DewTwo: a transparent PCS with small proofs from falsifiable assumptions

Benedikt Bünz
New York University

Tushar Mopuri
University of Pennsylvania

Alireza Shirzad
University of Pennsylvania

Sriram Sridhar
University of California, Berkeley

Polynomial Commitment Schemes

Polynomial Commitment Schemes

1. $\mathsf{Setup}(1^\lambda, N) \to (\mathsf{ck}, \mathsf{vk})$.

Polynomial Commitment Schemes

1. $\mathsf{Setup}(1^\lambda, N) \to (\mathsf{ck}, \mathsf{vk})$.

2. $\mathsf{Commit}(\mathsf{ck}, p) \to \mathsf{cm}$.

Polynomial Commitment Schemes

1. $\mathsf{Setup}(1^\lambda, N) \to (\mathsf{ck}, \mathsf{vk})$.

2. $\mathsf{Commit}(\mathsf{ck}, p) \to \mathsf{cm}$.

3. $\mathsf{Open}(\mathsf{ck}, \mathsf{cm}, p, z) \to \pi$.

Polynomial Commitment Schemes

1. $\mathsf{Setup}(1^\lambda, N) \to (\mathsf{ck}, \mathsf{vk})\,.$

2. $\mathsf{Commit}(\mathsf{ck}, p) \to \mathsf{cm}\,.$

3. $\mathsf{Open}(\mathsf{ck}, \mathsf{cm}, p, z) \to \pi\,.$

4. $\mathsf{Verify}(\mathsf{vk}, \mathsf{cm}, z, y, \pi) \to \{0,1\}\,.$

Polynomial Commitment Schemes

1. $\text{Setup}(1^\lambda, N) \to (\text{ck}, \text{vk})$.
2. $\text{Commit}(\text{ck}, p) \to \text{cm}$.
3. $\text{Open}(\text{ck}, \text{cm}, p, z) \to \pi$.
4. $\text{Verify}(\text{vk}, \text{cm}, z, y, \pi) \to \{0,1\}$.

- **Completeness:** If $p(z) = y$, Verify accepts honestly generated π.

Polynomial Commitment Schemes

1. $\mathsf{Setup}(1^\lambda, N) \to (\mathsf{ck}, \mathsf{vk})$.

2. $\mathsf{Commit}(\mathsf{ck}, p) \to \mathsf{cm}$.

3. $\mathsf{Open}(\mathsf{ck}, \mathsf{cm}, p, z) \to \pi$.

4. $\mathsf{Verify}(\mathsf{vk}, \mathsf{cm}, z, y, \pi) \to \{0,1\}$.

- **Completeness:** If $p(z) = y$, Verify accepts honestly generated π.
- **Knowledge soundness:** If $\mathsf{Verify}(\mathsf{vk}, \mathsf{cm}, z, y, \pi) = 1$, we can extract a polynomial p such that $\mathsf{cm} = \mathsf{Commit}(\mathsf{ck}, p)$ and $p(z) = y$.

Polynomial Commitment Schemes

1. $\mathsf{Setup}(1^\lambda, N) \to (\mathsf{ck}, \mathsf{vk})$.

2. $\mathsf{Commit}(\mathsf{ck}, p) \to \mathsf{cm}$.

3. $\mathsf{Open}(\mathsf{ck}, \mathsf{cm}, p, z) \to \pi$.

4. $\mathsf{Verify}(\mathsf{vk}, \mathsf{cm}, z, y, \pi) \to \{0,1\}$.

<u>Efficiency Measures:</u>

- **Completeness:** If $p(z) = y$, Verify accepts honestly generated π.
- **Knowledge soundness:** If $\mathsf{Verify}(\mathsf{vk}, \mathsf{cm}, z, y, \pi) = 1$, we can extract a polynomial p such that $\mathsf{cm} = \mathsf{Commit}(\mathsf{ck}, p)$ and $p(z) = y$.

Polynomial Commitment Schemes

1. $\mathsf{Setup}(1^\lambda, N) \to (\mathsf{ck}, \mathsf{vk})$.

2. $\mathsf{Commit}(\mathsf{ck}, p) \to \mathsf{cm}$.

3. $\mathsf{Open}(\mathsf{ck}, \mathsf{cm}, p, z) \to \pi$.

4. $\mathsf{Verify}(\mathsf{vk}, \mathsf{cm}, z, y, \pi) \to \{0,1\}$.

<u>Efficiency Measures:</u>

- Transparent setup

- **Completeness:** If $p(z) = y$, Verify accepts honestly generated π.
- **Knowledge soundness:** If $\mathsf{Verify}(\mathsf{vk}, \mathsf{cm}, z, y, \pi) = 1$, we can extract a polynomial p such that $\mathsf{cm} = \mathsf{Commit}(\mathsf{ck}, p)$ and $p(z) = y$.

Polynomial Commitment Schemes

1. $\mathsf{Setup}(1^\lambda, N) \to (\mathsf{ck}, \mathsf{vk})$.

2. $\mathsf{Commit}(\mathsf{ck}, p) \to \mathsf{cm}$.

3. $\mathsf{Open}(\mathsf{ck}, \mathsf{cm}, p, z) \to \pi$.

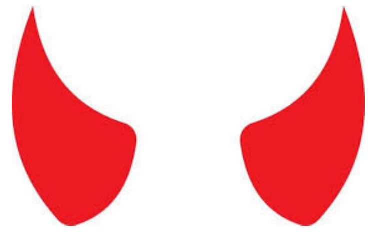
4. $\mathsf{Verify}(\mathsf{vk}, \mathsf{cm}, z, y, \pi) \to \{0,1\}$.

<u>Efficiency Measures:</u>

- Transparent setup

- **Completeness:** If $p(z) = y$, Verify accepts honestly generated π.
- **Knowledge soundness:** If $\mathsf{Verify}(\mathsf{vk}, \mathsf{cm}, z, y, \pi) = 1$, we can extract a polynomial p such that $\mathsf{cm} = \mathsf{Commit}(\mathsf{ck}, p)$ and $p(z) = y$.

Polynomial Commitment Schemes



1. $\mathsf{Setup}(1^\lambda, N) \to (\mathsf{ck}, \mathsf{vk})$.

2. $\mathsf{Commit}(\mathsf{ck}, p) \to \mathsf{cm}$.

3. $\mathsf{Open}(\mathsf{ck}, \mathsf{cm}, p, z) \to \pi$.

4. $\mathsf{Verify}(\mathsf{vk}, \mathsf{cm}, z, y, \pi) \to \{0,1\}$.

<u>Efficiency Measures:</u>

- Transparent setup

- **Completeness:** If $p(z) = y$, Verify accepts honestly generated π.
- ~~**Knowledge soundness:** If $\mathsf{Verify}(\mathsf{vk}, \mathsf{cm}, z, y, \pi) = 1$, we can extract a polynomial p such that $\mathsf{cm} = \mathsf{Commit}(\mathsf{ck}, p)$ and $p(z) = y$.~~

Polynomial Commitment Schemes

4. $\mathsf{Verify}(\mathsf{vk}, \mathsf{cm}, z, y, \pi) \to \{0,1\}$.

<u>Efficiency Measures:</u>

- Transparent setup

- **Completeness:** If $p(z) = y$, Verify accepts honestly generated π.
- ~~**Knowledge soundness:** If $\mathsf{Verify}(\mathsf{vk}, \mathsf{cm}, z, y, \pi) = 1$, we can extract a polynomial p such that $\mathsf{cm} = \mathsf{Commit}(\mathsf{ck}, p)$ and $p(z) = y$.~~

Polynomial Commitment Schemes

1. $\mathsf{Setup}(1^\lambda, N) \to (\mathsf{ck}, \mathsf{vk})$.

2. $\mathsf{Commit}(\mathsf{ck}, p) \to \mathsf{cm}$.

3. $\mathsf{Open}(\mathsf{ck}, \mathsf{cm}, p, z) \to \pi$.

4. $\mathsf{Verify}(\mathsf{vk}, \mathsf{cm}, z, y, \pi) \to \{0,1\}$.

Efficiency Measures:

- Transparent setup

[CHMMVW20]:

Polynomial Commitment Schemes

1. $\mathsf{Setup}(1^\lambda, N) \to (\mathsf{ck}, \mathsf{vk})$.
2. $\mathsf{Commit}(\mathsf{ck}, p) \to \mathsf{cm}$.
3. $\mathsf{Open}(\mathsf{ck}, \mathsf{cm}, p, z) \to \pi$.
4. $\mathsf{Verify}(\mathsf{vk}, \mathsf{cm}, z, y, \pi) \to \{0,1\}$.

Efficiency Measures:

- Transparent setup

[CHMMVW20]: $\boxed{\text{PIOP}} + \boxed{\text{PCS}} \Longrightarrow \boxed{\text{SNARK}}$

Polynomial Commitment Schemes

1. $\mathsf{Setup}(1^\lambda, N) \to (\mathsf{ck}, \mathsf{vk})$.

2. $\mathsf{Commit}(\mathsf{ck}, p) \to \mathsf{cm}$.

3. $\mathsf{Open}(\mathsf{ck}, \mathsf{cm}, p, z) \to \pi$.

4. $\mathsf{Verify}(\mathsf{vk}, \mathsf{cm}, z, y, \pi) \to \{0,1\}$.

Efficiency Measures:

- Transparent setup

[CHMMVW20]: PIOP + PCS $\Longrightarrow$ SNARK

Efficiency of SNARK $\propto$ Efficiency of PCS

Polynomial Commitment Schemes

1. $\mathsf{Setup}(1^\lambda, N) \to (\mathsf{ck}, \mathsf{vk})$.
2. $\mathsf{Commit}(\mathsf{ck}, p) \to \mathsf{cm}$.
3. $\mathsf{Open}(\mathsf{ck}, \mathsf{cm}, p, z) \to \pi$.
4. $\mathsf{Verify}(\mathsf{vk}, \mathsf{cm}, z, y, \pi) \to \{0,1\}$.

<u>Efficiency Measures:</u>

- Transparent setup
- (Commit + Open) time

[CHMMVW20]: $\boxed{\text{PIOP}} + \boxed{\text{PCS}} \implies \boxed{\text{SNARK}}$

Efficiency of SNARK $\propto$ Efficiency of PCS

Polynomial Commitment Schemes

1. $\mathsf{Setup}(1^\lambda, N) \to (\mathsf{ck}, \mathsf{vk})$.
2. $\mathsf{Commit}(\mathsf{ck}, p) \to \mathsf{cm}$.
3. $\mathsf{Open}(\mathsf{ck}, \mathsf{cm}, p, z) \to \pi$.
4. $\mathsf{Verify}(\mathsf{vk}, \mathsf{cm}, z, y, \pi) \to \{0,1\}$.

Efficiency Measures:

- Transparent setup
- (Commit + Open) time
- Verify time

[CHMMVW20]: $\boxed{\text{PIOP}} + \boxed{\text{PCS}} \Longrightarrow \boxed{\text{SNARK}}$

Efficiency of SNARK $\propto$ Efficiency of PCS

Polynomial Commitment Schemes

1. Setup$(1^\lambda, N) \to (\mathsf{ck}, \mathsf{vk})$.
2. Commit$(\mathsf{ck}, p) \to \mathsf{cm}$.
3. Open$(\mathsf{ck}, \mathsf{cm}, p, z) \to \pi$.
4. Verify$(\mathsf{vk}, \mathsf{cm}, z, y, \pi) \to \{0,1\}$.

<u>Efficiency Measures:</u>

- Transparent setup
- (Commit + Open) time
- Verify time
- Proof size

[CHMMVW20]: $\boxed{\text{PIOP}} + \boxed{\text{PCS}} \Longrightarrow \boxed{\text{SNARK}}$

Efficiency of SNARK ∝ Efficiency of PCS

Transparent PC schemes

Scheme	Category	Prover time	Verifier time	Proof Size	Falsifiable?	Transparent?
WHIR [ACFY25]	Hash-Based	$O(N \log(N))$ $\mathbb{F}$	$O(\lambda \log(N) \log\log(N))$ $\mathbb{F}$	$\lambda \log(N) \log\log(N)$ $\mathbb{F}$ 107 KB	Yes	Yes
Bulletproofs [BBBPWM18]	DLOG-Based	$O(N)$ $\mathbb{G}_{EC}$	$O(N)$ $\mathbb{G}_{EC}$	$2 \log(N)$ $\mathbb{G}_{EC}$ 1.5 KB	Yes	Yes
Dory [Lee21]	DLOG-Based	$O(N)$ $\mathbb{G}$	$O(\log N)$ $\mathbb{G}$	$6 \log(N)$ $\mathbb{G}_T$ 37 KB	Yes	Yes
Dew, Behemoth [AGLMS23, SB23]	Groups of unknown order	$O(N)$ $\mathbb{G}_{GUO}$ $O(N^3 \log N)$ $\mathbb{B}$	$O(\log N)$ $\mathbb{F}$ $O(1)$ $\mathbb{G}_{GUO}$	$O(1)$ $\mathbb{G}_{GUO}$ 9~12 KB	No	Yes

Transparent PC schemes

Scheme	Category	Prover time	Verifier time	Proof Size	Falsifiable?	Transparent?
WHIR [ACFY25]	Hash-Based	$O(N \log(N))\ \mathbb{F}$	$O(\lambda \log(N) \log\log(N))\ \mathbb{F}$	$\lambda \log(N) \log\log(N)\ \mathbb{F}$ 107 KB	Yes	Yes
Bulletproofs [BBBPWM18]	DLOG-Based	$O(N)\ \mathbb{G}_{EC}$	$O(N)\ \mathbb{G}_{EC}$	$2 \log(N)\ \mathbb{G}_{EC}$ 1.5 KB	Yes	Yes
Dory [Lee21]	DLOG-Based	$O(N)\ \mathbb{G}$	$O(\log N)\ \mathbb{G}$	$6 \log(N)\ \mathbb{G}_T$ 37 KB	Yes	Yes
Dew, Behemoth [AGLMS23, SB23]	Groups of unknown order	$O(N)\ \mathbb{G}_{GUO}$ $O(N^3 \log N)\ \mathbb{B}$	$O(\log N)\ \mathbb{F}$ $O(1)\ \mathbb{G}_{GUO}$	$O(1)\ \mathbb{G}_{GUO}$ 9~12 KB	No	Yes
DewTwo **[This work]**	Groups of unknown order	$O(N)\ \mathbb{G}_{GUO}$ $O(N \log^2 N)\ \mathbb{B}$	$O(\log N)\ \mathbb{F}$ $O(\log N)\ \mathbb{G}_{GUO}$	$\log\log(N)\ \mathbb{G}_{GUO}$ 4.5 KB	Yes	Yes

Setup

Groups of Unknown Order

Groups of Unknown Order

$(G, (\mathbb{G}, +)) \leftarrow \mathsf{Setup}(1^\lambda, N)$

Groups of Unknown Order

$(G, (\mathbb{G}, +)) \leftarrow \mathsf{Setup}(1^\lambda, N)$

Hard problem: Finding $|\mathbb{G}|$

Groups of Unknown Order

$(G, (\mathbb{G}, +)) \leftarrow \mathsf{Setup}(1^\lambda, N)$

Hard problem: Finding $|\mathbb{G}|$

- Can be used to commit to an *unbounded integer* $v \in \mathbb{Z}$:

Groups of Unknown Order

$(G, (\mathbb{G}, +)) \leftarrow \mathsf{Setup}(1^\lambda, N)$ **Hard problem:** Finding $|\mathbb{G}|$

- Can be used to commit to an *unbounded integer* $v \in \mathbb{Z}$:

$$\mathrm{IntCommit}(v) := v \cdot G$$

Groups of Unknown Order

$(G, (\mathbb{G}, +)) \leftarrow \mathsf{Setup}(1^\lambda, N)$ **Hard problem:** Finding $|\mathbb{G}|$

- Can be used to commit to an *unbounded integer* $v \in \mathbb{Z}$:

$$\mathrm{IntCommit}(v) := v \cdot G$$

- Candidates:

Groups of Unknown Order

$(G, (\mathbb{G}, +)) \leftarrow \mathsf{Setup}(1^\lambda, N)$ **Hard problem:** Finding $|\mathbb{G}|$

- Can be used to commit to an *unbounded integer* $v \in \mathbb{Z}$:

$$\mathrm{IntCommit}(v) := v \cdot G$$

- Candidates:
  1. RSA groups

Groups of Unknown Order

$(G, (\mathbb{G}, +)) \leftarrow \mathsf{Setup}(1^\lambda, N)$ **Hard problem:** Finding $|\mathbb{G}|$

- Can be used to commit to an *unbounded integer* $v \in \mathbb{Z}$:

$$\mathrm{IntCommit}(v) := v \cdot G$$

- Candidates:
  1. RSA groups
  2. Class groups.

Groups of Unknown Order

$(G, (\mathbb{G}, +)) \leftarrow \mathsf{Setup}(1^\lambda, N)$ **Hard problem:** Finding $|\mathbb{G}|$

- Can be used to commit to an *unbounded integer* $v \in \mathbb{Z}$:

$$\mathrm{IntCommit}(v) := v \cdot G$$

- Candidates:
  1. RSA groups
  2. Class groups.
  3. Jacobians of hyperelliptic curves.

Groups of Unknown Order

$(G, (\mathbb{G}, +)) \leftarrow \mathsf{Setup}(1^\lambda, N)$ **Hard problem:** Finding $|\mathbb{G}|$

- Can be used to commit to an *unbounded integer* $v \in \mathbb{Z}$:

$$\mathsf{IntCommit}(v) := v \cdot G$$

- Candidates:
    1. RSA groups
    2. Class groups.
    3. Jacobians of hyperelliptic curves.

- **Our work:** New falsifiable assumption in GUO instead of Generic Group Model.

Commit

Commit: Encode p as an Integer [BFS20]

Commit: Encode p as an Integer [BFS20]

$$p(X) = p_0 \cdot X^0 + p_1 \cdot X^1 + \ldots + p_{N-1} \cdot X^{N-1} \in \mathbb{F}_q[X]$$

Commit: Encode p as an Integer [BFS20]

$$p(X) = p_0 \cdot X^0 + p_1 \cdot X^1 + \ldots + p_{N-1} \cdot X^{N-1} \in \mathbb{F}_q[X]$$

$$\tilde{p}(X) = \tilde{p}_0 \cdot X^0 + \tilde{p}_1 \cdot X^1 + \ldots + \tilde{p}_{N-1} \cdot X^{N-1} \in \mathbb{Z}[X]$$

Commit: Encode p as an Integer [BFS20]

$$p(X) = p_0 \cdot X^0 + p_1 \cdot X^1 + \ldots + p_{N-1} \cdot X^{N-1} \in \mathbb{F}_q[X]$$

$$\tilde{p}(X) = \tilde{p}_0 \cdot X^0 + \tilde{p}_1 \cdot X^1 + \ldots + \tilde{p}_{N-1} \cdot X^{N-1} \in \mathbb{Z}[X]$$

$\in [0, q-1]$

Commit: Encode p as an Integer [BFS20]

$$p(X) = p_0 \cdot X^0 + p_1 \cdot X^1 + \ldots + p_{N-1} \cdot X^{N-1} \in \mathbb{F}_q[X]$$

$$\tilde{p}(X) = \tilde{p}_0 \cdot X^0 + \tilde{p}_1 \cdot X^1 + \ldots + \tilde{p}_{N-1} \cdot X^{N-1} \in \mathbb{Z}[X]$$

Commit: Encode p as an Integer [BFS20]

$$p(X) = p_0 \cdot X^0 + p_1 \cdot X^1 + \ldots + p_{N-1} \cdot X^{N-1} \in \mathbb{F}_q[X]$$

$$\tilde{p}(X) = \tilde{p}_0 \cdot X^0 + \tilde{p}_1 \cdot X^1 + \ldots + \tilde{p}_{N-1} \cdot X^{N-1} \in \mathbb{Z}[X]$$

$\underline{\mathsf{Commit}(\mathsf{ck}, p)}$:

Commit: Encode p as an Integer [BFS20]

$$p(X) = p_0 \cdot X^0 + p_1 \cdot X^1 + \ldots + p_{N-1} \cdot X^{N-1} \in \mathbb{F}_q[X]$$

$$\tilde{p}(X) = \tilde{p}_0 \cdot X^0 + \tilde{p}_1 \cdot X^1 + \ldots + \tilde{p}_{N-1} \cdot X^{N-1} \in \mathbb{Z}[X]$$

$\underline{\mathsf{Commit}(\mathsf{ck}, p)}$:

1. Compute $\tilde{p}(X) \in \mathbb{Z}[X]$.

Commit: Encode p as an Integer [BFS20]

$$p(X) = p_0 \cdot X^0 + p_1 \cdot X^1 + \ldots + p_{N-1} \cdot X^{N-1} \in \mathbb{F}_q[X]$$

$$\tilde{p}(X) = \tilde{p}_0 \cdot X^0 + \tilde{p}_1 \cdot X^1 + \ldots + \tilde{p}_{N-1} \cdot X^{N-1} \in \mathbb{Z}[X]$$

$\underline{\mathsf{Commit}(\mathsf{ck}, p)}$:

1. Compute $\tilde{p}(X) \in \mathbb{Z}[X]$.
2. For $\alpha > q$, compute $\tilde{p}(\alpha)$.

Commit: Encode p as an Integer [BFS20]

$$p(X) = p_0 \cdot X^0 + p_1 \cdot X^1 + \ldots + p_{N-1} \cdot X^{N-1} \in \mathbb{F}_q[X]$$

$$\tilde{p}(X) = \tilde{p}_0 \cdot X^0 + \tilde{p}_1 \cdot X^1 + \ldots + \tilde{p}_{N-1} \cdot X^{N-1} \in \mathbb{Z}[X]$$

$\underline{\mathrm{Commit}(\mathrm{ck}, p)}$:

1. Compute $\tilde{p}(X) \in \mathbb{Z}[X]$.
2. For $\alpha > q$, compute $\tilde{p}(\alpha)$.
3. Output $\tilde{p}(\alpha) \cdot G$.

Commit: Encode p as an Integer [BFS20]

$$p(X) = p_0 \cdot X^0 + p_1 \cdot X^1 + \ldots + p_{N-1} \cdot X^{N-1} \in \mathbb{F}_q[X]$$

$$\tilde{p}(X) = \tilde{p}_0 \cdot X^0 + \tilde{p}_1 \cdot X^1 + \ldots + \tilde{p}_{N-1} \cdot X^{N-1} \in \mathbb{Z}[X]$$

<u>Commit(ck, p)</u> :

1. Compute $\tilde{p}(X) \in \mathbb{Z}[X]$.
2. For $\alpha > q$, compute $\tilde{p}(\alpha)$.
3. Output $\tilde{p}(\alpha) \cdot G$.

Example: $q = 7, \alpha = 10$

Commit: Encode p as an Integer [BFS20]

$$p(X) = p_0 \cdot X^0 + p_1 \cdot X^1 + \ldots + p_{N-1} \cdot X^{N-1} \in \mathbb{F}_q[X]$$

$$\tilde{p}(X) = \tilde{p}_0 \cdot X^0 + \tilde{p}_1 \cdot X^1 + \ldots + \tilde{p}_{N-1} \cdot X^{N-1} \in \mathbb{Z}[X]$$

<u>Commit(ck, p)</u> :

1. Compute $\tilde{p}(X) \in \mathbb{Z}[X]$.
2. For $\alpha > q$, compute $\tilde{p}(\alpha)$.
3. Output $\tilde{p}(\alpha) \cdot G$.

Example: $q = 7, \alpha = 10$

$$\tilde{p}(X) = 3 \cdot X^2 + 2 \cdot X + 4 \in \mathbb{Z}[X]$$

Commit: Encode p as an Integer [BFS20]

$$p(X) = p_0 \cdot X^0 + p_1 \cdot X^1 + \ldots + p_{N-1} \cdot X^{N-1} \in \mathbb{F}_q[X]$$

$$\tilde{p}(X) = \tilde{p}_0 \cdot X^0 + \tilde{p}_1 \cdot X^1 + \ldots + \tilde{p}_{N-1} \cdot X^{N-1} \in \mathbb{Z}[X]$$

$\underline{\mathrm{Commit}(\mathrm{ck}, p)}$:

1. Compute $\tilde{p}(X) \in \mathbb{Z}[X]$.
2. For $\alpha > q$, compute $\tilde{p}(\alpha)$.
3. Output $\tilde{p}(\alpha) \cdot G$.

Example: $q = 7, \alpha = 10$

$$\tilde{p}(X) = 3 \cdot X^2 + 2 \cdot X + 4 \in \mathbb{Z}[X]$$

$$\tilde{p}(\alpha) = 3 \cdot 10^2 + 2 \cdot 10 + 4$$

Commit: Encode p as an Integer [BFS20]

$$p(X) = p_0 \cdot X^0 + p_1 \cdot X^1 + \ldots + p_{N-1} \cdot X^{N-1} \in \mathbb{F}_q[X]$$

$$\tilde{p}(X) = \tilde{p}_0 \cdot X^0 + \tilde{p}_1 \cdot X^1 + \ldots + \tilde{p}_{N-1} \cdot X^{N-1} \in \mathbb{Z}[X]$$

<u>Commit(ck, p)</u> :

1. Compute $\tilde{p}(X) \in \mathbb{Z}[X]$.
2. For $\alpha > q$, compute $\tilde{p}(\alpha)$.
3. Output $\tilde{p}(\alpha) \cdot G$.

Example: $q = 7, \alpha = 10$

$$\tilde{p}(X) = 3 \cdot X^2 + 2 \cdot X + 4 \in \mathbb{Z}[X]$$

$$\tilde{p}(\alpha) = 3 \cdot 10^2 + 2 \cdot 10 + 4$$

$$= 324 \in \mathbb{Z}$$

Open and Verify

Products of Integer Encodings [AGLMS23]

Products of Integer Encodings [AGLMS23]

Given $z \xleftarrow{\$} \mathbb{F}_q$, consider $\mathbf{r} := \left(z^{N-1}, z^{N-2}, \ldots, z^1, 1\right)$.

Products of Integer Encodings [AGLMS23]

Given $z \xleftarrow{\$} \mathbb{F}_q$, consider $\mathbf{r} := \left(z^{N-1}, z^{N-2}, \ldots, z^{1}, 1\right).$

$$\tilde{p}(\alpha) \cdot \tilde{r}(\alpha) = \left(\tilde{p}_0 + \tilde{p}_1\alpha + \ldots + \tilde{p}_{N-1}\alpha^{N-1}\right) \cdot \left(\tilde{r}_0 + \tilde{r}_1\alpha + \ldots + \tilde{r}_{N-1}\alpha^{N-1}\right)$$

Products of Integer Encodings [AGLMS23]

Given $z \xleftarrow{\$} \mathbb{F}_q$, consider $\mathbf{r} := \left(z^{N-1}, z^{N-2}, \ldots, z^1, 1\right)$.

$$\tilde{p}(\alpha) \cdot \tilde{r}(\alpha) = \left(\tilde{p}_0 + \tilde{p}_1\alpha + \ldots + \tilde{p}_{N-1}\alpha^{N-1}\right) \cdot \left(\tilde{r}_0 + \tilde{r}_1\alpha + \ldots + \tilde{r}_{N-1}\alpha^{N-1}\right)$$

$$= \left(\tilde{p}_0\tilde{r}_0\right) \cdot \alpha^0 + \left(\tilde{p}_0\tilde{r}_1 + \tilde{p}_1\tilde{r}_0\right) \cdot \alpha^1 + \left(\tilde{p}_0\tilde{r}_2 + \tilde{p}_1\tilde{r}_1 + \tilde{p}_2\tilde{r}_0\right) \cdot \alpha^2 + \ldots$$

Products of Integer Encodings [AGLMS23]

Given $z \xleftarrow{\$} \mathbb{F}_q$, consider $\mathbf{r} := \left(z^{N-1}, z^{N-2}, \ldots, z^{1}, 1\right)$.

$$\tilde{p}(\alpha)\cdot\tilde{r}(\alpha) = \left(\tilde{p}_0 + \tilde{p}_1\alpha + \ldots + \tilde{p}_{N-1}\alpha^{N-1}\right)\cdot\left(\tilde{r}_0 + \tilde{r}_1\alpha + \ldots + \tilde{r}_{N-1}\alpha^{N-1}\right)$$

$$= \left(\tilde{p}_0\tilde{r}_0\right)\cdot\alpha^0 + \left(\tilde{p}_0\tilde{r}_1 + \tilde{p}_1\tilde{r}_0\right)\cdot\alpha^1 + \left(\tilde{p}_0\tilde{r}_2 + \tilde{p}_1\tilde{r}_1 + \tilde{p}_2\tilde{r}_0\right)\cdot\alpha^2 + \ldots$$

$$= \sum_{i=0}^{N-2}\left(\sum_{j+k=i}\tilde{p}_j\tilde{r}_k\right)\alpha^i + \sum_{j=0}^{N-1}\tilde{p}_j\tilde{r}_{N-1-j}\,\alpha^{N-1} + \sum_{i=N}^{2N-2}\left(\sum_{j+k=i}\tilde{p}_j\tilde{r}_k\right)\alpha^i$$

Products of Integer Encodings [AGLMS23]

Given $z \xleftarrow{\$} \mathbb{F}_q$, consider $\mathbf{r} := \left(z^{N-1}, z^{N-2}, \ldots, z^1, 1\right)$.

$$\tilde{p}(\alpha)\cdot\tilde{r}(\alpha) = \left(\tilde{p}_0 + \tilde{p}_1\alpha + \ldots + \tilde{p}_{N-1}\alpha^{N-1}\right)\cdot\left(\tilde{r}_0 + \tilde{r}_1\alpha + \ldots + \tilde{r}_{N-1}\alpha^{N-1}\right)$$

$$= \left(\tilde{p}_0\tilde{r}_0\right)\cdot\alpha^0 + \left(\tilde{p}_0\tilde{r}_1 + \tilde{p}_1\tilde{r}_0\right)\cdot\alpha^1 + \left(\tilde{p}_0\tilde{r}_2 + \tilde{p}_1\tilde{r}_1 + \tilde{p}_2\tilde{r}_0\right)\cdot\alpha^2 + \ldots$$

$$= \sum_{i=0}^{N-2}\left(\sum_{j+k=i}\tilde{p}_j\tilde{r}_k\right)\alpha^i + \sum_{j=0}^{N-1}\tilde{p}_j\tilde{r}_{N-1-j}\,\alpha^{N-1} + \sum_{i=N}^{2N-2}\left(\sum_{j+k=i}\tilde{p}_j\tilde{r}_k\right)\alpha^i$$

$$:= s + \left(\tilde{p}_0\tilde{r}_{N-1} + \tilde{p}_1\tilde{r}_{N-2} + \ldots + \tilde{p}_{N-1}\tilde{r}_0\right)\cdot\alpha^{N-1} + u\cdot\alpha^N$$

Products of Integer Encodings [AGLMS23]

Given $z \xleftarrow{\$} \mathbb{F}_q$, consider $\mathbf{r} := \left(z^{N-1}, z^{N-2}, \ldots, z^{1}, 1\right)$.

$$\tilde{p}(\alpha)\cdot\tilde{r}(\alpha) = \left(\tilde{p}_0 + \tilde{p}_1\alpha + \ldots + \tilde{p}_{N-1}\alpha^{N-1}\right)\cdot\left(\tilde{r}_0 + \tilde{r}_1\alpha + \ldots + \tilde{r}_{N-1}\alpha^{N-1}\right)$$

$$= \left(\tilde{p}_0\tilde{r}_0\right)\cdot\alpha^0 + \left(\tilde{p}_0\tilde{r}_1 + \tilde{p}_1\tilde{r}_0\right)\cdot\alpha^1 + \left(\tilde{p}_0\tilde{r}_2 + \tilde{p}_1\tilde{r}_1 + \tilde{p}_2\tilde{r}_0\right)\cdot\alpha^2 + \ldots$$

$$= \sum_{i=0}^{N-2}\left(\sum_{j+k=i}\tilde{p}_j\tilde{r}_k\right)\alpha^i + \sum_{j=0}^{N-1}\tilde{p}_j\tilde{r}_{N-1-j}\,\alpha^{N-1} + \sum_{i=N}^{2N-2}\left(\sum_{j+k=i}\tilde{p}_j\tilde{r}_k\right)\alpha^i$$

$$:= s + \left(\tilde{p}_0\tilde{r}_{N-1} + \tilde{p}_1\tilde{r}_{N-2} + \ldots + \tilde{p}_{N-1}\tilde{r}_0\right)\cdot\alpha^{N-1} + u\cdot\alpha^N$$

$$t = \langle\tilde{\mathbf{p}}, \tilde{\bar{\mathbf{r}}}\rangle$$

Products of Integer Encodings [AGLMS23]

Given $z \xleftarrow{\$} \mathbb{F}_q$, consider $\mathbf{r} := (z^{N-1}, z^{N-2}, \ldots, z^1, 1)$.

$$\tilde{p}(\alpha)\cdot\tilde{r}(\alpha) = \left(\tilde{p}_0 + \tilde{p}_1\alpha + \ldots + \tilde{p}_{N-1}\alpha^{N-1}\right)\cdot\left(\tilde{r}_0 + \tilde{r}_1\alpha + \ldots + \tilde{r}_{N-1}\alpha^{N-1}\right)$$

$$= \left(\tilde{p}_0\tilde{r}_0\right)\cdot\alpha^0 + \left(\tilde{p}_0\tilde{r}_1 + \tilde{p}_1\tilde{r}_0\right)\cdot\alpha^1 + \left(\tilde{p}_0\tilde{r}_2 + \tilde{p}_1\tilde{r}_1 + \tilde{p}_2\tilde{r}_0\right)\cdot\alpha^2 + \ldots$$

$$= \sum_{i=0}^{N-2}\left(\sum_{j+k=i}\tilde{p}_j\tilde{r}_k\right)\alpha^i + \sum_{j=0}^{N-1}\tilde{p}_j\tilde{r}_{N-1-j}\,\alpha^{N-1} + \sum_{i=N}^{2N-2}\left(\sum_{j+k=i}\tilde{p}_j\tilde{r}_k\right)\alpha^i$$

$$:= s + \underbrace{(\tilde{p}_0\tilde{r}_{N-1} + \tilde{p}_1\tilde{r}_{N-2} + \ldots + \tilde{p}_{N-1}\tilde{r}_0)}_{\tilde{p}(\tilde{z})} \cdot \alpha^{N-1} + u \cdot \alpha^N$$

Products of Integer Encodings [AGLMS23]

Given $z \xleftarrow{\$} \mathbb{F}_q$, consider $\mathbf{r} := \left(z^{N-1}, z^{N-2}, \ldots, z^{1}, 1\right)$.

$$\tilde{p}(\alpha)\cdot\tilde{r}(\alpha) = \left(\tilde{p}_0 + \tilde{p}_1\alpha + \ldots + \tilde{p}_{N-1}\alpha^{N-1}\right)\cdot\left(\tilde{r}_0 + \tilde{r}_1\alpha + \ldots + \tilde{r}_{N-1}\alpha^{N-1}\right)$$

$$= \left(\tilde{p}_0\tilde{r}_0\right)\cdot\alpha^0 + \left(\tilde{p}_0\tilde{r}_1 + \tilde{p}_1\tilde{r}_0\right)\cdot\alpha^1 + \left(\tilde{p}_0\tilde{r}_2 + \tilde{p}_1\tilde{r}_1 + \tilde{p}_2\tilde{r}_0\right)\cdot\alpha^2 + \ldots$$

$$= \sum_{i=0}^{N-2}\left(\sum_{j+k=i}\tilde{p}_j\tilde{r}_k\right)\alpha^i + \sum_{j=0}^{N-1}\tilde{p}_j\tilde{r}_{N-1-j}\,\alpha^{N-1} + \sum_{i=N}^{2N-2}\left(\sum_{j+k=i}\tilde{p}_j\tilde{r}_k\right)\alpha^i$$

$$:= s + \left(\tilde{p}_0\tilde{r}_{N-1} + \tilde{p}_1\tilde{r}_{N-2} + \ldots + \tilde{p}_{N-1}\tilde{r}_0\right)\cdot\alpha^{N-1} + u\cdot\alpha^N$$

$$\tilde{p}(\tilde{z}) \mod q = p(z)$$

Strawman Construction

Strawman Construction

$\underline{\text{Open}(\text{ck}, \text{cm}, p, z)}$:

Strawman Construction

$\underline{\mathsf{Open}(\mathsf{ck}, \mathsf{cm}, p, z)}$:

$\underline{\mathsf{Verify}(\mathsf{vk}, \mathsf{cm}, z, y, \pi)}$:

Strawman Construction

Open(ck, cm, p, z) :

Recall that cm $= \tilde{p}(\alpha) \cdot G$

Verify(vk, cm, z, y, π) :

Strawman Construction

$\underline{\mathsf{Open}(\mathsf{ck}, \mathsf{cm}, p, z)}$:

1. Define $\mathbf{r} := (z^{N-1}, z^{N-2}, \ldots, z^1, 1)$ and compute $\tilde{r}(\alpha)$.

$\underline{\mathsf{Verify}(\mathsf{vk}, \mathsf{cm}, z, y, \pi)}$:

1. Define $\mathbf{r} := (z^{N-1}, z^{N-2}, \ldots, z^1, 1)$ and compute $\tilde{r}(\alpha)$.

Strawman Construction

$\underline{\mathsf{Open}(\mathsf{ck}, \mathsf{cm}, p, z)}$:

1. Define $\mathbf{r} := (z^{N-1}, z^{N-2}, \ldots, z^1, 1)$ and compute $\tilde{r}(\alpha)$.
2. Output $\pi := (s, t, u)$.

$\underline{\mathsf{Verify}(\mathsf{vk}, \mathsf{cm}, z, y, \pi)}$:

1. Define $\mathbf{r} := (z^{N-1}, z^{N-2}, \ldots, z^1, 1)$ and compute $\tilde{r}(\alpha)$.

Strawman Construction

$\underline{\text{Open}(\text{ck}, \text{cm}, p, z)}$:

1. Define $\mathbf{r} := (z^{N-1}, z^{N-2}, \ldots, z^1, 1)$ and compute $\tilde{r}(\alpha)$.
2. Output $\pi := (s, t, u)$.

$\underline{\text{Verify}(\text{vk}, \text{cm}, z, y, \pi)}$:

1. Define $\mathbf{r} := (z^{N-1}, z^{N-2}, \ldots, z^1, 1)$ and compute $\tilde{r}(\alpha)$.
2. Check that:

Strawman Construction

$\underline{\mathsf{Open}(\mathsf{ck}, \mathsf{cm}, p, z)}$:

1. Define $\mathbf{r} := (z^{N-1}, z^{N-2}, \ldots, z^1, 1)$ and compute $\tilde{r}(\alpha)$.
2. Output $\pi := (s, t, u)$.

$\underline{\mathsf{Verify}(\mathsf{vk}, \mathsf{cm}, z, y, \pi)}$:

1. Define $\mathbf{r} := (z^{N-1}, z^{N-2}, \ldots, z^1, 1)$ and compute $\tilde{r}(\alpha)$.
2. Check that:

$$\tilde{r}(\alpha) \cdot \mathsf{cm} = s \cdot G + t \cdot G + \alpha^N \cdot u \cdot G$$

Strawman Construction

$\underline{\mathsf{Open}(\mathsf{ck}, \mathsf{cm}, p, z)}$:

1. Define $\mathbf{r} := (z^{N-1}, z^{N-2}, \ldots, z^1, 1)$ and compute $\tilde{r}(\alpha)$.
2. Output $\pi := (s, t, u)$.

$\underline{\mathsf{Verify}(\mathsf{vk}, \mathsf{cm}, z, y, \pi)}$:

1. Define $\mathbf{r} := (z^{N-1}, z^{N-2}, \ldots, z^1, 1)$ and compute $\tilde{r}(\alpha)$.
2. Check that:

$$\tilde{r}(\alpha) \cdot \mathsf{cm} = (s + t + \alpha^N \cdot u) \cdot G$$

Strawman Construction

$\underline{\mathsf{Open}(\mathsf{ck}, \mathsf{cm}, p, z)}$:

1. Define $\mathbf{r} := (z^{N-1}, z^{N-2}, \ldots, z^1, 1)$ and compute $\tilde{r}(\alpha)$.
2. Output $\pi := (s, t, u)$.

$\underline{\mathsf{Verify}(\mathsf{vk}, \mathsf{cm}, z, y, \pi)}$:

1. Define $\mathbf{r} := (z^{N-1}, z^{N-2}, \ldots, z^1, 1)$ and compute $\tilde{r}(\alpha)$.
2. Check that:

$$(\tilde{r}(\alpha) \cdot \tilde{p}(\alpha)) \cdot G = (s + t + \alpha^N \cdot u) \cdot G$$

Are We Done?

Are We Done?

- For *honest* $\tilde{\mathbf{p}}, \tilde{\mathbf{r}} \in [0, q-1]^N$:

Are We Done?

- For *honest* $\tilde{\mathbf{p}}, \tilde{\mathbf{r}} \in [0, q-1]^N$:

$$\tilde{p}(\alpha) \cdot \tilde{r}(\alpha) = \sum_{i=0}^{N-2} \left(\sum_{j+k=i} \tilde{p}_j \tilde{r}_k \right) \alpha^i + \sum_{j=0}^{N-1} \tilde{p}_j \tilde{r}_{N-1-j} \, \alpha^{N-1} + \sum_{i=N}^{2N-2} \left(\sum_{j+k=i} \tilde{p}_j \tilde{r}_k \right) \alpha^i$$

$$= s + (\tilde{p}_0 \tilde{r}_{N-1} + \tilde{p}_1 \tilde{r}_{N-2} + \ \ldots + \tilde{p}_{N-1} \tilde{r}_0) \cdot \alpha^{N-1} + u \cdot \alpha^N$$

Are We Done?

- For *honest* $\tilde{\mathbf{p}}, \tilde{\mathbf{r}} \in [0, q-1]^N$:

$$\tilde{p}(\alpha)\cdot\tilde{r}(\alpha) = \sum_{i=0}^{N-2}\left(\sum_{j+k=i}\tilde{p}_j\tilde{r}_k\right)\alpha^i + \sum_{j=0}^{N-1}\tilde{p}_j\tilde{r}_{N-1-j}\,\alpha^{N-1} + \sum_{i=N}^{2N-2}\left(\sum_{j+k=i}\tilde{p}_j\tilde{r}_k\right)\alpha^i$$

$$= s + (\tilde{p}_0\tilde{r}_{N-1} + \tilde{p}_1\tilde{r}_{N-2} + \ \dots + \tilde{p}_{N-1}\tilde{r}_0)\cdot\alpha^{N-1} + u\cdot\alpha^N$$

$0 \le s \le \alpha^{N-1}$

$0 \le t \le N \cdot q^2$

$0 \le u \le \alpha^{N-1}$

Are We Done?

- For *honest* $\tilde{\mathbf{p}}, \tilde{\mathbf{r}} \in [0, q-1]^N$:

$$\tilde{p}(\alpha)\cdot\tilde{r}(\alpha) = \sum_{i=0}^{N-2}\left(\sum_{j+k=i}\tilde{p}_j\tilde{r}_k\right)\alpha^i + \sum_{j=0}^{N-1}\tilde{p}_j\tilde{r}_{N-1-j}\,\alpha^{N-1} + \sum_{i=N}^{2N-2}\left(\sum_{j+k=i}\tilde{p}_j\tilde{r}_k\right)\alpha^i$$

$$= s + (\tilde{p}_0\tilde{r}_{N-1} + \tilde{p}_1\tilde{r}_{N-2} + \ \dots + \tilde{p}_{N-1}\tilde{r}_0)\cdot\alpha^{N-1} + u\cdot\alpha^N$$

$0 \le s \le \alpha^{N-1}$

$0 \le t \le N \cdot q^2$

$0 \le u \le \alpha^{N-1}$

s, t, u are too large to send in the clear!

Range Proofs

Range Proofs

- Can the Prover instead directly send $s \cdot G, t \cdot G, u \cdot G$?

Range Proofs

- Can the Prover instead directly send $s \cdot G, t \cdot G, u \cdot G$?

- For soundness to hold, the Verifier needs to be convinced that s, t, u satisfy the appropriate bounds.

Range Proofs

- Can the Prover instead directly send $s \cdot G, t \cdot G, u \cdot G$?
- For soundness to hold, the Verifier needs to be convinced that s, t, u satisfy the appropriate bounds.
- Solution: Range Proofs

Range Proofs

- Can the Prover instead directly send $s \cdot G, t \cdot G, u \cdot G$?

- For soundness to hold, the Verifier needs to be convinced that s, t, u satisfy the appropriate bounds.

- Solution: Range Proofs

$$H = t \cdot G$$

Range Proofs

- Can the Prover instead directly send $s \cdot G, t \cdot G, u \cdot G$?
- For soundness to hold, the Verifier needs to be convinced that s, t, u satisfy the appropriate bounds.
- Solution: Range Proofs

$$H = t \cdot G$$

$$t \in [a, b]$$

Range Proofs

- Can the Prover instead directly send $s \cdot G, t \cdot G, u \cdot G$?
- For soundness to hold, the Verifier needs to be convinced that s, t, u satisfy the appropriate bounds.
- Solution: Range Proofs

$$H = t \cdot G$$

$$t \in [a, b]$$

$$\boxed{t \in [a, b]}$$

Range Proofs

- Can the Prover instead directly send $s \cdot G, t \cdot G, u \cdot G$?

- For soundness to hold, the Verifier needs to be convinced that s, t, u satisfy the appropriate bounds.

- Solution: Range Proofs

$$H = t \cdot G$$

$$t \in [a, b]$$

$$t \in [a, b]$$

$$\Updownarrow$$

$$(t - a)(b - t) \geq 0$$

Proof of Positive Exponent

Proof of Positive Exponent

$$H = t \cdot G$$

Proof of Positive Exponent

$$H = t \cdot G$$

$$t \geq 0$$

Proof of Positive Exponent

Proof of Positive Exponent

Tool: An efficient PoSE [BBF19] that $H = x^2 \cdot G$, with $|\pi| = O(\lambda)$ bits.

Proof of Positive Exponent

Tool: An efficient PoSE [BBF19] that $H = x^2 \cdot G$, with $|\pi| = O(\lambda)$ bits.

- Prior work: Lagrange four square theorem

Proof of Positive Exponent

Tool: An efficient PoSE [BBF19] that $H = x^2 \cdot G$, with $|\pi| = O(\lambda)$ bits.

- Prior work: Lagrange four square theorem

$$t \geq 0 \iff \exists [x_i]_{i \in [4]} \text{ s.t. } t = x_1^2 + x_2^2 + x_3^2 + x_4^2$$

Proof of Positive Exponent

Tool: An efficient PoSE [BBF19] that $H = x^2 \cdot G$, with $|\pi| = O(\lambda)$ bits.

- Prior work: Lagrange four square theorem

$$\boxed{t \geq 0 \iff \exists [x_i]_{i \in [4]} \text{ s.t. } t = x_1^2 + x_2^2 + x_3^2 + x_4^2}$$

- For $H = t \cdot G$, Prover sends $\pi := [(H_i := (x_i^2 \cdot G), \pi_i)]_{i \in [4]}$.

Proof of Positive Exponent

Tool: An efficient PoSE [BBF19] that $H = x^2 \cdot G$, with $|\pi| = O(\lambda)$ bits.

- Prior work: Lagrange four square theorem

$$t \geq 0 \iff \exists [x_i]_{i \in [4]} \text{ s.t. } t = x_1^2 + x_2^2 + x_3^2 + x_4^2$$

- For $H = t \cdot G$, Prover sends $\pi := [(H_i := (x_i^2 \cdot G), \pi_i)]_{i \in [4]}$.

- Verifier checks that the PoSE proof π_i for H_i is valid, for all $i \in [4]$, and that

Proof of Positive Exponent

Tool: An efficient PoSE [BBF19] that $H = x^2 \cdot G$, with $|\pi| = O(\lambda)$ bits.

- Prior work: Lagrange four square theorem

$$\boxed{t \geq 0 \iff \exists [x_i]_{i \in [4]} \text{ s.t. } t = x_1^2 + x_2^2 + x_3^2 + x_4^2}$$

- For $H = t \cdot G$, Prover sends $\pi := [(H_i := (x_i^2 \cdot G), \pi_i)]_{i \in [4]}$.

- Verifier checks that the PoSE proof π_i for H_i is valid, for all $i \in [4]$, and that

$$H = H_1 + H_2 + H_3 + H_4$$

Proof of Positive Exponent

Tool: An efficient PoSE [BBF19] that $H = x^2 \cdot G$, with $|\pi| = O(\lambda)$ bits.

- Prior work: Lagrange four square theorem

$$\boxed{t \geq 0 \iff \exists [x_i]_{i \in [4]} \text{ s.t. } t = x_1^2 + x_2^2 + x_3^2 + x_4^2}$$

- For $H = t \cdot G$, Prover sends $\pi := [(H_i := (x_i^2 \cdot G), \pi_i)]_{i \in [4]}$.

- Verifier checks that the PoSE proof π_i for H_i is valid, for all $i \in [4]$, and that

$$t \cdot G = x_1^2 \cdot G + x_2^2 \cdot G + x_3^2 \cdot G + x_4^2 \cdot G$$

Proof of Positive Exponent

Tool: An efficient PoSE [BBF19] that $H = x^2 \cdot G$, with $|\pi| = O(\lambda)$ bits.

- Prior work: Lagrange four square theorem

$$\boxed{t \geq 0 \iff \exists [x_i]_{i \in [4]} \text{ s.t. } t = x_1^2 + x_2^2 + x_3^2 + x_4^2}$$

- For $H = t \cdot G$, Prover sends $\pi := [(H_i := (x_i^2 \cdot G), \pi_i)]_{i \in [4]}$.

- Verifier checks that the PoSE proof π_i for H_i is valid, fo
that

$$t \cdot G = x_1^2 \cdot G + x_2^2 \cdot G + x_3^2 \cdot G + x_4^2$$

Takes $\tilde{O}(N^3)$ time!

Proof of Positive Exponent

Proof of Positive Exponent

- Our new theorem:

Proof of Positive Exponent

- Our new theorem:

$$\boxed{t \geq 0 \iff \exists [x_i]_{i \in [\log N]} \text{ s.t. } t = \sum_{i=1}^{\log N} x_i^2}$$

Proof of Positive Exponent

- Our new theorem:

$$\boxed{t \geq 0 \iff \exists [x_i]_{i\in[\log N]} \text{ s.t. } t = \sum_{i=1}^{\log N} x_i^2}$$

Takes only $\tilde{O}(N)$ time!

Proof of Positive Exponent

- Our new theorem:

$$\boxed{t \geq 0 \iff \exists [x_i]_{i \in [\log N]} \text{ s.t. } t = \sum_{i=1}^{\log N} x_i^2}$$

- But now, sending $\pi = [(H_i := x_i^2 \cdot G, \pi_i)]_{i \in [\log N]}$ requires $O(\log N)$ communication.

Proof of Positive Exponent

- Our new theorem:

$$\boxed{t \geq 0 \iff \exists [x_i]_{i \in [\log N]} \text{ s.t. } t = \sum_{i=1}^{\log N} x_i^2}$$

Proof of Positive Exponent

- Our new theorem:

$$\boxed{t \geq 0 \iff \exists [x_i]_{i \in [\log N]} \text{ s.t. } t = \sum_{i=1}^{\log N} x_i^2}$$

- Instead, Prover sends a single commitment to all x_i's:

Proof of Positive Exponent

- Our new theorem:

$$\boxed{t \geq 0 \iff \exists [x_i]_{i \in [\log N]} \text{ s.t. } t = \sum_{i=1}^{\log N} x_i^2}$$

- Instead, Prover sends a single commitment to all x_i's:

$$C = x_1 \cdot G_1 + x_2 \cdot G_2 + \cdots + x_{\log N} \cdot G_{\log N}$$

Proof of Positive Exponent

- Our new theorem:

$$\boxed{t \geq 0 \iff \exists [x_i]_{i \in [\log N]} \text{ s.t. } t = \sum_{i=1}^{\log N} x_i^2}$$

- Instead, Prover sends a single commitment to all x_i's:

$$C = x_1 \cdot G_1 + x_2 \cdot G_2 + \cdots + x_{\log N} \cdot G_{\log N}$$

- Then uses our new 'Self Inner Product Argument' to convince Verifier that for $H = t \cdot G$, the following holds:

Proof of Positive Exponent

- Our new theorem:

$$\boxed{t \geq 0 \iff \exists [x_i]_{i \in [\log N]} \text{ s.t. } t = \sum_{i=1}^{\log N} x_i^2}$$

- Instead, Prover sends a single commitment to all x_i's:

$$C = x_1 \cdot G_1 + x_2 \cdot G_2 + \cdots + x_{\log N} \cdot G_{\log N}$$

- Then uses our new 'Self Inner Product Argument' to convince Verifier that for $H = t \cdot G$, the following holds:

$$\left\langle (x_1, \ldots, x_{\log N}), (x_1, \ldots, x_{\log N}) \right\rangle = t$$

Proof of Positive Exponent

- Our new theorem:

$$\boxed{t \geq 0 \iff \exists [x_i]_{i \in [\log N]} \text{ s.t. } t = \sum_{i=1}^{\log N} x_i^2}$$

- Instead, Prover sends a single commitment to all x_i's:

$$C = x_1 \cdot G_1 + x_2 \cdot G_2 + \cdots + x_{\log N} \cdot G_{\log N}$$

- Then uses our new 'Self Inner Product Argument' to convince Verifier that for $H = t \cdot G$, the following holds:

$$x_1^2 + \ldots + x_{\log N}^2 = t$$

Proof of Positive Exponent

- Our new theorem:

$$t \geq 0 \iff \exists [x_i]_{i \in [\log N]} \text{ s.t. } t = \sum^{\log N} \ldots$$

Proof size $O(\log \log N)$

- Instead, Prover sends a single commitment to ...

$$C = x_1 \cdot G_1 + x_2 \cdot G_2 + \cdots + x_{\log N} \cdot G_{\log N}$$

- Then uses our new ‘Self Inner Product Argument’ to convince Verifier that for $H = t \cdot G$, the following holds:

$$x_1^2 + \ldots + x_{\log N}^2 = t$$

Conclusion

Summary and Open Questions

Summary and Open Questions

- DewTwo: a transparent multilinear PC scheme with $O(N \log^2 N)$ prover time, $O(\log N)$ verifier time and $O(\log \log N)$/4.5KB proof size.

Summary and Open Questions

- DewTwo: a transparent multilinear PC scheme with $O(N \log^2 N)$ prover time, $O(\log N)$ verifier time and $O(\log \log N)$/4.5KB proof size.

- Removed dependence on the Generic Group Model.

Summary and Open Questions

- DewTwo: a transparent multilinear PC scheme with $O(N \log^2 N)$ prover time, $O(\log N)$ verifier time and $O(\log \log N)$/4.5KB proof size.

- Removed dependence on the Generic Group Model.

Open Questions:

Summary and Open Questions

- DewTwo: a transparent multilinear PC scheme with $O(N \log^2 N)$ prover time, $O(\log N)$ verifier time and $O(\log \log N)$/4.5KB proof size.

- Removed dependence on the Generic Group Model.

<u>Open Questions:</u>

- Improve the prover time to O(N) while maintaining small proofs.

Summary and Open Questions

- DewTwo: a transparent multilinear PC scheme with $O(N \log^2 N)$ prover time, $O(\log N)$ verifier time and $O(\log \log N)$/4.5KB proof size.

- Removed dependence on the Generic Group Model.

Open Questions:

- Improve the prover time to O(N) while maintaining small proofs.

- Construct a plausibly post-quantum secure 'Proof of Squared Exponent'.

Summary and Open Questions

- DewTwo: a transparent multilinear PC scheme with $O(N \log^2 N)$ prover time, $O(\log N)$ verifier time and $O(\log \log N)$/4.5KB proof size.
- Removed dependence on the Generic Group Model.

Open Questions:

- Improve the prover time to O(N) while maintaining small proofs.
- Construct a plausibly post-quantum secure 'Proof of Squared Exponent'.
  - Would imply very efficient post-quantum PCS.

Thank you for listening!

ia.cr/2025/129