Zero-knowledge proofs

Slides for the Cryptographic Protocols course

December 2020
Zero-knowledge proofs

- There is relation $R \subseteq \{0, 1\}^* \times \{0, 1\}^*$, $R \in \mathcal{P}$
- Two parties: prover $P$ and verifier $V$
- $P$ knows $x, w \in \{0, 1\}^*$. $V$ knows $x$
- $P$ wants to convince $V$ that he knows $w$, such that $(x, w) \in R$

**Functionality $\mathcal{F}^R_{\text{ZK}}$**

- Receive $(\text{prove}, \text{sessionId}, x, w)$ from $P$. Ignore, if $(x, w) \notin R$
- Send $(\text{proofReceived}, \text{sessionId}, |x|)$ to $A$
- Receive $(\text{sendProof}, \text{sessionId})$ from $A$
- Send $(\text{proven}, \text{sessionId}, x)$ to $V$

https://zkp.science

December 2020
Stating differently: the properties we want

- Interactive proofs:
  - **Completeness**: if \((x, w) \in R\) and \(P\) follows the protocol, then honest \(V\) is convinced
  - **Soundness**: if a (malicious) \(P\) "does not know" \(w\), then \(V\) is not convinced
    - Easy to understand, if "does not know \(w\)" means \(\neg \exists w : (x, w) \in R\)

- **Zero-knowledge**: given \(x\), the traces of the protocol can be generated without access to \(w\)
  - I.e. there exists a generation algorithm for simulated traces
  - Simulated traces and real traces are undistinguishable for \(V\)
  - We may consider malicious \(V\), or (semi-)honest \(V\)
Σ-protocols
Σ-protocols

\[(x, w) \overset{P}{\rightarrow} x \quad V \quad \]

\[(\alpha, \text{state}) \overset{\$}{\leftarrow} \text{A}(x, w) \quad \beta \overset{\$}{\in} \langle \text{some set} \rangle \]

\[\gamma \leftarrow \text{R}(x, w, \text{state}, \beta) \quad \overset{\gamma}{\rightarrow} \quad \text{V}(x, \alpha, \beta, \gamma) \rightarrow 0/1\]
\( \Sigma \)-protocols

- \( P \) has \( x, w \). \( V \) has \( x \)
- \( P \) sends \( \alpha \). At the same time, \( V \) sends the challenge \( \beta \). \( P \) sends response \( \gamma \). \( V \) accepts or rejects.
- **Completeness**: if \((x, w) \in R\), then \( V \) accepts
- **Special soundness**: if \((\alpha, \beta, \gamma)\) and \((\alpha, \beta', \gamma')\) are both accepting transcripts, then \( w \) can be found from them
  - A possible definition for “\( P \) knows \( w \)”
- **Simulatability**: Given \((x, \beta)\), can generate \((\alpha, \gamma)\) so, that \((\alpha, \beta, \gamma)\) is indistinguishable from conversations between honest \( P \) and \( V \) on \( x \)

\( \Sigma \)-protocols are interactive proofs with honest-verifier zero-knowledge (HVZK)
Fiat-Shamir heuristic

- Turns $\Sigma$-protocols to non-interactive ZK proofs
  - $P$ has $(x, w)$, $V$ has $x$
  - $P$ computes some proof string $\pi$ and makes it public
  - $V$ looks at $x$ and $\pi$, becomes convinced that $\exists w$, learns nothing about $w$
- Compute the verifier’s challenge with the random oracle, applied to the first message of the protocol.
  - It is important that the verifier’s step is just “generate a random value, send it to the prover”
  - I.e. it is a public coin protocol
- Can be generalized to multi-round public coin protocols
Σ-protocol for proving knowledge of a discrete logarithm

- Let $G$ be a group with hard DLP. Let $|G| = p \in \mathbb{P}$.
- Consider the following $R \subseteq (G \times G) \times \mathbb{Z}_p$:
  \[ R = \{(g, h, x) \mid g^x = h\} \]

Protocol

- $P$ picks $r \xleftarrow{\$} \mathbb{Z}_p$. Sets $\alpha \leftarrow g^r$
- $V$ picks $\beta \xleftarrow{\$} \mathbb{Z}_p$
- $P$ sets $\gamma \leftarrow r + \beta x$
- $V$ checks if $g^\gamma = \alpha h^\beta$
Check the properties

- **Completeness.**  \( g^\gamma = g^{r+\beta x} = g^r \cdot (g^x)^\beta = \alpha \cdot h^\beta \)

- **Special soundness.** We have \((\alpha, \beta_1, \gamma_1)\) and \((\alpha, \beta_2, \gamma_2)\), satisfying
  
  \[
  g^{\gamma_1} = \alpha h^{\beta_1} \quad \text{and} \quad g^{\gamma_2} = \alpha h^{\beta_2}
  \]
  
  \[
  \gamma_1 = \log_g \alpha + x\beta_1 \quad \text{and} \quad \gamma_2 = \log_g \alpha + x\beta_2
  \]
  
  \[
  \gamma_1 - x\beta_1 = \gamma_2 - x\beta_2
  \]
  
  \[
  x = (\gamma_1 - \gamma_2)/(\beta_1 - \beta_2)
  \]

- **Zero-knowledge.** Given \( g, h, \beta \), generate \( \gamma \stackrel{\$}{\leftarrow} \mathbb{Z}_p \) and set \( \alpha \leftarrow g^\gamma / h^\beta \)
  
  Has the same distribution as a real transcript, because \( \alpha \) is a uniformly random element of \( \mathbb{G} \)
Generalize...

\[ R = \{(g_1, \ldots, g_n, h_1, \ldots, h_n, x) | \forall i : g_i^x = h_i \} \subseteq \mathbb{G}_2^2 \times \mathbb{Z}_p \]

Protocol

- \( P \) picks \( r \leftarrow \mathbb{Z}_p \). Sets \( \alpha = (\alpha_1, \ldots, \alpha_n) \), where \( \alpha_i \leftarrow g_i^r \)
- \( V \) picks \( \beta \leftarrow \mathbb{Z}_p \)
- \( P \) sets \( \gamma \leftarrow r + \beta x \)
- \( V \) checks if \( g_i^\gamma = \alpha_i h_i^\beta \) for all \( i \)
Generalize more...

- Let $V \leq \mathbb{Z}_p^n$ (as vector spaces)
- Let $\dim V = k$ and $\phi : \mathbb{Z}_p^k \rightarrow \mathbb{Z}_p^n$ be a vector space isomorphism between $\mathbb{Z}_p^k$ and $V$
- Consider the following $R \subseteq \mathbb{G}_p^2 \times \mathbb{Z}_p^n$:

$$R = \{(x_1, \ldots, x_n) \mid (x_1, \ldots, x_n) \in V \land \forall i : g_i^x_i = h_i\}$$

**Protocol**

- $P$ picks $\vec{r} \leftarrow \mathbb{Z}_p^k$. Sets $\alpha = (\alpha_1, \ldots, \alpha_n)$, where $\alpha_i \leftarrow g_i^{s_i}$ and $\vec{s} = \phi(\vec{r})$
- $V$ picks $\beta \leftarrow \mathbb{Z}_p$
- $P$ sets $\gamma = (\gamma_1, \ldots, \gamma_n)$, where $\gamma_i \leftarrow s_i + \beta x_i$
- $V$ checks if $g_i^{\gamma_i} = \alpha_i h_i^\beta$ for all $i$, and if $\vec{\gamma} \in V
Generalize even more... 

- Let $V \leq \mathbb{Z}_p^{m \times n}$ (as vector spaces). Let $k$ and $\phi$ be as before.

$$R = \{((g_1, \ldots, g_n, h_1, \ldots, h_m), X) \mid X \in V \land \forall i : h_i = \prod_{j=1}^{n} g_j^{X_{ij}}\}$$

### Protocol

- $P$ picks $\vec{r} \leftarrow \mathbb{Z}_p^k$. Sets $S = \phi(\vec{r})$. Sets $\alpha = (\alpha_1, \ldots, \alpha_m)$, where $\alpha_i \leftarrow \prod_{j=1}^{n} g_j^{S_{ij}}$

- $V$ picks $\beta \leftarrow \mathbb{Z}_p$

- $P$ sets $\gamma = (\gamma_{1,1}, \ldots, \gamma_{m,n})$, where $\gamma_{i,j} \leftarrow S_{ij} + \beta X_{ij}$

- $V$ checks if $\prod_{j=1}^{n} g_j^{\gamma_{i,j}} = \alpha_i h_i^{\beta}$ for all $i$, and if $\gamma \in V$
Pedersen’s commitments
Commitments

- Cryptographic analogue to “a thing in locked box”

Methods

- **Commit.** \((c, d) \leftarrow \text{Com}(m).\) \([m\) is a message to be temporarily hidden]
  - \(m\) cannot be found from \(c\)
- **Open (or decommit).** \(0/1 \leftarrow \text{Open}(m, c, d)\)
  - Difficult to find \(c, m_1, d_1, m_2, d_2,\) such that \(m_1 \neq m_2,\) but \(\text{Open}(m_1, c, d_1) = \text{Open}(m_2, c, d_2) = 1\)

- Com creates the box \(c\) with the thing \(m\) inside. \(d\) is the key that opens it
- We think of the parties called “committer” and “verifier”
Pedersen’s commitments

- Let $g$ generate $\mathbb{G}$
- Let $h \in \mathbb{G}$ be another element, such that nobody knows $\log_g h$.
- To commit $m \in \mathbb{Z}_p$, the committer randomly generates $r \in \mathbb{Z}_p$ and sends $g^m h^r$ to the verifier.
- To open the commitment, send $(m, r)$ to the verifier.
- The commitment is unconditionally hiding, because $g^m h^r$ is a random element of $\mathbb{G}$.
- The commitment is computationally binding, because the ability to open a commitment in two different ways allows to compute $\log_g h$.
- Commitments are additively homomorphic:

  $$g^{m_1} h^{r_1} \cdot g^{m_2} h^{r_2} = g^{m_1 + m_2} h^{r_1 + r_2}$$
Proving the knowledge of opening

- There is a commitment $c$. $P$ wants to prove that he knows how to open it
  - $P$ knows committed value $m$ and blinding exponent $r$

Protocol

- $P$ picks random $m', r'$, computes $c' = g^{m'} h^{r'}$, sends it to $V$
- $V$ picks $\beta \leftarrow \mathbb{Z}_p$, sends it to $P$
- Both compute $c'' \leftarrow c^\beta \cdot c'$
- $P$ opens $c''$ to $V$

...same, as showing the knowledge of a discrete logarithm
Proving the knowledge of many openings

- There are commitments \( c_1, \ldots, c_k \). \( P \) wants to prove that he knows how to open them all.
- \( V \) picks random values \( \zeta_1, \ldots, \zeta_k \). 
  - Or: sends a random seed. \( \zeta_1, \ldots, \zeta_k \) are generated from that seed.
- Both compute \( c' = \prod_{i=1}^{k} c_i^{\zeta_i} \).
- \( P \) proves that he knows how to open \( c' \).
- No longer a \( \Sigma \)-protocol (because it has four moves, or five if you count sending of \( c_1, \ldots, c_k \)).
- Special soundness still holds.
Multi-round arguments

- We have a protocol, where $P$ and $V$ exchange many messages
- Similarly to $\Sigma$-protocols:
  - $P$ sends the first and the last message
  - Each time, $V$ reacts by generating a random value and sending it to $P$
- ZK — given the instance and $V$’s challenges in all rounds, generate a transcript
- Soundness: by rewinding many times at different places, extract the witness
  - Total number of rew windings must be “small”
    - The “fork” must have only a polynomial number of prongs
- Fiat-Shamir heuristic is applicable
Special soundness of knowledge of many openings

- At point, where $V$ sends $(\zeta_1, \ldots, \zeta_k)$, rewind $(k - 1)$ times
  - So we have $(\zeta_{11}, \ldots, \zeta_{1k}), \ldots, (\zeta_{k1}, \ldots, \zeta_{kk})$
- At each of $k$ branches, where $V$ sends $\beta$, rewind once
  - So we have $\beta_{11}, \beta_{12}, \ldots, \beta_{k1}, \beta_{k2}$
- Using $\beta_{i1}, \beta_{i2}$, extract $m'_i$
  - It is equal to $\zeta_{i1} m_1 + \cdots + \zeta_{ik} m_k$
- With $k$ linear equations for $m_1, \ldots, m_k$, find them
Commit-and-prove
Committed computations

- There is a function $f: \mathbb{Z}_p^n \rightarrow \mathbb{Z}_p$, given by its circuit
- $P$ has created the commitments $c_1, \ldots, c_n, c_\bullet$, sent them to $V$
- $P$ wants to show that he knows $x_1, r_1, \ldots, x_n, r_n, y, r_\bullet$, such that
  - $c_i = g^{x_i} h^{r_i}$
  - $c_\bullet = g^y h^{r_\bullet}$
  - $y = f(x_1, \ldots, x_n)$

The $\Sigma$-protocol

- $P$ commits to the outputs of all intermediate gates
- $P$ proves that he knows what has been committed
- In parallel for each gate: $P$ proves that the committed inputs and output of the gate are in the correct relationship

December 2020
Proofs for gates computing linear combinations

Task

- $P$ and $V$ know $c_1, \ldots, c_n$. For each $i$, $P$ knows $x_i, r_i$, s.t. $c_i = g^{x_i}h^{r_i}$
- There are $s_1, \ldots, s_n \in \mathbb{Z}_p$. Both $P$ and $V$ know them
- $P$ wants to prove to $V$ that $\sum_i s_i x_i = 0$

Reduce to “discrete logarithm”

- Let $u = \prod_i c_i^{s_i}$
- $P$ proves to $V$ that he knows $\log_h u$
  - ...which is equal to $\sum_i s_i r_i$
Proof for multiplication gate

Task

- Let $P$ and $V$ know $c_1, c_2, c_3$. Let $P$ know $x_i, r_i$, such that $c_i = g^{x_i} h^{r_i}$
- $P$ wants to prove to $V$ that $x_1 x_2 = x_3$

Reduce to “subspace discrete logarithm”

\[
\begin{align*}
  g_1 &= g \\
  g_2 &= h \\
  g_3 &= c_1 \\
  h_1 &= c_2 \\
  h_2 &= c_1^{x_2} \cdot h^s \\
  h_3 &= h_2 / c_3
\end{align*}
\]

- $P$ picks $s \leftarrow \mathbb{Z}_p$, sends $h_2$ to $V$, both compute $h_3$
- $P$ shows knowledge of $s_1, s_2, s_3, s_4 \in \mathbb{Z}_p$, such that

\[
\begin{align*}
  h_1 &= g_1^{s_1} g_2^{s_2} \\
  h_2 &= g_3^{s_1} g_2^{s_3} \\
  h_3 &= g_2^{s_4}
\end{align*}
\]
Proof for multiplication gate

Task

- Let $P$ and $V$ know $c_1, c_2, c_3$. Let $P$ know $x_i, r_i$, such that $c_i = g^{x_i} h^{r_i}$
- $P$ wants to prove to $V$ that $x_1 x_2 = x_3$

Reduce to “subspace discrete logarithm”

\[
\begin{align*}
g_1 &= g \\
g_2 &= h \\
g_3 &= c_1 \\
h_1 &= c_2 \\
h_2 &= c_1^{x_2} \cdot h^s \\
h_3 &= h_2 / c_3
\end{align*}
\]

Coefficients generated by

<table>
<thead>
<tr>
<th></th>
<th>$g_1 h_1$</th>
<th>$g_1 h_2$</th>
<th>$g_1 h_3$</th>
<th>$g_2 h_1$</th>
<th>$g_2 h_2$</th>
<th>$g_2 h_3$</th>
<th>$g_3 h_1$</th>
<th>$g_3 h_2$</th>
<th>$g_3 h_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

December 2020
Combining ς-protocols
Proving a disjunction (1/3)

- Suppose there are two relations $R_0, R_1$ and $\Sigma$-protocols $(A_i, R_i, V_i)$ for them
  - In both protocols, let challenge $\beta$ come from some field $F$
  - Let the protocols be secure, i.e. there are extractors $\text{Extr}_i$ and simulators $\text{Sim}_i$
- Let $P$ and $V$ have instances $x_0, x_1$
- Let $P$ have a single witness $w_h, h \in \{0, 1\}$, s.t. $(x_h, w_h) \in R_h$
- $P$ wants to prove to $V$ that he knows $w_h$
- $P$ does not want to reveal the value of $h$
Proving a disjunction (2/3)

- \( P \) randomly generates \( \beta_{1-h} \leftarrow \mathbb{F} \)
- \( P \) computes \( (\alpha_{1-h}, \gamma_{1-h}) \) by running \( \text{Sim}_{1-h}(x_{1-h}, \beta_{1-h}) \)
- \( P \) computes \( \alpha_h \) by running \( A_h(x_h, w_h) \)
- \( P \rightarrow V : \alpha_0, \alpha_1 \)
- \( V \rightarrow P : \beta \)
- \( P \) computes \( \beta_h = \beta - \beta_{1-h} \)
- \( P \) computes \( \gamma_h \) by running \( R_h(x_h, w_h, \alpha_h, \beta_h) \)
- \( P \rightarrow V : \beta_0, \beta_1, \gamma_0, \gamma_1 \)
- \( V \) checks both claims, using \( V_i(x_i, \alpha_i, \beta_i, \gamma_i) \). Also checks \( \beta_0 + \beta_1 = \beta \)

Completeness

Yes
Proving a disjunction (3/3)

Zero-knowledge

- Simulator gets $x_0$, $x_1$, $\beta$
- Picks $\beta_0, \beta_1$, such that $\beta_0 + \beta_1 = \beta$. Runs $\text{Sim}_0(x_0, \beta_0)$ and $\text{Sim}_1(x_1, \beta_1)$

Special soundness

- Forking transcript:
  $$\alpha_0, \alpha_1, \beta, \beta_0, \beta_1, \gamma_0, \gamma_1, \beta', \beta'_0, \beta'_1, \gamma'_0, \gamma'_1$$
- $\exists i \in \{0, 1\},$ such that $\beta_i = \beta'_i$ and $\gamma_i = \gamma'_i$
- $h = 1 - i$
- Use $\text{Extr}_i(x_h, \alpha_h, \beta_h, \gamma_h, \beta'_h, \gamma'_h)$ to find $w_h$
Thresholds

- $P$ and $V$ have $x_1, \ldots, x_n$. Prover has $\{w_i\}_{i \in I}$, where $I \subseteq \{1, \ldots, n\}$, $|I| = k$, $I$ is private
- $P$ wants to show that he has witnesses for at least $k$ of $x_1, \ldots, x_n$
- $P$ randomly chooses $\beta_j \in \mathbb{F}$ for all $j \notin I$, simulates $\alpha_j$, $\gamma_j$.
- $P$ picks $\alpha_j$ for $j \in I$ as needed. Sends $\alpha = (\alpha_1, \ldots, \alpha_n)$ to $V$
- $V$ responds with $\beta \in \mathbb{F}$.
- $P$ picks polynomial $f$ so, that $f(0) = \beta$, $f(j) = \beta_j$ for all $j \notin I$ and $\deg f \leq n - k$
- $P$ defines $\beta_i = f(i)$ and computes the response $\gamma_i$ for all $i \in I$
- $P$ sends $\gamma = (f, \gamma_1, \ldots, \gamma_n)$ to $V$
- $V$ checks $\deg f$ and $f(0)$, recomputes $\beta_i$, checks $\gamma_i$ for all $i$

Exercise. The three properties?
Exercise. A circuit of threshold gates?
Universally composable zero-knowledge proofs
Trapdoor commitments

- A commitment scheme has two methods — “commit” and “open”
- A third one as well — “initialize”. Returns public parameters
- In a trapdoor commitment scheme, initialization also returns a secret key
  - …but no party receives it
- $sk$ allows to create fake commitments
  - Indistinguishable from real commitments (if do not know $sk$)
  - Can be opened as any value
- Pedersen’s commitments have the trapdoor $\log_g h$
Ω-protocols

Like Σ-protocols, but...

- There is a common reference string (CRS) σ
  - Additional input to all steps of the protocol
- A simulator can generate σ together with a trapdoor τ
- If there exist two accepting conversations \((α, β, γ)\) and \((α, β', γ')\) for some \(x\), then can find \(w\) from \(τ\), and a single conversation \((α, β, γ)\)

From Σ-protocol to Ω-protocol

- σ is the public key for an asymmetric encryption scheme
- \(P\) sends \(e \leftarrow \mathcal{E}_σ(w)\) to \(V\) (as part of \(α\))
- \(P\) proves that exists \(w\), such that \(e\) encrypts \(w\) and \((x, w) \in R\)
UC ZK

- Need an $\Omega$-protocol and a trapdoor commitment scheme
- There are $x, w, (\sigma_\Omega, \sigma_{TC})$ (latter output by $F_{CRS}$)
- $P$ constructs $\alpha$. Let $(com, dec) \leftarrow \text{commit}(\alpha)$
- $P$ sends $com$ to $V$
- $V$ generates and sends $\beta$ to $P$
- $P$ constructs $\gamma$. Sends $\alpha, \gamma, \text{dec}$ to $V$
- $V$ verifies the commitment and the transcript $(\alpha, \beta, \gamma)$

Exercise: do the simulators

- For corrupt prover, must use $\tau_\Omega$
- For corrupt verifier, must use $\tau_{TC}$
ZK from MPC techniques
Garbled circuits

- $V$ becomes the garbler for the circuit for $R$
  - Outputs “0” and “1” have secret encodings
- $V$ and $P$ run OT protocols for $P$ to learn the keys corresponding to the bits of $w$
- $V$ sends the keys corresponding to the bits of $x$ to $P$
- $P$ evaluates the circuit and obtains the result $r$; commits to it
- $V$ sends all keys to $P$; $P$ checks that the circuit was correctly garbled
  - ZK is a variant of 2PC, where $V$ has no secrets
- $P$ opens the commitment of $r$ to $V$
“MPC in the head”

- Consider the computation \( g(x; w_1, \ldots, w_n) = R(x, w_1 + \cdots + w_n) \)
- Let \( \Pi \) be an MPC protocol for \( g \) that tolerates semi-honest coalitions of size 2
- \( P \), with \( x, w \), selects \( w_1, \ldots w_n \) that add up to \( w \), and plays \( \Pi \)
- \( P \) commits to the views of all parties and sends them to \( V \)
- \( V \) asks \( P \) to open the views of two parties
- \( V \) accepts if these parties received “1” and their views are consistent with each other
“MPC in the head”

- Consider the computation \( g(x; w_1, \ldots, w_n) = R(x, w_1 + \cdots + w_n) \)
- Let \( \Pi \) be an MPC protocol for \( g \) that tolerates semi-honest coalitions of size 2
- \( P \), with \( x, w \), selects \( w_1, \ldots w_n \) that add up to \( w \), and plays \( \Pi \)
- \( P \) commits to the views of all parties and sends them to \( V \)
- \( V \) asks \( P \) to open the views of two parties
- \( V \) accepts if these parties received “1” and their views are consistent with each other
- If \( \Pi \) tolerates malicious coalitions of size \( t = \Theta(n) \), then
  - \( V \) gets the views of \( t \) parties
  - The soundness error of the protocol is negligible
MPC protocols for MPC-in-the-head

- "Normally", MPC protocols consist of two kinds of operations:
  - Computations by a single party
  - The two-party operation \((x, \bot) \mapsto (\bot, x)\)
    - I.e. send a message
- After opening, \(V\) checks that the two parties have done both kinds of operations correctly
- MPC-in-the-head can handle any two-party operation \((x, y) \mapsto (f(x, y), g(x, y))\) equally well
  - E.g. \(((x, r), y) \mapsto (\bot, xy - r)\)
    - Called oblivious linear evaluation
- Privacy properties are still important to establish
  - In example above, 2nd party only learns \(xy - r\). Does not learn \(x\)
A 3-party MPC-in-the-head protocol

- There’s a ring $R$. Private values are additively shared
- Addition: every party by himself
- Multiplication of $[u] = ([u]_1, [u]_2, [u]_3)$ and $[v] = ([v]_1, [v]_2, [v]_3)$:
  - $[u]_i \cdot [v]_i$ is computed by the $i$-th party
  - A secret-sharing of $[u]_i \cdot [v]_j$ is computed as follows:
    - $P_i$ generates a random $r \in R$
    - $P_i$ and $P_j$ perform $(([u]_i, r), [v]_j) \mapsto (\bot, [u]_i \cdot [v]_j - r)$
    - $P_j$ uses obtained value as his share. $P_i$ uses $r$
- Each party adds up the shares of the products of components
- The joint view of any two parties is random
  - Whenever one of the interacts with the 3rd party, it either gets nothing, or something masked with fresh randomness
Sum-Check
Verifiable computation

- A computation $C$, given e.g. as an arithm. circuit over a field $\mathbb{F}$
- Parties: prover $P$ and verifier $V$
- Both know the input $\vec{x}$ to $C$, and the corresponding output $y$
- Prover wants to convince the verifier that indeed $C(\vec{x}) = y$
- Optimize
  - Verifier’s computation and “access to resources”
  - Prover’s computation (beyond computing $C$)
  - Communication

Completeness. Protocol convinces the verifier

Soundness. If $C(\vec{x}) \neq y$, then verifier cannot be convinced
  - Except for a small soundness error
Facts about polynomials over $\mathbb{F}$

Univariate
- A non-zero polynomial of degree at most $d$ has at most $d$ roots
  - Two polynomials of degree at most $d$ that agree on at least $(d + 1)$ points, are equal
  - To test whether $f \equiv 0$, evaluate $f(r)$ on a random $r \in \mathbb{F}$
    - Error: at most $(\deg f)/|\mathbb{F}|$
- If $f(c) = 0$, then $(X - c)$ divides $f(X)$

Multivariate
- We can speak about total degree and individual degree in a particular variable
- Schwartz-Zippel lemma: (see Wikipedia for proof)
  - Let $f$ be non-zero $n$-variate polynomial of total degree $\leq d$
  - Let $S \subseteq \mathbb{F}$
  - Pick $v_1, \ldots, v_n$ uniformly randomly from $S$
  - Then $\Pr[f(v_1, \ldots, v_n) = 0] \leq d/|S|$
Let $f \in \mathbb{F}[X_1, \ldots, X_n]$, with $\deg_{X_i} f \leq d_i$ (for each $i$)

Let $B \subseteq \mathbb{F}$ be a “small” set (e.g. $\{0, 1\}$). Let $z \in \mathbb{F}$

**Sum-Check**: a verifiable computation protocol for

$$
z \overset{?}{=} \sum_{v_1 \in B} \sum_{v_2 \in B} \cdots \sum_{v_n \in B} f(v_1, v_2, \ldots, v_n).
$$
Sum-Check protocol

- $P$ sends $f_1(X) = \sum_{v_2 \in B} \cdots \sum_{v_n \in B} f(X, v_2, \ldots, v_n)$ to $V$
  - i.e. sends the coefficients of the polynomial $f_1$
- $V$ checks that $z = \sum_{v \in B} f_1(v)$
- $V$ randomly picks $r_1 \in \mathbb{F}$, sends it to $P$
- $P$ and $V$ use Sum-Check to verify that

$$f_1(r_1) = \sum_{v_2 \in B} \cdots \sum_{v_n \in B} f(r_1, v_2, \ldots, v_n).$$
Sum-Check protocol

- $P$ sends $f_1(X) = \sum_{v_2 \in B} \cdots \sum_{v_n \in B} f(X, v_2, \ldots, v_n)$ to $V$
  - i.e. sends the coefficients of the polynomial $f_1$
- $V$ checks that $z = \sum_{v \in B} f_1(v)$
- $V$ randomly picks $r_1 \in \mathbb{F}$, sends it to $P$
- $P$ and $V$ use Sum-Check to verify that
  $$f_1(r_1) = \sum_{v_2 \in B} \cdots \sum_{v_n \in B} f(r_1, v_2, \ldots, v_n).$$

**Base of the recursion** ($V$ evaluates $f$ only here)

- $P$ sends $f_n(X) = f(r_1, \ldots, r_{n-1}, X)$ to $V$
- $V$ checks that $f_{n-1}(r_{n-1}) = \sum_{v \in B} f_n(v)$
- $V$ randomly picks $r_n \in \mathbb{F}$, checks that $f_n(r_n) = f(r_1, \ldots, r_n)$
Description without recursion

- At the beginning: define $z_0 := z$
- Do $n$ rounds. In the $i$-th round:
  - $P \rightarrow V : f_i(X) = \sum_{v_{i+1} \in B} \cdots \sum_{v_n \in B} f(r_1, \ldots, r_{i-1}, X, v_{i+1}, \ldots, v_n)$
  - $V$ checks that $z_{i-1} = \sum_{v \in B} f_i(v)$
  - $V \rightarrow P : r_i \leftarrow \mathcal{F}$
  - Define $z_i := f_i(r_i)$
- $V$ checks that $z_n = f(r_1, \ldots, r_n)$
  - ...the only place where $V$ evaluates $f$
**Example**

\[ 4 \equiv \sum_{v_1 \in \{0,1\}} \cdots \sum_{v_5 \in \{0,1\}} v_1 v_2 + 3v_3 + v_1 v_4 - v_2 v_5 + 2v_1 v_2 v_3 v_5 - 4v_4 v_5 + 12 \pmod{17} \]

<table>
<thead>
<tr>
<th>(i)</th>
<th>(f_i)</th>
<th>(?)</th>
<th>(r_i)</th>
<th>(z_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>N/A</td>
<td>N/A</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>9 + 3X</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>(\ldots + \ldots X)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>(\ldots + \ldots X)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>(\ldots + \ldots X)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>(\ldots + \ldots X)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Check that \(z_5 = f(r_1, r_2, r_3, r_4, r_5)\)
Soundness of Sum-Check

**Theorem.** The soundness error of Sum-Check is \( \leq \frac{d_1 + \cdots + d_n}{|\mathbb{F}|} \)

**Induction base**

Check of \( f_n(X) \overset{?}{=} f(r_1, \ldots, r_{n-1}, X) \) errs with prob. \( \leq d_n / |\mathbb{F}| \)

**Induction step**

- Suppose \( f_1 \) sent by \( P \) is wrong. Let \( \bar{f}_1 \) be correct
- \( f_1(r_1) = \bar{f}_1(r_1) \) with prob. \( \leq \frac{d_1}{|\mathbb{F}|} \)
  - First source of soundness error
- If \( f_1(r_1) \neq \bar{f}_1(r_1) \), then recursive Sum-Check passes with probability \( \leq \frac{d_2 + \cdots + d_n}{|\mathbb{F}|} \)
  - Second source of soundness error
GKR protocol
Goldwasser-Kalai-Rothblum protocol

- Computation \( C \) given by an arithmetic circuit over field \( \mathbb{F} \). Depth \( d \)
  - Addition and multiplication nodes of fan-in 2
  - Circuit is \textit{layered}, wires go from one layer to the next
    - Outputs at layer 0. Inputs at layer \( d \)
  - Verifier “understands” circuit without reading it all
    - There’s a uniform description of gates and wires
  - Let number of gates at layer \( i \) be between \( 2^{k_i - 1} + 1 \) and \( 2^{k_i} \)
  - For each \( i \in \{1, \ldots, d\} \), wiring is described by
    \[ \text{add}_i, \text{mult}_i : \{0, 1\}^{k_i - 1} \times (\{0, 1\}^{k_i})^2 \rightarrow \{0, 1\} \]
    - \( \text{add}_i(\vec{a}, \vec{b}, \vec{c}) = 1 \) means that \( \vec{a} \)-th gate on \((i - 1)\)-st layer
      - ...is an addition gate
      - ...gets its inputs from \( \vec{b} \)-th and \( \vec{c} \)-th gates on \( i \)-th layer
    (similar for \( \text{mult}_i \))
  - \( V \) has descriptions of all \( \text{add}_i, \text{mult}_i \)
Example circuit and functions $\text{add}_i$, $\text{mult}_i$

![Circuit Diagram]

- **Layer 0**
  - $\text{add}_1 = \{(1,10,11)\}$
  - $\text{mult}_1 = \{(0,00,01)\}$

- **Layer 1**
  - $\text{add}_2 = \{(10,01,11), (11,00,11)\}$
  - $\text{mult}_2 = \{(00,00,10), (01,01,11)\}$

- **Layer 2**
  - $\text{add}_3 = \{(00,00,10), (11,01,10)\}$
  - $\text{mult}_3 = \{(01,00,01), (10,00,10)\}$
Assignment of values to gates

- For \( i \in \{0, \ldots, d\} \), let \( W_i : \{0, 1\}^{k_i} \to \mathbb{F} \) give the values at the gates in \( i \)-th layer.
- \( V \) has \( W_0 \) and \( W_d \).
- Each layer is computed from the next: \( \forall i \in \{1, \ldots, d\} : \)

\[
W_{i-1}(\vec{a}) = \sum_{\vec{b}, \vec{c} \in \{0, 1\}^{k_i}} \left( \text{add}_i(\vec{a}, \vec{b}, \vec{c}) \cdot (W_i(\vec{b}) + W_i(\vec{c})) + \right.
\]
\[
\left. \quad \text{mult}_i(\vec{a}, \vec{b}, \vec{c}) \cdot (W_i(\vec{b}) \cdot W_i(\vec{c})) \right)
\]
Multilinear extensions

Theorem

Let $f : \{0, 1\}^k \rightarrow \mathbb{F}$. There is a unique multilinear $\tilde{f} : \mathbb{F}^k \rightarrow \mathbb{F}$ that extends $f$.

Existence

- If $k = 0$ then $f$ is constant function. Take $\tilde{f} = f$
- $\tilde{f} = \tilde{f}_{X_1=0} + X_1 \cdot \tilde{f}_{X_1=1}$. Multilinear by induction
Multilinear extensions

Uniqueness

- let $h : \mathbb{F}^k \to \mathbb{F}$ be multilinear, let it be non-zero
- Let $M = cX_{i_1}X_{i_2} \cdots X_{i_n}$ be its monomial of minimal degree
- Consider the point $\vec{x} \in \{0, 1\}^k$, $x_i = 1$ iff $i \in \{i_1, \ldots, i_n\}$
- Then $h(\vec{x}) = c \neq 0$
  - All other monomials except $M$ contain other variables beside $X_{i_1}, \ldots, X_{i_n}$, hence become 0 at $\vec{x}$
- If $g_1, g_2 : \mathbb{F}^k \to \mathbb{F}$ are two multilinear extensions of $f$, then:
  - $(g_1 - g_2)$ is also multilinear
  - $(g_1 - g_2)$ is zero on all points in $\{0, 1\}^k$.
- Hence $(g_1 - g_2)$ is zero everywhere in $\mathbb{F}$, i.e. $g_1 = g_2$, i.e. the multilinear extension is unique
Alternative proof of being zero everywhere

Inducution (step) over number of variables

- $h(r_1, \ldots, r_k)$ can be computed from $h(r_1, \ldots, r_{k-1}, 0)$ and $h(r_1, \ldots, r_{k-1}, 1)$
  - Using interpolation. Because $\deg_{X_k} h \leq 1$

- $h(X_1, \ldots, X_{k-1}, b)$ (where $b \in \{0, 1\}$) is:
  - multilinear in $k - 1$ variables
  - zero on the hypercube

  hence zero everywhere, by the induction assumption

- Hence $h(r_1, \ldots, r_k) = 0$, by interpolation

higher-degree polynomials and larger sets

Let $H \subseteq \mathbb{F}$. Any function $H^k \rightarrow \mathbb{F}$ can be uniquely extended to a polynomial $\mathbb{F}^k \rightarrow \mathbb{F}$, where each individual degree is $< |H|$
Evaluating multilinear extensions

Let $\vec{x} \in \mathbb{F}^k$

$$\tilde{f}(\vec{x}) = \sum_{\vec{w} \in \{0,1\}^k} f(\vec{w}) \cdot \chi_{\vec{w}}(\vec{x})$$

$$\chi_{\vec{w}}(\vec{x}) := \prod_{i=1}^{k} (w_i ? x_i : (1 - x_i))$$

- Given the values of $f$ on the whole $\{0,1\}^k$, the value of any $\tilde{f}(\vec{x})$ can be computed in time $O(2^k)$
- If $f$ is sparse, then $\tilde{f}(\vec{x})$ can be computed in time proportional to support of $f$
Assignment of values to gates

**Theorem**

\[ \widetilde{W}_{i-1}(\vec{X}) = \sum_{\vec{b}, \vec{c} \in \{0,1\}^{k_i}} \left( \widetilde{\text{add}}_i(\vec{X}, \vec{b}, \vec{c}) \cdot (\widetilde{W}_i(\vec{b}) + \widetilde{W}_i(\vec{c})) + \right) \]

\[ \widetilde{\text{mult}}_i(\vec{X}, \vec{b}, \vec{c}) \cdot (\widetilde{W}_i(\vec{b}) \cdot \widetilde{W}_i(\vec{c})) \]

**Proof.**

- Polynomials at both sides of the \(=\)-sign are multilinear
- These polynomials agree at the set \(\{0,1\}^{k_i-1}\)
GKR protocol

- Verifies equation on previous slide (for V’s $W_0$ and $W_d$)
- Suppose there is some $i$, a random $\vec{r}_{i-1} \in \mathbb{F}^{k_{i-1}}$, and $w_{i-1} \in \mathbb{F}$
  - Verifier believes $\widehat{W}_{i-1}(\vec{r}_{i-1}) = w_{i-1}$ (knows it for $i = 1$)
- Sum-Check this:
  \[
  w_{i-1} = \sum_{\vec{b}, \vec{c}} \left( \widehat{\text{add}}_i(\vec{r}_{i-1}, \vec{b}, \vec{c}) \cdot (\widehat{W}_i(\vec{b}) + \widehat{W}_i(\vec{c})) + \widehat{\text{mult}}_i(\vec{r}_{i-1}, \vec{b}, \vec{c}) \cdot (\widehat{W}_i(\vec{b}) \cdot \widehat{W}_i(\vec{c})) \right)
  \]

- At the end of sum-check, for some random $\vec{s}_i, \vec{t}_i \in \mathbb{F}^{k_i}$, V needs to compute
  \[
  \widehat{\text{add}}_i(\vec{r}_{i-1}, \vec{s}_i, \vec{t}_i), \widehat{\text{mult}}_i(\vec{r}_{i-1}, \vec{s}_i, \vec{t}_i), \widehat{W}_i(\vec{s}_i), \widehat{W}_i(\vec{t}_i)
  \]
- First two are OK. But V does not know $\widehat{W}_i$ (except when $i = d$)
GKR protocol

- Define $\ell_i : \mathbb{F} \rightarrow \mathbb{F}^{k_i}$ by $\ell_i(x) = \vec{s}_i + x \cdot (\vec{t}_i - \vec{s}_i)$
- $q_i := \tilde{W}_i \circ \ell_i$ is a polynomial $\mathbb{F} \rightarrow \mathbb{F}$, deg $q_i \leq k_i$. $P$ tells it to $V$
- $V$ completes the Sum-Check, taking $\tilde{W}_i(\vec{s}_i) = q_i(0)$ and $\tilde{W}_i(\vec{t}_i) = q_i(1)$
- $V$ picks a random $r_i^\# \in \mathbb{F}$, defines $\vec{r}_i = \ell_i(r_i^\#)$ and $w_i = q_i(r_i^\#)$. Goes to next round
  - That’s like $V$ checking whether $q_i = \tilde{W}_i \circ \ell_i$, where $\tilde{W}_i$ is given by the Theorem above
- At the end of $d$-th round:
  - $P$ still defines $q_d$ and sends to $V$
  - $V$ still takes $\tilde{W}_d(\vec{s}_d) = q_d(0)$ and $\tilde{W}_d(\vec{t}_d) = q_d(1)$
  - $V$ picks a random $r_d^\# \in \mathbb{F}$, checks if $q_i(r_d^\#) = \tilde{W}_d(\ell_d(r_d^\#))$

so $V$ evaluates $\tilde{W}_d$ only once
Soundness

Cheating probability at $i$-th round

- Sum-Check: $\frac{2k_i}{|F|}$
- Comparison of $q_i$ and $\tilde{W}_i \circ \ell_i$: $\frac{k_i}{|F|}$
  - Also present for $\tilde{W}_0$
- $k_i$ is $O(\log |C|)$

Total soundness error: $O(d \log |C|/|F|)$
Zero-knowledge GKR

- Do everything with Pedersen’s commitments, i.e.:
  - There is an arithmetic circuit $C : \mathbb{Z}_p^n \times \mathbb{Z}_p^m \rightarrow \mathbb{Z}_p$ expressing the relation
    - Instance length: $n$. Witness length: $m$
    - Accept, if output is e.g. 0
  - Both compute the commitments to instance and output
  - Prover commits to witness
  - Verifier does all its computations in the GKR product with the commitments it has
    - If multiplication or equality check is necessary, then Prover helps
Costs of the GKR protocol (for verifier)

- In round $i$, does Sum-Check for $2k_i$-variate polynomial with individual degrees $\leq 2$
  - $2k_i$ rounds, 3 elements sent per round, $V$ computes a linear combination of them, checks equality
- At the end of the round, $V$
  - Gets $(k_i + 1)$ elements, computes two linear combinations of them
  - Does a multiplication, (a constant-size linear combination,) and an equality check
- At the last round, $V$ evaluates $\tilde{W}_d$ on a random point
Costs of the GKR protocol (for verifier)

- In round $i$, does Sum-Check for $2k_i$-variate polynomial with individual degrees $\leq 2$
  - $2k_i$ rounds, 3 elements sent per round, $V$ computes a linear combination of them, checks equality
- At the end of the round, $V$
  - Gets $(k_i + 1)$ elements, computes two linear combinations of them
  - Does a multiplication, (a constant-size linear combination,) and an equality check
- At the last round, $V$ evaluates $\tilde{W}_d$ on a random point
- So, $V$ has to do some work for each layer of the circuit:
  - For all but the input layer, the cost is logarithmic to the size of the layer
  - For the input layer, the cost is proportional to the size (of the witness)
- $V$ only needs $\tilde{W}_d$ evaluated on a single point. Could this be more efficient?
Evaluating $\tilde{\text{add}}_i$ and $\tilde{\text{mult}}_i$

- Verifier has to evaluate $\tilde{\text{add}}_i$ and $\tilde{\text{mult}}_i$ on random points
  - Generally, the costs for this are proportional to the number of gates in layer $(i - 1)$
- For faster evaluation, need regularity in the circuit. For example
  - Many identical circuits running in parallel (+ some pre- and postprocessing)
  - Computable by read-once ordered binary decision diagram
- For certain useful circuits, faster ways of evaluating $\tilde{\text{add}}_i$ and $\tilde{\text{mult}}_i$ are known, e.g.
  - Fast Fourier Transform
  - $\text{add}$ and $\text{mult}$ for certain universal circuits
Privacy from extra randomness
Polynomial commitments

- $P$ has a polynomial $f$ of degree $\leq d$
- $P$ sends some value $c$ to $V$
- Later, $V$ sends an element $x \in \mathbb{F}$ to $P$
- $P$ sends some $y \in \mathbb{F}$ and some opening information to $V$
  - Or perhaps they will run a longer protocol
- $V$ becomes convinced that
  - $P$ had in mind a polynomial $f'$ of degree $\leq d$, when it prepared $c$
  - $f'(x) = y$
- Zero-knowledge may or may not be required
Polynomial commitments

- $P$ has a polynomial $f$ of degree $\leq d$
- $P$ sends some value $c$ to $V$
- Later, $V$ sends an element $x \in \mathbb{F}$ to $P$
- $P$ sends some $y \in \mathbb{F}$ and some opening information to $V$
  - Or perhaps they will run a longer protocol
- $V$ becomes convinced that
  - $P$ had in mind a polynomial $f'$ of degree $\leq d$, when it prepared $c$
  - $f'(x) = y$
- Zero-knowledge may or may not be required
- The two convictions can be separate protocols; second one may be executed repeatedly
- This is a possibility for evaluating $\widetilde{\text{add}}_i$, $\widetilde{\text{mult}}_i$
Sum-Check with privacy (1/4)

- To compute

\[ S = \sum_{v_1 \in \{0,1\}} \cdots \sum_{v_n \in \{0,1\}} f(v_1, \ldots, v_n) \]

we considered a multilinear extension of \( f \)

- We want to make private the values of \( f \) on the hypercube (except for \( S \))
  - Still, there’s some commitment to \( f \)

- In Sum-Check,
  1. \( P \) sent to \( V \) (linear) polynomials
     \[ f_i(X) = \sum_{v_{i+1}} \cdots \sum_{v_n} \tilde{f}(r_1, \ldots, r_{i-1}, X, v_{i+1}, \ldots, v_n) \]
  2. \( V \) itself computed \( \tilde{f}(r_1, \ldots, r_n) \)

They both leak information about \( f \)
Sum-Check with privacy (2/4)

Fixing the 2nd leak

Do not commit to $\tilde{f}$. Instead, $P$ randomly picks $r_1 \in \mathbb{F}$ and commits to

$$\hat{f}(\vec{X}) := \tilde{f}(\vec{X}) + r_1 \cdot X_1(1 - X_1)$$

- For all $\vec{x} \in \{0, 1\}^n$: $\hat{f}(\vec{x}) = f(\vec{x})$
- For all $\vec{x} \in \mathbb{F}^n$, where $x_1 \not\in \{0, 1\}$: $\hat{f}(\vec{x})$ is independent of the values of $f$ on the hypercube

If some outer protocol (e.g. GKR) requires $\hat{f}$ to be evaluated in more than 1 point, then add more random terms
Sum-Check with privacy (3/4)

Fixing the 1st leak (1/2)

- $P$ commits to a random polynomial $p$ with the same individual degrees as $\hat{f}$
  - Commitment has to be ZK. It fixes the individual degrees of $p$
  - $P$ can take $p(\vec{X}) = p_1(X_1) + \cdots + p_n(X_n)$, where $p_i$ are random univariate polynomials of given individual degree
    - So $P$ separately commits to $p_1, \ldots, p_n$
- $P$ computes and sends to $V$

$$T = \sum_{v_1 \in \{0,1\}} \cdots \sum_{v_n \in \{0,1\}} p(v_1, \ldots, v_n)$$
Sum-Check with privacy (4/4)

Fixing the 1st leak (2/2)

- $V$ picks a random $\rho \in \mathbb{F}$ and sends it to $P$
- $P$ and $V$ run Sum-Check for $\rho \cdot \hat{f} + p$. The result must be $\rho S + T$
- In the end, when $V$ wants to evaluate $(\rho \cdot \hat{f} + p)(r_1, \ldots, r_n)$,
  - it gets the value of $p(r_1, \ldots, r_n)$ from the opening of the commitment
  - it gets the value of $\hat{f}(r_1, \ldots, r_n)$ “normally”
Polynomial commitments
Probabilistically checkable proofs (PCP)

- There’s a relation $R \subseteq \{0, 1\}^* \times \{0, 1\}^*$ with $R \in \mathbb{P}$
- $V$ knows $x$. $P$ knows $(x, w) \in R$, wants to convince $V$
- $P$ comes up with a proof string $\pi \in \Sigma^\ell$
  - $\Sigma$ is some proof alphabet. Typically, $\Sigma$ is $\mathbb{F}$
- $V$ gets *oracle access* to $\pi$
  - $i \mapsto \pi[i]$  
- $V$ looks at $x$ and makes oracle queries. Accepts or rejects
- Want: *completeness* and *soundness*
- Minimize: number of $V$’s queries & length of $\pi$
PCPs in cryptographic setting

- $P$ comes up with a proof string $\pi \in \Sigma^\ell$
- $P$ builds a Merkle tree on top of $\pi$, sends the root to $V$
- Whenever $V$ wants to get $\pi[i]$:  
  - $V$ sends $i$ to $P$
  - $P$ responds with $\pi[i]$ and the hash path
- Hence $\pi$ never has to be communicated
- But $P$ still has to materialize it
Low-degree tests

- Let $V$ have oracle access to some $f : \mathbb{F}^m \rightarrow \mathbb{F}$
  - E.g. as a proof string of length $|\mathbb{F}|^m$
- How can $V$ verify that $f$ is a polynomial of degree $\leq d$?
Low-degree tests

- Let $V$ have oracle access to some $f : \mathbb{F}^m \rightarrow \mathbb{F}$
  - E.g. as a proof string of length $|\mathbb{F}|^m$
- How can $V$ verify that $f$ is a polynomial of degree $\leq d$?

**Line vs. point test**

- Pick a line $\ell : \mathbb{F} \rightarrow \mathbb{F}^m$
- Check if $f \circ \ell$ is a polynomial of degree $\leq d$
  - Look at $|\mathbb{F}|$ points of the proof string
  - Or: let $P$ give that polynomial. Verify equality at a single point
    - That’s where the name comes from
- Does this give good confidence that $\deg f \leq d$?
Low-degree tests

- Let $V$ have oracle access to some $f : \mathbb{F}^m \rightarrow \mathbb{F}$
  - E.g. as a proof string of length $|\mathbb{F}|^m$
- How can $V$ verify that $f$ is a polynomial of degree $\leq d$?

Line vs. point test

- Pick a line $\ell : \mathbb{F} \rightarrow \mathbb{F}^m$
- Check if $f \circ \ell$ is a polynomial of degree $\leq d$
  - Look at $|\mathbb{F}|$ points of the proof string
  - Or: let $P$ give that polynomial. Verify equality at a single point
    - That’s where the name comes from
- Does this give good confidence that $\deg f \leq d$?
  - No. $f$ could differ from a low-degree polynomial in few places
  - But $f$ is close to a low-degree polynomial
“Differing in a few places”

- Let $\vec{a}, \vec{b}$ have equal length. Their relative Hamming distance is
  \[ \Delta(\vec{a}, \vec{b}) = \frac{|\{i \mid a_i \neq b_i\}|}{|\vec{a}|} \]

- For a set $B$ of vectors, define $\Delta(\vec{a}, B) = \min_{\vec{b} \in B} \Delta(\vec{a}, \vec{b})$

**Guarantee from line-vs-point test**

If $\Pr[\text{line-point test rejects}] \leq \delta$, then $\Delta(f, \mathbb{F}^{\leq d}[X_1, \ldots, X_m]) \leq \delta + m^{c_1} d^{c_2} / |\mathbb{F}|^{c_3}$, for some constants $c_1$, $c_2$, and $c_3$

**Guarantee from plane-vs-point test**

If $\Pr[\text{plane-point test rejects}] \leq \delta$, then $\Delta(f, \mathbb{F}^{\leq d}[X_1, \ldots, X_m]) \leq \delta + m^{c_1} (d / |\mathbb{F}|)^{c_2}$, for some constants $c_1$ and $c_2$
Combining low-degree tests

- Let $V$ have access to polynomials $f_1, \ldots, f_k : \mathbb{F} \to \mathbb{F}$
- Let $d_1, \ldots, d_k \in \mathbb{N}$. $V$ wants to verify that $\forall i : \deg f_i \leq d_i$
- Let $d = \max\{d_1, \ldots, d_k\}$
- $V$ generates random $r_1, \ldots, r_k \in \mathbb{F}$. Defines the polynomial

$$f(X) := \sum_{i=1}^{k} r_i \cdot X^{d-d_i} f_i(X)$$

- The nature of access to $f$ depends a lot on the nature of accesses to $f_i$
- One can always find $f(x)$ by finding all $f_i(x)$ and then combining
- Sometimes accesses to $f_i$ may be more homomorphic
- Check that $f$ has degree $\leq d$
Individual-degree testing

- $V$ wants to verify that $f : \mathbb{F}^m \to \mathbb{F}$ has degree $d$ in each variable

The test

- Test that $f$ has total degree $\leq dm$
- For each coordinate $i \in \{1, \ldots, m\}$
  - Select random $r_1, \ldots, r_{i-1}, r_{i+1}, \ldots, r_m \in \mathbb{F}$
  - i.e. select a random line parallel to the $i$-th axis
  - Test that $\deg f(r_1, \ldots, r_{i-1}, X, r_{i+1}, \ldots, r_m) \leq d$
PCP for CIRCUIT-SAT
Polynomial remainder theorem for multivariate polynomials

**Theorem**

Let \( f : \mathbb{F}^m \to \mathbb{F} \) and \( \vec{r} = (r_1, \ldots, r_m) \in \mathbb{F}^m \). Then

\[
f(\vec{r}) = 0 \iff \exists (g_1, \ldots, g_m : \mathbb{F}^m \to \mathbb{F}) : f(\vec{X}) = \sum_{i=1}^{m} (X_i - r_i) \cdot g_i(\vec{X})
\]

(Direction "\( \Leftarrow \)" is trivial. Direction "\( \Rightarrow \)": first show for \( \vec{r} = \vec{0} \), and then shift the variables)

- To show that \( f(\vec{r}) = v \), show that \( f(\vec{X}) - v \) has a root at \( \vec{r} \):
  - \( P \) commits to \( g_1, \ldots, g_m \)
  - \( V \) checks their degrees, and the equality of polynomials
A different proof for previous Theorem

- Let $f_0 := f$
- Define $f_1, \ldots, f_m : \mathbb{F}^m \rightarrow \mathbb{F}$ as reminders in polynomial division:
  \[
  f_{i-1}(\vec{X}) = \left( \sum_{i=1}^{m} (X_i - r_i) \cdot g_i(\vec{X}) \right) + f_i(\vec{X})
  \]
  where $g_i(\vec{X})$ is the quotient and $(X_i - r_i)$ is the divisor.
- $f_i$ has degree 0 in $X_1, \ldots, X_i$
  - Hence $f_m$ is a constant polynomial
- We have
  \[
  f(\vec{X}) = \left( \sum_{i=1}^{m} (X_i - r_i) \cdot g_i(\vec{X}) \right) + f_m(\vec{X})
  \]
- Left-hand side and the sum vanish at $\vec{X} \leftarrow \vec{r}$. Hence $f_m(\vec{r}) = 0$. Hence $f_m \equiv 0$. \(\square\)
Generalization to ranges

Let \( R \subseteq \mathbb{F} \). Define the vanishing polynomial \( Z_R : \mathbb{F} \to \mathbb{F} \) by

\[
Z_R(X) := \prod_{r \in R} (X - r).
\]

**Theorem**

Let \( f : \mathbb{F}^m \to \mathbb{F} \) and \( R_1, \ldots, R_m \subseteq \mathbb{F} \). Then

\[
\forall r_1 \in R_1, \ldots, r_m \in R_m : f(r_1, \ldots, r_m) = 0
\]

\[
\iff
\exists (g_1, \ldots, g_m : \mathbb{F}^m \to \mathbb{F}) : f(\vec{X}) = \sum_{i=1}^{m} Z_{R_i}(X_i) \cdot g_i(\vec{X})
\]

- **Proof**: as in previous slide
- Instead of \((X_i - r_i)\), we have \( Z_{R_i}(X_i) \)
- \( f_m \equiv 0 \), because
  - it has individual degree \(< |R_i| \) in variable \( X_i \), and
  - it equals 0 on the whole \( R_1 \times \cdots \times R_m \).
PCP for boolean circuit satisfiability

- Boolean circuit. $2^n$ gates (inputs+internals). One output
- Encode the circuit by the following $C : \{0, 1\}^{3n+3} \rightarrow \{0, 1\}$:
  $C(x, y, z, b_x, b_y, b_z) = 1$ iff the following things hold
  - Gate no. $z$ gets its inputs from gates no. $x$ and $y$
  - The output of gate no. $z$ on inputs $\neg b_x$ and $\neg b_y$ is $b_z$
  ($C$ is a public function, $\tilde{C} : \mathbb{F}^{3n+3} \rightarrow \mathbb{F}$ is public polynomial)
- Prover holds private assignment $A : \{0, 1\}^n \rightarrow \{0, 1\}$
- Prover commits to $\tilde{A} : \mathbb{F}^n \rightarrow \mathbb{F}$
- Prover also commits to the polynomial $D : \mathbb{F}^{3n+3} \rightarrow \mathbb{F}$:

$$D(x, y, z, b_x, b_y, b_z) := \tilde{C}(x, y, z, b_x, b_y, b_z) \cdot (\tilde{A}(x) - b_x) \cdot (\tilde{A}(y) - b_y) \cdot (\tilde{A}(z) - b_z)$$
The polynomial $D$

$C(\vec{x}, \vec{y}, \vec{z}, b_x, b_y, b_z) = 1$ iff the following things hold
- Gate no. $\vec{z}$ gets its inputs from gates no. $\vec{x}$ and $\vec{y}$
- The output of gate no. $\vec{z}$ on inputs $\neg b_x$ and $\neg b_y$ is $b_z$

$D(\vec{x}, \vec{y}, \vec{z}, b_x, b_y, b_z) := \tilde{C}(\vec{x}, \vec{y}, \vec{z}, b_x, b_y, b_z) \cdot (\tilde{A}(\vec{x}) - b_x) \cdot (\tilde{A}(\vec{y}) - b_y) \cdot (\tilde{A}(\vec{z}) - b_z)$

**Theorem**

*A is a valid assignment to the gates, iff $D$ is zero on the entire hypercube $\{0, 1\}^{3n+3}$.*

Proof: a simple case analysis
Zero-on-subcube test

- $P$ has committed to $f : \mathbb{F}^m \to \mathbb{F}$ of degree $\leq d$
- $H \subseteq \mathbb{F}$. $P$ wants to show that $f$ is zero on $H^m$.
- In this case $f(\vec{X}) = \sum_{i=1}^{m} Z_H(X_i) \cdot g_i(\vec{X})$
- $P$ computes $g_1, \ldots, g_m$ and commits to them, too
- $V$ checks that each $g_i$ has degree $\leq d - |H|$
- $V$ checks that the equality between polynomials holds
Proof string encodes $\tilde{A}$, $D$, $g_1, \ldots, g_n$

Verifier picks a random line $\ell : \mathbb{F} \to \mathbb{F}^{3n+3}$. Checks that

- All committed polynomials, when restricted to $\ell$ have appropriately bounded degrees;
- Zero-on-subcube equation for $H = \{0, 1\}$ is satisfied on all points of $\ell$;
- Definition of $D$ is satisfied on all points of $\ell$;
- $\tilde{A}$ assigns 1 to the output gate of the circuit

Note that we could use some other $H$ as the alphabet for indexing the gates. That would reduce $n$
ZK-PCP for CIRCUIT-SAT
Zero-knowledge?

- A number $\mu$ of values will be opened during these tests
- Try to encode the private values so, that the opening of any $\mu$ of them will not yet reveal the actual values
- A bit similar to having MPC-in-the-head
Zero-knowledge?

- A number $\mu$ of values will be opened during these tests
- Try to encode the private values so, that the opening of any $\mu$ of them will not yet reveal the actual values
- A bit similar to having MPC-in-the-head
- E.g. change the circuit. Instead of a each wire, have a bundle of them
  - The values on each bundle XOR together to the original value

Simulating Verifier’s view

- First, pick the line $\ell$
- Come up with the values of $\tilde{A}, D, g_1, \ldots, g_n'$ that satisfy the equations on the line $\ell$
  - Hopefully there’s enough freedom for that...I haven’t checked...
3CNF-SAT

- 3CNF: formula of the form \( \bigwedge_{i=1}^{n} (X_{i1}^{b_i} \lor X_{i2}^{b_i} \lor X_{i3}^{b_i}) \)
  - \( X_i \) — Boolean variables. \( b_i \in \{-1, 1\} \). \( X^1 := X \), \( X^{-1} := \neg X \)
  - CIRCUIT-SAT is simple to reduce to 3CNF-SAT
    - Introduce a variable for each wire
    - For each gate, add constraints stating that the output wire is the boolean operation applied to input wires
    - Whole circuit \( \iff \) conjunction of constraints

### Operations \( \rightarrow \) Constraints

<table>
<thead>
<tr>
<th>Operation</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>( y \leftarrow \neg x )</td>
<td>((x \lor y) \land (\bar{x} \lor \bar{y}))</td>
</tr>
<tr>
<td>( z \leftarrow x \lor y )</td>
<td>((\bar{x} \lor \bar{y} \lor z) \land (x \lor \bar{y} \lor z) \land (\bar{x} \lor y \lor z) \land (x \lor y \lor z))</td>
</tr>
<tr>
<td>( z \leftarrow x \land y )</td>
<td>((\bar{x} \lor \bar{y} \lor z) \land (x \lor \bar{y} \lor \bar{z}) \land (\bar{x} \lor y \lor \bar{z}) \land (x \lor y \lor \bar{z}))</td>
</tr>
</tbody>
</table>
Barrington’s transformation

Branching programs over a group $G$

- **Input:** a bit-string of length $n$
- **Program** $B$: a sequence of triples $[\iota_1, g_{1,0}, g_{1,1}; \iota_2, g_{2,0}, g_{2,1}; \ldots; \iota_m, g_{m,0}, g_{m,1}]$
  - $\iota_i \in \{1, \ldots, n\}$. $g_{ij} \in G$. $m$ — length of the program
- Defines a function $[B] : \{0, 1\}^n \rightarrow G$ by $[B](b_1 \cdots b_n) := g_1, b_{\iota_1} \cdot g_2, b_{\iota_2} \cdots g_m, b_{\iota_m}$.

Branching programs over $G$ with output $g \in G$

$B$, such that $[B](\{0, 1\}^n) \subseteq \{1, g\}$. Think of $g$ as “yes” and 1 as “no”

Theorem (D. A. Barrington)

*For any boolean circuit of depth $d$ with gates of fan-in 2, there exists an equivalent branching program over group $S_5$ of length $4^d$ with output $\langle$ some element of $S_5 \rangle$.***

December 2020
Permutation cycles

- An element of $S_5$ is something like $\left( \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \\ 3 & 5 & 4 & 1 & 2 \end{array} \right)$
- This element has two cycles: $1 \rightarrow 3 \rightarrow 4 \rightarrow 1$ and $2 \rightarrow 5 \rightarrow 2$
- An alternative way of writing: $(1\,3\,4)(2\,5)$
- The cycle type of a permutation is the count of its cycles of each possible length
- $\sigma \in S_5$ is a five-cycle if it consists of a single cycle (of length 5)
  - Five-cycles look like $(1\,x\,y\,z\,w)$, where $\{x, y, z, w\} = \{2, 3, 4, 5\}$
  - There are 24 five-cycles. Let their set be $C_5$
Permutation cycles

- An element of $S_5$ is something like $\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 5 & 4 & 1 & 2 \end{pmatrix}$
- This element has two cycles: $1 \rightarrow 3 \rightarrow 4 \rightarrow 1$ and $2 \rightarrow 5 \rightarrow 2$
- An alternative way of writing: $(1 \ 3 \ 4)(2 \ 5)$
- The cycle type of a permutation is the count of its cycles of each possible length
- $\sigma \in S_5$ is a five-cycle if it consists of a single cycle (of length 5)
  - Five-cycles look like $(1 \ x \ y \ z \ w)$, where $\{x, y, z, w\} = \{2, 3, 4, 5\}$
  - There are 24 five-cycles. Let their set be $C_5$
- Elements $g, g' \in G$ are conjugate, if $\exists h \in G : g' = h^{-1}gh$
- Theorem. Two elements of a symmetric group are conjugate iff they have the same cycle type
  - Suppose that $\sigma \in S_n$ contains a cycle $(x_1 \ x_2 \cdots x_k)$. Let $\tau \in S_n$. Then $\tau^{-1} \sigma \tau$ contains the cycle $(\tau(x_1) \tau(x_2) \cdots \tau(x_k))$
Proof of Barrington’s theorem (1/2)

- For any $\sigma \in C_5$, a circuit of depth 0 has an equivalent branching program of length 1 with output $\sigma$.

- Let $B$ be a branching program of length $d$ with output $\sigma \in C_5$. Let $\varsigma \in C_5$. There exists an equivalent branching program $B'$ of length $d$ with output $\varsigma$.
  - There exists $\rho \in S_5$, such that $\varsigma = \rho^{-1}\sigma\rho$. Replace any element $\tau$ in $B$ with $\rho^{-1}\tau\rho$.

- Let $B$ be a branching program of length $d$ with output $\sigma \in C_5$. There exists a branching program $B'$ (with output $\sigma^{-1}$) of length $d$ that accepts a bit-string iff $B$ rejects it.
  - Let $B$'s last step be $\langle \iota, \rho, \tau \rangle$. Let $B'$ be equal to $B$, except the last step is $\langle \iota, \rho\sigma^{-1}, \tau\sigma^{-1} \rangle$.

- There exist $\phi_1, \phi_2 \in C_5$, such that $\phi_1\phi_2\phi_1^{-1}\phi_2^{-1} \in C_5$.
  - Let $\phi_1 = (1 \ 2 \ 3 \ 4 \ 5)$ and $\phi_2 = (1 \ 3 \ 5 \ 4 \ 2)$.
  - $(1 \ 2 \ 3 \ 4 \ 5) \cdot (1 \ 3 \ 5 \ 4 \ 2) \cdot (5 \ 4 \ 3 \ 2 \ 1) \cdot (2 \ 4 \ 5 \ 3 \ 1) = (1 \ 3 \ 2 \ 5 \ 4) =: \psi$.
Proof of Barrington’s theorem (2/2)

The proof only handles negations (do not contribute to the depth) and conjunctions

- If $B$ with output $\sigma$ is equivalent to a circuit $A$, then there exists $B'$ of same length with output $\sigma^{-1}$ that is equivalent to $\neg A$
  - This is stated in previous slide

- If $B_i$ of length $d_i$ with output $\phi_i$ is equivalent to circuit $A_i$ ($i \in \{1, 2\}$), then there exists a branching program $B$ with output $\psi$ of length $2(d_1 + d_2)$ that is equivalent to $A_1 \land A_2$
  - Let $B'_i$ of length $d_i$ with output $\phi_i^{-1}$ be also equivalent to circuit $A_i$
  - Let $B = B_1; B_2; B'_1; B'_2$
Representing the disjuncts

- Replace each variable in the 3CNF formula with two variables:
  - \( X \mapsto X_1 \oplus X_2 \)

- Apply Barrington’s transformation to each disjunct, which now has the form
  \[
  (X_1^{b_1} \oplus X_2) \lor (Y_1^{b_2} \oplus Y_2) \lor (Z_1^{b_3} \oplus Z_2)
  \]
  - In a branching program, negation is expressed by swapping the two group elements

- We now have a set of branching programs of some length \( d \)
  - They all refer to the same input bits

- Everything above is the instance. The witness is a bit-string \( b_1 \cdots b_n \)

- The witness expands to sequences from \( (S_5)^d \), one for each disjunct
  - Additionally, there is the element \( \sigma \in S_5 \) meaning “yes”

- Verification — compute the product of each sequence; make sure the result is “yes”
Expanding a sequence \( g_1; g_2; \cdots; g_d \)

Table \( T \in (S_5)^{2 \times d} \) and vector \( \vec{r} \in (S_5)^{d-1} \)

- Let \( \vec{r} \leftarrow (S_5)^{d-1} \). Also define \( r_0 = r_d = 1 \)
- Let the top row \( T[1, \star] \) of \( T \) be \( g_1, g_2, \ldots, g_d \)
- Define \( T[2, i] \leftarrow r_{i-1}^{-1} \cdot T[1, i] \cdot r_i \)

Note that each both rows of \( T \) multiply to \( \sigma \) (public)

The proof string \( \pi \), constructed by \( P \)

- The witness \( b_1 \cdots b_n \) (one bit per cell)
- For each disjunct: vectors \( T[2, \star] \) and \( \vec{r} \) (one element of \( S_5 \) per cell)
  - The witness gives the first row of \( T \)
Possible queries of the verifier

- First, the verifier randomly selects a clause, fixing $T$ and $\vec{r}$ which he will read.
- The verifier now checks one of the following:
  - Gets the entire $T[2, \star]$ and checks that it multiplies to $\sigma$.
  - Picks $j \leftarrow \{1, \ldots, d\}$. Gets $T[1, j]$ (by querying a bit in the witness), $T[2, j]$, $r_{j-1}$ and $r_j$ and checks that $T[2, j] = r_{j-1}^{-1} \cdot T[1, j] \cdot r_i$.
- The completeness of this protocol is obvious.
Soundness and zero-knowledge

- **Soundness**: if the formula is not satisfiable, then there is always a clause that evaluates to “false”. The verifier may choose it.
  - In this case, $T[1,\star]$ does not multiply to $\sigma$.
  - If $T[2,\star]$ multiplies to $\sigma$, then it was not correctly constructed.
    - This is discovered when comparing $T[1,j]$ and $T[2,j]$.

- **Soundness error** — proportional to the fraction of satisfiable clauses in the formula.

- **Zero-knowledge**: verifier’s view is easy to simulate.
  - If it asked for $T[2,\star]$, give $d$ random elements multiplying to $\sigma$.
  - If it asked for $T[1,j]$, $T[2,j]$, $r_{j-1}$, $r_j$, give random elements with correct relationship.
    - Simulator has to come up with the value of one “new” variable. This is OK.

- Select other elements of $\pi$ randomly and create the Merkle tree and openings.
Interactive Oracle Proofs
Evaluating a polynomial and PCPs

- $\pi$ — a proof string encoding values of polynomial $f : \mathbb{F} \rightarrow \mathbb{F}$
- $V$ may ask to evaluate $f$ at any element of $\mathbb{F}$
- Must $\pi$ contain values $f(v)$ for all $v \in \mathbb{F}$?
Evaluating a polynomial and PCPs

- \( \pi \) — a proof string encoding values of polynomial \( f : \mathbb{F} \rightarrow \mathbb{F} \)
- \( V \) may ask to evaluate \( f \) at any element of \( \mathbb{F} \)
- Must \( \pi \) contain values \( f(v) \) for all \( v \in \mathbb{F} \)?
- No. If \( V \) wants \( f(r) \), \( P \) can just give it \( v = f(r) \). To prove it:
  - \( f(r) - v = 0 \). I.e. \( g(X) \equiv f(X) - v \) has a zero at \( r \)
  - Hence \( g(X) = (X - r) \cdot w(X) \) for some polynomial \( w \)
  - \( P \) commits to \( w \). \( V \) checks degree of \( w \) and the equality
    - “Interactive Oracle Proof” (IOP)
- \( \pi \) contains values of \( f \) on a sufficiently large subset of \( \mathbb{F} \)
IOP for low-degree test (1/2)

- Let $f : \mathbb{F} \to \mathbb{F}$ be committed, $V$ wants to check that $\deg f < 2^d$
- Let $\pi$ contain values of $f$ on a set $L \subseteq \mathbb{F}$
- Let $L \leq \mathbb{F}^*$, $|L| = 2^n$ ($n > d$). Note: $L$ is a cyclic group
  - I.e. $|\mathbb{F}| - 1$ must be divisible by $2^n$
IOP for low-degree test (1/2)

- Let $f : \mathbb{F} \to \mathbb{F}$ be committed, $V$ wants to check that $\deg f < 2^d$
- Let $\pi$ contain values of $f$ on a set $L \subseteq \mathbb{F}$
- Let $L \leq \mathbb{F}^*$, $|L| = 2^n$ ($n > d$). Note: $L$ is a cyclic group
  - I.e. $|\mathbb{F}| - 1$ must be divisible by $2^n$
- Let $f(X) = \sum_{i=0}^{2^d-1} a_i X^i$. $P$ defines following polynomials:
  
  \[
  f_0(X) := \sum_{i=0}^{2^d-1-1} a_{2i} X^i \quad f_1(X) := \sum_{i=0}^{2^d-1-1} a_{2i+1} X^i
  \]
  
  \[
  q(X, Y) := f_0(X) + Y \cdot f_1(X)
  \]

Note: $f(X) = q(X^2, X)$
IOP for low-degree test (1/2)

- Let $f : \mathbb{F} \rightarrow \mathbb{F}$ be committed, $V$ wants to check that $\deg f < 2^d$
- Let $\pi$ contain values of $f$ on a set $L \subseteq \mathbb{F}$
- Let $L \leq \mathbb{F}^*, |L| = 2^n$ ($n > d$). Note: $L$ is a cyclic group
  - i.e. $|\mathbb{F}| - 1$ must be divisible by $2^n$
- Let $f(X) = \sum_{i=0}^{2^d-1} a_i X^i$. $P$ defines following polynomials:
  
  $f_0(X) := \sum_{i=0}^{2^{d-1}-1} a_{2i} X^i$
  $f_1(X) := \sum_{i=0}^{2^{d-1}-1} a_{2i+1} X^i$
  
  $q(X, Y) := f_0(X) + Y \cdot f_1(X)$

  Note: $f(X) = q(X^2, X)$
- $V$ sends a random $r \in \mathbb{F}$ to $P$
- $P$ commits to $f'(X) := q(X, r)$ on the set $L' = \{r^2 | r \in L\}$
- $P$ proves to $V$ that $\deg f' < 2^{d-1}$. **Recursion!**
IOP for low-degree test (2/2)

Verifying relationship of \( f' \), \( q \), \((f)\). Do the following multiple times:

- \( V \) picks a random \( s \in L \)
  - Denote \( s' = -s \). Remember \( f(X) = q(X^2, X), f'(X) = q(X, r), q \) is linear in second argument
  - Denote \( g(X) = q(s^2, X) \). Then \( g \) is linear

- \( V \) verifies that

\[
\frac{(f(s) - f(-s))/(2s)}{f(s) - f'(s^2))/(s - r)}
\]

Base of the recursion. \( d = 0 \)

- Want to show \( \deg f < 1 \), i.e. \( f \) is constant. \( P \) sends that constant to \( V \)
IOP for low-degree test (2/2)

Verifying relationship of $f'$, $q$, $(f)$. Do the following multiple times:

- $V$ picks a random $s \in L$
  - Denote $s' = -s$. Remember $f(X) = q(X^2, X)$, $f'(X) = q(X, r)$, $q$ is linear in second argument
  - Denote $g(X) = q(s^2, X)$. Then $g$ is linear

- $V$ verifies that
  \[
  \frac{(f(s) - f(-s))}{(2s)} = \frac{(f(s) - f'(s^2))}{(s - r)}
  \]
  \[
  \frac{(q(s^2, s) - q(s^2, s'))}{(s - s')} = \frac{(q(s^2, s) - q(s^2, r))}{(s - r)}
  \]

Base of the recursion. $d = 0$

- Want to show $\deg f < 1$, i.e. $f$ is constant. $P$ sends that constant to $V$
IOP for low-degree test (2/2)

Verifying relationship of \( f', q, (f) \). Do the following multiple times:

- \( V \) picks a random \( s \in L \)
  - Denote \( s' = -s \). Remember \( f(X) = q(X^2, X) \), \( f'(X) = q(X, r) \), \( q \) is linear in second argument
  - Denote \( g(X) = q(s^2, X) \). Then \( g \) is linear

- \( V \) verifies that

\[
\frac{(f(s) - f(-s))}{(2s)} = \frac{(f(s) - f'(s^2))}{(s - r)}
\]
\[
\frac{(q(s^2, s) - q(s^2, s'))}{(s - s')} = \frac{(q(s^2, s) - q(s^2, r))}{(s - r)}
\]
\[
\frac{(g(s) - g(s'))}{(s - s')} = \frac{(g(s) - g(r))}{(s - r)}
\]

Base of the recursion. \( d = 0 \)

- Want to show \( \deg f < 1 \), i.e. \( f \) is constant. \( P \) sends that constant to \( V \)
Analysis of the low-degree test

- If $f$ is far from $\mathbb{F}^{<2^d}[X]$, then $f'$ is far from $\mathbb{F}^{<2^{d-1}}[X]$
- Precise analysis is complex. $f'$ is a random linear combination of $f_0$ and $f_1$. If at least one of them is far from $2^{d-1}$-degree polynomials, then so is $f'$
- The consistency check — $f'$ is given linear combination of $f_0$ and $f_1$ — fails with probability depending on the distance of $f'$ from this combination

Theorem

Let the relationship checking be done $\ell$ times in each round. Let $\rho = 2^{(d-n)/2}$. Let $\Delta(f, \mathbb{F}^{<2^d}[X]) \geq \delta$. Then the verifier accepts with probability at most

$$(\rho + \eta)^\ell + \left(\frac{2^d + 1}{2\eta}\right)^7 \cdot |\mathbb{F}|$$

for any $\eta \in (0, \rho/20)$. 
ZK IOP for low degree

Theorem

Let \( x, y \in \mathbb{F}^n \), let \( S \leq \mathbb{F}^n \), let \( \varepsilon \) be such, that \( \Delta(x, S) > \varepsilon \). Then exists at most a single \( \alpha \in \mathbb{F} \), such that \( \Delta(\alpha x + y, S) \leq \varepsilon/2 \).

Proof.

- Suppose there exist \( \alpha_1, \alpha_2 \in \mathbb{F} \), such that \( \Delta(\alpha_i x + y, S) \leq \varepsilon/2 \)
- Then \( \Delta((\alpha_1 - \alpha_2)x, S) \leq \varepsilon \)
  - Because \( S - S := \{ s_1 - s_2 \mid s_1, s_2 \in S \} = S \)
- Then also \( \Delta(x, S) \leq \varepsilon \)
ZK IOP for low degree

- $P$ sends to $V$ the commitment for a random $g: \mathbb{F} \rightarrow \mathbb{F}$
  - Commitment encodes values of $g$ on the set $L$
  - For the protocol to work, $\deg g < 2^d$ must hold
    - But that condition is not necessary for soundness
- $V$ picks a random $\rho \in \mathbb{F}$ and sends to $P$
- $P$ and $V$ run the low-degree test protocol for $\rho \cdot f + g$

This is ZK \textit{modulo} $V$’s queries to $f$’s values
Univariate Sum-Check (1/2)

Let \( p : \mathbb{F} \rightarrow \mathbb{F} \) (prover committed to it) and \( H \leq \mathbb{F}^* \). Show that

\[
\sum_{x \in H} p(x) = 0
\]

Let \( \deg p = d \) and \( |H| = n \)

**Theorem**

*Previous equality holds iff* \( p(X) = h(X) \cdot Z_H(X) + X \cdot f(X)\), *for some polynomials* \( h \) *and* \( f \), *where* \( \deg h \leq d - n \) *and* \( \deg f \leq n - 2 \)

Proof of the theorem consists of:

- Polynomial division with remainder
- Some (or a bit more) group theory to establish that the remainder has no free term
Some group theory

Let \( H \leq \mathbb{F}^* \), let \( |H| = n \)

- **Fact:** \( \sum_{a \in H} a = 0 \)
  - \( X^n - 1 = Z_H(X) = \prod_{a \in H}(X - a) \). Consider the coefficient of \( X^{n-1} \)

- **Fact:** \( \sum_{a \in H} a^m = 0 \), if \( m \) is not a multiple of \( n \)
  - If \( m \perp n \), then \( \{a^m | a \in H\} = H \)
  - If \( d = \gcd(m, n) > 1 \), then the sum passes several times through a subgroup \( H' \leq H \) of size \( n/d \)

- **Fact:** If \( \deg f < n \), then \( \sum_{a \in H} f(a) = n \cdot f(0) \)
  - Indeed, all terms of \( f \), except the free term, sum to 0
Polynomial division with remainder

**Theorem**

Let $p \in \mathbb{F}[X]$. Then $\sum_{a \in H} p(a) = 0$ iff $p(X) = h(X) \cdot Z_H(X) + X \cdot f(X)$, for some polynomials $h$ and $f$, where $\deg h \leq d - n$ and $\deg f \leq n - 2$

- If such polynomials exist, then $\sum_{a \in H} p(a) = \sum_{a \in H} a \cdot f(a) = 0$ by previous slide
- Other direction: $p(X) = h(X) \cdot Z_H(X) + r(X)$ for some $r \in \mathbb{F}^{\leq n-1}[X]$. If $\sum_{a \in H} p(a) = 0$, then also $n \cdot r(0) = \sum_{a \in H} r(a) = 0$, i.e. $r(0) = 0$, i.e. $r$ has no free term.
Univariate Sum-Check (2/2)

Protocol
- Prover commits to $h$. Verifier checks that its degree is at most $d - n$.
- Run the check that
  \[ f(X) = (p(X) - h(X) \cdot Z_H(X)) \cdot X^{-1} \]
  has degree at most $n - 1$.
- Whenever verifier has to compute some $f(r)$, it will find it from $p(r)$ and $h(r)$.
- The two checks for degree bounds are combined into one, as described previously.
- Note that $Z_H(X) = X^n - 1$, hence is easy to evaluate.
IOP for Rank-1 Constraint Systems (R1CS)
Rank-1 Constraint Systems

Definition

- R1CS is given by matrices $A, B, C \in \mathbb{F}^{m \times n}$
  - We say that there are $n$ variables and $m$ constraints
- A solution to R1CS is a vector $\vec{s} \in \mathbb{F}^n$, such that $s_1 = 1$ and $A\vec{s} \odot B\vec{s} = C\vec{s}$
  - Here “$\odot$” denotes componentwise multiplication

From arithmetic circuits to R1CS

- Each input or gate (addition, multiplication, constant) is a variable
- Each gate $g$ is a constraint:
  - For “$x_3 = x_1 + x_2$”: Let $A[g, x_1] = A[g, x_2] = B[g, 0] = C[g, x_3] = 1$
  - For “$x_3 = x_1 \cdot x_2$”: Let $A[g, x_1] = B[g, x_2] = C[g, x_3] = 1$
  - For “$x = c$”: Let $A[g, x_1] = B[g, 0] = 1$, $C[g, 0] = c$
Commitment

- Let $H \leq \mathbb{F}^*$, $|H| = m = n$. Let $\phi : H \to \{1, \ldots, n\}$ be bijective.
- Let $\vec{s}_A, \vec{s}_B, \vec{s}_C \in \mathbb{F}^m$, such that $\vec{s}_M = M \cdot \vec{s}$ for $M \in \{A, B, C\}$.
- Let $\hat{s}, \hat{s}_A, \hat{s}_B, \hat{s}_C : \mathbb{F} \to \mathbb{F}$ be polynomials of degree $\leq n$, such that $\hat{s}(h) = s_{\phi(h)}$ and $\hat{s}_M(h) = (s_M)_{\phi(h)}$ for all $h \in H$.
- The prover commits to $\hat{s}, \hat{s}_A, \hat{s}_B, \hat{s}_C$ over some group $L$, $H \leq L \leq \mathbb{F}^*$.
- Verifier checks the degree of committed $\hat{s}, \hat{s}_A, \hat{s}_B, \hat{s}_C$.
- Note that it is possible to get the evaluation of the polynomials at any point in $\mathbb{F}$.
  - This, and the other steps require the low-degree checking of more polynomials.
  - All these checks can be combined into one.
Checking the R1CS equation

- Want to check that $\hat{s}_A(h) \cdot \hat{s}_B(h) - \hat{s}_C(h) = 0$ for all $h \in H$
- i.e. $\exists w \in \mathbb{F}[X]$, such that $\hat{s}_A \cdot \hat{s}_B - \hat{s}_C = Z_H \cdot w$
- Prover finds $w$, commits to it. Verifier checks the degree
- Verifier picks random $r \in \mathbb{F}$, sends to prover
- They evaluate $\hat{s}_A, \hat{s}_B, \hat{s}_C, w$ on point $r$
- Verifier evaluates $Z_H(r)$
- Verifier checks that $\hat{s}_A(r) \cdot \hat{s}_B(r) - \hat{s}_C(r) = Z_H(r) \cdot w(r)$
Checking matrix-vector multiplication

We want to check, that for all \( h \in H \):

\[
\hat{s}_M(h) = \sum_{j \in H} M[\phi(h), \phi(j)] \cdot \hat{s}(j)
\]

This is the same as

\[
\hat{s}_M(X) = \sum_{j \in H} \widetilde{M}(X, j) \cdot \hat{s}(j)
\]

where \( \widetilde{M} \) is the polynomial extension of \( M[\cdot, \cdot] \) to the whole \( \mathbb{F}^2 \).

Verifier picks \( r \xleftarrow{\$} \mathbb{F} \) and does the following Sum-Check:

\[
0 \overset{?}{=} \sum_{j \in H} q(j) \quad \text{where} \quad q(Y) = \widetilde{M}(r, Y) \cdot \hat{s}(Y) - \hat{s}_M(r)/n
\]

In the end, verifier has to evaluate \( \widetilde{M}(r, r') \) for some \( r' \in \mathbb{F} \). How?
Trusted commitments for computing $\tilde{M}(r, r')$}

- Let $M$ have $k = |K|$ non-zero entries for some $K \leq F^*$
- Let $row, col : K \to H$ give the locations of non-zero entries
- Let $u \in \mathbb{F}[X, Y]$ satisfy $u(h, h) \neq 0$ and $u(h, h') = 0$ for all $h, h' \in H$, $h \neq h'$
  - ... and let individual degrees of $u$ be $\leq (n - 1)$
  - ... and let $u$ be easy to evaluate on the whole $\mathbb{F}^2$
- Define $val : K \to \mathbb{F}$ by $val(\kappa) = \frac{M[row(\kappa), col(\kappa)]}{u(row(\kappa), row(\kappa)) \cdot u(col(\kappa), col(\kappa))}$. Then
  \[
  \tilde{M}(X, Y) = \sum_{\kappa \in K} u(X, \tilde{row}(\kappa)) \cdot u(Y, \tilde{col}(\kappa)) \cdot \tilde{val}(\kappa)
  \]
  equals with $M[\cdot, \cdot]$ at all positions in $H \times H$
- For given $r, r'$, define polynomial $p(X) = u(r, \tilde{row}(X)) \cdot u(r, \tilde{col}(X)) \cdot \tilde{val}(X)$
  - Prover could give claimed value $\tilde{M}(r, r')$ and Sum-Check on $p$ could verify it
  - A trusted party has to commit to $\tilde{row}, \tilde{col}, \tilde{val}$
Sum-Checking $p$

- Degree of $p$ is $< kn$. For Sum-Check, prover has to compute and commit to a polynomial of degree $< kn - k$
- Instead, $P$ commits to the unique \( f \in \mathbb{F}^{<k}[X] \), where \( f(\kappa) = p(\kappa) \) for all $\kappa \in K$, and runs the Sum-Check with $f$ instead
Sum-Checking $p$

- Degree of $p$ is $< kn$. For Sum-Check, prover has to compute and commit to a polynomial of degree $< kn - k$
- Instead, $P$ commits to the unique $f \in \mathbb{F}^{<k}[X]$, where $f(\kappa) = p(\kappa)$ for all $\kappa \in K$, and runs the Sum-Check with $f$ instead
- Verifier must be able to verify that $f(\kappa) = p(\kappa)$ for all $\kappa \in K$
- Below we will get $p(\kappa) = \xi(\kappa)/\psi(\kappa)$ for some $O(k)$-degree polynomials $\xi$ and $\psi$, and for all $\kappa \in K$
  - ...such that $\xi$ and $\psi$ are easy to compute for the Verifier
Sum-Checking $p$

- Degree of $p$ is $< kn$. For Sum-Check, prover has to compute and commit to a polynomial of degree $< kn - k$
- Instead, $P$ commits to the unique $f \in \mathbb{F}^{<k}[X]$, where $f(\kappa) = p(\kappa)$ for all $\kappa \in K$, and runs the Sum-Check with $f$ instead
- Verifier must be able to verify that $f(\kappa) = p(\kappa)$ for all $\kappa \in K$
- Below we will get $p(\kappa) = \xi(\kappa)/\psi(\kappa)$ for some $O(k)$-degree polynomials $\xi$ and $\psi$, and for all $\kappa \in K$
  - ...such that $\xi$ and $\psi$ are easy to compute for the Verifier
- Equality check is then: for all $\kappa \in K : \psi(\kappa) \cdot f(\kappa) - \xi(\kappa) = 0$
- Standard method for checking this: $P$ commits to some $\varphi(X)$, such that

$$\psi(X) \cdot f(X) - \xi(X) = Z_K(X) \cdot \varphi(X)$$

and verifier checks that equality on a point, and the degree of $\varphi(X)$
The polynomial \( u(X, Y) \)

- Put \( u(X, Y) = (Z_H(X) - Z_H(Y))/(X - Y) \). It is a polynomial.
- Indeed, as \( H \leq \mathbb{F}^* \), we have \( Z_H(X) = X^n - 1 \) (recall \(|H| = n\))
- Hence
  \[
  u(X, Y) = X^{n-1} + X^{n-2}Y + X^{n-3}Y^2 + \cdots + XY^{n-2} + Y^{n-1}
  \]
- Also, \( u(X, X) = n \cdot X^{n-1} \)
- \( u(r, r') \) is easy (\( O(\log n) \) field operations) to evaluate for any \( r, r' \)
The polynomials $\xi$ and $\psi$

For all $\kappa \in K$:

\[
\begin{align*}
  u(r, \text{row}(\kappa)) &= \frac{Z_H(r) - Z_H(\text{row}(\kappa))}{r - \text{row}(\kappa)} = \frac{Z_H(r)}{r - \text{row}(\kappa)} \\
  u(r', \text{col}(\kappa)) &= \frac{Z_H(r') - Z_H(\text{col}(\kappa))}{r' - \text{col}(\kappa)} = \frac{Z_H(r')}{r' - \text{col}(\kappa)}
\end{align*}
\]

Hence

\[
p(\kappa) = u(r, \text{row}(\kappa)) \cdot u(r, \text{col}(\kappa)) \cdot \text{val}(\kappa) = \frac{Z_H(r)Z_H(r') \cdot \text{val}(\kappa)}{(r - \text{row}(\kappa))(r' - \text{col}(\kappa))}
\]

Define $\xi(X) = Z_H(r)Z_H(r') \cdot \text{val}(X)$ and $\psi(X) = (r - \text{row}(X))(r' - \text{col}(X))$
Fast Fourier Transform and other computations with polynomials
Motivation

- In the protocols we have seen, the Prover (and sometimes also the Verifier) has to perform complex computations with polynomials:
  - Evaluate polynomials at a large number of points
  - Multiply polynomials
  - Divide polynomials
- The polynomials themselves are large as well
- If we are not careful, these operations could easily take time $O(d^2)$, where $d$ is the degree of the polynomials
Discrete Fourier transformation (DFT)

- We work in a finite field $\mathbb{F}$
- Let $\omega \in \mathbb{F}$ be a primitive $n$-th root of unity, i.e.
  - $\omega^n = 1$
  - $\omega^k \neq 1$ for $1 \leq k < n$
- Such $\omega$ exists iff $n$ divides $|\mathbb{F}^\ast|$
- Theorem: the multiplicative group of a finite field is cyclic
- Such $\omega$ satisfies $\sum_{j=0}^{n-1} \omega^{kj} = 0$ for all $1 \leq k < n$
- Indeed, $(\sum_{j=0}^{n-1} \omega^{kj})(\omega^k - 1) = \omega^{kn} - 1 = 0$, but $\omega^k \neq 1$
- DFT maps the sequence $(v_0, \ldots, v_{n-1}) \in \mathbb{F}^n$ to the sequence $(v'_0, \ldots, v'_{n-1})$, where

$$v'_i = \sum_{j=0}^{n-1} v_j \cdot \omega^{ij}$$
DFT and polynomials

- Let $f(X) = \sum_{i=0}^{n-1} a_i X^i$
- DFT of $(a_0, \ldots, a_{n-1})$ corresponds to evaluating $f(1), f(\omega), f(\omega^2), \ldots, f(\omega^{n-1})$
Inverse DFT

Let \( v_0, \ldots, v_{n-1}, v'_0, \ldots, v'_{n-1} \in \mathbb{F} \)

\[
v'_i = \sum_{j=0}^{n-1} v_j \cdot \omega^{ij} \quad \Leftrightarrow \quad v_i = \frac{1}{n} \sum_{j=0}^{n-1} v'_j \cdot \omega^{-ij}
\]

I.e. inverse DFT is the same as DFT, except

- a different root of unity is used (\( \omega^{-1} \) vs. \( \omega \))
- The result is scaled by \( 1/n \)

In terms of polynomials, IDFT corresponds to interpolation
FFT (Cooley-Tukey alg. for DFT)

- Let $n = n_1 \cdot n_2$. It is OK, if $\gcd(n_1, n_2) > 1$
  - Typically, $n$ is a power of 2 and $n_1 = 2$
- We had $i \in \{0, \ldots, n - 1\}$. Let $i = n_2 i_1 + i_2$, where $i_1 \in \{0, \ldots, n_1 - 1\}$ and $i_2 \in \{0, \ldots, n_2 - 1\}$
- Let $\omega_1 = \omega^{n_2}$ and $\omega_2 = \omega^{n_1}$
- Denote $v[j_1, j_2] = v_{j_1 + n_1 j_2}$ and $v'[i_1, i_2] = v'_{n_2 i_1 + i_2}$
FFT (Cooley-Tukey alg. for DFT)

\[ v'[i_1, i_2] = \sum_{j_1=0}^{n_1-1} \sum_{j_2=0}^{n_2-1} v[j_1, j_2] \cdot \omega^{(n_2 i_1 + i_2)(j_1 + n_1 j_2)} = \]

\[ \sum_{j_1=0}^{n_1-1} \sum_{j_2=0}^{n_2-1} v[j_1, j_2] \cdot \omega^{i_2 j_1} \omega^{i_1 j_1} \omega^{i_2 j_2} = \sum_{j_1=0}^{n_1-1} \left[ \omega^{i_2 j_1} \left( \sum_{j_2=0}^{n_2-1} v[j_1, j_2] \cdot \omega^{i_2 j_2} \right) \right] \omega^{i_1 j_1} \]

 FFT of \( v[j_1, \star] \)

scaling / “rotation”

FFT giving \( v'[\star, i_2] \)
FFT (Cooley-Tukey alg. for DFT)

1. Populate $V \in F^{n_1 \times n_2}$ by $V[i, j] \leftarrow v_{i+n_1j}$
2. Compute $W \in F^{n_1 \times n_2}$ by $W[i, \star] \leftarrow \text{FFT}(V[i, \star])$
3. Compute $W' \in F^{n_1 \times n_2}$ by $W'[i, j] \leftarrow \omega^{ij} W[i, j]$
4. Compute $V' \in F^{n_1 \times n_2}$ by $V'[\star, j] \leftarrow \text{FFT}(W'[\star, j])$
5. Read off $v'_{n_2i+j} \leftarrow V'[i, j]$

- Time complexity: $T(n) = n_1 T(n_2) + n_2 T(n_1) + O(n)$
- If $n_1 = 2$ then $T(n) = 2 T(n/2) + O(n)$. Hence $T(n) = O(n \log n)$
Cooley-Tukey alg. for \( n = 2 \cdot (n/2) \)
Multiplication of polynomials

- $f(X) = \sum_{i=0}^{n} a_{i}X^{i}$ and $g(X) = \sum_{i=0}^{n} b_{i}X^{i}$

- “Usual” algorithm requires the computation of $a_{i}b_{j}$ for each $i$ and $j$, giving $O(n^2)$ complexity

- Instead of that, we could
  - Evaluate $f$ and $g$ on (at least) $2n + 1$ points $(x_1, \ldots, x_{2n+1})$, using FFT
  - Multiply the evaluations: $h_i = f(x_i) \cdot g(x_i)$
  - Interpolate $h_1, \ldots, h_{2n+1}$, using inverse FFT

- Time complexity: $O(n \log n)$
Division of polynomials

- Given \( f, g \in \mathbb{F}[X] \). Let \( g \) be monic (coeff. of \( X^{\deg g} \) is 1)
- Find \( q, r \in \mathbb{F}[X] \), such that \( \deg r < \deg g \) and \( f = qg + r \)
Division of polynomials

- Given $f, g \in \mathbb{F}[X]$. Let $g$ be monic (coeff. of $X^{\deg g}$ is 1).
- Find $q, r \in \mathbb{F}[X]$, such that $\deg r < \deg g$ and $f = qg + r$.
- For $p(X) = \sum_{i=0}^{n} a_i X^i$ define its reverse $R[p](X) = \sum_{i=0}^{n} a_{n-i} X^i$.
- $R[p](x) = x^n p(1/x)$. Hence $R[p_1 \cdot p_2] = R[p_1] \cdot R[p_2]$.
Division of polynomials

- Given \( f, g \in \mathbb{F}[X] \). Let \( g \) be monic (coeff. of \( X^{\deg g} \) is 1)
- Find \( q, r \in \mathbb{F}[X] \), such that \( \deg r < \deg g \) and \( f = qg + r \)
- For \( p(X) = \sum_{i=0}^{n} a_i X^i \) define its reverse \( R[p](X) = \sum_{i=0}^{n} a_{n-i} X^i \)
- \( R[p](x) = x^n p(1/x) \). Hence \( R[p_1 \cdot p_2] = R[p_1] \cdot R[p_2] \)

\[
R[f] = R[qg + r] = R[q]R[g] + X^{\deg f - \deg r} R[r]
\]

Consider this modulo \( X^{\deg f - \deg g + 1} \). This modulus divides \( X^{\deg f - \deg r} \)

\[
R[f] \equiv R[q]R[g] \pmod{X^{\deg f - \deg g + 1}}
\]

\[
R[q] \equiv R[f]R[g]^{-1} \pmod{X^{\deg f - \deg g + 1}}
\]

- We are looking for ways to invert polynomials modulo \( X^l \)
Hensel lifting

- **Input:** $h \in \mathbb{F}[X]$ and $p = h^{-1} \pmod{X^l}$
- **Output:** $h^{-1} \pmod{X^{2l}}$
Hensel lifting

- Input: $h \in \mathbb{F}[X]$ and $p = h^{-1} \pmod{X^l}$
- Output: $h^{-1} \pmod{X^{2l}}$
- Looking for result in the form $p + qX^l$ for some $q \in \mathbb{F}[X]$
- Let $h = h_0 + h_1X^l$ with $\deg h_0 < l$. Then $ph_0 = 1 + rX^l$ and
  \[(p + qX^l)(h_0 + h_1X^l) \equiv 1 + (r + qh_0 + ph_1)X^l \pmod{X^{2l}}\]
- pick $q$ so, that $(r + qh_0 + ph_1)$ is a multiple of $X^l$. I.e.
  \[
  q \equiv (ph_1 + r)/(-h_0) \pmod{X^l}
  \]
  \[
  q \equiv -p(ph_1 + r) \pmod{X^l}
  \]
- Time complexity of inverting $h$ modulo $X^n$: $O(n \log n)$
  - Indeed, $T(n) = T(n/2) + O(n \log n)$
Commitment to multilinear polynomials
Commitment to multilinear polynomials

- To commit to multilinear $f : \mathbb{F}^m \rightarrow \mathbb{F}$:
  - Pick $H \leq \mathbb{F}^*, |H| = 2^m$, and a bijection $\beta : H \rightarrow \{0, 1\}^m$
  - Commit to $q := f \circ \beta$ over $H$

- To evaluate $f(\vec{x})$ for $\vec{x} \in \mathbb{F}^m$:
  $$f(\vec{x}) = \sum_{a \in H} q(a) \cdot \chi_{\beta^{-1}(a)}(\vec{x})$$

Define $u_{\vec{x}} : \mathbb{F} \rightarrow \mathbb{F}$ as the polynomial (of degree $< 2^m$) satisfying
\[ \forall a \in H : u_{\vec{x}}(a) = \chi_{\beta^{-1}(a)}(\vec{x}) \]
Commitment to multilinear polynomials

To prove that $f(\vec{x}) = v$:

- Define (but don’t commit to)
  $$g(X) := q(X) \cdot u_{\vec{x}}(X) - v \cdot |H|^{-1},$$
  then $\sum_{a \in H} g(a) = 0$ iff $f(\vec{x}) = v$.

- Run the univariate Sum-Check protocol for $g$ and $H$

- During the run, $V$ may need to evaluate $q(r), h(r)$ (from the univariate Sum-Check protocol), $u_{\vec{x}}(r)$
  - $q$ and $h$ have been committed
  - $u_{\vec{x}}(r)$ has to be interpolated from the values of $u_{\vec{x}}$ on $H$
    - The values of $u_{\vec{x}}$ on $H$ also have to be computed
    - This is doable with a circuit of size $O(2^m)$, depth $O(m)$
    - Prover can help with the GKR protocol
Correcting the committed polynomials
Reed-Solomon codes

- Given: field $\mathbb{F}$, numbers $d, n \in \mathbb{N}$, $d \leq n$, (pairwise different) values $\alpha_1, \ldots, \alpha_n \in \mathbb{F}$
- The Reed-Solomon code of block length $n$ and message length $d$ is the following set:
  \[
  \{(f(\alpha_1), \ldots, f(\alpha_n)) \mid r_0, \ldots, r_{d-1} \in \mathbb{F}, f(X) := r_0 + r_1 X + \cdots + r_{d-1} X^{d-1}\}
  \]
- Berlekamp-Welch algorithm — an efficient algorithm for error correction:
  - Given $\vec{c}' = (c'_1, \ldots, c'_n) \in \mathbb{F}$, such that
  - exists $\vec{c} = (c_1, \ldots, c_n)$ in the code, such that
  - $\vec{c}'$ and $\vec{c}$ differ at most $\left\lfloor (n - d)/2 \right\rfloor$ positions,
  the algorithm returns $\vec{c}$
Suppose that the original codeword was \((s_1, \ldots, s_n)\), corresponding to the polynomial \(p\).

But we received \((\tilde{s}_1, \ldots, \tilde{s}_n)\).

- We assume it has at most \((n - d)/2\) errors.

Find the coefficients for polynomials \(q_0\) and \(q_1\), such that
  - Degree of \(q_0\) is at most \((n + d - 2)/2\). Degree of \(q_1\) is at most \((n - d)/2\).
  - For all \(i \in \{1, \ldots, n\}\): \(q_0(c_i) - q_1(c_i) \cdot \tilde{s}_i = 0\).
  - \(q_0\) and \(q_1\) are not both equal to 0.

Then \(p = q_0/q_1\).

In general, there are more equations than variables, but \(\tilde{s}_i\) are not arbitrary.
Correctness of decoding

Such polynomials $q_0, q_1$ exist:

- $(s_1, \ldots, s_n), (\tilde{s}_1, \ldots, \tilde{s}_n)$ — original and received codewords. Let $E$ be the set of $i$, where $s_i \neq \tilde{s}_i$. Then $|E| \leq (n - d)/2$.
- Let $k(x) = \prod_{i \in E}(x - c_i)$. Then $\deg k \leq (n - d)/2$.
- Take $q_1 = k$ and $q_0 = p \cdot k$. Then $\deg q_0 \leq (n + d - 2)/2$.
- For all $i \in \{1, \ldots, n\}$ we have

$$q_0(c_i) - q_1(c_i) \cdot \tilde{s}_i = k(c_i)(p(c_i) - \tilde{s}_i) = k(c_i)(s_i - \tilde{s}_i) = \begin{cases} k(c_i)(s_i - s_i) = 0, & i \notin E \\ 0 \cdot (s_i - \tilde{s}_i) = 0, & i \in E \end{cases}$$
Correctness of decoding

If $q_0$ and $q_1$ satisfy the equalities and upper bounds on degrees, then $p = q_0 / q_1$:

- Let $q'(x) = q_0(x) - q_1(x)p(x)$. Degree of $q'$ is at most $(n + d - 2)/2$.
- For each $i \not\in E$, $q'(c_i) = q_0(c_i) - q_1(c_i)p(c_i) = q_0(c_i) - q_1(c_i)\tilde{s}_i = 0$.
  - $1 \leq i \leq n$.
- The number of such $i$ is at least $n - (n - d)/2 = (n + d)/2$.
- Thus the number of roots of $q'$ is larger than its degree. Hence $q' = 0$.
- $q_0 - q_1 \cdot p = 0$. 
Correcting polynomials (1/2)

The task

- Given
  - Field $\mathbb{F}$, numbers $d, m \in \mathbb{N}$, rate $\delta \in (1/2, 1]$, unknown polynomial $f : \mathbb{F}^m \to \mathbb{F}$ with $\deg f \leq d$
  - Access to oracle $f^* : \mathbb{F}^m \to \mathbb{F}$, that agrees with $f$ on at least $\delta$ fraction of $\mathbb{F}^m$
  - A point $\vec{x} \in \mathbb{F}^m$ (Also known to whoever prepared $f^*$)
- Compute $f(\vec{x})$

Solution idea

- Randomly sample $\vec{r} \in \mathbb{F}^m$, define the line $\ell(X) := \vec{x} + X \cdot \vec{r}$
- Sample $f^* \circ \ell$ at sufficiently many points, run error correction

Exercise. Why doesn’t this idea work?
Correcting polynomials (2/2)

An idea that works

- Randomly sample $\vec{r}_1, \vec{r}_2 \in \mathbb{F}^m$, define the parabola $\ell(X) := \vec{x} + X \cdot \vec{r}_1 + X^2 \cdot \vec{r}_2$
- Sample $f^* \circ \ell$ at sufficiently many points, run error correction
  - The degree of $f \circ \ell$ would be $2d$
  - The probability of error on a randomly sampled location of $\ell$ is not much higher than $(1 - \delta)$

Theorem

Let $\alpha_1, \ldots, \alpha_n \in \mathbb{F}$. Let $I = \{i \in \{1, \ldots, n\} \mid f^*(\ell(i)) = f(\ell(i))\}$. Then

$$\Pr_{\vec{r}_1, \vec{r}_2} [|I| \leq \delta(n - c\sqrt{n})] \leq 1/c^2$$

for any positive number $c$
Proof of the theorem

- For any \( \vec{x} \in \mathbb{F}^m \) let \( I(x) \in \{0, 1\} \) indicate whether \( f^*(\vec{x}) = f(\vec{x}) \)
- Let \( A_1, \ldots, A_n \) be random variables, where \( A_i = I(\ell(\alpha_i)) \)
  - \( A_i \) are pairwise independent, because so are the random points \( \ell(\alpha_i) \)
- Let \( B = A_1 + \cdots + A_n \). Find its average \( \mathbb{E}[B] \) and variance \( \mathbb{V}[B] = \mathbb{E}[(B - \mathbb{E}[B])^2] \):
  \[
  \mathbb{E}[B] = \mathbb{E}[A_1 + \cdots + A_n] = \mathbb{E}[A_1] + \cdots + \mathbb{E}[A_n] = \delta n
  \]
  \[
  \mathbb{V}[B] = \mathbb{V}[A_1] + \cdots + \mathbb{V}[A_n] + \sum_{i \neq j} \text{Cov}[A_i, A_j] = n\delta(1 - \delta) \leq n\delta^2
  \]

Chebyshev inequality: \( \Pr[|B - \mathbb{E}[B]| > c\sqrt{\mathbb{V}[B]}] < 1/c^2 \). If \( B \leq \mathbb{E}[B] \), then
\[
\delta n - |I| > c\delta \sqrt{n} \quad \text{i.e.} \quad |I| < \delta(n - c\sqrt{n})
\]
Polynomial commitments from the hardness of Discrete Logarithm
First example

- Let $P$ have $f : \mathbb{Z}_p \to \mathbb{Z}_p$, $f(X) = \sum_{i=0}^{d} a_i X^i$
- Let $G$, $g$, $h$ be the set-up for Pedersen’s commitments
- $P \to V : c_0, \ldots, c_d$, where $c_i = g^{a_i} h^{r_i}$ for $r_i \leftarrow \mathbb{Z}_p$
  - Same size, as the whole $f$, but gives privacy
- To compute a commitment to $f(x)$, $V$ computes $\prod_{i=0}^{d} c_i^{x_i}$
  - $P$ is able to open it, if necessary
- This generalizes to certain classes of multivariate polynomials
  - dense polynomials in this class must not have too many coefficients
Commitments to vectors

**Pedersen commitments**

- Group $G$, size $p$, elements $g, h \in G$ with unknown $\log_g h$
- $\text{Com}(x; r) = g^x h^r$
- To open, give $x$ and $r$

**Pedersen vector commitments**

- Commitments to elements of $\mathbb{Z}_p^n$
- Elements $g_1, \ldots, g_n, h \in G$ with no known non-trivial discrete log relations
- $\text{Com}(x_1, \ldots, x_n; r) = g_1^{x_1} \cdots g_n^{x_n} h^r$
- Opening: give $x_1, \ldots, x_n, r$
- Homomorphic (for operations on vectors)
Discrete Log Relations

Fix $n$. Suppose that we have a machine $\mathcal{O}$ that takes $n$ elements of $\mathbb{G}$ and outputs $n$ elements of $\mathbb{Z}_p$, such that

$$\Pr \left[ \begin{array}{c} g_1^{x_1} \cdots g_n^{x_n} = 1 \\
\exists i : x_i \neq 0 \end{array} \right] = 1$$

$$g_1, \ldots, g_n \leftarrow \mathbb{G}$$

$$(x_1, \ldots, x_n) \leftarrow \mathcal{O}(g_1, \ldots, g_n)$$

is non-negligible, where probabilities are over the choice of $g_1, \ldots, g_n$ and the randomness used by $\mathcal{O}$.

Exercise. You are given some $g, h \in \mathbb{G}$. You have access to $\mathcal{O}$. Find $\log_g h$.
Solution to exercise

- Generate random \( r_1, \ldots, r_n, s_1, \ldots, s_n \leftarrow \mathbb{Z}_p \)
- Call \((x_1, \ldots, x_n) \leftarrow \mathcal{O}(g^{r_1} h^{s_1}, g^{r_2} h^{s_2}, \ldots, g^{r_n} h^{s_n})\)
  - Inputs are uniformly random elements of \( \mathbb{G} \)
  - Hence the output is a non-trivial DL relation (with non-negligible probability)
- Denote \( z = \log_g h \). Then \( \sum_{i=1}^{n} x_i (r_i + z s_i) = 0 \). Find:
  \[
  \log_g h = z = - \left( \sum_{i=1}^{n} x_i r_i \right) / \left( \sum_{i=1}^{n} x_i s_i \right)
  \]
- This fails only if the denominator is 0
- But the inputs to \( \mathcal{O} \) are independent of \( s_1, \ldots, s_n \)
- Hence the denominator is a random linear combination of \( x_1, \ldots, x_n \)
Committing to the vector of coefficients

- Let \( g_1, \ldots, g_n, h \) be fixed. Let \( P \) commit to \( \vec{u} \in \mathbb{Z}_p^n \) by \( c_u \leftarrow h^{r_u} \cdot \prod_{i=1}^{n} g_i^{u_i} \)
- Later, we get a public vector \( \vec{y} \in \mathbb{Z}_p^n \). \( P \) computes \( v = \langle \vec{u}, \vec{y} \rangle \) and \( c_v \leftarrow h^{r_v} g_1^v \)
- \( V \) has \( c_u, c_v, \vec{y} \). \( P \) wants to prove that \( v = \langle \vec{u}, \vec{y} \rangle \)

Protocol — similar to knowledge of a DL

- \( P \) samples \( \vec{s} \in \mathbb{Z}_p^n, r_1, r_2 \in \mathbb{Z}_p \), sends \( (\alpha_1, \alpha_2) = (h^{r_1} \prod_{i=1}^{n} g_i^{s_i}, h^{r_2} g_1^{\langle \vec{s}, \vec{y} \rangle}) \) to \( V \)
- \( V \) samples and sends a challenge \( \beta \in \mathbb{Z}_p \)
- \( P \) sends \( (\vec{\gamma}_1, \gamma_2, \gamma_3) = (\beta \vec{u} + \vec{s}, \beta r_u + r_1, \beta r_v + r_2) \)
- \( V \) checks that \( c_u^\beta \cdot \alpha_1 = h^{\gamma_2} \prod_{i=1}^{n} g_i^{\gamma_1,i} \) and \( c_v^\beta \cdot \alpha_2 = h^{\gamma_3} g_1^{\langle \vec{\gamma}_1, \vec{y} \rangle} \)

Hence, commitments are short. Unfortunately, \( \vec{\gamma}_1 \) is long
Trade-off: square-root lengths

Hadamard product

Let \( \vec{u} \) and \( \vec{v} \) be two vectors over \( \mathbb{F} \), with length \( m \) and \( n \). Their Hadamard product is
\[
\vec{u} \otimes \vec{v} := (u_i v_j)_{i,j=1,1}^{m,n} \quad \text{(a vector of length } mn)\]

- Note that \((1, r, r^2, \ldots, r^{mn-1}) = (1, r, r^2, \ldots, r^{n-1}) \otimes (1, r^n, r^{2n}, \ldots, r^{(m-1)n})\)

- \( P \) knows \( \vec{u} \in \mathbb{Z}_{p}^{n^2}. \) Creates \( n \) commitments:
  \[
c_{u,j} \leftarrow h^{r_{u,j} \cdot \prod_{i=1}^{n} g_i^{u_i+n \cdot j}}
  \]
- Later, there are public \( \vec{y}, \vec{z} \in \mathbb{Z}_p^{n}. \) \( P \) computes \( v = \langle \vec{u}, (\vec{y} \otimes \vec{z}) \rangle \) and \( c_v \leftarrow h^{r_v g_1^v} \)

- \( V \) can compute \( \hat{c} \leftarrow \prod_{j=1}^{n} c_{u,j}^{y_j} = h^{\sum_{j=1}^{n} r_{u,j} y_i} \cdot \prod_{i=1}^{n} g_i^{\sum_{j=1}^{n} u_i+n \cdot j y_j} \) himself
- \( P \) and \( V \) use the previous protocol on \( c_v \) and \( \hat{c} \) to show that

\[
c_v \text{ stores } \sum_{i=1}^{n} \left( \sum_{j=1}^{n} u_{i+n \cdot j} y_j \right) \cdot z_i = \langle \vec{u}, (\vec{y} \otimes \vec{z}) \rangle
\]
Committing to multilinear polynomials

- A dense multilinear polynomial with \( n \) variables has \( 2^n \) monomials
- Or it is a linear combination of \( 2^n \) Lagrange basis polynomials
  \[
  \chi_{\bar{w}}(\bar{X}) = (\prod_{i:w_i=1} X_i) \cdot (\prod_{i:w_i=0} (1 - X_i))
  \]
- The list of either of them can be represented as Hadamard product of two 
  \( 2^{n/2} \)-length vectors
  - Everything involving first \( n/2 \) variables only vs. everything involving last \( n/2 \) variables only
- The construction on previous slide will work
Bilinear pairings
Bilinear pairings

- $G_1, G_2, G_T$ — three cyclic groups of size $p \in \mathbb{P}$, with hard DLP
- let $g$ generate $G_1$ and $h$ generate $G_2$

**Definition**

$\hat{e} : G_1 \times G_2 \rightarrow G_T$ is a (non-degenerate) bilinear pairing, if

- $\hat{e}(g_1 g_2, h_1) = \hat{e}(g_1, h_1) \cdot \hat{e}(g_2, h_1)$ and $\hat{e}(g_1, h_1 h_2) = \hat{e}(g_1, h_1) \cdot \hat{e}(g_1, h_2)$
- $\hat{e}(g, h) \neq 1$, i.e. $\hat{e}(g, h)$ generates $G_T$

Hence

$$\hat{e}(g^x, h^y) = \hat{e}(g, h)^{xy}$$
Recall: exponential ElGamal

- A group $\mathbb{G}$ of size $p$ with generator $g$
- Secret key: $sk \leftarrow \mathbb{Z}_p$. Public key: $G = g^{sk}$
- Encrypting $m \in \mathbb{Z}_p$:
  - Generate $r \leftarrow \mathbb{Z}_p$
  - Output $(c_1, c_2) = (g^r, g^m G^r)$
- Decryption: let $g' = c_2/c_1^{sk}$, brute-force to find $m = \log_g g'$

Additively homomorphic

$$\mathcal{E}(m_1; r_1) \cdot \mathcal{E}(m_2; r_2) = \mathcal{E}(m_1 + m_2; r_1 + r_2)$$

(multiplication is componentwise)
The use of pairings

- We are “committing” to values in exponents:
  \[ g^x, h^y, \ldots \]
- We can do linear combinations with the committed values
- Pairing allows us to do one multiplication with them
- Getting an encryption scheme out of here takes some extra work
Boneh-Goh-Nissim cryptosystem

- Cyclic groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of size $n = pq$, secret factorization
- Public key: elements $G \in \mathbb{G}_1$, $H \in \mathbb{G}_2$ of order $q$
- Encryption of $m \in \mathbb{Z}_p$: $g^m G^r \in \mathbb{G}_1$ or $h^m H^r \in \mathbb{G}_2$
- Decryption of $c \in \mathbb{G}_1$: Compute $c' = c^q = g^{qm}$. Find $\log_{g^q} c'$
  - Same in $\mathbb{G}_2$
- Homomorphic addition: yes
- Homomorphic multiplication:
  $$\hat{e}(g^{m_1} G^{r_1}, h^{m_2} H^{r_2}) = \hat{e}(g, h)^{m_1 m_2} \hat{e}(g, H)^{m_1 r_2} \hat{e}(G, h)^{r_1 m_2} \hat{e}(G, H)^{r_1 r_2}$$

  I.e. when decrypting, we get $\hat{e}(g, h)^{qm_1 m_2}$ and have to find its discrete log. to the base $\hat{e}(g, h)^q$
Typical security properties

Bilinear Diffie-Hellman (for $G_1 = G_2$)
Given $g^a, g^b, g^c$, find $\hat{e}(g, g)^{abc}$

Bilinear Decisional Diffie-Hellman (for $G_1 = G_2$)
Distinguish $(g^a, g^b, g^c, \hat{e}(g, g)^{abc})$ from $(g^a, g^b, g^c, \hat{e}(g, g)^r)$

- Recent number-theoretic advances have obsoleted all instances of pairings, where $G_1 = G_2$
  - These instances were called symmetric
- For asymmetric instances, some elements in these assumptions come from $G_1$, and some from $G_2$
Where do the groups come from?

- $G_1$ is some elliptic curve group $E(\mathbb{F}_q)$
  - $q$ is the power of some prime. $p \approx q$
  - $E \equiv y^2 = x^3 + ax + b$. $a, b \in \mathbb{F}_q$
- $G_2$ is a subgroup of $E(\mathbb{F}_{q^k})$. $G_T$ is a subgroup of $\mathbb{F}_{q^k}^*$
  - The same $E$
  - The embedding degree $k$ is such, that $p$ divides $q^k - 1$
  - $k$ could be e.g. 12
- Computations in $G_1$ are cheaper than in $G_2$ or $G_T$
- Not every combination of $p, q, k$ works
  - Different design choices than for “usual” ECC

https://medium.com/@VitalikButerin/exploring-elliptic-curve-pairings-c73c1864e627
Generic group model (GGM)

- Access the elements of the group only through handles
- Have an API for performing group operations

The functionality $\mathcal{F}_{\text{gengroup}}^p$, $p \in \mathbb{P}$

- Internal state: $S \subseteq \mathbb{Z}_p \times \{0, 1\}^*$, initially $\{(0, 00 \cdots 0)\}$
  - Injective in both directions
- On input “op($w_1, \ldots, w_k$)”, where “op” is “mult” or “inv”:
  - Look up $e_i = S^{-1}(w_i)$
  - If “op” is “mult”, then put $r = \sum_i e_i$. If “op” is “inv”, then put $r = -e_1$
  - Return $S(r)$
- If $S(e)$ or $S^{-1}(w)$ is undefined, then
  - randomly pick $e \leftarrow \mathbb{Z}_p$ or $w \leftarrow \{0, 1\}^*$, avoiding collisions
  - Insert $(e, r)$ into $S$
DL is hard in the generic group

- Attacker $\mathcal{A}$ is given $g, h \in \{0, 1\}^*$
- Random $e_g = S^{-1}(g)$ and $e_h = S^{-1}(h)$ get defined by $\mathcal{F}_p^{\text{gengroup}}$
- We want to find $X = e_h/e_g$. Assume w.l.o.g. that $e_g = 1$. 

For given $f_1, f_2$, and a random $X$, this happens with probability $1/p$. There are $O(n^2)$ possible pairs $f_1, f_2$. 
DL is hard in the generic group

- Attacker $\mathcal{A}$ is given $g, h \in \{0, 1\}^*$
- Random $e_g = S^{-1}(g)$ and $e_h = S^{-1}(h)$ get defined by $\mathcal{F}_p^{gengroup}$
- We want to find $X = e_h/e_g$. Assume w.l.o.g. that $e_g = 1$.
- $\mathcal{A}$ queries $\mathcal{F}_p^{gengroup}$ ($n$ times). To each argument and answer, we can assign a linear polynomial in $\mathbb{Z}_p[X, Y_1, Y_2, \ldots]$:
  - $g \mapsto 1$. $h \mapsto X$
  - If $k_1 \mapsto f_1$ and $k_2 \mapsto f_2$, then $\text{mult}(k_1, k_2) \mapsto f_1 + f_2$
  - If $k \mapsto f$ then $\text{inv}(k) \mapsto -f$
  - If $\mathcal{A}$ submits a new $k$ to $\mathcal{F}_p^{gengroup}$, then $k \mapsto Y_i$, where $Y_i$ is new

There has to be a collision: $k \mapsto f_1$ and $k \mapsto f_2$ I.e. the linear polynomial $f_1 - f_2$ is 0 at $X$ For given $f_1, f_2$, and a random $X$, this happens with probability $1/p$ There are $O(n^2)$ possible pairs $f_1, f_2$
DL is hard in the generic group

- Attacker $\mathcal{A}$ is given $g, h \in \{0, 1\}^*$
- Random $e_g = S^{-1}(g)$ and $e_h = S^{-1}(h)$ get defined by $\mathcal{F}^p_{\text{gengroup}}$
- We want to find $X = e_h/e_g$. Assume w.l.o.g. that $e_g = 1$.
- $\mathcal{A}$ queries $\mathcal{F}^p_{\text{gengroup}}$ ($n$ times). To each argument and answer, we can assign a linear polynomial in $\mathbb{Z}_p[X, Y_1, Y_2, \ldots]$:  
  - $g \mapsto 1$. $h \mapsto X$
  - If $k_1 \mapsto f_1$ and $k_2 \mapsto f_2$, then $\text{mult}(k_1, k_2) \mapsto f_1 + f_2$
  - If $k \mapsto f$ then $\text{inv}(k) \mapsto -f$
  - If $\mathcal{A}$ submits a new $k$ to $\mathcal{F}^p_{\text{gengroup}}$, then $k \mapsto Y_i$, where $Y_i$ is new
- To find $X$, $\mathcal{A}$ need a non-trivial equation containing it  
  - There has to be a collision: $k \mapsto f_1$ and $k \mapsto f_2$
  - I.e. the linear polynomial $f_1 - f_2$ is 0 at $X$
  - For given $f_1, f_2$, and a random $X$, this happens with probability $1/p$
- There are $O(n^2)$ possible pairs $f_1, f_2$
DDH is hard in a generic group

- Given random $g, g^A, g^B, g^C, g^D$ with either $C = AB$ or $D = AB$
- $A$ must figure out, whether $C = AB$ or $D = AB$
DDH is hard in a generic group

- Given random $g, g^A, g^B, g^C, g^D$ with either $C = AB$ or $D = AB$
- $A$ must figure out, whether $C = AB$ or $D = AB$
- We get linear polynomials $f_i \in \mathbb{Z}_p[A, B, C, D, \ldots]$
- $A$ “wins”, if exist $i \neq j$, such that
  - $f_i(x, y, xy, z) = f_j(x, y, xy, z)$, or
  - $f_i(x, y, z, xy) = f_j(x, y, z, xy)$
- Each equality happens with probability $\leq 2/p$
  - Two equalities per pair $(i, j)$. Number of pairs: $O(n^2)$
- If $A$ “wins”, then it can make an informed choice. Otherwise it just guesses randomly
Generic bilinear group model (GBGM)

- Same as GGM, except that
  - Internal state has three partial functions $S_1, S_2, S_T$
  - There are operations $\text{mult}_i$ and $\text{inv}_i$ for $i \in \{1, 2, T\}$
  - There is an operation “pair”

$$\text{pair}(g, h) = S_T(S_1^{-1}(g) \cdot S_2^{-1}(h))$$
Polynomial commitments from pairings
A binding scheme

- There’s a CRS: $g, g^\tau, g^{\tau^2}, \ldots, g^{\tau^d}, h, h^\tau$ for committing to polynomials of degree at most $d$
  - $\tau \overset{S}{\leftarrow} \mathbb{Z}_p$ is random, must be remain hidden

- Commitment to $f : \mathbb{Z}_p \rightarrow \mathbb{Z}_p$: the value $c = g^{f(\tau)}$

- To open as $f(x) = y$:
  - $P$ computes $w(X) = (f(X) - y)/(X - x)$, sends $q = g^{w(\tau)}$ to $V$
  - $V$ checks: $\hat{e}(c \cdot g^{-y}, h) \overset{?}{=} \hat{e}(q, h^\tau \cdot h^{-x})$

Note that $V$ does not need the whole CRS, but only $h, h^\tau$.

Exercise.

Make the scheme hiding, too. Use Pedersen’s commitments.
A binding scheme

- There’s a CRS: $g, g^\tau, g^{\tau^2}, \ldots, g^{\tau^d}, h, h^\tau$ for committing to polynomials of degree at most $d$
  - $\tau \overset{\$}{\leftarrow} \mathbb{Z}_p$ is random, must be remain hidden
- Commitment to $f : \mathbb{Z}_p \rightarrow \mathbb{Z}_p$: the value $c = g^{f(\tau)}$
- To open as $f(x) = y$:
  - $P$ computes $w(X) = (f(X) - y)/(X - x)$, sends $q = g^{w(\tau)}$ to $V$
  - $V$ checks: $\hat{e}(c \cdot g^{-y}, h) \overset{?}{=} \hat{e}(q, h^\tau \cdot h^{-x})$
- I.e. $V$ checks whether $f(\tau) - y = w(\tau)(\tau - x)$
  - Checks the polynomial equation above at the random point $\tau$
- Note that $V$ does not need the whole CRS, but only $h, h^\tau$

**Exercise.** Make the scheme hiding, too. Use Pedersen's commitments
Binding

$d$-strong Diffie-Hellman ($d$-SDH) assumption

Given $g, g^\tau, g^{\tau^2}, \ldots, g^{\tau^d}$, the attacker cannot output $(c, g^{1/(\tau - c)}) \in \mathbb{Z}_p \times \mathbb{G}_1$

Suppose prover can open $f(x)$ as $y$ and as $y'$. I.e. he knows $q, q' \in \mathbb{G}_1$, such that

\[
\hat{e}(c \cdot g^{-y}h) = \hat{e}(q, h^{\tau} \cdot h^{-x}) \quad \text{and} \quad \hat{e}(c \cdot g^{-y'}h) = \hat{e}(q', h^{\tau} \cdot h^{-x})
\]

\[
\log_g c - y = (\log_g q) \cdot (\tau - x) \quad \text{and} \quad (\log_g c) - y' = (\log_g q') \cdot (\tau - x)
\]

\[
(\log_g q) \cdot (\tau - x) + y = (\log_g q') \cdot (\tau - x) + y'
\]

\[
((\log_g q) - (\log_g q')) \cdot (\tau - x) = y' - y
\]

\[
(q/q')^{\tau-x} = g^{y'-y}
\]

\[
(q/q')^{1/(y'-y)} = g^{1/(\tau-x)}
\]

December 2020
Bulletproofs
Inner product argument

- Cyclic group $G$ of size $p \in \mathbb{P}$
- Public elements $g_1, \ldots, g_n, h_1, \ldots, h_n, P \in G$, $c \in \mathbb{Z}_p$
  - $g_1, \ldots, g_n, h_1, \ldots, h_n$ come from the CRS
  - No known non-trivial discrete log relations among all $g_i, h_i$
- $P$ wants to convince $V$ that he knows $a_1, \ldots, a_n, b_1, \ldots, b_n \in \mathbb{Z}_p$, such that
  \[
  \prod_{i=1}^{n} g_i^{a_i} h_i^{b_i} = P \quad \text{and} \quad \sum_{i=1}^{n} a_i b_i = c
  \]
- Privacy is not important
- Can we be more efficient than $P$ just sending over all $a_i, b_i$?
Modified inner product argument

- Public elements \( g_1, \ldots, g_n, h_1, \ldots, h_n, P, u \in G \)
  - \( g_1, \ldots, g_n, h_1, \ldots, h_n, u \) come from the CRS
  - No known non-trivial discrete log relations among \( u \) and all \( g_i, h_i \)
- \( P \) wants to convince \( V \) that he knows \( a_1, \ldots, a_n, b_1, \ldots, b_n \in \mathbb{Z}_p \), such that
  \[
  u \sum_{i=1}^{n} a_i b_i \cdot \prod_{i=1}^{n} g_i^{a_i} h_i^{b_i} = P
  \]
- Privacy is still not important
Reduction from modified to original argument

To make the original argument:

- $V$ picks random $u \in \mathbb{G}$, sends it to $P$;
- Run the modified protocol with
  \[ P \leftarrow P_{\text{orig}} \cdot u^{\text{orig}} \]
  \[ \ldots \text{using the same } \vec{a}, \vec{b} \]
Soundness

- Run the modified protocol twice, with \( u_1 = g^{x(1)} \) and \( u_2 = g^{x(2)} \), for some \( g \in \mathbb{G} \).
- Extract the witnesses \( \vec{a}(1), \vec{b}(1), \vec{a}(2), \vec{b}(2) \). They satisfy

\[
g^{x(1)} \cdot \sum_{i=1}^{n} a_i^{(1)} b_i^{(1)} \cdot \prod_{i=1}^{n} g_i a_i^{(1)} b_i^{(1)} = P_{\text{orig}} \cdot g^{x(1)\text{orig}} \quad g^{x(2)} \cdot \sum_{i=1}^{n} a_i^{(2)} b_i^{(2)} \cdot \prod_{i=1}^{n} g_i a_i^{(2)} b_i^{(2)} = P_{\text{orig}} \cdot g^{x(2)\text{orig}}
\]
Soundness

- Run the modified protocol twice, with \( u_1 = g^{x(1)} \) and \( u_2 = g^{x(2)} \), for some \( g \in \mathbb{G} \).
- Extract the witnesses \( \vec{a}(1), \vec{b}(1), \vec{a}(2), \vec{b}(2) \). They satisfy

\[
g^{x(1)}(-c_{\text{orig}} + \sum_{i=1}^{n} a_i^{(1)} b_i^{(1)}). \prod_{i=1}^{n} g_i^{a_i^{(1)}} h_i^{b_i^{(1)}} = P_{\text{orig}}\]
\[
g^{x(2)}(-c_{\text{orig}} + \sum_{i=1}^{n} a_i^{(2)} b_i^{(2)}). \prod_{i=1}^{n} g_i^{a_i^{(2)}} h_i^{b_i^{(2)}} = P_{\text{orig}}\]
Soundness

- Run the modified protocol twice, with $u_1 = g^{x^{(1)}}$ and $u_2 = g^{x^{(2)}}$, for some $g \in G$.
- Extract the witnesses $\vec{a}^{(1)}, \vec{b}^{(1)}, \vec{a}^{(2)}, \vec{b}^{(2)}$. They satisfy

$$g^{x^{(1)}}(-c_{\text{orig}} + \sum_{i=1}^{n} a_i^{(1)} b_i^{(1)}) - x^{(2)}(-c_{\text{orig}} + \sum_{i=1}^{n} a_i^{(2)} b_i^{(2)}) \cdot \prod_{i=1}^{n} g_i^{a_i^{(1)} - a_i^{(2)}} h_i^{b_i^{(1)} - b_i^{(2)}} = 1$$

Hence $\vec{a}^{(1)} = \vec{a}^{(2)}$ and $\vec{b}^{(1)} = \vec{b}^{(2)}$. Otherwise, we have a non-trivial DL relation $\sum_{i=1}^{n} a_i^{(1)} b_i^{(1)} = c_{\text{orig}}$. The original equation now also gives $\prod_{i=1}^{n} g_i^{a_i^{(1)} - a_i^{(2)}} h_i^{b_i^{(1)} - b_i^{(2)}} = 1$.
Soundness

- Run the modified protocol twice, with \( u_1 = g^{x,(1)} \) and \( u_2 = g^{x,(2)} \), for some \( g \in \mathbb{G} \).
- Extract the witnesses \( \vec{a}^{(1)}, \vec{b}^{(1)}, \vec{a}^{(2)}, \vec{b}^{(2)} \). They satisfy

\[
g^{x,(1)}(-c_{\text{orig}}+\sum_{i=1}^{n} a_i^{(1)} b_i^{(1)}) - x^{(2)}(-c_{\text{orig}}+\sum_{i=1}^{n} a_i^{(2)} b_i^{(2)}) \cdot \prod_{i=1}^{n} g_i^{a_i^{(1)}-a_i^{(2)}} h_i^{b_i^{(1)}-b_i^{(2)}} = 1
\]

- Hence \( \vec{a}^{(1)} = \vec{a}^{(2)} \) and \( \vec{b}^{(1)} = \vec{b}^{(2)} \). Otherwise, we have a non-trivial DL relation.
Soundness

- Run the modified protocol twice, with \( u_1 = g^{x(1)} \) and \( u_2 = g^{x(2)} \), for some \( g \in \mathbb{G} \).
- Extract the witnesses \( \vec{a}(1), \vec{b}(1), \vec{a}(2), \vec{b}(2) \). They satisfy
  \[
  g^{x(1)}(-c_{\text{orig}} + \sum_{i=1}^{n} a_i^{(1)} b_i^{(1)}) - x^{(2)}(-c_{\text{orig}} + \sum_{i=1}^{n} a_i^{(2)} b_i^{(2)}) \prod_{i=1}^{n} g_i^{a_i^{(1)}-a_i^{(2)}} h_i^{b_i^{(1)}-b_i^{(2)}} = 1
  \]
- Hence \( \vec{a}(1) = \vec{a}(2) \) and \( \vec{b}(1) = \vec{b}(2) \). Otherwise, we have a non-trivial DL relation
  \[
  g^{x(1)}(-c_{\text{orig}} + \sum_{i=1}^{n} a_i^{(1)} b_i^{(1)}) - x^{(2)}(-c_{\text{orig}} + \sum_{i=1}^{n} a_i^{(1)} b_i^{(1)}) = 1
  \]
Soundness

- Run the modified protocol twice, with $u_1 = g^{x(1)}$ and $u_2 = g^{x(2)}$, for some $g \in \mathbb{G}$.
- Extract the witnesses $\vec{a}'(1)$, $\vec{b}'(1)$, $\vec{a}'(2)$, $\vec{b}'(2)$. They satisfy

$$g^{x(1)}(-c_{\text{orig}} + \sum_{i=1}^{n} a_i^{(1)} b_i^{(1)}) - x^{(2)}(-c_{\text{orig}} + \sum_{i=1}^{n} a_i^{(2)} b_i^{(2)}) \cdot \prod_{i=1}^{n} g_i^{a_i^{(1)} - a_i^{(2)}} h_i^{b_i^{(1)} - b_i^{(2)}} = 1$$

- Hence $\vec{a}'(1) = \vec{a}'(2)$ and $\vec{b}'(1) = \vec{b}'(2)$. Otherwise, we have a non-trivial DL relation

$$(x^{(1)} - x^{(2)})(-c_{\text{orig}} + \sum_{i=1}^{n} a_i^{(1)} b_i^{(1)}) = 0$$
Soundness

- Run the modified protocol twice, with $u_1 = g^{x(1)}$ and $u_2 = g^{x(2)}$, for some $g \in \mathbb{G}$.
- Extract the witnesses $\vec{a}(1)$, $\vec{b}(1)$, $\vec{a}(2)$, $\vec{b}(2)$. They satisfy

$$g^{x(1)}(-c_{\text{orig}} + \sum_{i=1}^{n} a_i^{(1)} b_i^{(1)}) - x^{(2)}(-c_{\text{orig}} + \sum_{i=1}^{n} a_i^{(2)} b_i^{(2)}) \cdot \prod_{i=1}^{n} g_i^{a_i^{(1)} - a_i^{(2)}} h_i^{b_i^{(1)} - b_i^{(2)}} = 1$$

- Hence $\vec{a}(1) = \vec{a}(2)$ and $\vec{b}(1) = \vec{b}(2)$. Otherwise, we have a non-trivial DL relation

$$(x^{(1)} - x^{(2)})(-c_{\text{orig}} + \sum_{i=1}^{n} a_i^{(1)} b_i^{(1)}) = 0$$

- Hence $\sum_{i=1}^{n} a_i^{(1)} b_i^{(1)} = c_{\text{orig}}$. The original equation now also gives

$$\prod_{i=1}^{n} g_i^{a_i^{(1)}} h_i^{b_i^{(1)}} = P_{\text{orig}}$$
Modified inner product argument (again)

- Public elements $g_1, \ldots, g_n, h_1, \ldots, h_n, P, u \in G$
  - $g_1, \ldots, g_n, h_1, \ldots, h_n, u$ come from the CRS
  - No known non-trivial discrete log relations among $u$ and all $g_i, h_i$
- $P$ wants to convince $V$ that he knows $a_1, \ldots, a_n, b_1, \ldots, b_n \in \mathbb{Z}_p$, such that
  \[
  u \sum_{i=1}^{n} a_i b_i \cdot \prod_{i=1}^{n} g_i^{a_i} h_i^{b_i} = P
  \]
- Privacy is still not important
The protocol

Let $m = n/2$. $P$ computes and sends to $V$:

\[
L = u \sum_{i=1}^{m} a_i b_{i+m} \cdot \prod_{i=1}^{m} g_i^{a_i} h_i^{b_{i+m}}
\]

\[
R = u \sum_{i=1}^{m} a_{i+m} b_i \cdot \prod_{i=1}^{m} g_i^{a_{i+m}} h_i^{b_i}
\]

$V$ sends random challenge $x \xleftarrow{\$} \mathbb{Z}_p$

$P$ sends $a'_i = xa_i + x^{-1}a_{i+m}$ and $b'_i = x^{-1}b_i + xb_{i+m}$ to $V$ ($1 \leq i \leq m$)

$V$ checks

\[
L^{x^2} \text{PR}^{x^{-2}} \stackrel{?}{=} u \sum_{i=1}^{m} a'_i b'_i \cdot \prod_{i=1}^{m} g_i^{x^{-1}a'_i} g_{i+m}^{xa'_i} h_i^{xb'_i} h_{i+m}^{x^{-1}b'_i}
\]
Correctness

$$L^{x^2}PR^{x^{-2}} = u \sum_{i=1}^{m} a_i b_{i+m} x^2 \cdot \prod_{i=1}^{m} g_i^{a_i x^2} h_i^{b_i x^2} \times$$

$$u \sum_{i=1}^{m} (a_i b_i + a_{i+m} b_{i+m}) \cdot \prod_{i=1}^{m} g_i^{a_i} g_i^{a_{i+m}} h_i^{b_i} h_i^{b_{i+m}} \cdot u \sum_{i=1}^{m} a_{i+m} b_i x^{-2} \cdot \prod_{i=1}^{m} g_i^{a_{i+m} x^{-2}} h_i^{b_i x^{-2}} =$$

$$u \sum_{i=1}^{m} (a_i x + a_{i+m} x^{-1})(b_i x^{-1} + b_{i+m} x) \cdot \prod_{i=1}^{m} g_i^{a_i + a_{i+m} x^{-2}} g_i^{a_{i+m} x^2} h_i^{b_i + b_{i+m} x^2} h_i^{b_i x^{-2} + b_{i+m}} =$$

$$u \sum_{i=1}^{m} a_i' b_i' \cdot \prod_{i=1}^{m} g_i^{x^{-1} a_i'} g_i^{a_i'} g_i^{b_i'} g_i^{b_i x^{-1} b_i'}$$

Because \( a_i' = a_i x + a_{i+m} x^{-1} \), \( b_i' = b_i x^{-1} + b_{i+m} x \), \( P = u \sum_{j=1}^{n} a_j b_j \cdot \prod_{j=1}^{n} g_j^{a_j} h_j^{b_j} \)
Recursion

- $P$ has to convince $V$ that he knows $a'_i, b'_i$, such that

$$L^x PR^{x-2} \equiv u \sum_{i=1}^m a'_i b'_i \cdot \prod_{i=1}^m g_i^{-1} a'_i x a'_i g_{i+m}^{-1} x b'_i x^{-1} b'_i$$

$$= u \sum_{i=1}^m a'_i b'_i \cdot \prod_{i=1}^m (g_i^{x-1} g_i^{x+m}) a'_i (h_i^{x} h_i^{x+m})^{-1} b'_i$$

- The same inner product argument, same $u$, changed $P$, new $g_i, h_i$, halved $n$

- Do log $n$ steps:
  - $P$ sends two elements of $G$ at each step
  - $V$ sends an element of $\mathbb{Z}_p$ (except for the last step)
  - After the last step, $V$ does all verifications (the computations can be optimized)
Soundness

- Get a forked transcript
  \[ L, R, x_I, \vec{a}_I, \vec{b}_I, x_{II}, \vec{a}_{II}, \vec{b}_{II}, x_{III}, a'_{III}, b'_{III}, x_{IV}, \vec{a}_{IV}, \vec{b}_{IV} \]
  where \( x_I^2, x_{II}^2, x_{III}^2, x_{IV}^2 \) are all different

- They satisfy (for \( q \in \{I, II, III, IV\} \))
  \[
  L x_q^2 PR x_q^{−2} = \mu \sum_{i=1}^{m} a'_{q,i} b'_{q,i} \prod_{i=1}^{m} \left( \frac{x_q^{-1} x_q}{g_i g_i+m} \right)^{a'_{q,i}} \left( \frac{h_i x_q x_q^{-1}}{h_i h_i+m} \right)^{b'_{q,i}}
  \]

- Let \( \nu_I, \nu_{II}, \nu_{III} \) satisfy
  \[
  \sum_{q=1}^{III} \nu_q x_q^2 = 1 \quad \sum_{q=1}^{III} \nu_q = 0 \quad \sum_{q=1}^{III} \nu_q x_q^{−2} = 0
  \]
Linear combination gives...

\[
L = \prod_{q=1}^{III} L^{\nu_q x_q^2} P^{\nu_q R^{\nu_q x_q^{-2}}}
\]
\[
= \prod_{q=1}^{III} \left( u \sum_{i=1}^{m} a'_q, i b'_q, i \cdot \prod_{i=1}^{m} \left( g_i x_i^{-1} g_i + m \right)^{a'_q, i} \left( h_i x_i h_i + m \right)^{b'_q, i} \right)^{\nu_q}
\]
\[
= u \sum_{q=1}^{III} \sum_{i=1}^{m} \nu_q a'_q, i b'_q, i
\]
\[
\times \prod_{i=1}^{m} g_i \sum_{q=1}^{III} \nu_q x_q^{-1} a'_q, i h_i \sum_{q=1}^{III} \nu_q x_q b'_q, i \prod_{i=1}^{m} g_i + m \sum_{q=1}^{III} \nu_q x_q a'_q, i h_i + m \sum_{q=1}^{III} \nu_q x_q^{-1} b'_q, i
\]
\[
=: u^{c_L} \cdot \prod_{j=1}^{n} g_j^{a_{L, j}} h_j^{b_{L, j}}
\]
Representations of $L$, $R$, $P$

If we let $\nu_I$, $\nu_{II}$, $\nu_{III}$ satisfy different systems of linear equations, we will also get

$$R = u^c_R \cdot \prod_{j=1}^{n} g_{j}^{a_{R,j}} h_{j}^{b_{R,j}}$$

$$P = u^c_P \cdot \prod_{j=1}^{n} g_{j}^{a_{P,j}} h_{j}^{b_{P,j}}$$

The representation of $P$ almost looks like a witness.

It would be a witness, if $c_P = \sum_{j=1}^{n} a_{P,j} b_{P,j}$

A convention (until the end of discussing the inner product argument)

$i$ ranges from 1 to $m$; $j$ ranges from 1 to $n = 2m$; $q$ ranges over \{I, II, III, IV\}
The powers of $u$, $g_j$, $h_j$ have to be equal (or we have a non-trivial discrete log relation)

We get a number of equations out of this
Equal exponents

\[ c_L x_q^2 + c_P + c_R x_q^{-2} = \sum_{i=1}^{m} a'_{q,i} b'_{q,i} \]

(exponents of \( u \))

\[ a_L, i x_q^2 + a_P, i + a_R, i x_q^{-2} = a'_{q,i} x_q^{-1} \]

(exponents of \( g_i \))

\[ a_{L,i+m} x_q^2 + a_{P,i+m} + a_{R,i+m} x_q^{-2} = a'_{q,i} x_q \]

(exponents of \( g_{i+m} \))

\[ b_L, i x_q^2 + b_P, i + b_R, i x_q^{-2} = b'_{q,i} x_q \]

(exponents of \( h_i \))

\[ b_{L,i+m} x_q^2 + b_{P,i+m} + b_{R,i+m} x_q^{-2} = b'_{q,i} x_q^{-1} \]

(exponents of \( h_{i+m} \))
Equal exponents

\[ c_L x_q^2 + c_P + c_R x_q^{-2} = \sum_{i=1}^{m} a'_{q,i} b'_{q,i} \]  (exponents of \( u \))

\[ a_{L,i} x_q^2 + a_{P,i} + a_{R,i} x_q^{-2} = a'_{q,i} x_q^{-1} \]  (exponents of \( g_i \))

\[ a_{L,i+m} x_q^2 + a_{P,i+m} + a_{R,i+m} x_q^{-2} = a'_{q,i} x_q \]  (exponents of \( g_{i+m} \))

\[ b_{L,i} x_q^2 + b_{P,i} + b_{R,i} x_q^{-2} = b'_{q,i} x_q \]  (exponents of \( h_i \))

\[ b_{L,i+m} x_q^2 + b_{P,i+m} + b_{R,i+m} x_q^{-2} = b'_{q,i} x_q^{-1} \]  (exponents of \( h_{i+m} \))

Take \( x_q \) times the 2nd [5th] equation, \( x_q^{-1} \) times the 3rd [4th] equation and subtract:

\[ a_{L,i} x_q^3 + (a_{P,i} - a_{L,i+m}) x_q + (a_{R,i} - a_{P,i+m}) x_q^{-1} - a_{R,i+m} x_q^{-3} = 0 \]

\[ b_{L,i+m} x_q^3 + (b_{P,i+m} - b_{L,i}) x_q + (b_{R,i+m} - b_{P,i}) x_q^{-1} - b_{R,i} x_q^{-3} = 0 \]

These must be zero polynomials
“These must be zero polynomials…”

For four different values of $x_q$, we have

$$a_{L,i}x_q^3 + (a_{P,i} - a_{L,i+m})x_q + (a_{R,i} - a_{P,i+m})x_q^{-1} - a_{R,i+m}x_q^{-3} = 0$$
“These must be zero polynomials...”

For four different values of $x_q$, we have

\[
a_{L,i} x_q^3 + (a_{P,i} - a_{L,i+m})x_q + \left(a_{R,i} - a_{P,i+m}\right)x_q^{-1} - a_{R,i+m}x_q^{-3} = 0
\]

\[
a_{L,i} x_q^6 + (a_{P,i} - a_{L,i+m})x_q^4 + \left(a_{R,i} - a_{P,i+m}\right)x_q^2 - a_{R,i+m} = 0
\]
“These must be zero polynomials...”

For four different values of $x_q^2$, we have

\[ a_{L,i}x_q^3 + (a_{P,i} - a_{L,i+m})x_q + (a_{R,i} - a_{P,i+m})x_q^{-1} - a_{R,i+m}x_q^{-3} = 0 \]

\[ a_{L,i}x_q^6 + (a_{P,i} - a_{L,i+m})x_q^4 + (a_{R,i} - a_{P,i+m})x_q^2 - a_{R,i+m} = 0 \]

\[ a_{L,i}(x_q^2)^3 + (a_{P,i} - a_{L,i+m})(x_q^2)^2 + (a_{R,i} - a_{P,i+m})x_q^2 - a_{R,i+m} = 0 \]
“These must be zero polynomials...”

For four different values of \(x^2_q\), we have

\[
a_{L,i}x^3_q + (a_{P,i} - a_{L,i+m})x_q + (a_{R,i} - a_{P,i+m})x^{-1}_q - a_{R,i+m}x^{-3}_q &= 0 \\
a_{L,i}x^6_q + (a_{P,i} - a_{L,i+m})x^4_q + (a_{R,i} - a_{P,i+m})x^2_q - a_{R,i+m} &= 0 \\
a_{L,i}(x^2_q)^3 + (a_{P,i} - a_{L,i+m})(x^2_q)^2 + (a_{R,i} - a_{P,i+m})x^2_q - a_{R,i+m} &= 0
\]

A non-zero cubic polynomial can have at most three roots over a field
Equal exponents, again

2nd–5th equations

\[ a_{L,i}x_q^2 + a_{P,i} + a_{R,i}x_q^{-2} = a'_{q,i}x_q^{-1} \]
\[ a_{L,i+m}x_q^2 + a_{P,i+m} + a_{R,i+m}x_q^{-2} = a'_{q,i}x_q \]
\[ b_{L,i}x_q^2 + b_{P,i} + b_{R,i}x_q^{-2} = b'_{q,i}x_q \]
\[ b_{L,i+m}x_q^2 + b_{P,i+m} + b_{R,i+m}x_q^{-2} = b'_{q,i}x_q^{-1} \]

Zero polynomials

\[ a_{L,i} = 0 \quad a_{R,i} = a_{P,i+m} \]
\[ a_{R,i+m} = 0 \quad a_{L,i+m} = a_{P,i} \]
\[ b_{R,i} = 0 \quad b_{L,i} = b_{P,i+m} \]
\[ b_{L,i+m} = 0 \quad b_{R,i+m} = b_{P,i} \]
Equal exponents, again

2nd–5th equations

\[
\begin{align*}
  a_{L,i}x_q^2 + a_{P,i} + a_{R,i}x_q^{-2} &= a'_{q,i}x_q^{-1} \\
  a_{L,i+m}x_q^2 + a_{P,i+m} + a_{R,i+m}x_q^{-2} &= a'_{q,i}x_q \\
  b_{L,i}x_q^2 + b_{P,i} + b_{R,i}x_q^{-2} &= b'_{q,i}x_q \\
  b_{L,i+m}x_q^2 + b_{P,i+m} + b_{R,i+m}x_q^{-2} &= b'_{q,i}x_q^{-1}
\end{align*}
\]

Zero polynomials

\[
\begin{align*}
  a_{L,i} &= 0 & a_{R,i} &= a_{P,i+m} \\
  a_{R,i+m} &= 0 & a_{L,i+m} &= a_{P,i} \\
  b_{R,i} &= 0 & b_{L,i} &= b_{P,i+m} \\
  b_{L,i+m} &= 0 & b_{R,i+m} &= b_{P,i}
\end{align*}
\]
Equal exponents, again

2nd–5th equations

\[ a_{P,i} + a_{R,i} x_q^{-2} = a'_{q,i} x_q^{-1} \]
\[ a_{L,i+m} x_q^2 + a_{P,i+m} = a'_{q,i} x_q \]
\[ b_{L,i} x_q^2 + b_{P,i} = b'_{q,i} x_q \]
\[ b_{P,i+m} + b_{R,i+m} x_q^{-2} = b'_{q,i} x_q^{-1} \]

Zero polynomials

\[ a_{L,i} = 0 \]
\[ a_{R,i} = a_{P,i+m} \]
\[ a_{R,i+m} = 0 \]
\[ a_{L,i+m} = a_{P,i} \]
\[ b_{R,i} = 0 \]
\[ b_{L,i} = b_{P,i+m} \]
\[ b_{L,i+m} = 0 \]
\[ b_{R,i+m} = b_{P,i} \]
Equal exponents, again

2nd–5th equations
\[ a_{P,i} + a_{R,i} x_q^{-2} = a'_{q,i} x_q^{-1} \]
\[ a_{L,i+m} x_q^2 + a_{P,i+m} = a'_{q,i} x_q \]
\[ b_{L,i} x_q^2 + b_{P,i} = b'_{q,i} x_q \]
\[ b_{P,i+m} + b_{R,i+m} x_q^{-2} = b'_{q,i} x_q^{-1} \]

Zero polynomials
\[ a_{L,i} = 0 \]
\[ a_{R,i} = a_{P,i+m} \]
\[ a_{R,i+m} = 0 \]
\[ a_{L,i+m} = a_{P,i} \]
\[ b_{R,i} = 0 \]
\[ b_{L,i} = b_{P,i+m} \]
\[ b_{L,i+m} = 0 \]
\[ b_{R,i+m} = b_{P,i} \]
Equal exponents, again

2nd–5th equations
\[a_{P,i} + a_{P,i+m} x_q^{-2} = a'_{q,i} x_q^{-1}\]
\[a_{P,i} x_q^2 + a_{P,i+m} = a'_{q,i} x_q\]
\[b_{P,i+m} x_q^2 + b_{P,i} = b'_{q,i} x_q\]
\[b_{P,i+m} + b_{P,i} x_q^{-2} = b'_{q,i} x_q^{-1}\]

Zero polynomials
\[a_{L,i} = 0 \quad a_{R,i} = a_{P,i+m}\]
\[a_{R,i+m} = 0 \quad a_{L,i+m} = a_{P,i}\]
\[b_{R,i} = 0 \quad b_{L,i} = b_{P,i+m}\]
\[b_{L,i+m} = 0 \quad b_{R,i+m} = b_{P,i}\]
Equal exponents, again

- 2nd and 5th equation:
  \[ a'_q,ix_q^{-1} = a_{P,i} + a_{P,i+m}x_q^{-2} \quad b'_q,ix_q^{-1} = b_{P,i+m} + b_{P,i}x_q^{-2} \]

- Multiply both sides by \( x_q \)
  \[ a'_q,i = a_{P,i}x_q + a_{P,i+m}x_q^{-1} \quad b'_q,i = b_{P,i}x_q^{-1} + b_{P,i+m}x_q \]

(would get the same from 3rd and 4th equations)
\( \vec{a}_P, \vec{b}_P \) is the witness

\[
a'_{q,i} = a_{P,i} x_q + a_{P,i + m} x_q^{-1} \quad b'_{q,i} = b_{P,i} x_q^{-1} + b_{P,i + m} x_q
\]

The inner product of \( \vec{a}'_q \) and \( \vec{b}'_q \) is

\[
\sum_{i=1}^{m} a'_{q,i} b'_{q,i} = \sum_{i=1}^{m} (a_{P,i} x_q + a_{P,i + m} x_q^{-1})(b_{P,i} x_q^{-1} + b_{P,i + m} x_q)
\]

\[
= x_q^2 \sum_{i=1}^{m} a_{P,i} b_{P,i + m} + \sum_{j=1}^{n} a_{P,j} b_{P,j} + x_q^{-2} \sum_{i=1}^{m} a_{P,i + m} b_{P,i}
\]
$\vec{a}_P, \vec{b}_P$ is the witness

\[ a'_{q,i} = a_P, i x_q + a_P, i + m x_q^{-1} \quad b'_{q,i} = b_P, i x_q^{-1} + b_P, i + m x_q \]

The inner product of $a'_q$ and $b'_q$ is

\[
\sum_{i=1}^{m} a'_{q,i} b'_{q,i} = \sum_{i=1}^{m} (a_P, i x_q + a_P, i + m x_q^{-1})(b_P, i x_q^{-1} + b_P, i + m x_q) \\
= x_q^2 \sum_{i=1}^{m} a_P, i b_P, i + m + \sum_{j=1}^{n} a_P, j b_P, j + x_q^{-2} \sum_{i=1}^{m} a_P, i + m b_P, i \\
= \sum_{i=1}^{m} a'_{q,i} b'_{q,i} = c_L x_q^2 + c_P + c_R x_q^{-2} \quad (1st \ equation) \\
\]

These polynomials have to be equal. Free terms give $c_P = \sum_{j=1}^{n} a_P, j b_P, j$
Soundness of recursive protocol

- To get a witness of length $n$, we need four executions (and witnesses) of length $n/2$
- To get a witness of length $n/2$, we need four executions (and witnesses) of length $n/4$
- etc.
- To get a witness of length $n$, we need $4^\log_2 n \approx n^2$ executions
Representing arithmetic circuits

- There are $n$ (binary) multiplication gates
  - $i$-th one has inputs $a_{L,i}$ and $a_{R,i}$, output $a_{O,i}$
  - These three values per multiplication gate are the witness

- There are $Q$ affine relationships between $a_{L,i}$, $a_{R,i}$, $a_{O,i}$

$$\sum_{i=1}^{n} w_{L,q,i} a_{L,i} + \sum_{i=1}^{n} w_{R,q,i} a_{R,i} + \sum_{i=1}^{n} w_{O,q,i} a_{O,i} = c_q \quad (1 \leq q \leq Q)$$

- The coefficients $w_{L,q,i}$, $w_{R,q,i}$, $w_{O,q,i}$ and $c_q$ are part of the instance
Representing arithmetic circuits

- There are $n$ (binary) multiplication gates
  - $i$-th one has inputs $a_{L,i}$ and $a_{R,i}$, output $a_{O,i}$
  - These three values per multiplication gate are the witness

- There are $Q$ affine relationships between $a_{L,i}$, $a_{R,i}$, $a_{O,i}$
  \[
  \sum_{i=1}^{n} w_{L,q,i} a_{L,i} + \sum_{i=1}^{n} w_{R,q,i} a_{R,i} + \sum_{i=1}^{n} w_{O,q,i} a_{O,i} = c_q \quad (1 \leq q \leq Q)
  \]

- The coefficients $w_{L,q,i}$, $w_{R,q,i}$, $w_{O,q,i}$ and $c_q$ are part of the instance

- Some Pedersen commitments $C_1, \ldots, C_m$ could be a part of the instance
  - Messages $v_1, \ldots, v_m$ and randomnesses are part of the witness
  - Messages can show up in the affine relationships
Representing arithmetic circuits

- There are $n$ (binary) multiplication gates
  - $i$-th one has inputs $a_{L,i}$ and $a_{R,i}$, output $a_{O,i}$
  - These three values per multiplication gate are the witness

- There are $Q$ affine relationships between $a_{L,i}$, $a_{R,i}$, $a_{O,i}$

\[ \sum_{i=1}^{n} w_{L,q,i} a_{L,i} + \sum_{i=1}^{n} w_{R,q,i} a_{R,i} + \sum_{i=1}^{n} w_{O,q,i} a_{O,i} + \sum_{j=1}^{m} w_{V,q,j} v_j = c_q \quad (1 \leq q \leq Q) \]

- The coefficients $w_{L,q,i}$, $w_{R,q,i}$, $w_{O,q,i}$, $w_{V,q,j}$ and $c_q$ are part of the instance

- Some Pedersen commitments $C_1, \ldots, C_m$ could be a part of the instance
  - Messages $v_1, \ldots, v_m$ and randomnesses are part of the witness
  - Messages can show up in the affine relationships
  - (I won’t talk about them here)
Start of the protocol

- CRS contains \( g_1, \ldots, g_n, h_1, \ldots, h_n, h \in \mathbb{G} \)
- \( P \) picks \( \alpha, \beta \leftarrow \mathbb{Z}_p \); computes and sends to \( V \)

\[
A_I = h^\alpha \cdot \prod_{i=1}^{n} g_i^{a_{L,i}} h_i^{a_{R,i}}
\]

\[
A_O = h^\beta \cdot \prod_{i=1}^{n} g_i^{a_{O,i}}
\]

(Pedersen vector commitments)
Many equations to one

\[ a_{L,i} a_{R,i} - a_{O,i} = 0 \quad (1 \leq i \leq n) \]

\[ \sum_{i=1}^{n} w_{L,q,i} a_{L,i} + \sum_{i=1}^{n} w_{R,q,i} a_{R,i} + \sum_{i=1}^{n} w_{O,q,i} a_{O,i} = c_q \quad (1 \leq q \leq Q) \]
Many equations to one

\[ a_{L,i}a_{R,i} - a_{O,i} = 0 \quad (1 \leq i \leq n) \]

\[ \sum_{i=1}^{n} w_{L,q,i}a_{L,i} + \sum_{i=1}^{n} w_{R,q,i}a_{R,i} + \sum_{i=1}^{n} w_{O,q,i}a_{O,i} = c_{q} \quad (1 \leq q \leq Q) \]

Turn it to a single polynomial equation (variables \( Y, Z \))

\[
\sum_{i=1}^{n} \left( a_{L,i}a_{R,i} - a_{O,i} \right) Y^{i-1} + \\
\sum_{q=1}^{Q} \left( \sum_{i=1}^{n} w_{L,q,i}a_{L,i} + \sum_{i=1}^{n} w_{R,q,i}a_{R,i} + \sum_{i=1}^{n} w_{O,q,i}a_{O,i} \right) Z^{q} = \sum_{q=1}^{Q} c_{q} Z^{q}
\]

\( \text{V picks } y, z \overset{\$}{\leftarrow} \mathbb{Z}_p, \text{ sends them to } P \)
## Committing to a polynomial

### Functionality
- $P$ becomes bound to a polynomial $f \in \mathbb{Z}_p[X]
- V$ picks a value $x \in X$
- $P$ gives $f(x)$ to $V$ and convinces him of its correctness

### A naïve implementation (sufficient for us)
- $P$ commits to all coefficients of $f$, using Pedersen commitments
- $V$ sends $x$ to $P$
- Both compute commitment to $f(x)$, as the linear combination of commitments to coefficients
- $P$ opens $f(x)$ to $V$
What happens next? Arguments with polynomials...

- $P$ substitutes $y, z$ for $Y, Z$
- Define polynomials $\ell_i(X), r_i(X)$ $(1 \leq i \leq n)$ so, that
  - denote $t(X) = \sum_i \ell_i(X) r_i(X)$
  - The coefficient of $X^2$ in $t(X)$ is the LHS of the equation three slides ago (almost)
  - For given $x \in \mathbb{Z}_p$, the verifier (using $A_I, A_O$) can compute smth. like

$$C = h^{\text{smth}} \cdot \prod_{i=1}^{n} g_i^{\ell_i(x)} h_i^{r_i(x)}$$ (like a vector commitment to $\{ \ell_i(x), r_i(x) \}_{i=1}^{n}$)

- $P$ commits to $t(X)$. Shows, the coefficient of $X^2$ is almost $\sum q z^q c_q$
- $V$ challenges with $x \overset{\$}{\leftarrow} \mathbb{Z}_p$
- $P$ opens $\ell_i(x), r_i(x)$ for all $i$ (i.e. opens $C$)
- $P$ also opens $t(x)$. $V$ checks that $t(x) = \sum_i \ell_i(x) r_i(x)$
The polynomials

\[
\ell_i(X) = a_{L,i}X + a_{O,i}X^2 + y^{-i+1}\left(\sum_{q=1}^{Q} w_{R,q,i}z^q\right)X \\
r_i(X) = y^{i-1}a_{R,i}X - y^{i-1} + \left(\sum_{q=1}^{Q} w_{L,q,i}z^q\right)X + \left(\sum_{q=1}^{Q} w_{O,q,i}z^q\right)
\]

The coefficient of \(X^2\) in \(t(X) = \sum_i \ell_i(X)r_i(X)\) is

\[
\sum_{i=1}^{n} \left[ \left( a_{L,i} + y^{-i+1}\left(\sum_{q=1}^{Q} w_{R,q,i}z^q\right) \right) \left( y^{i-1}a_{R,i} + \left(\sum_{q=1}^{Q} w_{L,q,i}z^q\right) \right) + a_{O,i} \left( -y^{i-1} + \left(\sum_{q=1}^{Q} w_{O,q,i}z^q\right) \right) \right]
\]
The polynomials

The coefficient of $X^2$ in $t(X) = \sum_i \ell_i(X) r_i(X)$ is

$$\sum_{i=1}^{n} \left[ \left( a_{L,i} + y^{-i+1} \left( \sum_{q=1}^{Q} w_{R,q,i} z^q \right) \right) \left( y^{i-1} a_{R,i} + \left( \sum_{q=1}^{Q} w_{L,q,i} z^q \right) \right) + a_{O,i} \left( -y^{i-1} + \left( \sum_{q=1}^{Q} w_{O,q,i} z^q \right) \right) \right]$$

Which equals

$$\sum_{q=1}^{Q} \left( \sum_{i=1}^{n} w_{L,q,i} a_{L,i} + \sum_{i=1}^{n} w_{R,q,i} a_{R,i} + \sum_{i=1}^{n} w_{O,q,i} a_{O,i} \right) z^q + \sum_{i=1}^{n} (a_{L,i} a_{R,i} - a_{O,i}) y^{i-1} + \sum_{i=1}^{n} y^{-i+1} \left( \sum_{q=1}^{Q} w_{R,q,i} z^q \right) \left( \sum_{q=1}^{Q} w_{L,q,i} z^q \right)$$
The polynomials

The coefficient of $X^2$ in $t(X) = \sum_i \ell_i(X) r_i(X)$ is

$$\sum_{i=1}^n \left[ \left( a_{L,i} + y^{-i+1} \left( \sum_{q=1}^Q w_{R,q,i} z_q \right) \right) \left( y^{-1} a_{R,i} + \left( \sum_{q=1}^Q w_{L,q,i} z_q \right) \right) + a_{O,i} \left( -y^{-i+1} + \left( \sum_{q=1}^Q w_{O,q,i} z_q \right) \right) \right]$$

Which equals

$$\sum_{q=1}^Q \left( \sum_{i=1}^n w_{L,q,i} a_{L,i} + \sum_{i=1}^n w_{R,q,i} a_{R,i} + \sum_{i=1}^n w_{O,q,i} a_{O,i} \right) z^q + \sum_{i=1}^n (a_{L,i} a_{R,i} - a_{O,i}) y^{-i+1} + \sum_{i=1}^n y^{-i+1} \left( \sum_{q=1}^Q w_{R,q,i} z_q \right) \left( \sum_{q=1}^Q w_{L,q,i} z_q \right)$$
Committing to $t$ and opening

- $P$ commits to coefficients of $X, X^3$
- $V$ computes the commitment to the coefficient of $X^2$ himself
  - This coefficient is
    \[
    \sum_{q=1}^{Q} c_q z^q + \sum_{i=1}^{n} y^{-i+1} \left( \sum_{q=1}^{Q} w_{R,q,i} z^q \right) \left( \sum_{q=1}^{Q} w_{L,q,i} z^q \right)
    \]
  - Using $h^0$ as the blinding factor
- $V$ sends the challenge $x$
- $P$ sends $t(x)$ to $V$, as well as the blinding exponent
  - Computed from the blinding exponents of the coefficients
Commitment to points on polynomials $\ell_i, r_i$

\[
\ell_i(x) = a_{L,i}x + a_{O,i}x^2 + y^{-i+1}\left(\sum_{q=1}^{Q} w_{R,q}i^q\right)x
\]

\[
r_i(x) = y^{i-1}a_{R,i}x - y^{i-1} + \left(\sum_{q=1}^{Q} w_{L,q}i^q\right)x + \left(\sum_{q=1}^{Q} w_{O,q}i^q\right)
\]

The commitment, computed by $V$, is

\[
A_1^x \cdot A_2^{x^2} \cdot \prod_{i=1}^{n} g_i^{y^{-i+1}\left(\sum_{q=1}^{Q} w_{R,q}i^q\right)}h_i^{y^{i-1}\left(\sum_{q=1}^{Q} w_{L,q}i^q\right)} + \left(\sum_{q=1}^{Q} w_{O,q}i^q\right)
\]

...but not quite...
Change the CRS

- Think of the CRS containing $h'_i = h_i^{y^{-i+1}}$, instead of $h_i$
- We had $A_1 = h^\alpha \cdot \prod_{i=1}^n g_i^{a_L,i} h_i^{a_R,i}$. This equals $A_1 = h^\alpha \cdot \prod_{i=1}^n g_i^{a_L,i} h_i^{y^{-i+1}a_R,i}$
- The whole commitment $C$ is

$$C = A_1^x \cdot A_0^{x^2} \cdot \prod_{i=1}^n g_i^{y^{-i+1}} \left( \sum_{q=1}^Q w_{R,q,i} z^q \right) x - y^{-i+1} + \left( \sum_{q=1}^Q w_{L,q,i} z^q \right) x + \left( \sum_{q=1}^Q w_{O,q,i} z^q \right)$$

- The blinding exponent of Pedersen’s commitment is $\alpha x + \beta x^2$
- $P$ opens $C$ as $\ell_1(x), r_1(x), \ldots, \ell_n(x), r_n(x)$
- $V$ checks correct opening, also checks that $t(x) = \sum_{i=1}^n \ell_i(x) r_i(x)$
Blinding

Problem: \( \ell_i(x), r_i(x), t(x) \) leak about \( a_{L,i}, a_{R,i}, a_{O,i} \)

Solution

- In the beginning, \( P \) also generates \( \vec{s}_L, \vec{s}_R \in \mathbb{Z}_p^n \)
- Commits to them:
  - Generates \( \rho \xleftarrow{\$} \mathbb{Z}_p \)
  - Sends \( A_S = h^\rho \cdot \prod_{i=1}^{n} g_i^{s_{L,i}} h_i^{s_{R,i}} \) to \( V \), together with \( A_I \) and \( A_O \)
Blinding of $\ell_i$, $r_i$

$$
\ell_i(X) = a_{L,i}X + a_{O,i}X^2 + y^{-i+1} \left( \sum_{q=1}^{Q} w_{R,q,i}z^q \right) X \\
+ s_{L,i}X^3
$$

$$
r_i(X) = y^{i-1}a_{R,i}X - y^{i-1} + \left( \sum_{q=1}^{Q} w_{L,q,i}z^q \right) X + \left( \sum_{q=1}^{Q} w_{O,q,i}z^q \right) \\
+ y^{i-1}s_{R,i}X^3
$$
Changes to the construction, due to blinding

- Polynomial $t$: now has degree 6
- No change to coefficient of $X^2$
- Commitment $C$ includes the factor $A_S^3$

$$C = A_I^x \cdot A_O^x^2 \cdot A_S^x^3 \cdot \prod_{i=1}^{n} g_i \cdot y^{-i+1} \left( \sum_{q=1}^{Q} w_{R,q,i} z^q \right) x - y^{-i-1} + \left( \sum_{q=1}^{Q} w_{L,q,i} z^q \right) x + \left( \sum_{q=1}^{Q} w_{O,q,i} z^q \right)$$

- ...and the blinding exponent adds $\rho x^3$
Whole protocol

- The CRS contains \( g_1, \ldots, g_n, h_1, \ldots, h_n, h \in \mathbb{G} \)
- \( P \) computes and sends \( A_1, A_O, A_S \)
- \( V \) sends \( y, z \); both can now compute \( h'_1, \ldots, h'_n \)
- \( P \) sends commitments to the coefficients of \( X, X^3, X^4, X^5, X^6 \) in \( t(X) \)
- \( V \) sends \( x \); both compute commitment to \( t(x) \); both compute \( C \)
- \( P \) opens commitment to \( t(x) \)
- \( P \) sends \( \ell_i(x), r_i(x) \) for \( 1 \leq i \leq n \), and the blinding exponent \( \sigma = \alpha x + \beta x^2 + \rho x^3 \)
- \( V \) checks that

\[
t(x) = \sum_{i=1}^{n} \ell_i(x) r_i(x) \quad \text{and} \quad C/h^\sigma = \prod_{i=1}^{n} g_i^{\ell_i(x)} h_i' r_i(x)
\]
Whole protocol

- The CRS contains $g_1, \ldots, g_n, h_1, \ldots, h_n, h \in \mathbb{G}$
- $P$ computes and sends $A_I, A_O, A_S$
- $V$ sends $y, z$; both can now compute $h'_1, \ldots, h'_n$
- $P$ sends commitments to the coefficients of $X, X^3, X^4, X^5, X^6$ in $t(X)$
- $V$ sends $x$; both compute commitment to $t(x)$; both compute $C$
- $P$ opens commitment to $t(x)$
- $P$ sends $\ell_i(x), r_i(x)$ for $1 \leq i \leq n$, and the blinding exponent $\sigma = \alpha x + \beta x^2 + \rho x^3$
- $V$ checks that

$$t(x) = \sum_{i=1}^{n} \ell_i(x) r_i(x) \quad \text{and} \quad C / h^\sigma = \prod_{i=1}^{n} g_i^{\ell_i(x)} h'_i r_i(x)$$

...using the inner product argument
About the security proof

- Completeness — hopefully I did convince you in this
- Zero-knowledge — easy. There’s sufficient randomization everywhere
- Soundness — similar to the inner product proof:
  - Find $\bar{a}_L, \bar{a}_R, \bar{a}_O$ as before
  - Get so many transcripts with different witnesses, that the equations between values of polynomials become equations between polynomials
  - 7 different $x$-s, $n$ different $y$-s, $(Q + 1)$ different $z$-s
    - $7(Q + 1)n \approx O(n^2)$ in total, still a small number
About the security proof

- Completeness — hopefully I did convince you in this
- Zero-knowledge — easy. There’s sufficient randomization everywhere
- Soundness — similar to the inner product proof:
  - Find $\vec{a}_L, \vec{a}_R, \vec{a}_O$ as before
  - Get so many transcripts with different witnesses, that the equations between values of polynomials become equations between polynomials
  - 7 different $x$-s, $n$ different $y$-s, $(Q + 1)$ different $z$-s
    - $7(Q + 1)n \approx O(n^2)$ in total, still a small number

Efficiency of the witness extractor

- Together with inner product proof, needs $O(n^2) \cdot O(n^2) = O(n^4)$ transcripts
  - Disc. log. in $\mathbb{G}$ must survive the attack of complexity $O(n^4)$
- I wonder if instead of $(Q + 1)n$ different $y$-s and $z$-s, we could manage with $(Q + 1 + n)$ different $y$-s and $z$-s...
Linear PCPs
Linear PCPs (LPCP)

- The prover prepares a proof string $\vec{\pi}$ of length $n$
- Each entry of $\pi$ is from the field $\mathbb{F}$
- The verifier’s queries are vectors $\vec{q} \in \mathbb{F}^n$
- The answers are the inner products $\langle \vec{\pi}, \vec{q} \rangle$
Linear PCPs (LPCP)

- The prover prepares a proof string $\vec{\pi}$ of length $n$.
- Each entry of $\pi$ is from the field $\mathbb{F}$.
- The verifier’s queries are vectors $\vec{q} \in \mathbb{F}^n$.
- The answers are the inner products $\langle \vec{\pi}, \vec{q} \rangle$.
- Depending on the cryptographic realization, $V$ has or has not to be ready for
  - The answers not being computed linearly from queries.
  - Different answers being computed using different linear functions.
Committing to a linear PCP

- Let there be an additively homomorphic encryption scheme \((E, D)\)
  - Only a single keypair is in use. Verifier has the private key
  - Plaintext space is \(F\)
  - Let \(\boxplus\) denote addition and \(\Box\) constant multiplication

Commitment

- The prover has \(\vec{\pi} = (\pi_1, \ldots, \pi_n)\)
- The verifier randomly generates \(r_1, \ldots, r_n \leftarrow F\)
- \(V \rightarrow P : E(r_1), \ldots, E(r_n)\). Denote this operation by \(E(\vec{r})\)
- \(P \rightarrow V : \boxplus_{i=1}^n \pi_i \Box E(r_i)\). Denote this operation by \([\langle \vec{\pi}, E(\vec{r}) \rangle]\)
- Verifier decrypts. Denote \(s = \langle \vec{\pi}, \vec{r} \rangle\)
Querying a committed PCP

- $V$ wants to make $k$ queries $\vec{q}_1, \ldots, \vec{q}_k$
- Let all queries be made at the same time. I.e. $V$ is non-adaptive
- $V$ picks $\alpha_1, \ldots, \alpha_k \xleftarrow{\$} F$, defines $\vec{q}_{k+1} := \vec{r} + \sum_{i=1}^{k} \alpha_i \cdot \vec{q}_i$
- $V \to P : \vec{q}_1, \ldots, \vec{q}_{k+1}$
- $P \to V : a_1, \ldots, a_{k+1}$, where $a_i = \langle \vec{r}, \vec{q}_i \rangle$
- $V$ checks that $a_{k+1} = s + \sum_{i=1}^{k} \alpha_i a_i$

Soundness follows from $\vec{r}$ computationally masking $\vec{q}_{k+1}$
Soundness

- From $P$’s point of view, $\vec{r}$ could be any $\vec{q}_{k+1} - \sum_{i=1}^{k} \alpha_i \cdot \vec{q}_i$, for $\alpha_1, \ldots, \alpha_k \in \mathbb{F}$
- The corresponding $s$ is $\langle \vec{\pi}, \vec{q}_{k+1} \rangle - \sum_{i=1}^{k} \alpha_i \langle \vec{\pi}, \vec{q}_i \rangle + \pi_0$
- $\pi_0$ and $\vec{\pi}$ are defined by $P$’s actions during the commitment
- $P$ must come up with $a_1, \ldots, a_{k+1}$ that satisfy

$$a_{k+1} = \langle \vec{\pi}, \vec{q}_{k+1} \rangle - \sum_{i=1}^{k} \alpha_i \langle \vec{\pi}, \vec{q}_i \rangle + \pi_0 + \sum_{i=1}^{k} \alpha_i a_i$$

$$a_{k+1} - \langle \vec{\pi}, \vec{q}_{k+1} \rangle = \sum_{i=1}^{k} \alpha_i (a_i - \langle \vec{\pi}, \vec{q}_i \rangle) + \pi_0$$

for a significant fraction of possible $(\alpha_1, \ldots, \alpha_k)$
- Hence $P$ should pick $a_i = \langle \vec{\pi}, \vec{q}_i \rangle$ for $i \in \{1, \ldots, k\}$
Linear PCP for CIRCUIT-SAT

- Circuit over $\mathbb{F}$, $m$ input and internal gates ($+$ and $\times$), $t$ outputs, $\ell$ fixed inputs
  - Let $\text{in}_1, \text{in}_2 : \{1, \ldots, m\} \rightarrow \{1, \ldots, m\}$ give the inputs of internal gates
  - Let $\vec{w} \in \mathbb{F}^m$ be a satisfying assignment to gates

- Constraints:
  - For any addition gate $a$: $w_a - w_{\text{in}_1(a)} - w_{\text{in}_2(a)} = 0$
  - For any multiplication gate $a$: $w_a - w_{\text{in}_1(a)} \cdot w_{\text{in}_2(a)} = 0$
  - For input $a$ fixed to the value $x_a$: $w_a - x_a = 0$
  - For output $a$ fixed to the value $y_a$: $w_a - y_a = 0$

The proof string is $\vec{\pi} = \vec{w} \parallel (\vec{w} \otimes \vec{w})$ (where $\parallel$ is concatenation)
Queries

- If \( V \) has to be ready for non-linearity from cheating \( P \):
  - Check for linearity: query \( \vec{\pi} \) at 3 linearly dependent points \( \in \mathbb{F}^{m+m^2} \), compare answers
  - Turn each query below to two queries at random points on a random line through original query, interpolate the answer

- Check Hadamard product: Let \( \vec{q}_1, \vec{q}_2 \leftarrow \mathbb{F}^m \)
  - make the queries \( \vec{q}_1 \| 0^{m^2} \mapsto a_1 \), \( \vec{q}_2 \| 0^{m^2} \mapsto a_2 \), \( 0^m (\vec{q}_1 \otimes \vec{q}_2) \mapsto a_3 \)
  - Check that \( a_1 \cdot a_2 = a_3 \)

- Check the constraints of the circuit
  - Each constraint \( c \) corresponds to a simple query string \( \vec{q}_c \in \mathbb{F}^{m+m^2} \)
    - The expected answer is 0 or \( x_a \) or \( y_a \)
    - Query a single random linear combination of \( \vec{q}_c \)-s
A linear-size LPCP for R1CS
Quadratic Arithmetic Programs (QAP) over $\mathbb{Z}_p$

- A QAP with variables $a_0 = 1, a_1, \ldots, a_m$ is a set of equations of the form

$$\left(\sum_{i=0}^{m} u_{i,q} \cdot a_i\right) \cdot \left(\sum_{i=0}^{m} v_{i,q} \cdot a_i\right) = \left(\sum_{i=0}^{m} w_{i,q} \cdot a_i\right)$$

- $u_{i,q}, v_{i,q}, w_{i,q} \in \mathbb{Z}_p$
- $0 \leq i \leq m$. Let there be $n$ equations, i.e. $1 \leq q \leq n$

Very similar to Rank-1 constraint systems
QAPs with polynomials — motivation

- Let \( r_1, \ldots, r_n \) be distinct elements of \( \mathbb{Z}_p \)
- Define polynomials \( u_i, v_i, w_i \) (\( 0 \leq i \leq m \)) by
  \[
  u_i(r_q) = u_{i,q} \quad v_i(r_q) = v_{i,q} \quad w_i(r_q) = w_{i,q}
  \]
- We want that for each \( r_1, \ldots, r_n \)
  \[
  \left( \sum_{i=0}^{m} a_i u_i(r_q) \right) \cdot \left( \sum_{i=0}^{m} a_i v_i(r_q) \right) - \left( \sum_{i=0}^{m} a_i w_i(r_q) \right) = 0
  \]
- Hence we want the polynomial
  \[
  \left( \sum_{i=0}^{m} a_i u_i(X) \right) \cdot \left( \sum_{i=0}^{m} a_i v_i(X) \right) - \left( \sum_{i=0}^{m} a_i w_i(X) \right)
  \]
  to be divisible with the polynomial \( t(X) = \prod_{i=1}^{n} (X - r_i) \)
QAPs with polynomials — syntax

Components

- Field \( \mathbb{Z}_p \). Numbers \( \ell, m, n \)
- Polynomial \( t \in \mathbb{Z}_p[X] \) of degree \( n \)
- Polynomials \( u_i, v_i, w_i \in \mathbb{Z}_p[X] \) of degree at most \( n - 1 \)
  - \( 0 \leq i \leq m \)

The relation

- Instance: \( (a_0, \ldots, a_\ell) \in \mathbb{Z}_p^{\ell+1} \). Witness: \( (a_{\ell+1}, \ldots, a_m) \in \mathbb{Z}_p^{m-\ell} \)
- Relation: \( a_0 = 1 \) and
  \[
  \left( \sum_{i=0}^{m} a_i u_i(X) \right) \cdot \left( \sum_{i=0}^{m} a_i v_i(X) \right) \equiv \left( \sum_{i=0}^{m} a_i w_i(X) \right) \pmod{t(X)}
  \]
The linear proof

- The proof string (only the witness part)
  - First part: the vector $\vec{a}$
  - Second part: coefficients of the polynomial $h(X)$ of degree $\leq n - 2$, satisfying

$$\left( \sum_{i=0}^{m} a_i u_i(X) \right) \cdot \left( \sum_{i=0}^{m} a_i v_i(X) \right) - \left( \sum_{i=0}^{m} a_i w_i(X) \right) = t(X) \cdot h(X)$$

- Verifier picks a random $r \in \mathbb{F}$, computes and queries:
  - computes $u = (u_i(r))_{i=0}^{m}$, $v = (v_i(r))_{i=0}^{m}$, $w = (w_i(r))_{i=0}^{m}$, queries the first part of the proof string with their suffixes of length $(m - \ell)$, computes the left hand side of the equation above
  - queries the second part of the proof string with $(1, r, r^2, \ldots, r^{n-2})$ thus learning $h(r)$, computes $t(r)$, computes the right hand side of the equation above
Adding zero-knowledge (1/3)

- We checked whether $A(r) \cdot B(r) - C(r) = t(r) \cdot h(r)$, where

$$
A(X) = \sum_{i=0}^{m} a_i u_i(X)
$$

$$
B(X) = \sum_{i=0}^{m} a_i v_i(X)
$$

$$
C(X) = \sum_{i=0}^{m} a_i w_i(X)
$$

This leaked $A(r)$, $B(r)$, $C(r)$, $h(r)$, which may have been dependent on the witness $(a_{\ell+1}, \ldots, a_m)$.

- The prover hides these values by adding a random multiple of $t(X)$ to each of $A$, $B$, $C$
Adding zero-knowledge (2/3)

Prover picks three random values \( r_A, r_B, r_C \) \( \leftarrow \mathbb{F} \). Defines

\[
A^*(X) := A(X) + r_A \cdot t(X) \quad B^*(X) := B(X) + r_B \cdot t(X) \quad C^*(X) := C(X) + r_C \cdot t(X)
\]

and \( h^*(X) = \frac{(A^*(X) \cdot B^*(X) - C^*(X))}{t(X)} \)

\( P \) appends \( r_A, r_B, r_C \) to the first part of the proof string. Replaces the second part with coefficients of \( h^* \)

\( V \) makes the following queries against the first part of the proof string:

\[
(u_{\ell+1}, \ldots, u_m, t(r), 0, 0) \mapsto z_1
\]

\[
(v_{\ell+1}, \ldots, v_m, 0, t(r), 0) \mapsto z_2
\]

\[
(w_{\ell+1}, \ldots, w_m, 0, 0, t(r)) \mapsto z_3
\]

and the same old query against the second part, and does the same verification
Adding zero-knowledge (3/3)

Zero-knowledge

- $z_1, z_2, z_3$ are masked with (non-zero multiples of) $r_A, r_B, r_C$
- The value $h^*(r)$ is determined as $(z_1 \cdot z_2 - z_3)/t(r)$ in an accepting transcript
[Groth16] zk-SNARK
Non-interactive linear proofs (NILP)

- A relation $R$ is given (over any math. structure). $\phi$ — instance. $w$ — witness

### Syntax

- $(\vec{\sigma}, \vec{\tau}) \leftarrow \text{Setup()} \in \mathbb{F}^m \times \mathbb{F}^n$
- $\Pi \leftarrow \text{ProofMatrix}(\phi, w) \in \mathbb{F}^{k \times m}$
  - The actual proof is $\vec{\pi} = \Pi \vec{\sigma} \in \mathbb{F}^k$
- $\vec{t} \leftarrow \text{Test}(\phi) \in (\mathbb{F}[x_1, \ldots, x_{m+k}])^\eta$, where each polynomial in $\vec{t}$ has the total degree at most 2
  - $\vec{t}$ is used to verify. Proof $\vec{\pi}$ is accepted, if $t(\vec{\sigma}, \vec{\pi}) = 0$ for each $t \in \vec{t}$
- $\vec{\pi} \leftarrow \text{Sim}(\vec{\tau}, \phi)$
Affine attacks (by prover)

Soundness

There exists an extractor $X$, such that if

- Attacker (seeing $\vec{\sigma}$) comes up with some $(\phi, \Pi)$
- $\Pi \vec{\sigma}$ is a good proof for $\phi$, i.e. $\text{Test}(\phi)(\vec{\sigma}, \Pi \vec{\sigma}) = \vec{0}$

then $X(\phi, \Pi) \in R(\phi)$, i.e. is a good witness for $\phi$

Disclosure-freeness

Adversary cannot distinguish different $\vec{\sigma}$-s with valid (i.e. quadratic) tests:

- Let $A$ generate $\vec{t}_{\text{adv}} \in (\mathbb{F}[x_1, \ldots, x_m])^\eta$
- Generate $\vec{\sigma}_0, \vec{\sigma}_1$ by running Setup() twice
- Then, with high probability, $\vec{t}_{\text{adv}}(\vec{\sigma}_0) = \vec{0}$ iff $\vec{t}_{\text{adv}}(\vec{\sigma}_1) = \vec{0}$
From QAP to NILP: Setup()

○ Recall: relation $R$ was given by
  ○ Polynomials $u_i, v_i, w_i$ of degree $\leq (n - 1)$, where $0 \leq i \leq m$
  ○ Polynomial $t$ of degree $n$
  ○ The number $\ell$: the length (+1) of the instance

○ Pick the elements $\alpha, \beta, \gamma, \delta, x \leftarrow \mathbb{F}^*$. These are the trapdoor $\vec{\tau}$

○ The CRS $\vec{\sigma}$ consists of the following elements:
  ○ $\alpha, \beta, \gamma, \delta$
  ○ $1, x, x^2, \ldots, x^{n-1}$
  ○ $\Gamma_0/\gamma, \ldots, \Gamma_\ell/\gamma, \Gamma_{\ell+1}/\delta, \ldots, \Gamma_m/\delta$
    ○ ...where $\Gamma_i = \beta u_i(x) + \alpha v_i(x) + w_i(x)$
  ○ $t(x)/\delta, xt(x)/\delta, x^2t(x)/\delta, \ldots, x^{n-2}t(x)/\delta$
Disclosure-freeness of the CRS

- Let $t_{\text{adv}}$ be a test polynomial
- $T = t_{\text{adv}}(\vec{\sigma})$ is a multi-variate Laurent polynomial in $\alpha, \beta, \gamma, \delta, x$
- Adversary knows the coefficients of $T$
  - The total degree of $T$ is less than $4n$
- $T$ may evaluate to 0, because
  - $T \equiv 0$. Such $t_{\text{adv}}$ cannot be used to distinguish different CRSs
  - $T(\alpha, \beta, \gamma, \delta, x) = 0$ for the given values of $\alpha, \beta, \gamma, \delta, x$
    - Happens with negligible probability
Laurent polynomials

- A field $\mathbb{F}$. The variables $X_1, \ldots, X_n$
- A Laurent monomial has the form $X_1^{d_1} \cdots X_n^{d_n}$, where $d_1, \ldots, d_n \in \mathbb{Z}$
- A Laurent polynomial is a linear combination (over $\mathbb{F}$) of a finite number of Laurent monomials
- Schwartz-Zippel lemma also applies to Laurent polynomials:
  - Let $(-\delta_i)$ be the least power of $X_i$ in $f$ (let $\delta_i = 0$, if $X_i$ does not have negative powers in $f$)
  - $f \cdot X_1^{\delta_1} \cdots X_n^{\delta_n}$ is a “normal” polynomial
    - This multiplication can only increase the number of roots
From QAP to NILP:

\[ \text{ProofMatrix}\left( (a_0, \ldots, a_\ell), (a_{\ell+1}, \ldots, a_m) \right) \]

- Find the private polynomial \( h \) of degree \( \leq (n-2) \), satisfying
  \[
  \left( \sum_{i=0}^{m} a_i u_i(X) \right) \cdot \left( \sum_{i=0}^{m} a_i v_i(X) \right) = \left( \sum_{i=0}^{m} a_i w_i(X) \right) + h(X) t(X)
  \]

- Pick \( r, s \leftarrow \mathbb{F} \), let \( \Pi \sigma = (A, B, C) \), where
  \[
  A = \alpha + \sum_{i=0}^{m} a_i u_i(x) + r\delta \quad B = \beta + \sum_{i=0}^{m} a_i v_i(x) + s\delta
  \]
  (make use of \( 1, x, \ldots, x^{n-1} \) in the CRS)
  \[
  C = \sum_{i=\ell+1}^{m} a_i \frac{\Gamma_i}{\delta} + h(x) \frac{t(x)}{\delta} + sA + rB - rs\delta
  \]
  (This part uses \( t(x)/\delta, \ldots, x^{n-2} t(x)/\delta \) in the CRS)
From QAP to NILP: Test\((a_0, \ldots, a_\ell)\) and simulation

- Test returns a single polynomial, corresponding to the test

\[ A \cdot B = \alpha \cdot \beta + \sum_{i=0}^{\ell} a_i \frac{\Gamma_i}{\gamma} \cdot \gamma + C \cdot \delta \]

- To simulate a proof, randomly generate \(A, B\) and compute \(C\) so, that the previous equation is satisfied
  - \(C\) can be computed with the values in \(\vec{\tau}\). Computation does not have to be linear
  - We have perfect zero-knowledge: in the real proof, \(A\) and \(B\) are also uniformly distributed
Correctness

\[ A \cdot B = \left( \alpha + \sum_{i=0}^{m} a_i u_i(x) + r\delta \right) \cdot \left( \beta + \sum_{i=0}^{m} a_i v_i(x) + s\delta \right) = \]

\[ \alpha \cdot \beta + \left( \sum_{i=0}^{m} a_i (\beta u_i(x) + \alpha v_i(x)) \right) + \left( \sum_{i=0}^{m} a_i u_i(x) \right) \cdot \left( \sum_{i=0}^{m} a_i v_i(x) \right) + \]

\[ r\delta B + s\delta A - rs\delta^2 = \alpha \cdot \beta + \sum_{i=0}^{m} a_i \Gamma_i + h(x) t(x) + r\delta B + s\delta A - rs\delta^2 = \]

\[ \alpha \cdot \beta + \sum_{i=0}^{\ell} a_i \Gamma_i + \left( \sum_{i=\ell+1}^{m} a_i \frac{\Gamma_i}{\delta} + h(x) \frac{t(x)}{\delta} + sA + rB - rs\delta \right) \delta = \alpha \cdot \beta + \sum_{i=0}^{\ell} a_i \Gamma_i + C \cdot \delta \]
Soundness

- The row of $\Pi$ corresponding to $C$ contains $a_i$, $(\ell + 1 \leq i \leq m)$ as the coefficients for $\frac{\Gamma_i}{\delta}$.
- But that’s for an honest $P$ only. We have to show that whatever $A, B, C$ are, these $a_i$ must occur there.
- We start from the Test equation, where $A, B, C$ are unknown linear combinations of elements in CRS.
- Think of it as the equality between Laurent polynomials with variables $\alpha, \beta, \gamma, \delta, x$.
- From the coefficients, find $a_{\ell+1}, \ldots, a_m$, such that
  $$\left( \sum_{i=0}^{m} a_i u_i(X) \right) \cdot \left( \sum_{i=0}^{m} a_i v_i(X) \right) \equiv \left( \sum_{i=0}^{m} a_i w_i(X) \right) \pmod{t(X)}$$
Equations

\[ A \cdot B = \alpha \cdot \beta + \sum_{i=0}^{\ell} a_i \frac{\Gamma_i}{\gamma} \cdot \gamma + C \cdot \delta \]

\[ \Gamma_i = \beta u_i(x) + \alpha v_i(x) + w_i(x) \]

\[ A = A_\alpha \alpha + A_\beta \beta + A_\gamma \gamma + A_\delta \delta + A(x) + \sum_{i=0}^{\ell} A_i \Gamma_i/\gamma + \sum_{i=\ell+1}^{m} A_i \Gamma_i/\delta + A_h(x)t(x)/\delta \]

\[ B = B_\alpha \alpha + B_\beta \beta + B_\gamma \gamma + B_\delta \delta + B(x) + \sum_{i=0}^{\ell} B_i \Gamma_i/\gamma + \sum_{i=\ell+1}^{m} B_i \Gamma_i/\delta + B_h(x)t(x)/\delta \]

\[ C = C_\alpha \alpha + C_\beta \beta + C_\gamma \gamma + C_\delta \delta + C(x) + \sum_{i=0}^{\ell} C_i \Gamma_i/\gamma + \sum_{i=\ell+1}^{m} C_i \Gamma_i/\delta + C_h(x)t(x)/\delta \]
Coefficient of $\alpha^2$

$$A \cdot B = \alpha \cdot \beta + \sum_{i=0}^{\ell} a_i \Gamma_i + C \cdot \delta$$

- 0 in RHS
- $A_{\alpha} B_{\alpha}$ in LHS
  - $\Gamma_i$ also contains $\alpha$, but always comes with $\gamma^{-1}$ or $\delta^{-1}$
  - There is no monomial $\alpha \gamma$ or $\alpha \delta$ in $A$ or $B$
- Hence $A_{\alpha} B_{\alpha} = 0$. Either $A_{\alpha} = 0$ or $B_{\alpha} = 0$
- W.l.o.g. $B_{\alpha} = 0$
Equations

\[
A \cdot B = \alpha \cdot \beta + \sum_{i=0}^{\ell} a_i \Gamma_i + C \cdot \delta
\]

\[
\Gamma_i = \beta u_i(x) + \alpha v_i(x) + w_i(x)
\]

\[
A = A_\alpha \alpha + A_\beta \beta + A_\gamma \gamma + A_\delta \delta + A(x) +
\sum_{i=0}^{\ell} A_i \Gamma_i/\gamma + \sum_{i=\ell+1}^{m} A_i \Gamma_i/\delta + A_h(x) t(x)/\delta
\]

\[
B = B_\beta \beta + B_\gamma \gamma + B_\delta \delta + B(x) +
\sum_{i=0}^{\ell} B_i \Gamma_i/\gamma + \sum_{i=\ell+1}^{m} B_i \Gamma_i/\delta + B_h(x) t(x)/\delta
\]

\[
C = C_\alpha \alpha + C_\beta \beta + C_\gamma \gamma + C_\delta \delta + C(x) +
\sum_{i=0}^{\ell} C_i \Gamma_i/\gamma + \sum_{i=\ell+1}^{m} C_i \Gamma_i/\delta + C_h(x) t(x)/\delta
\]
Coefficient of $\alpha\beta$

- 1 in RHS
- $A_\alpha B_\beta$ in LHS
  - Again, cannot introduce monomial $\alpha\beta$ through $\Gamma_i$
- Hence $A_\alpha B_\beta = 1$
- W.l.o.g. $A_\alpha = 1$ and $B_\beta = 1$
  - Otherwise, rescale coefficients of $A$ by $1/A_\alpha$ and coefficients of $B$ by $1/B_\beta$
  - This does not change the LHS nor the RHS of the Test equation
Equations

\[ A \cdot B = \alpha \cdot \beta + \sum_{i=0}^{\ell} a_i \Gamma_i + C \cdot \delta \]

\[ \Gamma_i = \beta u_i(x) + \alpha v_i(x) + w_i(x) \]

\[ A = \alpha + A_\beta \beta + A_\gamma \gamma + A_\delta \delta + A(x) + \]
\[ \sum_{i=0}^{\ell} A_i \Gamma_i /\gamma + \sum_{i=\ell+1}^{m} A_i \Gamma_i /\delta + A_h(x) t(x) /\delta \]

\[ B = \beta + B_\gamma \gamma + B_\delta \delta + B(x) + \]
\[ \sum_{i=0}^{\ell} B_i \Gamma_i /\gamma + \sum_{i=\ell+1}^{m} B_i \Gamma_i /\delta + B_h(x) t(x) /\delta \]

\[ C = C_\alpha \alpha + C_\beta \beta + C_\gamma \gamma + C_\delta \delta + C(x) + \]
\[ \sum_{i=0}^{\ell} C_i \Gamma_i /\gamma + \sum_{i=\ell+1}^{m} C_i \Gamma_i /\delta + C_h(x) t(x) /\delta \]
Coefficient of $\beta^2$

- 0 in RHS
- $A_\beta$ in LHS
  - No contribution from the coefficients of $\Gamma_i$
- Hence $A_\beta = 0$
Equations

\[ A \cdot B = \alpha \cdot \beta + \sum_{i=0}^{\ell} a_i \Gamma_i + C \cdot \delta \]

\[ \Gamma_i = \beta u_i(x) + \alpha v_i(x) + w_i(x) \]

\[ A = \alpha + A_{\gamma \gamma} + A_{\delta \delta} + A(x) + \sum_{i=0}^{\ell} A_i \Gamma_i / \gamma + \sum_{i=\ell+1}^{m} A_i \Gamma_i / \delta + A_h(x) t(x) / \delta \]

\[ B = \beta + B_{\gamma \gamma} + B_{\delta \delta} + B(x) + \sum_{i=0}^{\ell} B_i \Gamma_i / \gamma + \sum_{i=\ell+1}^{m} B_i \Gamma_i / \delta + B_h(x) t(x) / \delta \]

\[ C = C_{\alpha} \alpha + C_{\beta} \beta + C_{\gamma} \gamma + C_{\delta} \delta + C(x) + \sum_{i=0}^{\ell} C_i \Gamma_i / \gamma + \sum_{i=\ell+1}^{m} C_i \Gamma_i / \delta + C_h(x) t(x) / \delta \]
Coefficient of $\delta^{-2}$
(with constants $\alpha, \beta, x$)

- 0 in RHS
- In LHS, it is

$$\left( A_h(x)t(x) + \sum_{i=\ell+1}^{m} A_i \Gamma_i \right) \cdot \left( B_h(x)t(x) + \sum_{i=\ell+1}^{m} B_i \Gamma_i \right)$$

- One of the factors is 0
- W.l.o.g. $B_h(x)t(x) + \sum_{i=\ell+1}^{m} B_i \Gamma_i = 0$
Equations

\[ A \cdot B = \alpha \cdot \beta + \sum_{i=0}^{\ell} a_i \Gamma_i + C \cdot \delta \]

\[ \Gamma_i = \beta u_i(x) + \alpha v_i(x) + w_i(x) \]

\[ A = \alpha + A\gamma \gamma + A\delta \delta + A(x) + \]
\[ \sum_{i=0}^{\ell} A_i \Gamma_i/\gamma + \sum_{i=\ell+1}^{m} A_i \Gamma_i/\delta + A_h(x) t(x)/\delta \]

\[ B = \beta + B\gamma \gamma + B\delta \delta + B(x) + \sum_{i=0}^{\ell} B_i \Gamma_i/\gamma \]

\[ C = C_\alpha \alpha + C_\beta \beta + C_\gamma \gamma + C_\delta \delta + C(x) + \]
\[ \sum_{i=0}^{\ell} C_i \Gamma_i/\gamma + \sum_{i=\ell+1}^{m} C_i \Gamma_i/\delta + C_h(x) t(x)/\delta \]
Coefficient of $\delta^{-1}$
(with constants $\alpha, \beta, \gamma, x$)

- 0 in RHS
- In LHS, it is

$$
\left( A_h(x) t(x) + \sum_{i=\ell+1}^{m} A_i \Gamma_i \right) \cdot \left( \beta + B_\gamma \gamma + B(x) + \sum_{i=0}^{\ell} B_i \Gamma_i / \gamma \right)
$$

- One of the factors is 0
- The right factor is not 0
- Hence $A_h(x) t(x) + \sum_{i=\ell+1}^{m} A_i \Gamma_i = 0$
\[ A \cdot B = \alpha \cdot \beta + \sum_{i=0}^{\ell} a_i \Gamma_i + C \cdot \delta \]
\[ \Gamma_i = \beta u_i(x) + \alpha v_i(x) + w_i(x) \]
\[ A = \alpha + A \gamma \gamma + A \delta \delta + A(x) + \sum_{i=0}^{\ell} A_i \Gamma_i / \gamma \]
\[ B = \beta + B \gamma \gamma + B \delta \delta + B(x) + \sum_{i=0}^{\ell} B_i \Gamma_i / \gamma \]
\[ C = C_\alpha \alpha + C_\beta \beta + C_\gamma \gamma + C_\delta \delta + C(x) + \]
\[ \sum_{i=0}^{\ell} C_i \Gamma_i / \gamma + \sum_{i=\ell+1}^{m} C_i \Gamma_i / \delta + C_h(x) t(x) / \delta \]
Coefficients of $\gamma^{-2}$ (with constants $\alpha, \beta, x$)

- $0$ in RHS
- In LHS, it is

$$\left(\sum_{i=0}^{\ell} A_i \Gamma_i\right) \cdot \left(\sum_{i=0}^{\ell} B_i \Gamma_i\right)$$

- One of the factors is $0$
- W.l.o.g. $\sum_{i=0}^{\ell} A_i \Gamma_i = 0$
Equations

\[ A \cdot B = \alpha \cdot \beta + \sum_{i=0}^{\ell} a_i \Gamma_i + C \cdot \delta \]

\[ \Gamma_i = \beta u_i(x) + \alpha v_i(x) + w_i(x) \]

\[ A = \alpha + A\gamma \gamma + A\delta \delta + A(x) \]

\[ B = \beta + B\gamma \gamma + B\delta \delta + B(x) + \sum_{i=0}^{\ell} B_i \Gamma_i / \gamma \]

\[ C = C_{\alpha} \alpha + C_{\beta} \beta + C_{\gamma} \gamma + C_{\delta} \delta + C(x) + \]

\[ \sum_{i=0}^{\ell} C_i \Gamma_i / \gamma + \sum_{i=\ell+1}^{m} C_i \Gamma_i / \delta + C_h(x) t(x) / \delta \]
Coefficients of $\gamma^{-1}$
(with constants $\alpha, \beta, x$)

- 0 in RHS
- In LHS, it is

$$\left(\sum_{i=0}^{\ell} B_i \Gamma_i\right) \cdot (\alpha + A_\delta \delta + A(x))$$

- One of the factors is 0
- The right factor is not 0
- Hence $\sum_{i=0}^{\ell} B_i \Gamma_i = 0$
Equations

\[ A \cdot B = \alpha \cdot \beta + \sum_{i=0}^{\ell} a_i \Gamma_i + C \cdot \delta \]

\[ \Gamma_i = \beta u_i(x) + \alpha v_i(x) + w_i(x) \]

\[ A = \alpha + A \gamma + A \delta \delta + A(x) \]

\[ B = \beta + B \gamma + B \delta \delta + B(x) \]

\[ C = C_\alpha \alpha + C_\beta \beta + C_\gamma \gamma + C_\delta \delta + C(x) + \]

\[ \sum_{i=0}^{\ell} C_i \Gamma_i / \gamma + \sum_{i=\ell+1}^{m} C_i \Gamma_i / \delta + C_h(x) t(x) / \delta \]
Coefficients of $\beta \gamma$ and $\alpha \gamma$

- $0$ and $0$ in RHS
- $A_\gamma$ and $B_\gamma$ in LHS
- Hence these coefficients are equal to $0$
Equations

\[ A \cdot B = \alpha \cdot \beta + \sum_{i=0}^{\ell} a_i \Gamma_i + C \cdot \delta \]

\[ \Gamma_i = \beta u_i(x) + \alpha v_i(x) + w_i(x) \]

\[ A = \alpha + A_\delta \delta + A(x) \]

\[ B = \beta + B_\delta \delta + B(x) \]

\[ C = C_\alpha \alpha + C_{\beta} \beta + C_\gamma \gamma + C_\delta \delta + C(x) + \sum_{i=0}^{\ell} C_i \Gamma_i / \gamma + \sum_{i=\ell+1}^{m} C_i \Gamma_i / \delta + C_h(x) t(x) / \delta \]
Coefficients of $\alpha$ and $\beta$
(with constant $x$)

\[ \alpha : \quad B(x) = \sum_{i=0}^{\ell} a_i v_i(x) + \sum_{i=\ell+1}^{m} C_i v_i(x) \]

\[ \beta : \quad A(x) = \sum_{i=0}^{\ell} a_i u_i(x) + \sum_{i=\ell+1}^{m} C_i u_i(x) \]

Define $a_i = C_i$ for $\ell + 1 \leq i \leq m$
Equations

\[ A \cdot B = \alpha \cdot \beta + \sum_{i=0}^{\ell} a_i \Gamma_i + C \cdot \delta \]

\[ \Gamma_i = \beta u_i(x) + \alpha v_i(x) + w_i(x) \]

\[ A = \alpha + A \delta \delta + \sum_{i=0}^{m} a_i u_i(x) \]

\[ B = \beta + B \delta \delta + \sum_{i=0}^{m} a_i v_i(x) \]

\[ C = C_\alpha \alpha + C_\beta \beta + C_\gamma \gamma + C_\delta \delta + C(x) + \]

\[ \sum_{i=0}^{\ell} C_i \Gamma_i / \gamma + \sum_{i=\ell+1}^{m} a_i \Gamma_i / \delta + C_h(x) t(x) / \delta \]
Coefficients of $1, x, x^2, \ldots$

i.e. take $\alpha = \beta = \gamma = \delta = 0$

\[
\text{RHS} = \sum_{i=0}^{\ell} a_i w_i(x) + \sum_{i=\ell+1}^{m} a_i w_i(x) + C_h(x) t(x)
\]

\[
\text{LHS} = \left( \sum_{i=0}^{m} a_i u_i(x) \right) \cdot \left( \sum_{i=0}^{m} a_i v_i(x) \right)
\]

Hence $(a_{\ell+1}, \ldots, a_m) = (C_{\ell+1}, \ldots, C_m)$ is a witness
A notation for exponentiation

- Pairing-based setup:
  - Cyclic groups $G_1, G_2, G_T$ of size $p$;
  - Pairing $\hat{\epsilon} : G_1 \times G_2 \rightarrow G_T$;
  - Groups generated by $g, h, \hat{\epsilon}(g, h)$

- Let $x \in \mathbb{Z}_p$. Denote

\[
\begin{align*}
[x]_1 &= g^x \\
[x]_2 &= h^x \\
[x]_T &= \hat{\epsilon}(g, h)^x
\end{align*}
\]
From NILP to NIZK proof for QAP

The CRS

- \([\alpha]_1, [\beta]_1, [\beta]_2, [\gamma]_2, [\delta]_1, [\delta]_2\)
- \([1]_1, [x]_1, [x^2]_1, \ldots, [x^{n-1}]_1, [1]_2, [x]_2, [x^2]_2, \ldots, [x^{n-1}]_2\)
- \([\Gamma_0/\gamma]_1, \ldots, [\Gamma_\ell/\gamma]_1, [\Gamma_{\ell+1}/\delta]_1, \ldots, [\Gamma_m/\delta]_1\)
- \([t(x)/\delta]_1, [xt(x)/\delta]_1, [x^2 t(x)/\delta]_1, \ldots, [x^{n-2} t(x)/\delta]_1\)

Proof

\([A]_1, [B]_2, [C]_1\). The elements of proof matrix are used as exponents

Verification

\[\hat{e}( [A]_1, [B]_2 ) \overset{?}{=} \hat{e}( [\alpha]_1, [\beta]_2 ) \cdot \hat{e}( \prod_{i=0}^{\ell} [\Gamma_i/\gamma]_1^{a_i}, [\gamma]_2 ) \cdot \hat{e}( [C]_1, [\delta]_2 )\]
Fixed-base multi-exponentiation

Task: Compute $g_1^{x_1} \cdots g_n^{x_n}$

$g_1, \ldots, g_n$ are constants. $x_1, \ldots, x_n$ are $k$-bit long variables

Precomputation

$g_J \leftarrow \prod_{i \in J} g_i$ for all $J \subseteq \{1, \ldots, n\}$

Computation

$\text{res} := 1$
for $i := k - 1$ down to 0 do
    $\text{res} := \text{res}^2$
    $J \leftarrow \{ j \mid i\text{-th bit of } g_j \text{ is } 1 \}$
    $\text{res} := \text{res} \cdot g_J$
Security proof

- ...in generic bilinear group model
- Completeness — obvious
- Zero-knowledge — use trapdoor to simulate. Hence obvious
- Soundness
  - CRS does not tell the adversary anything interesting
    - Due to disclosure-freeness. Any test the adversary can do can be expressed as a quadratic polynomial in the elements of CRS
  - As CRS is uninteresting, the adversary generates \([A]_1, [B]_2, [C]_1\) independently of the elements of CRS
  - These can only be generated as linear combinations of the elements of CRS in \(G_1 / G_2\)
  - The coefficients can be found from the adversary’s calls
  - The witness can be extracted as before