ satisfies computational zero-knowledge.

Hint: You may assume that the extractable commitment scheme satisfies a property called *binding extractability*. That is, no polynomial-time adversary on input *trapdoor parameters pp* can compute with non-negligible probability a triple $(C, m, d)$ of commitment, message and decommitment information such that $\mathsf{Verify}(pp, C, d, m) = 1$ and $\mathsf{Extract}(pp, ek, C) \neq m$.

Now assume that the above extractable commitment scheme is also *equivocable*.[3] Namely, it includes two additional algorithms $(\mathsf{EqSetup}, \mathsf{EqOpen})$ where

- the "equivocation" trapdoor setup algorithm $\mathsf{EqSetup}$ generates public parameters $pp$ and an equivocation key $ek$ on input a security parameter $1^\lambda$; i.e., $(pp, ek) \leftarrow \mathsf{EqSetup}(1^\lambda)$. Similar to $\mathsf{ExtSetup}$ above, we have the distribution of parameters $pp$ that $\mathsf{EqSetup}$ generates to be indistinguishable from the output of the standard $\mathsf{Setup}$ algorithm.
- the "equivocal opening" algorithm $\mathsf{EqOpen}$ that can compute a valid decommitment to *any* valid commitment w.r.t. *any* message. More formally, given $(pp, ek) \leftarrow \mathsf{EqSetup}(1^\lambda)$, any valid commitment $C$ and any message $m$, we have $d \leftarrow \mathsf{EqOpen}(pp, ek, C, m)$ such that $\mathsf{Verify}(pp, C, d, m) = 1$.

**c)** Show that $(P', V')$ now also satisfies knowledge soundness.

Hint: You may further assume that the extractable commitment scheme satisfies a property called *equivocational indistinguishability*. It roughly states that no polynomial-time adversary on input *equivocation parameters pp* can distinguish with non-negligible probability a triple $(C, m, d)$, where $(C, d) \leftarrow \mathsf{Commit}(pp, m)$, from the triple $(C, m', d')$, where $d' \leftarrow \mathsf{EqOpen}(pp, ek, C, m')$ where $m$ and $m'$ are two uniformly random *and independent* messages.

## 7.3 Applications of the Sumcheck Protocol

**a)** Let $f\colon \{0,1\}^\ell \to \{0,1\}$ be a function. Let $\mathbb{F}$ be a field. Show that there is a unique polynomial $p \in \mathbb{F}[X_1, \ldots, X_\ell]$ of individual degree at most 1 whose evaluations match those of $f$ on $\{0,1\}^\ell$.

**b)** Let $G$ be a graph. Give an interactive proof that convinces the verifier that $G$ has $v$ triangles.

Hint: Encode the vertices of the graph as strings in $\{0,1\}^\ell$. Let $a\colon \{0,1\}^\ell \times \{0,1\}^\ell \to \{0,1\}$ be a function that returns 1 if two input vertices are connected by an edge and 0 if not. Use $a$ to design a function which counts the number of triangles in $G$.

[3] Examples of extractable and equivocable commitments exist in the literature [FLM11, ABB+13].

# References

[ABB$^+$13] Michel Abdalla, Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, and David Pointcheval. SPHF-friendly non-interactive commitments. pages 214–234, 2013.

[FLM11] Marc Fischlin, Benoît Libert, and Mark Manulis. Non-interactive and re-usable universally composable string commitments with adaptive security. pages 468–485, 2011.

[GMO16] Irene Giacomelli, Jesper Madsen, and Claudio Orlandi. Zkboo: Faster zero-knowledge for boolean circuits. In *Proceedings of the 25th USENIX Conference on Security Symposium*, SEC'16, page 1069–1083, USA, 2016. USENIX Association.

ETH Zurich, Department of Computer Science Dr. Jonathan Bootle
HS 2023 Karen Klein, Varun Maram, Antonio Merino Gallardo

# Zero-Knowledge Proofs
# Exercise 7

## 7.1 Arithmetic-Circuit Decomposition

The goal of this exercise is to analyse a concrete example of an MPC protocol and show that it satisfies suitable properties to be transformed into a ZK proof using a *variant* of the MPC-in-the-head method.

Consider a finite ring $(R, +, \cdot)$ and an arithmetic circuit $C$ (with $k$ input wires and 1 output wire) over $R$ built from addition and multiplication gates. Denote the total number of gates by $N$.

The protocol involves three parties $P_1$, $P_2$ and $P_3$ with respective private inputs (also referred to as shares) $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3 \in R^k$ and a common public input $y \in R$. The function the parties compute is whether $y = C(\mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3)$, i.e., the corresponding NP relation consists of the pairs $(y, \mathbf{x}) \in R \times R^k$ such that $y = C(\mathbf{x})$.

Each gate $C^j$ of the circuit for $j \in [\![N]\!]$[1] is decomposed into three "gate evaluation" functions $C_1^j, C_2^j, C_3^j$, where $C_i^j$ is to be interpreted as evaluating the gate $C^j$ from the point of view of party $P_i$.

$P_i$ maintains a vector $\mathbf{v}_i \in R^{k+N}$ containing their input share $x_i$ (in the first $k$ entries, starting at 0), as well as the intermediate values they compute (stored in the next $N$ entries) – i.e., $v_i[k+j-1]$ for $j \in [\![N]\!]$ contains the evaluation of $C_i^j$ corresponding to the $j$-th gate of the circuit $C^j$. More specifically, let $j_0, j_1 \in [\![k+j-1]\!]$ index the left and right input "gates" of $C^j$ respectively – i.e., if $1 \le j_b \le k$, then the "gate" just forwards the $j_b$-th input wire of the circuit, and if $k < j_b \le k+j-1$, then we have the corresponding input gate to be the additional/multiplication gate $C^{j_b-k}$. Then if $C^j$ is

- a unary addition gate to a constant $\alpha \in R$ (with no right gate) , then for all $i \in [\![3]\!]$

$$\mathbf{v}_i[k+j-1] \leftarrow C_i^j(\mathbf{v}_i[j_0]) \coloneqq \begin{cases} \mathbf{v}_i[j_0] + \alpha \text{ if } i = 1 \\ \mathbf{v}_i[j_0] \text{ otherwise} \end{cases}$$

- a unary multiplication gate to a constant $\alpha \in R$ (with no right gate), then for all $i \in [\![3]\!]$

$$\mathbf{v}_i[k+j-1] \leftarrow C_i^j(\mathbf{v}_i[j_0]) \coloneqq \alpha \cdot \mathbf{v}_i[j_0]$$

- a binary addition gate, then for all $i \in [\![3]\!]$

$$\mathbf{v}_i[k+j-1] \leftarrow C_i^j(\mathbf{v}_i[j_0], \mathbf{v}_i[j_1]) \coloneqq \mathbf{v}_i[j_0] + \mathbf{v}_i[j_1]$$

[1] Note that $[\![N]\!] = \{1, 2, \ldots, N\}$.

- a binary multiplication gate, then for all $i \in [\![3]\!]$

$$\begin{aligned}\mathbf{v}_i[k+j-1] \leftarrow C_i^j\left(\mathbf{v}_i[j_0,j_1], \mathbf{v}_{i+1 \bmod 3}[j_0,j_1]\right) &\coloneqq \mathbf{v}_i[j_0]\cdot\mathbf{v}_i[j_1] + \mathbf{v}_{i+1 \bmod 3}[j_0]\cdot\mathbf{v}_i[j_1] \\ &+ \mathbf{v}_i[j_0]\cdot\mathbf{v}_{i+1 \bmod 3}[j_1] + \mathbf{r}_i[j-1] - \mathbf{r}_{i+1 \bmod 3}[j-1]\end{aligned}$$

  for uniformly random values $\mathbf{r}_i[j-1]$ and $\mathbf{r}_{i+1 \bmod 3}[j-1]$ maintained by parties $P_i$, $P_{i+1}$ in vectors $\mathbf{r}_i, \mathbf{r}_{i+1 \bmod 3} \in R^N$ respectively.
  In other words, the function $C_i^j$ takes as input values from $\mathbf{v}_i$ and $\mathbf{v}_{i+1 \bmod 3}$ and its output is stored in $\mathbf{v}_i$ at position $k+j-1$.

During the protocol, $P_i$ evaluates gate by gate the $i$-th decomposition of the circuit $C_i^j$ and sends $P_{i-1 \bmod 3}$ the value the latter needs for each multiplication gate. At the end of the protocol, each party broadcasts their share $y_i \leftarrow \mathbf{v}_i[k+N-1]$ of the circuit output, and each party returns 1 if $y = y_1 + y_2 + y_3$ and otherwise returns 0.

Show that the protocol is correct and that it satisfies a *weakened* version of 2-privacy, namely that there exists a probabilistic polynomial-time simulator $S$ such that for all $T \in \binom{[\![3]\!]}{2}$, the output of $S(T, y, (\mathbf{x}_i)_{i\in T})$ has the same distribution as the views $(\{\mathbf{r}_i, \mathbf{v}_i\}_{i\in T}, y_{i\notin T})$ *given* $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ satisfy $C(\mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3) = y$.[2]

**Solution:** Correctness. To prove the protocol correct, it suffices to show that if $C^j$ is

- a unary addition gate to a constant $\alpha \in R$ with $j_0$ indexing the input "gate", then

$$\sum_i \mathbf{v}_i[k+j-1] = \sum_i \mathbf{v}_i[j_0] + \alpha$$

- a unary multiplication gate to a constant $\alpha \in R$ with $j_0$ indexing the input "gate", then

$$\sum_i \mathbf{v}_i[k+j-1] = \alpha \cdot \sum_i \mathbf{v}_i[j_0]$$

- a binary addition gate with $j_0$, $j_1$ indexing the left and right input "gates" respectively, then

$$\sum_i \mathbf{v}_i[k+j-1] = \sum_i \mathbf{v}_i[j_0] + \sum_i \mathbf{v}_i[j_1]$$

- a binary multiplication gate with $j_0$, $j_1$ indexing the left and right input "gates" respectively, then

$$\sum_i \mathbf{v}_i[k+j-1] = \left(\sum_i \mathbf{v}_i[j_0]\right)\cdot\left(\sum_i \mathbf{v}_i[j_1]\right).$$

[2] This definition of "weak" 2-privacy is weaker than the original definition of 2-privacy presented in the lectures on the following accounts:

- According to the original definition, the simulator $S$ is supposed to simulate views of the parties in $T$ for *any* final output of the MPC protocol; i.e., even when $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ satisfy $C(\mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3) \neq y$. However in the context of proving zero-knowledge of the "MPC-in-the-head" construction, recall from our SHVZK analysis in the lectures that we only need the MPC simulator $S$ to work when the parties in $T$ output 1, i.e., when $C(\mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3) = y$.
- Note that in an execution of the given MPC protocol, a party $P_i$ may obtain some values from $P_{i-1 \bmod 3}$ w.r.t. a multiplication gate. So technically, these values should also be a part of $P_i$'s view. However, it turns out that restricting our definition of $P_i$'s view to the vector $\mathbf{v}_i$ (along with $\mathbf{r}_i$) is sufficient to prove SHVZK property (and special-soundness) of the resulting $\Sigma$-protocol based on the "MPC-in-the-head" paradigm; see [GMO16] for a formal analysis.

The takeaway is that, in the context of constructing ZK proofs from MPC protocols via a *concrete* instantiation of the "MPC-in-the-head" paradigm, we should not consider the corresponding MPC properties (i.e., definitions of "views", "privacy", etc.) *in isolation* – and instead adapt the definitions in a way which allow us to establish the final properties (completeness, SHVZK, special-soundness) of our constructed "MPC-in-the-head" ZK proofs.

Indeed, by considering these equalities iteratively for all the gates it follows that $\sum_i \mathbf{v}_i[k+N-1] = C(\sum_i \mathbf{x}_i)$, which then implies that $y = \sum_i y_i = C(\mathbf{x})$.

The first three equalities immediately follow from the definitions of $C_i^j$. As for the last one, note that

$$\begin{aligned}\sum_i \mathbf{v}_i[k+j-1] &= \sum_i \mathbf{v}_i[j_0] \cdot (\mathbf{v}_i[j_1] + \mathbf{v}_{i+1 \bmod 3}[j_1] + \mathbf{v}_{i-1 \bmod 3}[j_1]) \\ &+ \sum_i \mathbf{r}_i - \sum_i \mathbf{r}_{i+1 \bmod 3} \\ &= \left(\sum_i \mathbf{v}_i[j_0]\right) \cdot \left(\sum_i \mathbf{v}_i[j_1]\right).\end{aligned}$$

2-Privacy. Fix $T = \{i, i+1 \bmod 3\} \in \binom{[\![3]\!]}{2}$ for some $i \in [\![3]\!]$ . Consider a simulator which, on input $(T, y, (\mathbf{x}_i)_{i\in T})$,

- generates uniformly random vectors $\mathbf{r}_i$ and $\mathbf{r}_{i+1 \bmod 3}$,
- computes $\mathbf{v}_i[k+j-1]$ and $\mathbf{v}_{i+1 \bmod 3}[k+j-1]$ respectively as $C_i^j$ and $C_{i+1 \bmod 3}^j$ if $C^j$ is an addition to or a multiplication by a constant or an addition gate,
- computes $\mathbf{v}_i[k+j-1]$ as $C_i^j$ and generates a uniformly random value $\mathbf{v}_{i+1 \bmod 3}[k+j-1]$ if $C^j$ is a multiplication gate,
- sets $y_i \leftarrow \mathbf{v}_i[k+N-1]$, $y_{i+1 \bmod 3} \leftarrow \mathbf{v}_{i+3 \bmod 3}[k+N-1]$ and $y_{i+2 \bmod 3} \leftarrow y - y_i - y_{i+1 \bmod 3}$,
- returns $(\{\mathbf{r}_i, \mathbf{v}_i\}_{i\in T}, y_{i\notin T})$.

Except for $\mathbf{v}_{i+1 \bmod 3}[k+j-1]$ when $C^j$ is a multiplication between two inputs, the values computed by the simulator are exactly as in the protocol. However, when $C^j$ is a multiplication between two inputs, $\mathbf{v}_{i+1 \bmod 3}[k+j-1]$ is computed by subtracting a uniformly random value $\mathbf{r}_{i+2 \bmod 3}$, which is independent of $\mathbf{r}_i$ and $\mathbf{r}_{i+1 \bmod 3}$, and therefore the output of the simulator is distributed as in a real protocol execution.

## 7.2 From Honest-Verifier to Full Zero-Knowledge via Extractable Commitments

An extractable commitment scheme $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Verify}, \mathsf{ExtSetup}, \mathsf{Extract})$ is a commitment scheme (as defined in the lectures) that additionally features

- an extraction trapdoor setup algorithm $\mathsf{ExtSetup}$ that generates public parameters $pp$ and an extraction key $ek$ on input a security parameter $1^\lambda$; namely, we have $(pp, ek) \leftarrow \mathsf{ExtSetup}(1^\lambda)$. The distribution of the parameters $pp$ that it generates is indistinguishable from the output of the standard $\mathsf{Setup}$ algorithm.
- an extraction algorithm $\mathsf{Extract}$ that can compute a committed message from any valid commitment (the algorithm returns $\perp$ if the commitment is invalid) given an extraction key $ek$. More formally, given $(pp, ek) \leftarrow \mathsf{ExtSetup}(1^\lambda)$ and a valid commitment $C$, we have $(m, d) \leftarrow \mathsf{Extract}(pp, ek, C)$ such that $\mathsf{Verify}(pp, C, d, m) = 1$.

Let $(P, V)$ be a $\Sigma$-Protocol for a relation $R$. Consider the following protocol $(P', V')$ for the same $R$, with challenge space $\mathcal{C}$:

**a)** Show the completeness of protocol $(P', V')$ with respect to language $L(R)$.

**Solution:** The completeness of $(P', V')$ immediately follows from the correctness of the commitment scheme and the completeness of $(P, V)$.

| $\underline{P'(x,w)}$ | $pp \leftarrow \mathsf{Setup}(1^\lambda)$ | $\underline{V'(x)}$ |
|---|---|---|
| | $\xleftarrow{\quad C \quad}$ | $c \leftarrow \mathcal{C}$<br>$(C,d) \leftarrow \mathsf{Commit}(pp,c)$ |
| $a \leftarrow P(x,w)$ | $\xrightarrow{\quad a \quad}$ | |
| | $\xleftarrow{\quad c,d \quad}$ | |
| Check $\mathsf{Verify}(pp,C,d,c)=1$<br>or abort.<br>$z \leftarrow P(x,w,a,c)$ | $\xrightarrow{\quad z \quad}$ | $b \leftarrow V(x,a,c,z)$<br>Output $b$ |

**b)** Show that $(P', V')$ satisfies computational zero-knowledge.

HINT: You may assume that the extractable commitment scheme satisfies a property called *binding extractability*. That is, no polynomial-time adversary on input *trapdoor parameters* $pp$ can compute with non-negligible probability a triple $(C, m, d)$ of commitment, message and decommitment information such that $\mathsf{Verify}(pp, C, d, m) = 1$ and $\mathsf{Extract}(pp, ek, C) \neq m$.

**Solution:** To prove the protocol zero-knowledge, consider a polynomial-time verifier $V^*$. Let $S$ be a simulator that runs the trapdoor set-up algorithm $\mathsf{ExtSetup}$ for the commitment scheme instead of the standard set-up algorithm. On an input $x \in L(R)$, simulator $S$ runs $V^*$ on the parameters for $(P, V)$, the resulting (trapdoor) commitment parameters $pp$ and $x$. If $V^*$ computes an initial commitment $C$, algorithm $S'$ runs $\mathsf{Extract}(pp, ek, C)$ to recover a committed challenge $c$ and corresponding decommitment information $d$. It then runs the SHVZK simulator of $(P, V)$ to obtain a transcript $(a, c, z)$ and sends $a$ to $V^*$. If $V^*$ computes a challenge $c'$ and decommitment information $d'$ such that $\mathsf{Verify}(pp, C, d', c') = 1$ and $c = c'$, then $S'$ returns $(C, a, c, z)$. If $V^*$ does not compute a commitment $C$ or a challenge $c'$ and a decommitment $d'$, then $S'$ returns $\perp$. If $V^*$ computes a challenge $c' \neq c$ and a decommitment $d'$ such that $\mathsf{Verify}(pp, C, d', c') = 1$, then $S'$ returns $\perp$. The latter event is the only one that can allow to distinguish the output of $S'$ from an execution between $P$ and $V^*$. But this event only occurs with a negligible probability since we assumed that the commitment scheme satisfies *binding extractability* (see the hint above), and the above tuple $(C, c', d')$ breaks such a property. The protocol is therefore computational zero-knowledge.

Now assume that the above extractable commitment scheme is also *equivocable*.[3] Namely, it includes two additional algorithms $(\mathsf{EqSetup}, \mathsf{EqOpen})$ where

- the "equivocation" trapdoor setup algorithm $\mathsf{EqSetup}$ generates public parameters $pp$ and an equivocation key $ek$ on input a security parameter $1^\lambda$; i.e., $(pp, ek) \leftarrow \mathsf{EqSetup}(1^\lambda)$. Similar to $\mathsf{ExtSetup}$ above, we have the distribution of parameters $pp$ that $\mathsf{EqSetup}$ generates to be indistinguishable from the output of the standard $\mathsf{Setup}$ algorithm.
- the "equivocal opening" algorithm $\mathsf{EqOpen}$ that can compute a valid decommitment to *any* valid commitment w.r.t. *any* message. More formally, given $(pp, ek) \leftarrow \mathsf{EqSetup}(1^\lambda)$, any valid commitment $C$ and any message $m$, we have $d \leftarrow \mathsf{EqOpen}(pp, ek, C, m)$ such that $\mathsf{Verify}(pp, C, d, m) = 1$.

**c)** Show that $(P', V')$ now also satisfies knowledge soundness.

[3] Examples of extractable and equivocable commitments exist in the literature [FLM11, ABB+13].

HINT: You may further assume that the extractable commitment scheme satisfies a property called *equivocational indistinguishability.* It roughly states that no polynomial-time adversary on input *equivocation parameters* $pp$ can distinguish with non-negligible probability a triple $(C, m, d)$, where $(C, d) \leftarrow \mathsf{Commit}(pp, m)$, from the triple $(C, m', d')$, where $d' \leftarrow \mathsf{EqOpen}(pp, ek, C, m')$ where $m$ and $m'$ are two uniformly random *and independent* messages.

**Solution:**

<u>*Intuitive Solution:*</u> It helps to first consider the "normal" soundness property. Consider a prover $P^*$ which is supposed to interact with $V'$. Now consider an algorithm $\tilde{V}$ interacting with $P^*$ which initially commits to a uniformly random value and later generates an independent challenge. Assuming the commitment scheme to be perfectly hiding, there must exist a decommitment that allows to open the initial commitment to the challenge that $\tilde{V}$ sends in its second message. $\tilde{V}$ then computes commitments to the challenge with every possible choice of randomness until it results in the initial commitment and a decommitment information, and sends the latter to $P^*$ together with the challenge. $P^*$ cannot distinguish $V'$ from $\tilde{V}$ and since the initial commitment $\tilde{V}$ sends is independent from its later challenge, the probability that $\tilde{V}$ accepts an interaction with $P^*$ on an input $x \notin L(R)$ is at most the probability that any prover convinces $V$ on the same input, which is at most $1/2$. Note that the runtime of $\tilde{V}$ is exponential if the randomness space for the commitment scheme is of polynomial size. It is not an issue to prove the (normal) soundness of the protocol, but one could not prove knowledge soundness by defining a knowledge extractor that proceeds as $\tilde{V}$ to decouple the initial commitment from the challenges that a knowledge extractor for the original protocol ($P$, $V$) would use. However, an extractor for ($P'$, $V'$) that runs in expected polynomial time could be defined if the commitment scheme were assumed to additionally be *equivocable* (and only computionally hiding instead of perfectly). Indeed, given a knowledge extractor $E$ for ($P$, $V$), a knowledge extractor $E'$ for ($P'$, $V'$) could be defined as an algorithm that initially commits to a uniformly random value, then runs $E$ and uses the equivocation key to open the initial commitment to the challenges independently generated by $E$. In other words, knowledge soundness can be achieved if the commitments can be efficiently equivocated, whereas inefficient equivocation (assuming the scheme to be perfectly hiding) suffices for soundness.

<u>*(More) Formal Solution:*</u> Consider a prover $P^*$ (with fixed randomness) interacting with $V'$ in the above protocol such that it is able to convince $V'$ with a non-negligible probability $\varepsilon$. Now consider a variant of the protocol where we generate public parameters $pp$ as $(pp, ek) \leftarrow \mathsf{EqSetup}(1^\lambda)$; note that $P^*$'s "success" probability changes by at-most a negligible quantity because of the indistinguishability of parameters generated by $\mathsf{Setup}$ and $\mathsf{EqSetup}$. We now further modify the protocol by having $P^*$ interact with the algorithm $\tilde{V}$ above, instead of $V'$; the only difference compared to the normal soundness case above is that $\tilde{V}$ now uses the equivocation key $ek$ to obtain the decommitment information which allows to open the initial commitment in its first message to the independent challenge later sampled by it (instead of $\tilde{V}$ using "brute-force"). To argue that $P^*$'s success probability in this case again differs from $\varepsilon$ by a negligible quantity, instead of relying on perfect hiding property of the commitment scheme, we rely on the computational property of *equivocational indistinguishability* (see the hint above). Finally, note that $P^*$'s success probability (with fixed randomness) now essentially depends on the randomness used by $\tilde{V}$ to sample its independent challenge in the second message. This allows us to use the knowledge soundness guarantees of the underlying protocol $(P, V)$. Namely, given a knowledge extractor $E$ for $(P, V)$, the knowledge extractor $E'$ for $(P', V')$ commits to a uniformly random value and sends it to $P^*$ first. Later, $E'$ forwards $P^*$'s first message to $E$, and then rewinds $P^*$

as $E$ does on different independent challenges in its second message (w.r.t. the *same* initial commitment) where $E'$ additionally uses the equivocation key $ek$ to obtain the corresponding decommitment information. $E'$ finally outputs the same witness (or $\perp$) that $E$ computes.

### 7.3 Applications of the Sumcheck Protocol

**a)** Let $f\colon \{0,1\}^\ell \to \{0,1\}$ be a function. Let $\mathbb{F}$ be a field. Show that there is a unique polynomial $p \in \mathbb{F}[X_1, \ldots, X_\ell]$ of individual degree at most 1 whose evaluations match those of $f$ on $\{0,1\}^\ell$.

**Solution:** The polynomial $\sum_{s\in\{0,1\}^\ell} \prod_{i=1}^{\ell} \left(s_i(2X_i - 1) + (1 - X_i)\right) f(s)$ is of individual degree 1 and matches the evaluations of $f$ on $\{0,1\}^\ell$. To prove its uniqueness, suppose that there are two such polynomials $P$ and $Q$. Their difference $Z := P - Q$ is of individual degree at most 1 so it can be written as $\sum_{\mathbf{i}\in\{0,1\}^\ell} z_{\mathbf{i}} X_1^{i_1} \cdots X_\ell^{i_\ell}$. Since it evaluates to 0 on $\{0,1\}^\ell$, it follows that $Z(\mathbf{i}) = z_{\mathbf{i}} = 0$ for all $\mathbf{i} \in \{0,1\}^\ell$ and $P = Q$ necessarily. This unique polynomial is often referred to as the *multi-linear extension* of $f$ in the literature.

**b)** Let $G$ be a graph. Give an interactive proof that convinces the verifier that $G$ has $v$ triangles.

Hint: Encode the vertices of the graph as strings in $\{0,1\}^\ell$. Let $a\colon \{0,1\}^\ell \times \{0,1\}^\ell \to \{0,1\}$ be a function that returns 1 if two input vertices are connected by an edge and 0 if not. Use $a$ to design a function which counts the number of triangles in $G$.

**Solution:** If $G$ has $n$ vertices, then set $\ell := \lfloor \log_2 n \rfloor + 1$. Let $a\colon \{0,1\}^\ell \times \{0,1\}^\ell \to \{0,1\}$ a function that interprets its input as the binary representation of integers in $[\![0, 2^\ell - 1]\!]$ and returns 1 if these vertices are connected in $G$ and 0 otherwise (the integers larger than $n$ are by convention assumed not to be connected to any other vertex). The number $\Delta$ of triangles in $G$ is then equal to $(1/6)\cdot\sum_{x,y,z\in\{0,1\}^\ell} a(x,y)a(y,z)a(z,x)$; the factor $1/6$ comes from the fact a triangle is an *unordered* triple of pairwise connected vertices. If $\tilde{a}$ denotes the multi-linear extension of $a$ in a field $\mathbb{F}$, $6\Delta = \sum_{x,y,z\in\{0,1\}^\ell} \tilde{a}(x,y)\tilde{a}(y,z)\tilde{a}(z,x) \bmod p$ and this equality also holds over the integers if $|\mathbb{F}| > 6\binom{n}{3}$.

The interactive protocol thus consists in running the sum-check protocol for the $3\ell$-variate polynomial $\sum_{x,y,z\in\{0,1\}^\ell} \tilde{a}(x,y)\tilde{a}(y,z)\tilde{a}(z,x)$ and the value $6v$.

# References

[ABB$^+$13] Michel Abdalla, Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, and David Pointcheval. SPHF-friendly non-interactive commitments. pages 214–234, 2013.

[FLM11] Marc Fischlin, Benoît Libert, and Mark Manulis. Non-interactive and re-usable universally composable string commitments with adaptive security. pages 468–485, 2011.

[GMO16] Irene Giacomelli, Jesper Madsen, and Claudio Orlandi. Zkboo: Faster zero-knowledge for boolean circuits. In *Proceedings of the 25th USENIX Conference on Security Symposium*, SEC'16, page 1069–1083, USA, 2016. USENIX Association.