Lecture 6: Σ-protocols wrap-up and the Sumcheck Protocol

Zero-knowledge proofs

263-4665-00L

Lecturer: Jonathan Bootle

Last time

- Making Σ-protocols zero-knowledge against malicious verifiers

Sigma protocols from DLOG

- Intro level: Schnorr and homomorphisms

- Medium level: multiplicative relations

- Advanced level: low-degree circuit proofs

Agenda

Sigma protocols from DLOG

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

New topic: ZK arguments with short proofs

Multiplication relation

Secret

Masked response $z = xa + b$ Mask

Challenge

- $\mathcal{R}_{Mult} := \left\{ \begin{array}{c} \text{Instance } \mathbb{x} \\ (\mathbb{G}, G, H, \{A_i\}_{i\in[3]}, p), \\ \{(a_i, r_i)\}_{i\in[3]} \\ \text{Witness } \mathbb{w} \end{array} : \begin{array}{c} G, H, A_1, A_2, A_3 \in \mathbb{G}, \\ a_1, a_2, a_3, r_1, r_2, r_3 \in \mathbb{Z}_p, \\ A_i = a_i \cdot G + r_i \cdot H \ \forall i \\ a_1 \cdot a_2 = a_3 \end{array} \right\}.$

Technique: computing on masked secrets

- Pedersen protocols for A_1, A_2 give masked z_1, z_2 with a_1, a_2 ‘inside’.
- Compute $a_3 = a_1 \cdot a_2$ but use z_i instead of a_i.
- $z_1 z_2 = x^2 a_1 a_2 + x(a_1 b_2 + a_2 b_1) + b_1 b_2 := x^2 a_3 + x \cdot m_1 + m_0$.
- P commits to m_1, m_0 before seeing x.
- V checks this equation using the homomorphic commitments.

Multiplication proof

$P(\mathbb{x}, \mathbb{w})$

$b_1, b_2, s_1, s_2, u_1, u_0 \leftarrow_\$ \mathbb{Z}$

$B_i \coloneqq b_i \cdot G + s_i \cdot H, i \in [2]$

$M_i \coloneqq m_i \cdot G + u_i \cdot H, i \in \{0,1\}$

$\xrightarrow{B_1, B_2, M_1, M_0}$

$V(\mathbb{x})$

$x \leftarrow_\$ \mathbb{Z}_p$

$\xleftarrow{x}$

$z_i \coloneqq x a_i + b_i, i \in [2]$

$t_i \coloneqq x r_i + s_i, i \in [2]$

$u_3 \coloneqq x^2 r_3 + x m_1 + m_0$

$\xrightarrow{z_1, z_2, t_1, t_2, u_3}$

Output 1 iff

$z_i \cdot G + t_i \cdot H == x \cdot A_i + B_i, i \in [2]$

$z_1 z_2 \cdot G + u_3 \cdot H == x^2 \cdot A_3 + x \cdot M_1 + M_0$

Completeness analysis

- The check that $z_1 \cdot G + t_1 \cdot H == x \cdot A_1 + B_1$ passes by completeness of Pedersen protocol.
- Similarly for $z_2 \cdot G + t_2 \cdot H == x \cdot A_2 + B_2$.

- The check that $z_1 z_2 \cdot G + u_3 \cdot H == x^2 \cdot A_3 + x \cdot M_1 + M_0$ passes because

$$z_1 z_2 = x^2 a_3 + x m_1 + m_0$$

Multiply by G

$+$

$$u_3 = x^2 r_3 + x u_1 + u_0$$

Multiply by H

$=$

$$z_1 z_2 \cdot G + u_3 \cdot H = x^2 \cdot A_3 + x \cdot M_1 + M_0$$

SHVZK analysis

$S(\mathbb{x}, x)$

1. $z_1, z_2, t_1, t_2, u_3, \leftarrow_\$ \mathbb{Z}_p$.
2. $B_i \coloneqq z_i \cdot G + t_i \cdot H - x \cdot A_i, i \in [2]$
3. $M_1 \leftarrow_\$ \mathbb{G}$.
4. $M_0 \coloneqq z_1 z_2 \cdot G + u_3 \cdot H$
   $-x^2 \cdot C_3 - x \cdot M_1$
5. Output
   $(B_1, B_2, M_1, M_0, x, z_1, z_2, t_1, t_2, u_3)$.

What is the verifier's view?

- $B_1, B_2, z_1, z_2, t_1, t_2$ from Pedersen proofs
- $u_1 \leftarrow_\$ \mathbb{Z}_p$ so $M_1 = m_1 G + u_1 H$ uniform
- $u_0 \leftarrow_\$ \mathbb{Z}_p$ so $u_3 = x^2 r_3 + x u_1 + u_0$ uniform
- M_0 uniquely determined as $M_0 = z_1 z_2 \cdot G + u_3 \cdot H - x^2 \cdot A_3 - x \cdot M_1$.
- **Why is the simulator valid?** (efficient, indistinguishable)
- Clearly, the simulator is efficient.
- The Pedersen proof simulations are perfect.
- M_1 and u_3 are correctly distributed, and M_0 is then uniquely determined.

3-soundness analysis I

- Consider a 3-tree of accepting transcripts.

$$
\begin{tikzcd}
 & B_1, B_2, M_1, M_0 \arrow[dl, "x"'] \arrow[d, "x'"] \arrow[dr, "x''"] & \\
\begin{array}{c} z_1, z_2 \\ t_1, t_2, u_3 \end{array} & \begin{array}{c} z_1', z_2' \\ t_1', t_2', u_3' \end{array} & \begin{array}{c} z_1'', z_2'' \\ t_1'', t_2'', u_3'' \end{array}
\end{tikzcd}
$$

distinct

Satisfying

$$z_1 \cdot G + t_1 \cdot H = x \cdot A_1 + B_1$$
$$z_2 \cdot G + t_2 \cdot H = x \cdot A_2 + B_2$$
$$z_1 z_2 \cdot G + u_3 \cdot H = x^2 \cdot A_3 + x \cdot M_1 + M_0$$

Similarly for e.g. z_i', z_i''.

$$
\begin{pmatrix} 1 & x & x^2 \\ 1 & x' & x'^2 \\ 1 & x'' & x''^2 \end{pmatrix}
\begin{pmatrix} B_1 & B_2 & M_0 \\ A_1 & A_2 & M_1 \\ 0 & 0 & A_3 \end{pmatrix}
=
\begin{pmatrix} z_1 & z_2 & z_1 z_2 \\ z_1' & z_2' & z_1' z_2' \\ z_1'' & z_2'' & z_1'' z_2'' \end{pmatrix} \cdot G +
\begin{pmatrix} t_1 & t_2 & u_3 \\ t_1' & t_2' & u_3' \\ t_1'' & t_2'' & u_3'' \end{pmatrix} \cdot H
$$

Rows: different branches of tree

Columns: different verifier checks

3-soundness analysis II

$$\begin{pmatrix} 1 & x & x^2 \\ 1 & x' & x'^2 \\ 1 & x'' & x''^2 \end{pmatrix}\begin{pmatrix} B_1 & B_2 & M_0 \\ A_1 & A_2 & M_1 \\ 0 & 0 & A_3 \end{pmatrix} = \begin{pmatrix} z_1 & z_2 & z_1 z_2 \\ z_1' & z_2' & z_1' z_2' \\ z_1'' & z_2'' & z_1'' z_2'' \end{pmatrix} \cdot G + \begin{pmatrix} t_1 & t_2 & u_3 \\ t_1' & t_2' & u_3' \\ t_1'' & t_2'' & u_3'' \end{pmatrix} \cdot H$$

Fact:

$$Q(x, x', x'') = \begin{pmatrix} 1 & x & x^2 \\ 1 & x' & x'^2 \\ 1 & x'' & x''^2 \end{pmatrix}$$ is invertible whenever x, x', x'' are distinct

https://en.wikipedia.org/wiki/Vandermonde_matrix

$k_1 \cdot G + l_1 \cdot H = 0 = 0 \cdot G + 0 \cdot H.$
$k_1 = l_1 = 0$ or break binding.
Similarly, $k_2 = l_2 = 0.$

Multiplying by Q^{-1} gives

$$\begin{pmatrix} B_1 & B_2 & M_0 \\ A_1 & A_2 & M_1 \\ 0 & 0 & A_3 \end{pmatrix} = \begin{pmatrix} b_1 & b_2 & m_0 \\ a_1 & a_2 & m_1 \\ k_1 & k_2 & a_3 \end{pmatrix} \cdot G + \begin{pmatrix} s_1 & s_2 & u_0 \\ r_1 & r_2 & u_1 \\ l_1 & l_2 & r_3 \end{pmatrix} \cdot H$$

Now we have openings to all commitments

$= (xa_1 + b_1) \cdot G$
$+ (xr_1 + s_1) \cdot H$

3-soundness analysis III

Verification equations:

$z_1 \cdot G + t_1 \cdot H = x \cdot A_1 + B_1$
$z_2 \cdot G + t_2 \cdot H = x \cdot A_2 + B_2$
$z_1 z_2 \cdot G + u_3 \cdot H = x^2 \cdot A_3 + x \cdot M_1 + M_0$
Similarly for e.g. z_i', z_i''.

Extractor output: $a_1, r_1, a_2, r_2, a_3, r_3 \in \mathbb{Z}_p$.

Why is the output a witness?

By construction, $A_i = a_i \cdot G + r_i \cdot H$.

$$\begin{pmatrix} B_1 & B_2 & M_0 \\ A_1 & A_2 & M_1 \\ 0 & 0 & A_3 \end{pmatrix} = \begin{pmatrix} b_1 & b_2 & m_0 \\ a_1 & a_2 & m_1 \\ k_1 & k_2 & a_3 \end{pmatrix} \cdot G + \begin{pmatrix} s_1 & s_2 & u_0 \\ r_1 & r_2 & u_1 \\ l_1 & l_2 & r_3 \end{pmatrix} \cdot H.$$

Substituting openings for A_i, B_i, M_i into the verification equations and applying binding implies

$z_1 = xa_1 + b_1$
$z_2 = xa_2 + b_2$
$z_1 z_2 = x^2 a_3 + xm_1 + m_0$
Similarly for e.g. x', x''.

$(xa_1 + b_1)(xa_2 + b_2)$
$= x^2 a_3 + xm_1 + m_0$
Similarly for e.g. x', x''.

Degree 2 polynomial
Three roots x, x', x''.
Identically zero
$\Rightarrow a_1 a_2 = a_3$

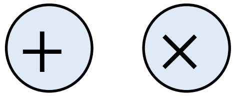
Application: proof that values are non-zero

- $\mathcal{R}_{\neq 0} \coloneqq \left\{ \left((\mathbb{G}, G, H, A, p\ ), a, r \right) : \begin{array}{c} G, H, A \in \mathbb{G},\ \ a, r \in \mathbb{Z}_p, \\ A = a \cdot G + r \cdot H, a \neq 0 \end{array} \right\}.$
- $a \in \mathbb{Z}_p$ is non-zero $\Leftrightarrow$ a is invertible, $\exists a_2$ with $a a_2 = 1.$

Protocol:

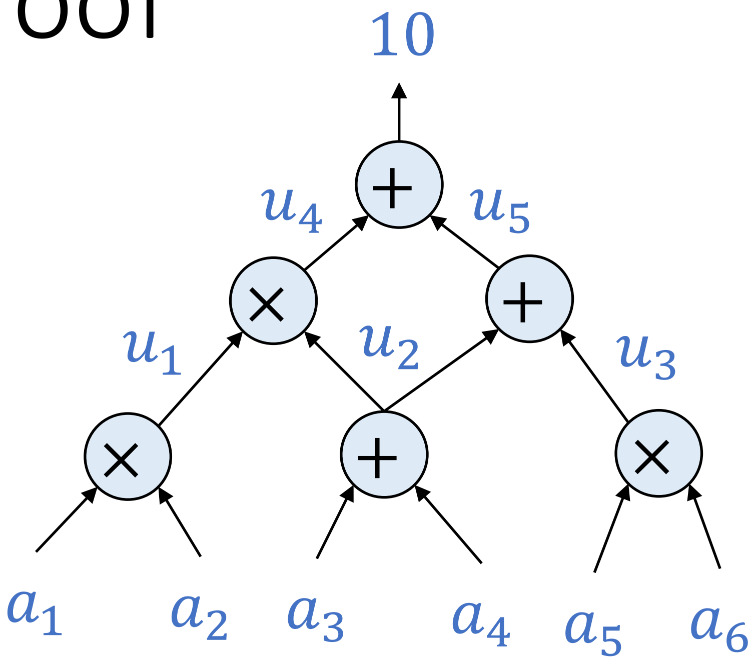
- P samples $r_2 \leftarrow \mathbb{Z}_p$ and sends commitment $A_2 \coloneqq a_2 \cdot G + r_2 \cdot H.$
- P, V commit to 1 without randomness i.e. $A_3 \coloneqq 1 \cdot g.$
- Use a multiplication proof to show that $a a_2 = 1.$

Application: circuit satisfiability proof



Instance: prime p, circuit over $\mathbb{Z}_p$ with output.

Witness: input wire values giving correct output.



Protocol:

- P sends commitments to *every* wire value.
- P, V commit to output without randomness i.e. $A \coloneqq 10 \cdot g$.
- Use multiplication and linear relation proofs for each gate.
- AND composition.
- Inherits 3-soundness.

Each subproof has $O(1)$ proof size, prover complexity and verifier complexity
Total is $O(N)$ for N gates

Can save on $+$ gates.

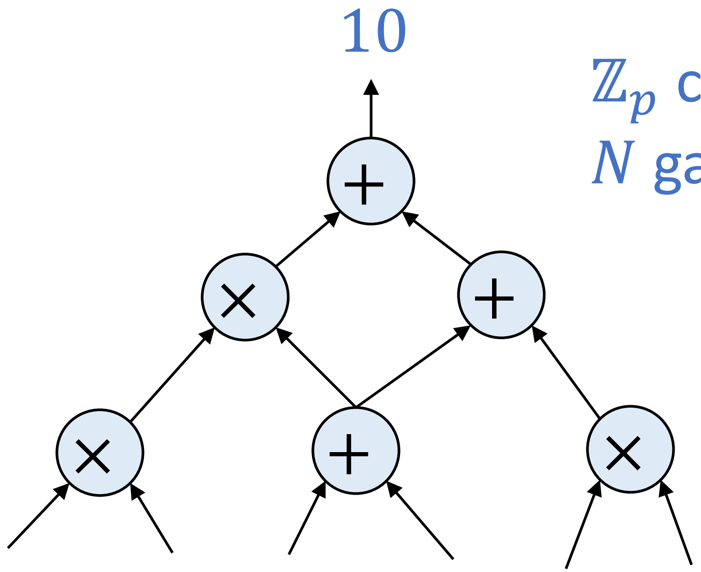
Agenda

Sigma protocols from DLOG

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

New topic: ZK arguments with short proofs

Relation for low-degree circuits



$\mathbb{Z}_p$ circuit
N gates

$\mathbb{Z}_p$ polynomial
Degree d

$q(a_1, \ldots, a_6) = 10$

- $$\mathcal{R}_q := \left\{ \begin{matrix} \left(\mathbb{G}, G, H, \{A_i\}_{i=1}^{l+1}, p\right) \\ \{(a_i, r_i)\}_{i=1}^{l+1} \end{matrix} : \begin{matrix} G, H, \{A_i\} \in \mathbb{G}, \{a_i, r_i\} \in \mathbb{Z}_p, \\ A_i = a_i \cdot G + r_i \cdot H, \forall i \in [l+1] \\ q(a_1, \ldots, a_l) = a_{l+1} \end{matrix} \right\}.$$

Goal: proof size $O(d + l)$ instead of N

Computing on masked secrets

Masked response

Secret

$z = xa + b$

Challenge

Mask

- Pedersen protocols for C_i give masked z_i with a_i 'inside'.
- If $a_3 = a_1 \cdot a_2$ then $z_1 z_2 \coloneqq x^2 a_3 + x \cdot m_1 + m_0$.

- What is $q(z_1, \dots, z_l)$?
- $q(X_1, X_2) = X_1 X_2 + X_1$,

Want $q(a_1, a_2)$ in x^2 term

- $q(z_1, z_2) = z_1 z_2 + z_1 = x^2 a_1 a_2 + x(a_1 b_2 + b_2 a_1 + a_1) + (b_1 b_2 + b_1)$

- Instead, ${}^{z_i}/_{x} = a_i + {}^{b_i}/_{x}$, $q\left({}^{z_1}/_{x}, \dots, {}^{z_l}/_{x}\right) = q(a_1, \dots, a_l)$ + -ve powers
- $q^*(x, z_1, \dots, z_l) \coloneqq x^d q\left({}^{z_1}/_{x}, \dots, {}^{z_l}/_{x}\right) = x^d q(a_1, \dots, a_l) + \sum_{i=0}^{d-1} x^i m_i$

Defined even when $x = 0$

Low degree polynomial proof

Trade soundness and proof size

$P(\mathbb{x}, \mathbb{w})$		$V(\mathbb{x})$
$B_1, \dots, B_l \leftarrow_\$$ Pedersen protocols for $A_i, i \in [l]$. $u_0, \dots, u_{d-1} \leftarrow_\$ \mathbb{Z}_p$ $M_i \coloneqq m_i \cdot G + u_i \cdot H,$ $i \in \{0, \dots, d-1\}$	$B_1, \dots, B_l$ $M_0, \dots, M_{d-1}$ $\longrightarrow$	
	$\xleftarrow{x}$	$x \leftarrow_\$ \mathbb{Z}_p$
$z_1, \dots, z_l, t_1, \dots, t_l \leftarrow_\$$ Pedersen protocols for $A_i, i \in [l]$. $u_d \coloneqq x^d r_d + \sum_{i=0}^{d-1} x^i u_i$	$z_1, \dots, z_l$ $t_1, \dots, t_l, u_d$ $\longrightarrow$	Output 1 iff $z_i \cdot G + t_i \cdot H == x \cdot A_i + B_i, i \in [l]$ $q^*(x, z_1, \dots, z_l) \cdot G + u_d \cdot H$ $== x^d \cdot A_{l+1} + \sum_{i=0}^{d-1} x^i \cdot M_i$

Proof size: l Pedersen protocols $+O(d)$ extra $= O(l+d)$

Verifier: l Pedersen protocols $+q^*$ evaluation $+O(d)$

Prover: efficient

Completeness analysis

- The checks that $z_i \cdot G + t_i \cdot H == x \cdot A_i + B_i$ pass by completeness of Pedersen protocol.

- Check that $q^*(x, z_1, \dots, z_l) \cdot G + u_d \cdot H == x^d \cdot A_{l+1} + \sum_{i=0}^{d-1} x^i \cdot M_i$ passes because

$$q^*(x, z_1, \dots, z_l) = x^d a_{l+1} + \sum_{i=0}^{d-1} x^i m_i$$

Multiply by G

$+$

$$u_d = x^d r_{l+1} + \sum_{i=0}^{d-1} x^i u_i$$

Multiply by H

$=$

$$q^*(x, z_1, \dots, z_l) \cdot G + u_d \cdot H = x^d \cdot A_{l+1} + \sum_{i=0}^{d-1} x^i \cdot M_i$$

SHVZK analysis

$S(\mathbb{x}, x)$
1. $z_1, \ldots, z_l, t_1, \ldots, t_l, u_d, \leftarrow_\$ \mathbb{Z}_p$.
2. $B_i \coloneqq z_i \cdot G + t_i \cdot H - x \cdot A_i, i \in [l]$
3. $M_{d-1}, \ldots, M_1 \leftarrow_\$ \mathbb{G}$.
4. $M_0 \coloneqq q^*(x, z_1, \ldots, z_l) \cdot G + u_d \cdot H - x^d \cdot A_{l+1} - \sum_{i=1}^{d-1} x^i \cdot M_i$
5. Output $(\{C_i\}, \{M_i\}, x, \{(z_i, t_i)\}, u_d)$.

What is the verifier's view?

- B_i, z_i, t_i from Pedersen proofs.
- $u_i \leftarrow_\$ \mathbb{Z}_p$ so $M_i = m_i G + u_i H$ uniform.
- $u_0 \leftarrow_\$ \mathbb{Z}_p$ so $u_d = x^d r_d + \sum_{i=0}^{d-1} x^i u_i$ uniform.
- M_0 uniquely determined as

$$M_0 = q^*(x, z_1, \ldots, z_l) \cdot G + u_d \cdot H - x^d \cdot A_{l+1} - \sum_{i=1}^{d-1} x^i \cdot M_i$$

- **Why is the simulator valid?** (efficient, indistinguishable)
- Clearly, the simulator is efficient. Pedersen proof simulations are perfect.
- M_i and u_d are correctly distributed, and M_0 is then uniquely determined.

$(d+1)$-soundness analysis I

- Consider a $(d+1)$-tree of accepting transcripts.

$$B_1, .., B_l, M_{d-1}, \dots, M_0$$

$x^{(0)}$ … $x^{(d)}$ distinct

$$\left\{z_i^{(0)}, t_i^{(0)}\right\}_{i\in[l]} \qquad \left\{z_i^{(d)}, t_i^{(d)}\right\}_{i\in[l]}$$

$$u_d^{(1)} \qquad u_d^{(d)}$$

Satisfying, $\forall j \in \{0, \dots, d\}$,

$$z_i^{(j)} \cdot G + t_i^{(j)} \cdot H = x^{(j)} \cdot A_i + B_i, \forall i \in [l],$$

$$q^*\left(x^{(j)}, z_1^{(j)}, \dots, z_l^{(j)}\right) \cdot G + u_d^{(j)} \cdot H$$

$$= \left(x^{(j)}\right)^d \cdot A_{l+1} + \sum_{i=0}^{d-1} \left(x^{(j)}\right)^i \cdot M_i$$

$$\begin{pmatrix} 1 & \cdots & \left(x^{(0)}\right)^d \\ \vdots & \ddots & \vdots \\ 1 & \cdots & \left(x^{(d)}\right)^d \end{pmatrix} \begin{pmatrix} B_1 \cdots & B_l & M_0 \\ A_1 \cdots & A_l & M_1 \\ 0 \cdots & 0 & \vdots \\ \vdots \ \ddots & \vdots & M_{d-1} \\ 0 \cdots & 0 & A_{l+1} \end{pmatrix} = \begin{pmatrix} z_1^{(0)} \cdots & z_l^{(0)} & q^*(x^{(0)}, z_1^{(0)}, \dots, z_l^{(0)}) \\ \vdots \ \ddots & \vdots & \vdots \\ z_1^{(d)} \cdots & z_l^{(d)} & q^*(x^{(d)}, z_1^{(d)}, \dots, z_l^{(d)}) \end{pmatrix} \cdot G + \begin{pmatrix} t_1^{(0)} \cdots & t_l^{(0)} & u_d^{(0)} \\ \vdots \ \ddots & \vdots & \vdots \\ t_1^{(d)} \cdots & t_l^{(d)} & u_d^{(d)} \end{pmatrix} \cdot H$$

Rows: different branches of tree

Columns: different verifier checks

$(d+1)$-soundness analysis II

$$\begin{pmatrix} 1 & \cdots & \left(x^{(0)}\right)^d \\ \vdots & \ddots & \vdots \\ 1 & \cdots & \left(x^{(d)}\right)^d \end{pmatrix} \begin{pmatrix} B_1 \cdots & B_l & M_0 \\ A_1 \cdots & A_l & M_1 \\ 0 \cdots & 0 & \vdots \\ \vdots \ddots & \vdots & M_{d-1} \\ 0 \cdots & 0 & A_{l+1} \end{pmatrix} = \begin{pmatrix} z_1^{(0)} \cdots & z_l^{(0)} & q^*\left(x^{(0)}, z_1^{(0)}, \ldots, z_l^{(0)}\right) \\ \vdots \ddots & \vdots & \vdots \\ z_1^{(d)} \cdots & z_l^{(d)} & q^*\left(x^{(d)}, z_1^{(d)}, \ldots, z_l^{(d)}\right) \end{pmatrix} \cdot G + \begin{pmatrix} t_1^{(0)} \cdots & t_l^{(0)} & u_d^{(0)} \\ \vdots \ddots & \vdots & \vdots \\ t_1^{(d)} \cdots & t_l^{(d)} & u_d^{(d)} \end{pmatrix} \cdot H$$

Fact:

$$Q\left(x^{(0)}, \ldots, x^{(d)}\right) = \begin{pmatrix} 1 & \cdots & \left(x^{(0)}\right)^d \\ \vdots & \ddots & \vdots \\ 1 & \cdots & \left(x^{(d)}\right)^d \end{pmatrix}$$ is invertible when $x^{(0)}, \ldots, x^{(d)}$ distinct.

https://en.wikipedia.org/wiki/Vandermonde_matrix

Multiplying by Q^{-1} gives

$$\begin{pmatrix} B_1 \cdots & B_l & M_0 \\ A_1 \cdots & A_l & M_1 \\ 0 \cdots & 0 & \vdots \\ \vdots \ddots & \vdots & M_{d-1} \\ 0 \cdots & 0 & A_{l+1} \end{pmatrix} = \begin{pmatrix} b_1 \cdots & b_l & m_0 \\ a_1 \cdots & a_l & m_1 \\ 0 \cdots & 0 & \vdots \\ \vdots \ddots & \vdots & m_{d-1} \\ 0 \cdots & 0 & a_{l+1} \end{pmatrix} \cdot G + \begin{pmatrix} s_1 \cdots & s_l & u_0 \\ r_1 \cdots & r_l & u_1 \\ 0 \cdots & 0 & \vdots \\ \vdots \ddots & \vdots & u_{d-1} \\ 0 \cdots & 0 & a_{l+1} \end{pmatrix} \cdot H$$

Now we have openings to all commitments

Note: zero entries or break binding

$(d+1)$-soundness analysis III

Extractor output: $a_1, r_1, \dots, a_{l+1}, r_{l+1} \in \mathbb{Z}_p$.

Verification equations $\forall j \in \{0, \dots, d\}$,

$$z_i^{(j)} \cdot G + t_i^{(j)} \cdot H = x^{(j)} \cdot A_i + B_i, \forall i \in [l],$$

$$q^*\left(x^{(j)}, z_1^{(j)}, \dots, z_l^{(j)}\right) \cdot G + u_d^{(j)} \cdot H$$

$$= \left(x^{(j)}\right)^d \cdot A_{l+1} + \sum_{i=0}^{d-1} \left(x^{(j)}\right)^i \cdot M_i$$

Why is the output a witness?

By construction, $A_i = a_i \cdot G + r_i \cdot H$.

Substituting openings for A_i, B_i, M_i into the verification equations and applying binding implies

$$z_i^{(j)} = x^{(j)} a_i + b_i$$

$$q^*\left(x^{(j)}, z_1^{(j)}, \dots, z_l^{(j)}\right)$$

$$= \left(x^{(j)}\right)^d a_{l+1} + \sum_{i=0}^{d-1} \left(x^{(j)}\right)^i \cdot m_i$$

Combining gives a degree d polynomial.

$(d+1)$ roots $x^{(0)}, \dots, x^{(d)}$.

Therefore identically zero.

Coefficient of x^d

$\Rightarrow q(a_1, \dots, a_l) = a_{l+1}$

Application: (non-)membership proofs

Prove you are in an accept list or not in a block list.

- $\mathcal{S} \coloneqq \{s_1, \dots, s_{n-1}\} \subseteq \mathbb{Z}_p$.
- $\mathcal{R}_{\mathcal{S}} = \left\{ \begin{matrix} (\mathbb{G}, G, H, C, p, \mathcal{S}) \\ (x, r) \end{matrix} : \begin{matrix} \mathcal{S} = \{s_1, \dots, s_{n-1}\} \subseteq \mathbb{Z}_p \\ C = x \cdot G + r \cdot H, x \in \mathcal{S} \end{matrix} \right\}$,
- $\mathcal{R}_{\bar{\mathcal{S}}}$ with $x \notin \mathcal{S}$.
- $v_{\mathcal{S}}(X) \coloneqq \prod_{i=1}^{n-1}(X - s_i)$, $v_{\mathcal{S}}(x) = 0 \Leftrightarrow x \in \mathcal{S}$.
- If $l + d \ll N$ then low-degree proof has smaller proofs than circuit proof.
- Task: design a low-degree circuit for computing $v_{\mathcal{S}}(x)$.

More efficient (non-)membership proofs

$n = 2^d$

Binary representation of i

$X_j := X^{2^j} = X_{j-1}^2$

- $v_{\mathcal{S}}(X) := \sum_{i=0}^{n-1} v_i X^i = \sum_{i_0,\ldots,i_{d-1}}^{n-1} v_{i_0,i_1,\ldots,i_{d-1}} X_0^{i_0} \cdots X_{d-1}^{i_{d-1}}$
$= q(X_0, \ldots, X_{d-1})$

P, V use 0 as output commitment in q-proof

- $\mathcal{R}_{\mathcal{S}} = \left\{ \begin{matrix} (\mathbb{G}, G, H, C, p, \mathcal{S}) \\ (x, r) \end{matrix} : \begin{matrix} \mathcal{S} = \{s_1, \ldots, s_{n-1}\} \subseteq \mathbb{Z}_p \\ C = x \cdot G + r \cdot H, x \in \mathcal{S} \end{matrix} \right\}.$

$\ell, d = O(\log n)$
$O(\log n)$ proof size

$P(\mathbb{x}, \mathbb{w})$
For $i \in \{0, \ldots, d-1\}$,
$r_i \leftarrow_\$ \mathbb{Z}_p, C_i = x_i \cdot G + r_i \cdot H$

Merge into first round

$\xrightarrow{C_1, \ldots, C_l}$

$V(\mathbb{x})$

Naïve circuit would give $O(n)$ proof size

Proof that $q(x_0, \ldots, x_{d-1}) = 0$ $\wedge \forall i$ Multiplication proof that $x_i = x_{i-1}^2$

Easily modify to prove
$q(x_0, \ldots, x_{d-1}) \neq 0$
$\Rightarrow x \notin \mathcal{S}$

New topic:
Zero-knowledge arguments with short proofs

Course Outline (13 lectures)

1. **Introduction and definitions** ~2 lectures

2. **Sigma protocols** ~3 lectures

3. **ZK arguments with short proofs** ~4 lectures

4. **Non-interactive zero-knowledge** ~3 lectures

5. **Bonus material?** ~1 lecture

Efficiency targets

		Proof size	Verifier complexity	
1. Introduction and definitions				
2. Sigma protocols	Practical, useful techniques	$\mathrm{poly}(\|x\|)$	$\mathrm{poly}(\|x\|)$	(SHV)ZK
3. ZK arguments with short proofs	Standard assumptions	$\mathrm{poly}\log(\|x\|)$	$\mathrm{poly}\log(\|x\|)$	(SHV)ZK
4. Non-interactive zero-knowledge				
5. Bonus material?				

Plan for the next few lectures

Information theoretic proof systems

+

Cryptography

=

ZK arguments with short proofs

Sumcheck protocol

Today

Interactive proof for layered arithmetic circuits

Interactive oracle proofs for general arithmetic circuits

Commitments and polynomial commitments

ZK arguments for layered arithmetic circuits

ZK arguments for general arithmetic circuits

About the sumcheck protocol

- Interactive proof with soundness against unbounded prover
- No witness or zero knowledge property
- $\mathcal{L}_{SC} := \left\{ (\mathbb{F}, H, p, u) : \begin{array}{c} H \subseteq \mathbb{F}, u \in \mathbb{F}, p(X_1, \ldots, X_\ell) \in \mathbb{F}[X_1, \ldots, X_\ell], \\ \sum_{\omega_1,\ldots,\omega_\ell \in H} p(\omega_1, \ldots, \omega_\ell) = u \end{array} \right\}.$
- Protocol provides *superfast verification*.

- Lund, Fortnow, Carloff, Nisan, 1992: **coNP** $\subseteq$ **IP** (today)
- Shamir, 1992, Shen: generalizes to **PSPACE** $\subseteq$ **IP**.

The sumcheck protocol is everywhere!

Probabilistic proofs
[BFL91,BFLS91,GKR08]

Sumcheck-based succinct arguments
[Thaler13]

[CMT13], [VSBW13], [W+17], [ZGKPP17], [WTSTW18], [XZZPS19], [Set20]

Univariate-sumcheck-based arguments
[BCRSVS19]

[BCGGRS19], [ZXZS20], [CHMVW20], [COS20], [CFQR20], [BFHVXZ20]

Sumchecks for tensor codes
[Meir13]

[RR20], [BCG20], [BCL20]

Useful properties:

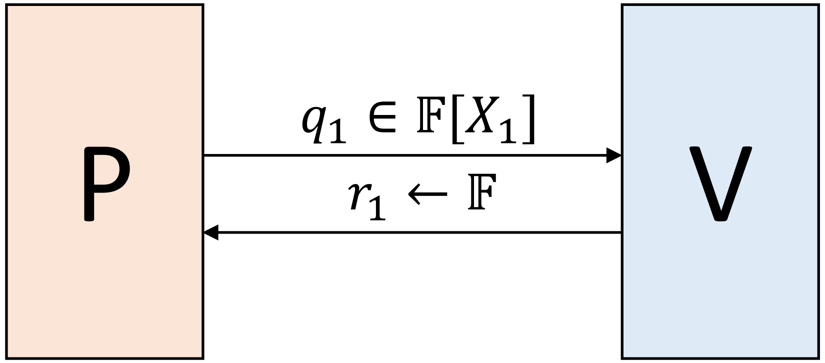
- Linear-time prover [Thaler13,ZXZS20]
- Small space [CMT13] (can be implemented with streaming access)
- Strong soundness properties [CCHLRR18] (can make non-interactive without random oracles)

The sumcheck protocol [LFKN92] (recursively)

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

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

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



Checks $\sum_{\omega_1 \in H} q_1(\omega_1) = u$
(and $\deg q_i \le d$)

$d := \max_i \deg_{X_i}(p)$

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

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

P

$\ell - 1$ more reductions

V

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

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

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

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

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

Zero-Knowledge Proofs
Exercise 6

6.1 Special Honest-Verifier Zero-Knowledge Σ-Protocols ($\star$)

The goal of this exercise is to show that an HVZK Σ-protocol can be assumed to be Special HVZK (SHVZK) without loss of generality or efficiency.

Let (P, V) be a HVZK Σ protocol for a relation R. Show that the following protocol (P', V') is a SHVZK protocol for the same relation.

1. On input x, P' compute the first message t that P computes on the same input. P' generates a uniformly random bit string c' of the same length as the challenges V computes. P' sends (t, c') to V'.
2. V' generates a uniformly random bit string c'' and sends it to P'.
3. P' computes $c \leftarrow c' \oplus c''$, computes the response r from P on challenge c and sends it to V'.
4. V' accepts if and only if V accepts r with respect to the t and c.

The new protocol is essentially the same as the old one, except that the new challenge is the XOR of a string chosen by the prover and another chosen by the verifier.

6.2 Pitfalls of the Fiat–Shamir Transform

The figure below shows the Fiat-Shamir transform, which can be used to make any Σ-protocol non-interactive by replacing the verifier's challenge c with the output of a hash function $H\colon \{0,1\}^* \to \mathcal{C}$.

$$
\begin{array}{lcl}
P(x,w) & & V(x) \\
a \leftarrow P(x,w) & \xrightarrow{\quad a \quad} & \\
 & \xleftarrow{\quad c \quad} & c \leftarrow_{\$} \mathcal{C} \\
z \leftarrow_{\$} P(x,w,c) & \xrightarrow{\quad z \quad} & b \leftarrow V(x,a,c,z) \\
 & & \text{return } b
\end{array}
$$

$$
\begin{array}{lcl}
P(x,w,H) & & V(x,H) \\
a \leftarrow P(x,w) & & \\
c \leftarrow H(x,a) & & \\
z \leftarrow P(x,w,a,c) & \xrightarrow{\quad (a,c,z) \quad} & b \leftarrow \big(V(x,a,c,z) \\
 & & \qquad \wedge (c \stackrel{?}{=} H(x,a))\big) \\
 & & \text{return } b
\end{array}
$$

a) Recall Schnorr's protocol to prove knowledge of $\mathrm{dlog}_g A$. Write down its Fiat–Shamir transform.

b) Consider a variant of the Fiat–Shamir transform for Schnorr's protocol in which A is not included in the hash computation of the verifier's challenge. Show that when the prover can choose the instance A, then they can convince the verifier without knowing $\mathrm{dlog}_g A$.
Remark: As will be seen later on in the lectures when we discuss non-interactive zero knowledge *(NIZK), the property of the modified Fiat-Shamir transform that the above prover is seemingly breaking is closely related to* adaptive soundness *of NIZK protocols. Roughly speaking, this "adaptive" definition allows a dishonest prover to pick an instance of its choice (in contrast to non-adaptive soundness where the prover does not have control over the instance).*

c) Let $\mathbb{G}$ be a finite abelian group of unknown order. We consider the verifiable delay function[1] by Wesolowski [Wes19], which is the Fiat-Shamir transform of the following interactive argument for relation

$$\mathcal{R}_{\mathsf{VDF}} = \left\{ ((\mathbb{G}, x, T, y), \emptyset) \ : \ x, y \in \mathbb{G} \wedge y = x^{2^T} \right\}.$$

$V \to P$. Sample a large prime ℓ uniformly at random and send ℓ to P.
$P \to V$. Compute $\pi = x^{\lfloor 2^T/\ell \rfloor}$ and send π to V.
V. Compute $r = 2^T \mod \ell$ and accept if and only if $\pi^\ell \cdot g^r = y$.

The idea of this protocol is that while the prover is required to do $\Omega(T)$ sequential multiplications, where T can be somewhat large, the verifier in contrast only requires $O(\log(T))$ computation time; we refer to [Wes19] for a precise treatment. To apply the Fiat-Shamir transform to this protocol, we consider a hash function $H : \{0,1\}^* \to \mathsf{Primes}(2\lambda)$ which maps to 2λ-bit primes and is modelled as a random oracle.

Show that if the time parameter T is not included in the hash computation, i.e. $\ell \leftarrow H(\mathbb{G}, x, y)$, then if the prover can choose T adaptively, it could succeed in the protocol for significantly larger T than the amount of sequential computation they indeed performed.

6.3 Proofs of One-out-of-Many Pedersen Commitments

In the following, let G be an algorithm that generates the description of a group of prime order $p = \Theta\left(2^\lambda\right)$ on input a security parameter 1^λ. For a given positive integer λ, let $(\mathbb{G}, g, h, p) \leftarrow \mathsf{G}\left(1^\lambda\right)$, with uniformly random $g, h \in \mathbb{G}$.

Let $k \geq 1$ be an integer and $n = 2^k < p$. Let $C_0, \ldots, C_{n-1}$ be pairwise-distinct Pedersen commitments in group $\mathbb{G}$ with commitment keys g and h.

a) Design a Σ-protocol to prove knowledge of an opening of C_i for some $i \in [\![0, n-1]\!]$, with linear communication complexity $O(n)$ for the prover.
Hint: Use OR composition of Σ-protocols.

b) Given a public list $x_0, \ldots, x_{n-1} \in \mathbb{Z}_p$, construct a polynomial function f in k variables such that for each $\mathbf{i} \in \{0,1\}^k$, $f(\mathbf{i}) = x_i$, where i is the integer represented by the bits of $\mathbf{i}$. Use this polynomial function to describe a sigma protocol proving that a Pedersen commitment C made using commitment key g and h opens to a value in the public list.
Remark (motivation for this exercise): Using the function f, but replacing the list

[1] A verifiable delay function (VDF) is a function whose evaluation requires a certain predetermined amount of time T — a delay — and allows for a proof of correct evaluation. VDFs are used in several blockchain-based cryptocurrencies to allow a prover to prove that they spent a certain time of computation.

$x_0, \ldots, x_{n-1} \in \mathbb{Z}_p$ *with the list of Pedersen commitments* $C_0, \ldots, C_{n-1}$*, one can design a* Σ*-protocol to prove knowledge of an opening of* C_i *for some* $i \in [\![0, n-1]\!]$ *with* sub-linear *prover communication complexity* $o(n)$*. This is outside the scope of this exercise and we refer to Groth and Kohlweiss [GK15].*

References

[GK15] Jens Groth and Markulf Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. pages 253–280, 2015.

[Wes19] Benjamin Wesolowski. Efficient verifiable delay functions. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019*, pages 379–407, Cham, 2019. Springer International Publishing.

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

Zero-Knowledge Proofs

Exercise 6

6.1 Special Honest-Verifier Zero-Knowledge Σ-Protocols ($\star$)

The goal of this exercise is to show that an HVZK Σ-protocol can be assumed to be Special HVZK (SHVZK) without loss of generality or efficiency.

Let (P, V) be a HVZK Σ protocol for a relation R. Show that the following protocol (P', V') is a SHVZK protocol for the same relation.

1. On input x, P' compute the first message t that P computes on the same input. P' generates a uniformly random bit string c' of the same length as the challenges V computes. P' sends (t, c') to V'.
2. V' generates a uniformly random bit string c'' and sends it to P'.
3. P' computes $c \leftarrow c' \oplus c''$, computes the response r from P on challenge c and sends it to V'.
4. V' accepts if and only if V accepts r with respect to the t and c.

The new protocol is essentially the same as the old one, except that the new challenge is the XOR of a string chosen by the prover and another chosen by the verifier.

Solution: The simulator S' for (P', V') is defined as follows. On input (x, c''), it runs the simulator S for (P, V) on input x. If S returns a failure symbol then so does S', otherwise S returns a transcript (t, c, r) (with c uniformly random) and S' computes $c' \leftarrow c \oplus c''$ and returns the transcript $((t, c'), c'', r)$. Clearly, S' returns a transcript which is identically distributed to an honest execution of (P', V'), and it meets our definition of SHVZK.

6.2 Pitfalls of the Fiat–Shamir Transform

The figure below shows the Fiat-Shamir transform, which can be used to make any Σ-protocol non-interactive by replacing the verifier's challenge c with the output of a hash function $H\colon \{0,1\}^* \to \mathcal{C}$.

$$
\begin{array}{lcl}
P(x, w) & & V(x) \\
a \leftarrow P(x, w) & \xrightarrow{\quad a \quad} & \\
 & \xleftarrow{\quad c \quad} & c \leftarrow_{\$} \mathcal{C} \\
z \leftarrow_{\$} P(x, w, c) & \xrightarrow{\quad z \quad} & b \leftarrow V(x, a, c, z) \\
 & & \text{return } b
\end{array}
$$

$$
\begin{array}{lcl}
P(x,w,H) & & V(x,H) \\
a \leftarrow P(x,w) & & \\
c \leftarrow H(x,a) & & \\
z \leftarrow P(x,w,a,c) & \xrightarrow{\quad (a,c,z) \quad} & b \leftarrow \big(V(x,a,c,z) \\
& & \quad \wedge (c \stackrel{?}{=} H(x,a))\big) \\
& & \text{return } b
\end{array}
$$

a) Recall Schnorr's protocol to prove knowledge of $\mathrm{dlog}_G A$. Write down its Fiat–Shamir transform.

Solution: In Schnorr's protocol, the instance is of the form $(\mathbb{G}, G, A, p)$, where $\mathbb{G}$ is a group of prime order p and $G, A \in \mathbb{G}$. The prover computes $B \leftarrow b \cdot G$ for $b \leftarrow_\$ \mathbb{Z}_p$ and sends it to the verifier. The verifier sends $x \leftarrow_\$ \mathbb{Z}_p$ to the prover, the prover replies with $y \leftarrow ax + b$ and the verifier accepts if and only if $y \cdot G = x \cdot A + B$.

Let $H \colon \{0,1\}^* \to \mathbb{Z}_p$ be a hash function. In the transformed protocol, the prover computes $B \leftarrow b \cdot G$ for $b \leftarrow_\$ \mathbb{Z}_p$, $x \leftarrow H((\mathbb{G}, G, A, p), B)$ and then $y \leftarrow ax + b$. The proof consists of the pair (B, y). The verifier computes $x = H((\mathbb{G}, G, A, p), B)$ and accepts the proof if and only if $y \cdot G = x \cdot A + B$. This can be slightly optimized by sending as the proof (x, y) and accepting if and only if $x = H((\mathbb{G}, G, A, p), y \cdot G - x \cdot A)$

b) Consider a variant of the Fiat–Shamir transform for Schnorr's protocol in which A is not included in the hash computation of the verifier's challenge. Show that when the prover can choose the instance A, then they can convince the verifier without knowing $\mathrm{dlog}_G A$.

Remark: As will be seen later on in the lectures when we discuss non-interactive zero knowledge *(NIZK), the property of the modified Fiat-Shamir transform that the above prover is seemingly breaking is closely related to* adaptive soundness *of NIZK protocols. Roughly speaking, this "adaptive" definition allows a dishonest prover to pick an instance of its choice (in contrast to non-adaptive soundness where the prover does not have control over the instance).*

Solution: Specifically, we show how a prover can cheat in this variant of the Fiat-Shamir transform by convincing an (honest) verifier that it knows the discrete log of an element $A \in \mathbb{G}$ – *of the prover's choice* – even when it doesn't't! The prover first samples a random group element $B \leftarrow_\$ \mathbb{G}$ and computes $x = H((\mathbb{G}, G, p), B)$. Then the prover generates $y \leftarrow_\$ \mathbb{Z}_p$ and computes the instance $A \leftarrow (1/x) \cdot (y \cdot G - B)$. Now the prover convinces the verifier that it knows $\mathrm{dlog}_G A$ by forwarding the pair (B, y) in this modified Fiat-Shamir protocol. Note that the verifier outputs 1 since we have $x = H((\mathbb{G}, G, p), B)$ and $y \cdot G = x \cdot A + B$ by construction.

c) Let $\mathbb{G}$ be a finite abelian group of unknown order. We consider the verifiable delay function[1] by Wesolowski [Wes19], which is the Fiat-Shamir transform of the following interactive argument for relation

$$\mathcal{R}_{\mathsf{VDF}} = \left\{ ((\mathbb{G}, x, T, y), \emptyset) \ : \ x, y \in \mathbb{G} \wedge y = x^{2^T} \right\}.$$

$V \to P$. Sample a large prime ℓ uniformly at random and send ℓ to P.

$P \to V$. Compute $\pi = x^{\lfloor 2^T / \ell \rfloor}$ and send π to V.

V. Compute $r = 2^T \mod \ell$ and accept if and only if $\pi^\ell \cdot x^r = y$.

[1] A verifiable delay function (VDF) is a function whose evaluation requires a certain predetermined amount of time T — a delay — and allows for a proof of correct evaluation. VDFs are used in several blockchain-based cryptocurrencies to allow a prover to prove that they spent a certain time of computation.

The idea of this protocol is that while the prover is required to do $\Omega(T)$ sequential multiplications, where T can be somewhat large, the verifier in contrast only requires $O(\log(T))$ computation time; we refer to [Wes19] for a precise treatment. To apply the Fiat-Shamir transform to this protocol, we consider a hash function $H : \{0,1\}^* \to \mathsf{Primes}(2\lambda)$ which maps to 2λ-bit primes and is modelled as a random oracle.

Show that if the time parameter T is not included in the hash computation, i.e. $\ell \leftarrow H(\mathbb{G}, x, y)$, then if the prover can choose T adaptively, it could succeed in the protocol for significantly larger T than the amount of sequential computation they indeed performed.

Solution: The following attack was discovered in [DMWG23]. The prover first chooses an arbitrary small time parameter t and computes $y = x^{2^t}$. They receive the challenge ℓ as $\ell \leftarrow H(\mathbb{G}, x, y)$ and compute a proof $\pi = x^{\lfloor 2^t/\ell \rfloor}$. The prover then sets $T = t+\ell-1$ and sends $((\mathbb{G}, x, T, y), \pi)$ to the verifier. The verifier now computes $r = 2^T \mod \ell$, where by choice of T we have

$$r = 2^T = 2^{t+\ell-1} = 2^t \cdot 2^{\ell-1} = 2^t \cdot 1 = 2^t \mod \ell.$$

The verifier therefore accepts, as it indeed holds that

$$\pi^\ell \cdot x^r = (x^{\lfloor 2^t/\ell \rfloor})^\ell \cdot x^r = x^{\ell \cdot \lfloor 2^t/\ell \rfloor + r} = y.$$

Thus, this attack could allow the prover to convince the verifier that they did T sequential computation, while they only did a computation of time delay $t \ll T$.

6.3 Proofs of One-out-of-Many Pedersen Commitments

In the following, let G be an algorithm that generates the description of a group of prime order $p = \Theta\left(2^\lambda\right)$ on input a security parameter 1^λ. For a given positive integer λ, let $(\mathbb{G}, G, H, p) \leftarrow \mathsf{G}\left(1^\lambda\right)$, with uniformly random $G, H \in \mathbb{G}$.

Let $k \geq 1$ be an integer and $n = 2^k < p$. Let $C_0, \ldots, C_{n-1}$ be pairwise-distinct Pedersen commitments in group $\mathbb{G}$ with commitment keys G and H.

a) Design a Σ-protocol to prove knowledge of an opening of C_i for some $i \in [\![0, n-1]\!]$, with linear communication complexity $O(n)$ for the prover.
Hint: Use OR composition of Σ-protocols.

Solution: Suppose that the prover is given $(i_0, (m_{i_0}, r_{i_0}))$ as private input for some $i_0 \in [\![0, n-1]\!]$, i.e., we have $C_{i_0} = m_{i_0} \cdot G + r_{i_0} \cdot H$. The protocol is similar to the OR proofs seen in the lectures. The main idea is to have the prover simulate proofs of knowledge (as the HVZK simulator of the Σ-protocol for Pedersen commitments, a.k.a. "the basic protocol" in the lectures) for the commitments for which it does *not* know the openings, and honestly follow the protocol for the commitment for which it knows the opening. Formally, the protocol is as follows.

$P \to V$. Generate $x_i, y_i, z_i \leftarrow_\$ \mathbb{Z}_p$ for $i \neq i_0$ and compute $D_i \leftarrow y_i \cdot G + z_i \cdot H - x_i \cdot C_i$. Generate $u_{i_0}, v_{i_0} \leftarrow_\$ \mathbb{Z}_p$ and compute $D_{i_0} \leftarrow u_{i_0} \cdot G + v_{i_0} \cdot H$. Send $(D_0, \ldots, D_{n-1})$ to the verifier.

$V \to P$. Generate $x \leftarrow_\$ \mathbb{Z}_p$ and send it to the prover.

$P \to V$. Compute $x_{i_0} \leftarrow x - \sum_{i \neq i_0} x_i$, $y_{i_0} \leftarrow x_{i_0} m_{i_0} + u_{i_0}$ and $z_{i_0} \leftarrow x_{i_0} r_{i_0} + v_{i_0}$, and send $(x_0, \ldots, x_{n-1}, (y_0, z_0) \ldots, (y_{n-1}, z_{n-1}))$ to the verifier.

V. The verifier accepts if and only if $\sum_i x_i = x$ and $y_i \cdot G + z_i \cdot H = x_i \cdot C_i + D_i$ for all $i \in [\![0, n-1]\!]$.

As the prover sends n group elements and $3n$ elements of $\mathbb{Z}_p$, the communication complexity of the prover is linear.

Completeness. The completeness follows from the fact that, by definition, $y_i \cdot G + z_i \cdot H = x_i \cdot C_i + D_i$ for all $i \in [\![0, n-1]\!]$ and $x_{i_0} = x - \sum_{i \neq i_0} x_i$.

Special Honest Verifier Zero-Knowledge. Consider a simulator which, on input x, generates uniformly random $x_0, \ldots, x_{n-1}$ conditioned on $\sum_{i=0}^{n-1} x_i = x$, and $y_0, \ldots, y_{n-1}$, $z_0, \ldots, z_{n-1} \leftarrow_\$ \mathbb{Z}_p$, computes $D_i \leftarrow y_i \cdot G + z_i \cdot H - x_i \cdot C_i$ and returns the transcript

$$\left((D_0, \ldots, D_{n-1}), \sum_{i=0}^{n-1} x_i, (x_0, \ldots, x_{n-1}, (y_0, z_0), \ldots, (y_{n-1}, z_{n-1})) \right).$$

Then, $(D_0, \ldots, D_{n-1})$ are uniformly distributed elements in $\mathbb{G}$, $x_0, \ldots, x_{n-1}$ are uniformly distributed in $\mathbb{Z}_p$ constrained to their sum being x, just as in an honest execution, and the last message similarly reveals uniformly distributed elements satisfying the verifying equations.

Knowledge Soundness. The knowledge soundness of the protocol follows from the 2-special soundness of the protocol, namely that from two accepting transcripts $((D_i)_i, x, ((x_i)_i, (y_i, z_i)_i))$ and $((D_i)_i, x', ((x'_i)_i, (y'_i, z'_i)_i))$ such that $x \neq x'$, one can conclude that there exists an index i_0 such that $x_{i_0} \neq x'_{i_0}$ since $x = \sum_i x_i$ and $x' = \sum_i x'_i$. Therefore,

$$\frac{(y_{i_0} - y'_{i_0})}{(x_{i_0} - x'_{i_0})} \cdot G + \frac{(z_{i_0} - z'_{i_0})}{(x_{i_0} - x'_{i_0})} \cdot H = C_{i_0}$$

i.e., a witness $(i_0, (m_{i_0}, r_{i_0}))$ such that $C_{i_0} = m_{i_0} \cdot G + r_{i_0} \cdot H$ can be efficiently computed.

b) Given a public list $x_0, \ldots, x_{n-1} \in \mathbb{Z}_p$, construct a polynomial function f in k variables such that for each $\mathbf{i} \in \{0,1\}^k$, $f(\mathbf{i}) = x_i$, where i is the integer represented by the bits of $\mathbf{i}$. Use this polynomial function to describe a sigma protocol proving that a Pedersen commitment C made using commitment key G and H opens to a value in the public list.

Remark (motivation for this exercise): Using the function f, but replacing the list $x_0, \ldots, x_{n-1} \in \mathbb{Z}_p$ with the list of Pedersen commitments $C_0, \ldots, C_{n-1}$, one can design a Σ-protocol to prove knowledge of an opening of C_i for some $i \in [\![0, n-1]\!]$ with sub-linear *prover communication complexity $o(n)$. This is outside the scope of this exercise and we refer to Groth and Kohlweiss [GK15].*

Solution: Let us consider any fixed $i \in \{0, 1, \ldots n-1\}$ and its binary decomposition $\mathbf{i} = (\mathbf{i}_{k-1}, \mathbf{i}_{k-2}, \ldots, \mathbf{i}_0) \in \{0,1\}^k$, i.e., $i = \sum_{\ell=0}^{k-1} \mathbf{i}_\ell \cdot 2^\ell$. Towards constructing the above polynomial function f, it helps to first express x_i in terms of the bits of $\mathbf{i}$ and all values $\{x_0, \ldots, x_{n-1}\}$. Considering a simple case of $n = 4$, and binary representation of integers $\{1, \ldots, 4\}$, it is not hard to see that

$$x_i = x_0 \cdot (1 - \mathbf{i}_1)(1 - \mathbf{i}_0) + x_1 \cdot (1 - \mathbf{i}_1)\mathbf{i}_0 + x_2 \cdot \mathbf{i}_1(1 - \mathbf{i}_0) + x_3 \cdot \mathbf{i}_1\mathbf{i}_0.$$

Generalizing this example to all integers $n \geq 1$, we obtain

$$x_i = \sum_{j=0}^{n-1} x_j \cdot \prod_{\ell=0}^{k-1} \mathbf{i}_{(\ell, \mathbf{j}_\ell)}$$

where $(\mathbf{j}_{k-1}, \mathbf{j}_{k-2}, \ldots, \mathbf{j}_0) \in \{0,1\}^k$ is the binary decomposition of j and the value "$\mathbf{i}_{(\ell,\mathbf{j}_\ell)}$" is defined as: $\mathbf{i}_{(\ell,0)} = (1 - \mathbf{i}_\ell)$ and $\mathbf{i}_{(\ell,1)} = \mathbf{i}_\ell$.

Hence, setting $Y_{(\ell,0)} = 1 - Y_\ell$ and $Y_{(\ell,1)} = Y_\ell$, the required polynomial $f \in \mathbb{Z}_p[Y_{k-1}, \ldots, Y_0]$ in k variables $Y_{k-1}, \ldots, Y_0$ can be constructed as

$$f(Y_{k-1}, \ldots, Y_0) = \sum_{j=0}^{n-1} x_j \cdot \prod_{\ell=0}^{k-1} Y_{(\ell, \mathbf{j}_\ell)}.$$

We indeed have for all $i \in [\![0, n-1]\!]$ and the corresponding bit decomposition $\mathbf{i}$ of i:

$$f(\mathbf{i}) = f(\mathbf{i}_{k-1}, \mathbf{i}_{k-2}, \dots, \mathbf{i}_0) = \sum_{j=0}^{n-1} x_j \cdot \prod_{\ell=0}^{k-1} \mathbf{i}_{(\ell, \mathbf{j}_\ell)} = x_i.$$

Using the polynomial f, we now derive a sigma protocol which proves that a Pedersen commitment C (w.r.t. the commitment keys G and H) opens to a value in the list $\{x_0, \dots, x_{n-1}\}$. The main building block that we will be using is the sigma protocol presented in the lectures for proving *low-degree circuit satisfiability*; specifically, satisfiability of the $(\log n)$-degree polynomial f in this context. Note that the statement "C opens to a value $x_i \in \{x_0, \dots, x_{n-1}\}$" is equivalent to "there exists an input $\mathbf{i} \in \{0,1\}^k$ to f such that C opens to the corresponding output $f(\mathbf{i})$".

(In the following, we only provide a sketch of the overall sigma protocol, but students are encouraged to fill in the details).

Hence, after committing to each of the k bits of $\mathbf{i}$, we can essentially run the "low-degree circuit" sigma protocol where the above k bit-commitments are interpreted as the *input commitments* to circuit/polynomial f and the given Pedersen commitment C is interpreted as the *output commitment* of f. Here the prover includes the k bit-commitments in its first message in the protocol. At the same time, the prover also needs to prove that each of the k commitments open to bits $\{0,1\}$. For this, we can use the "multiplication proofs" seen in the lectures. Recall that 0 and 1 are exactly the roots of the polynomial $X(1-X) \in \mathbb{Z}_p[X]$. Also, given a commitment $A = x \cdot G + r \cdot H$, a commitment to $x-1$ can be publicly computed as $A - G$. Therefore, one could then use $0_{\mathbb{G}}$ as a commitment to 0 mod p and the multiplication protocol from the lectures to prove that the values committed in A and $A - G$ multiply to 0. (A different, and more efficient, protocol to prove commitments to bits is provided further below.)

To summarize, the final sigma protocol is basically an AND-composition of the main "low-degree circuit" sigma protocol and k sigma protocols to prove that each of the input commitments is a bit commitment.

[Bonus] *A More Efficient Sigma Protocol for Bit Commitments:* Observe that if $x \in \{0,1\}$, then $x(A-G) = (xr) \cdot H$, i.e., $x(A-G)$ is a commitment to 0. Yet, in the protocol for proving opening to Pedersen commitments, the verifier is only given a response $z = y + cx$ for a challenge c and a random y chosen by the prover. The idea is then to let the verifier compute $z(A-G)$ instead of $x(A-G)$, and have the prover commit in its first flow to terms that the verifier is not able to compute on its own. As $z(A-G) = y(x-1) \cdot G + (zr) \cdot H$ if $x \in \{0,1\}$, the prover can then commit to $y(1-x)$ (with some randomness s') in the first flow, and add $t' := s' + zr$ to the response in the last flow. The overall protocol is then as below.

$P(A, x \in \{0,1\}, r \colon A = x \cdot G + r \cdot H)$		$V(A)$
$y, s, s' \leftarrow_\$ \mathbb{Z}_p$ $B \leftarrow y \cdot G + s \cdot H$ $B' \leftarrow (y(1-x)) \cdot G + s' \cdot H$	$\xrightarrow{B, B'}$	
	$\xleftarrow{c}$	$c \leftarrow_\$ \mathbb{Z}_p$
$z \leftarrow y + cx$ $t \leftarrow s + cr$ $t' \leftarrow s' + zr$	$\xrightarrow{z, t, t'}$	$z \cdot G + t \cdot H \stackrel{?}{=} c \cdot A + B$ $t' \cdot H \stackrel{?}{=} z(A-G) + B'$

The proof that it is an HVZK proof of knowledge is left to the reader.

References

[DMWG23] Quang Dao, Jim Miller, Opal Wright, and Paul Grubbs. Weak fiat-shamir attacks on modern proof systems. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 199–216, 2023.

[GK15] Jens Groth and Markulf Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. pages 253–280, 2015.

[Wes19] Benjamin Wesolowski. Efficient verifiable delay functions. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019*, pages 379–407, Cham, 2019. Springer International Publishing.