PointProofs, Revisited

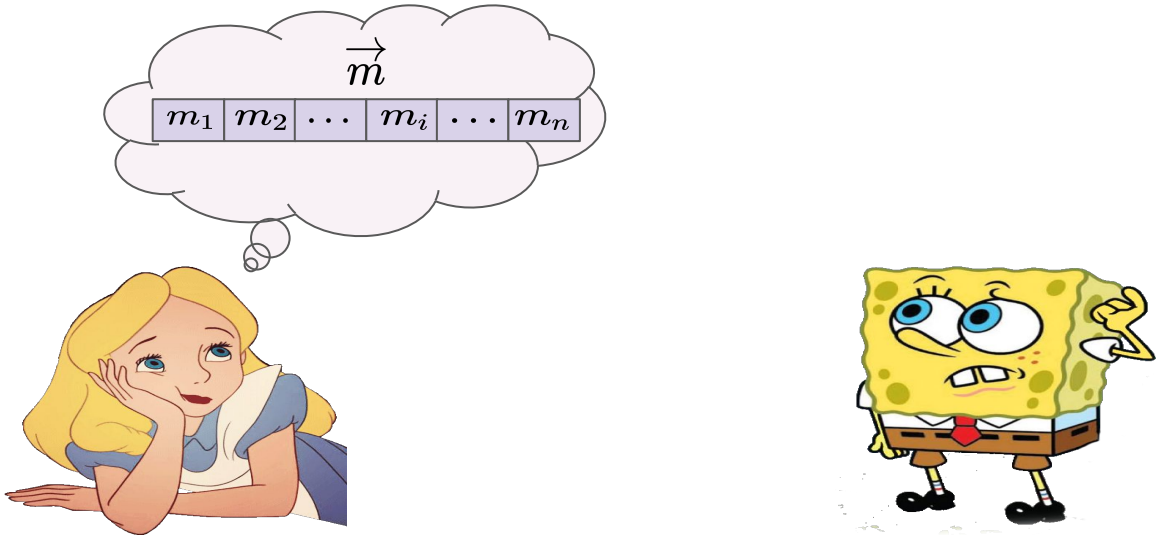
Benoît Libert[1,2], Alain Passelègue[2,3], Mahshid Riahinia[2]

CNRS, Laboratoire LIP, France
ENS de Lyon, Laboratoire LIP (U. Lyon, CNRS, ENSL, Inria, UCBL), France
Inria, France

Asiacrypt, 2022

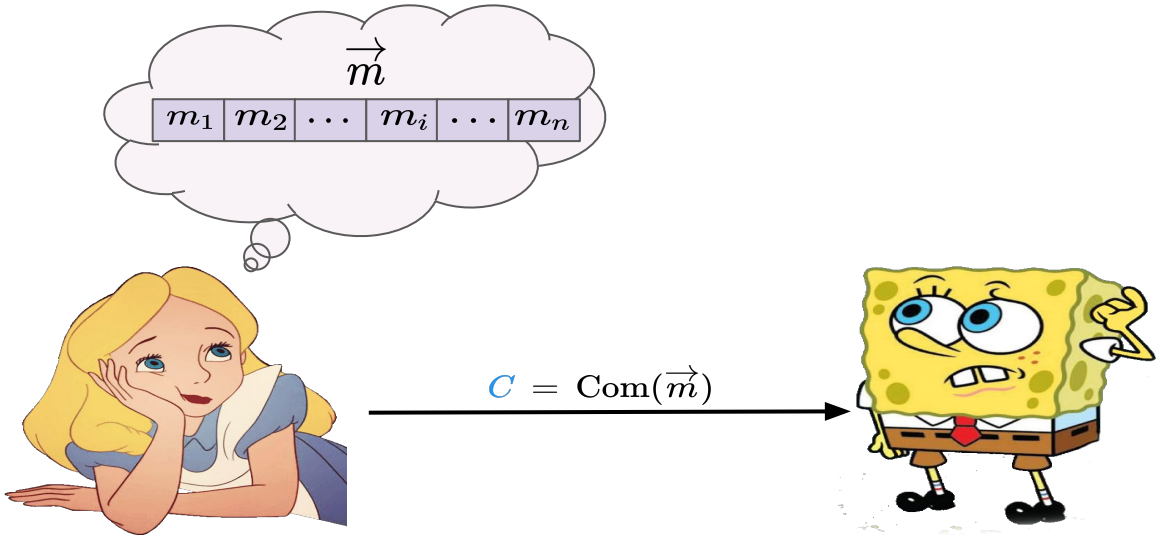
Vector Commitments

Definition. Commit to a vector, and selectively open single entries



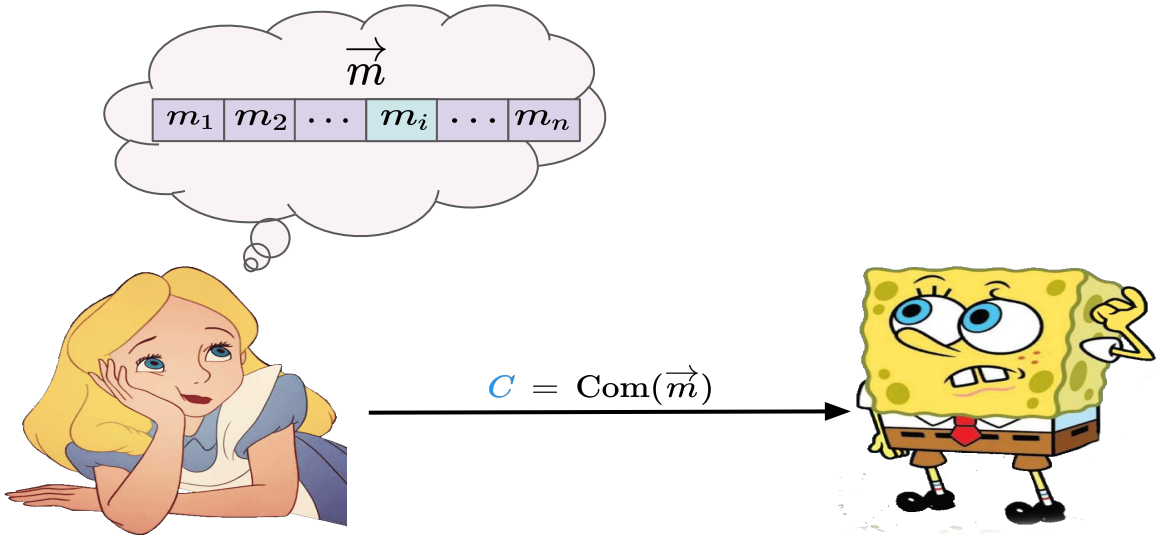
Vector Commitments

Definition. Commit to a vector, and selectively open single entries



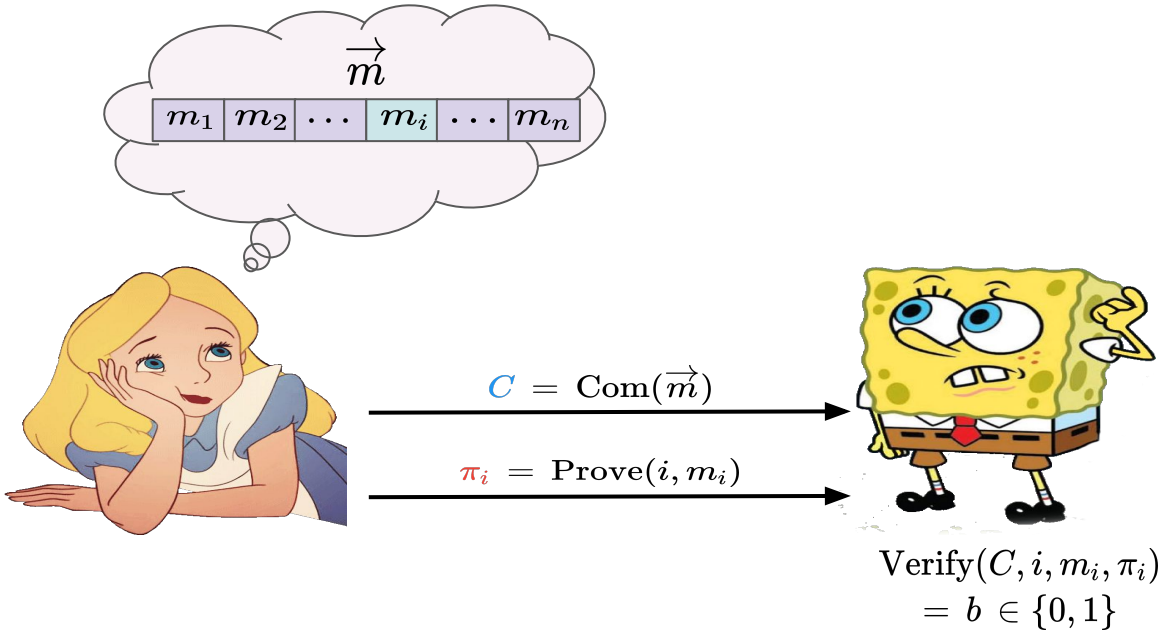
Vector Commitments

Definition. Commit to a vector, and selectively open single entries



Vector Commitments

Definition. Commit to a vector, and selectively open single entries

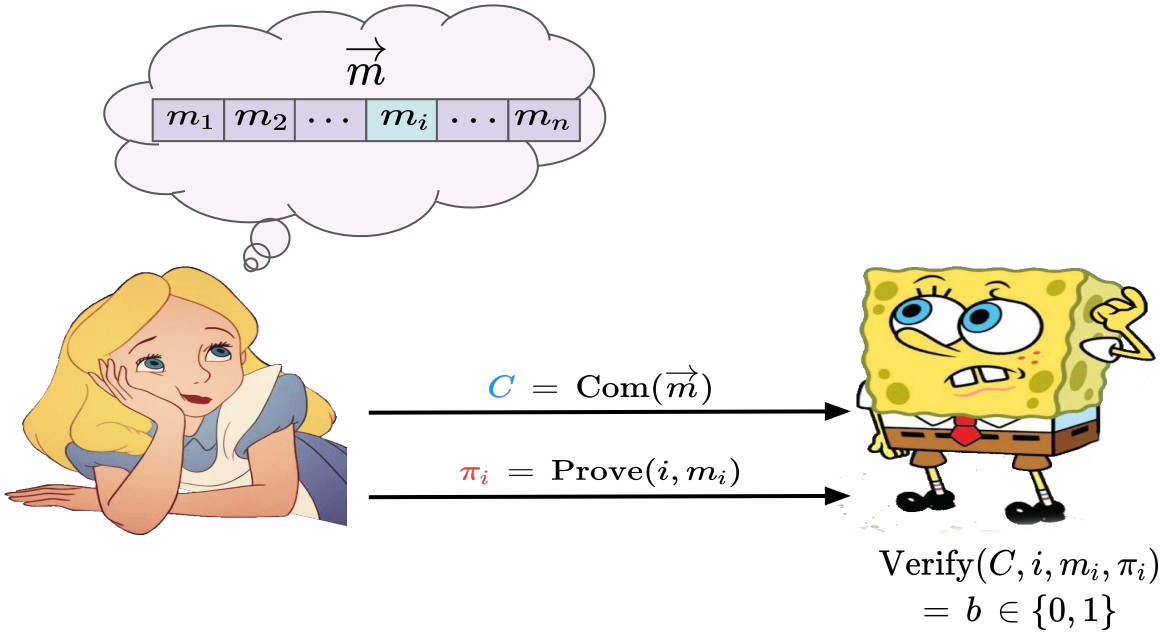


Vector Commitments

Definition. Commit to a vector, and selectively open single entries

Security. Position Binding ✗ Open to two different values at the same position ✗

Hiding ✗ Reveal information about the unopened positions ✗



PointProofs

Gorbunov, Reyzin, Wee, Zhang, 2020

(CCS'20)

PointProofs [Gorbunov, Reyzin, Wee, Zhang, 2020]

PointProofs:

- Vector commitment supporting non-interactive aggregations of proofs
- Supports commitment updates
- Very efficient
  - Applications: Blockchain smart contracts (reducing storage and bandwidth 60%)
- Security: Perfectly hiding & Computationally binding

Under n-wBDHE in AGM+ ROM

PointProofs [Gorbunov, Reyzin, Wee, Zhang, 2020]

PointProofs:

- Vector commitment supporting non-interactive aggregations of proofs
- Supports commitment updates
- Very efficient
  - Applications: Blockchain smart contracts (reducing storage and bandwidth 60%)
- Security: Perfectly hiding & Computationally binding

Under n-wBDHE in AGM+ ROM

PointProofs [Gorbunov, Reyzin, Wee, Zhang, 2020]

PointProofs:

- Vector commitment supporting non-interactive aggregations of proofs
- Supports commitment updates
- Very efficient
  - Applications: Blockchain smart contracts (reducing storage and bandwidth 60%)
- Security: Perfectly hiding & Computationally binding

Our contribution.

Under n-DHE in ~~AGM+~~ ROM

Removing AGM from the proof without changing the scheme

(& from a weaker assumption)

PointProofs [Gorbunov, Reyzin, Wee, Zhang, 2020]

PointProofs:

- Vector commitment supporting non-interactive aggregations of proofs
  - Same-commitment aggregations

    Aggregating single-position proofs for the same commitment

    $\Longrightarrow$ *Proofs for sub-vectors*
  - Cross-commitment aggregations

    Aggregating proofs across different commitments

PointProofs [Gorbunov, Reyzin, Wee, Zhang, 2020]

PointProofs:

- Vector commitment supporting non-interactive aggregations of proofs
  - Same-commitment aggregations

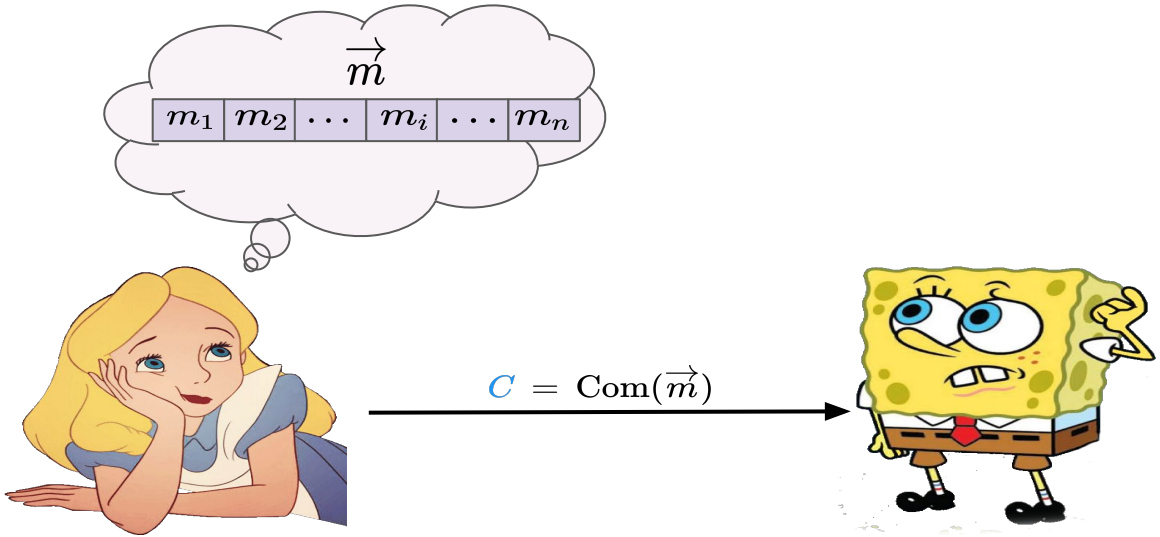
    Aggregating single-position proofs for the same commitment

    $\Longrightarrow$ *Proofs for sub-vectors*
  - Cross-commitment aggregations

    Aggregating proofs across different commitments

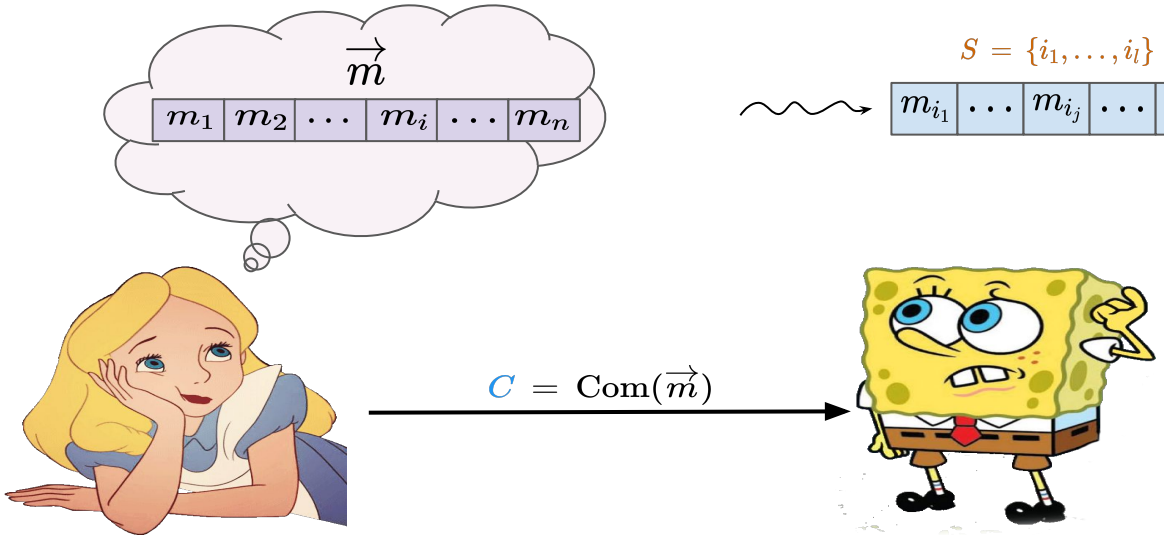
PointProofs [Gorbunov, Reyzin, Wee, Zhang, 2020]

Same-Commitment Aggregation



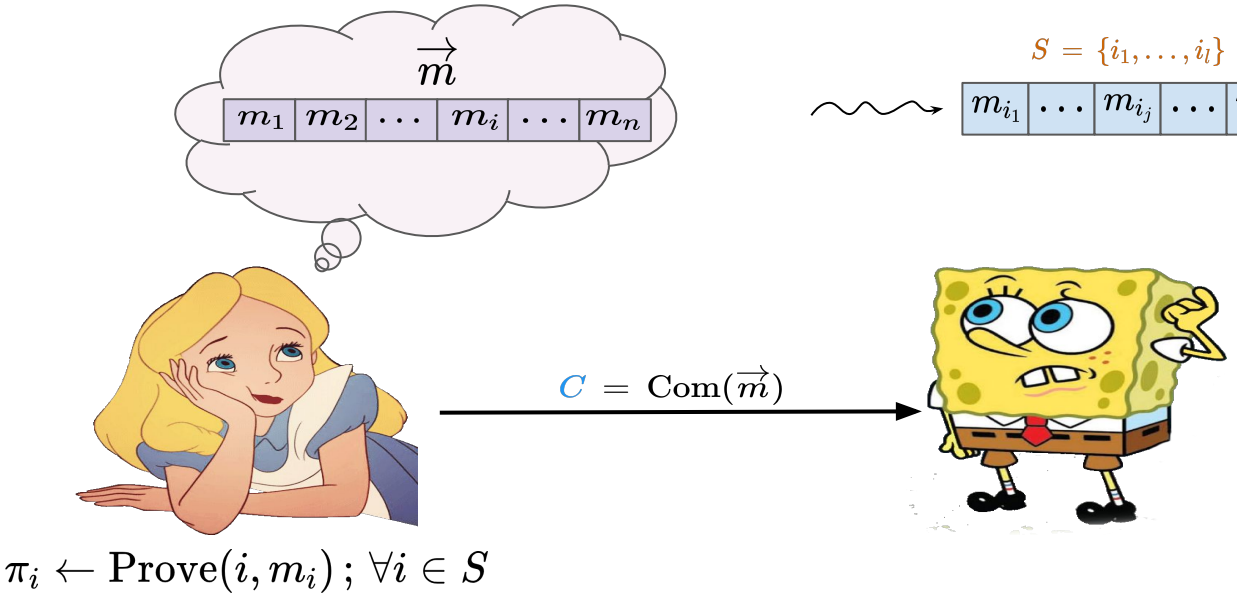
PointProofs [Gorbunov, Reyzin, Wee, Zhang, 2020]

Same-Commitment Aggregation



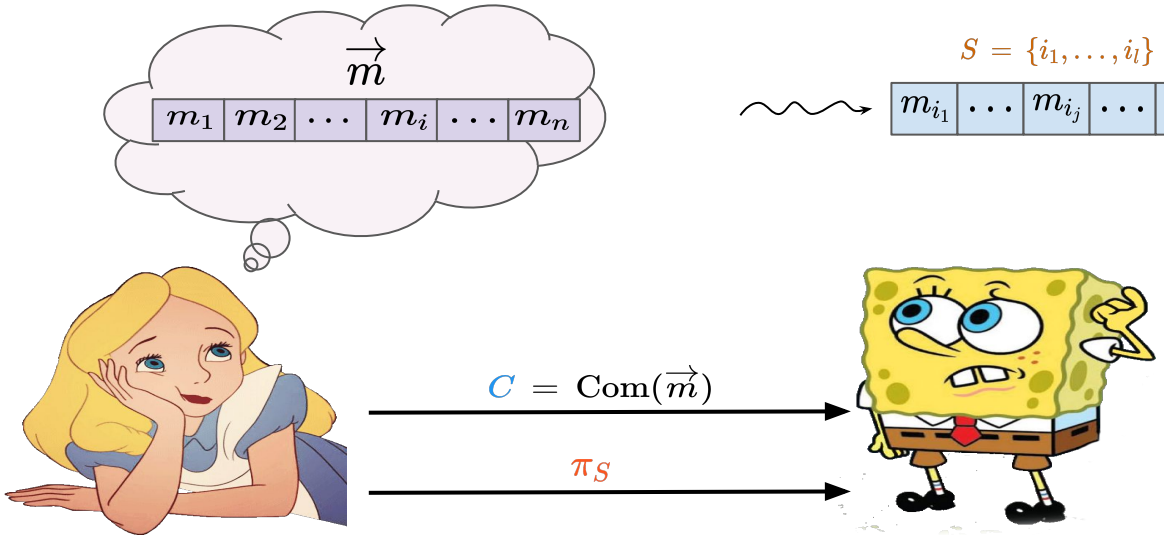
PointProofs [Gorbunov, Reyzin, Wee, Zhang, 2020]

Same-Commitment Aggregation



PointProofs [Gorbunov, Reyzin, Wee, Zhang, 2020]

Same-Commitment Aggregation



$\pi_i \leftarrow \mathrm{Prove}(i, m_i)\,;\, \forall i \in S$

$\pi_S = \mathrm{Aggregate}\big(C, S, m[S], \{\pi_i\}_{i \in S}\big)$

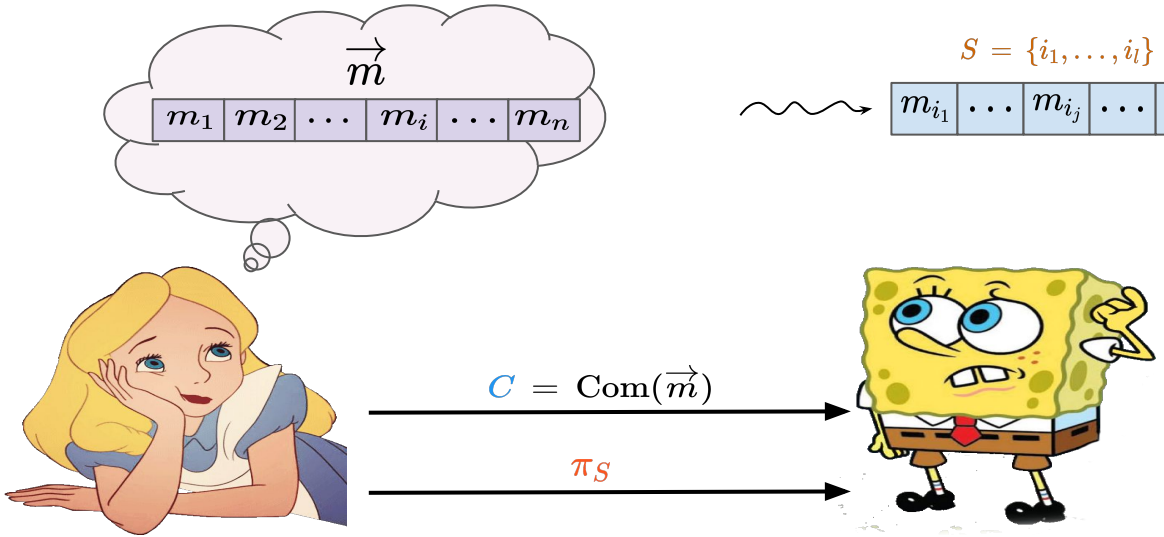
$\mathrm{Verify}(C, S, m[S], \pi_S) = \mathrm{b} \in \{0, 1\}$

PointProofs [Gorbunov, Reyzin, Wee, Zhang, 2020]

Same-Commitment Aggregation

Security. Batch Binding

✗ Open to two different batches with different values at their intersection ✗



$\pi_i \leftarrow \mathrm{Prove}(i, m_i)\,;\, \forall i \in S$

$\pi_S = \mathrm{Aggregate}\big(C, S, m[S], \{\pi_i\}_{i \in S}\big)$

$\mathrm{Verify}(C, S, m[S], \pi_S) = \mathrm{b} \in \{0, 1\}$

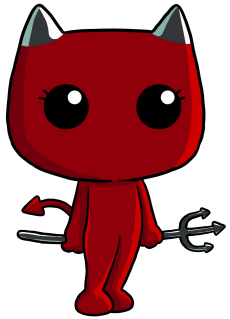
PointProofs [Gorbunov, Reyzin, Wee, Zhang, 2020]

Same-Commitment Aggregation

Security. Batch Binding

✗ Open to two different batches with different values at their intersection ✗

$\vec{m} = (m_1, \ldots, m_n)$, $S = \{i_1, \ldots, i_\ell\} \subseteq \{1, \ldots, n\}$
$m[S] = (m_i)_{i \in S}$



$C = \mathrm{Com}(\vec{m})$

π : $m[S]$ is the subvector on indices in S

π' : $m'[S']$ is the subvector on indices in S'

$\exists i^\star \in S \cap S'$ s.t. $m[i^\star] \neq m'[i^\star]$

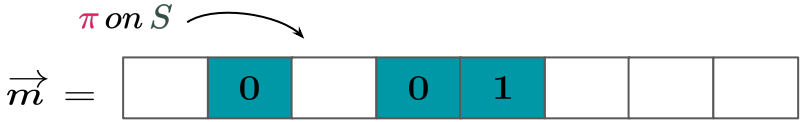
$\mathrm{Verify}(C, S, m[S], \pi) = 1$
$\wedge$
$\mathrm{Verify}(C, S', m'[S'], \pi') = 1$

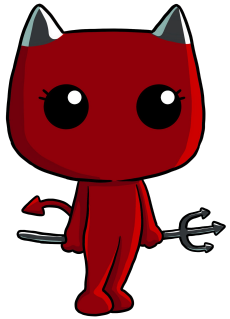
PointProofs [Gorbunov, Reyzin, Wee, Zhang, 2020]

Same-Commitment Aggregation

Security. Batch Binding

✗ Open to two different batches with different values at their intersection ✗





$C = \mathrm{Com}(\vec{m})$

π : $m[S]$ is the subvector on indices in S

π' : $m'[S']$ is the subvector on indices in S′

$\mathrm{Verify}(C, S, m[S], \pi) = 1$

$\wedge$

$\mathrm{Verify}(C, S', m'[S'], \pi') = 1$

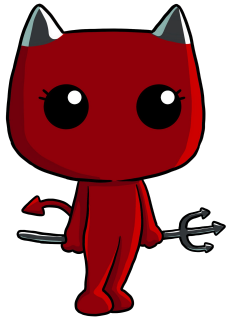
$\exists i^\star \in S \cap S'$ s.t. $m[i^\star] \neq m'[i^\star]$

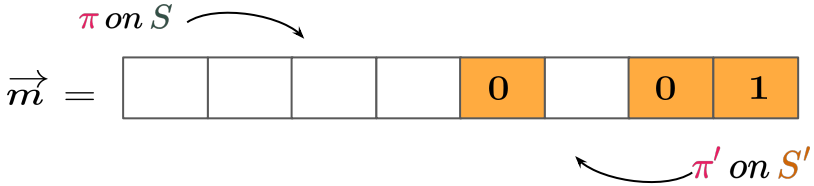
PointProofs [Gorbunov, Reyzin, Wee, Zhang, 2020]

Same-Commitment Aggregation

Security. Batch Binding

✗ Open to two different batches with different values at their intersection ✗





$C = \mathrm{Com}(\vec{m})$

π : $m[S]$ is the subvector on indices in S

π' : $m'[S']$ is the subvector on indices in S′

$\mathrm{Verify}(C, S, m[S], \pi) = 1$

$\wedge$

$\mathrm{Verify}(C, S', m'[S'], \pi') = 1$

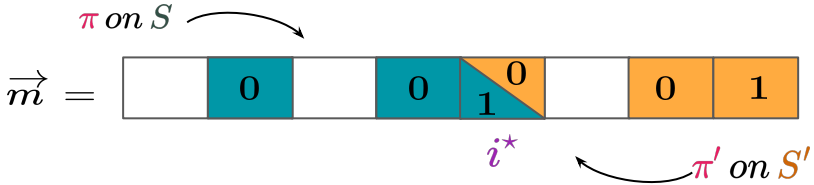
$\exists i^\star \in S \cap S'$ s.t. $m[i^\star] \neq m'[i^\star]$

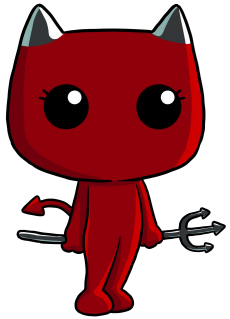
PointProofs [Gorbunov, Reyzin, Wee, Zhang, 2020]

Same-Commitment Aggregation

Security. Batch Binding

✗ Open to two different batches with different values at their intersection ✗





$$C = \mathrm{Com}(\vec{m})$$

π : $m[S]$ is the subvector on indices in S

π' : $m'[S']$ is the subvector on indices in S'

$$\mathrm{Verify}(C, S, m[S], \pi) = 1 \wedge \mathrm{Verify}(C, S', m'[S'], \pi') = 1$$

$$\exists i^\star \in S \cap S' \text{ s.t. } m[i^\star] \neq m'[i^\star]$$

PointProofs [Gorbunov, Reyzin, Wee, Zhang, 2020]

Our contribution.

A new security proof for the PointProofs in the ROM without AGM, from a weaker assumption.

+ perfectly-hiding polynomial commitment with constant size batch openings inspired by PointProofs techniques (full version).

PointProofs [Gorbunov, Reyzin, Wee, Zhang, 2020]

Our contribution.

A new security proof for the PointProofs in the ROM without AGM, from a weaker assumption.

Generic Group Model (GGM) [Sho97]

- Only ***generic*** adversaries are considered.

✗ Only giving access to a random representation of the group.

Computing group operations via oracle queries.

Black-Box Access To The Group

Algebraic Group Model (AGM) [FKL18]

- Only ***algebraic*** adversaries are considered.

✓ No restriction on the access to the group representation

✗ If $\mathcal{A} \to z \in G$, then

$$\mathcal{A} \to (a_1, \ldots, a_t) \in \mathbb{Z}_p^t \text{ s.t. } z = \prod_{i=1}^{t} L_i^{a_i}$$

for known $(L_1, \ldots, L_t)$.

PointProofs [Gorbunov, Reyzin, Wee, Zhang, 2020]

Our contribution.

A new security proof for the PointProofs in the ROM without AGM, from a weaker assumption.

Generic Group Model (GGM) [Sho97]

- Only ***generic*** adversaries are considered.

✗ Only giving access to a random representation of the group.

Computing group operations via oracle queries.

Black-Box Access To The Group

Algebraic Group Model (AGM) [FKL18]

- Only ***algebraic*** adversaries are considered.

✓ No restriction on the access to the group representation

✗ If $\mathcal{A} \to z \in G$, then

$$\mathcal{A} \to (a_1, \ldots, a_t) \in \mathbb{Z}_p^t \text{ s.t. } z = \prod_{i=1}^{t} L_i^{a_i}$$

for known $(L_1, \ldots, L_t)$.

PointProofs [GRWZ'20]

Improved Proof of Same-Commitment Aggregation

PointProofs [GRWZ'20]

Construction

- Public Parameters

  $\mathbb{G} = \langle g \rangle$ of prime order p, $\left(g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^n}, \text{✗}, g^{\alpha^{n+2}}, \ldots, g^{\alpha^{2n}}\right)$

- Commit to $\vec{m} = (m_1, \ldots, m_n)$

1	α^1		α^i		α^n	α^{n+1}		α^{2n}
γ	m_1	$\cdots$	m_i	$\cdots$	m_n	0	$\cdots$	0

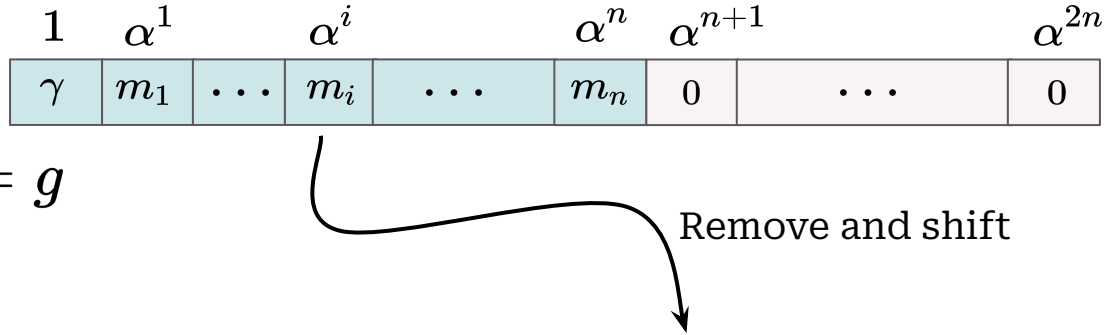
$$C = g^{\gamma} \cdot \prod_{j=1}^{n} g^{\alpha^j \cdot m_j} = g$$

- Public Parameters

$\mathbb{G} = \langle g \rangle$ of prime order p, $\left(g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^n}, \times, g^{\alpha^{n+2}}, \ldots, g^{\alpha^{2n}}\right)$

- Commit to $\vec{m} = (m_1, \ldots, m_n)$

$$C = g^{\gamma} \cdot \prod_{j=1}^{n} g^{\alpha^j \cdot m_j} = g$$



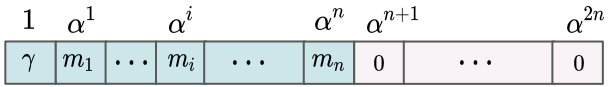
- Open the position i

$$\pi_i = \left(C / g^{\alpha^i \cdot m_i}\right)^{\alpha^{n+1-i}} = g$$

- Commit to $\vec{m} = (m_1, \ldots, m_n)$

$$C = g^{\gamma} \cdot \prod_{j=1}^{n} g^{\alpha^j \cdot m_j}$$

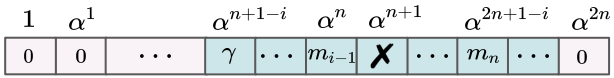
$$= g$$



- Open the position i

$$\pi_i = \left(C / g^{\alpha^i \cdot m_i}\right)^{\alpha^{n+1-i}}$$

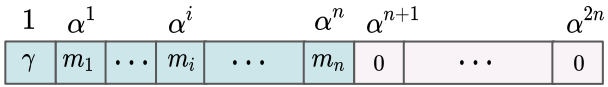
$$= g$$



- Commit to $\vec{m} = (m_1, \ldots, m_n)$

$$C = g^{\gamma} \cdot \prod_{j=1}^{n} g^{\alpha^j \cdot m_j}$$

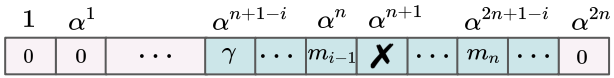
$$= g$$



- Open the position i

$$\pi_i = \left(C / g^{\alpha^i \cdot m_i}\right)^{\alpha^{n+1-i}}$$

$$= g$$



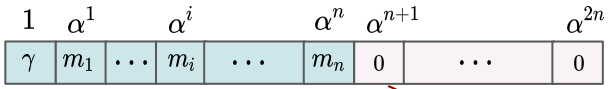
- Verify (C, i, π_i, m_i)

$$e\left(C, \hat{g}^{\alpha^{n+1-i}}\right) = e(\pi_i, \hat{g}) \cdot g_T^{m_i \cdot \alpha^{n+1}}$$

- Commit to $\vec{m} = (m_1, \ldots, m_n)$

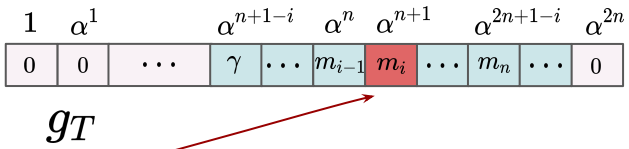
$$C = g^{\gamma} \cdot \prod_{j=1}^{n} g^{\alpha^j \cdot m_j}$$

$$= g$$



- Verify (C, i, π_i, m_i)

$$e\left(C, \hat{g}^{\alpha^{n+1-i}}\right) = e(\pi_i, \hat{g}) \cdot g_T^{m_i \cdot \alpha^{n+1}}$$

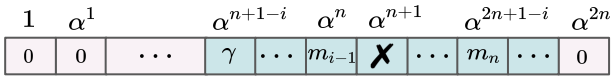


(LHS) Shifting the ***committed*** value

- Open the position i

$$\pi_i = \left(C / g^{\alpha^i \cdot m_i}\right)^{\alpha^{n+1-i}}$$

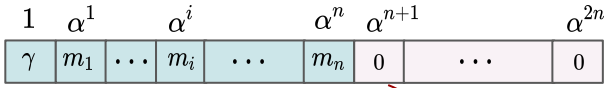
$$= g$$



- Commit to $\vec{m} = (m_1, \ldots, m_n)$

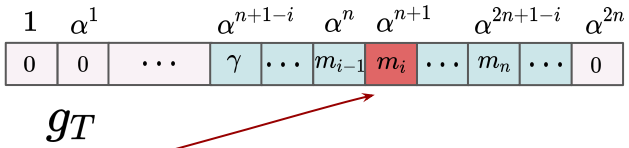
$$C = g^{\gamma} \cdot \prod_{j=1}^{n} g^{\alpha^j \cdot m_j}$$

$$= g$$



- Verify (C, i, π_i, m_i)

$$e\left(C, \hat{g}^{\alpha^{n+1-i}}\right) = e(\pi_i, \hat{g}) \cdot g_T^{m_i \cdot \alpha^{n+1}}$$

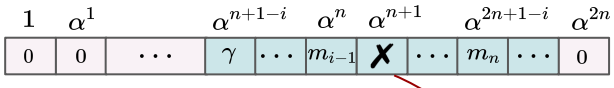


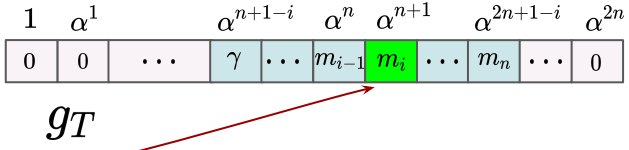
(LHS) Shifting the ***committed*** value

- Open the position i

$$\pi_i = \left(C / g^{\alpha^i \cdot m_i}\right)^{\alpha^{n+1-i}}$$

$$= g$$



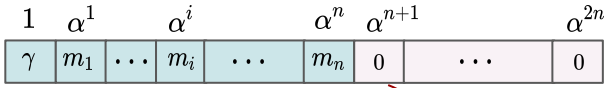


(RHS) Adding the ***opened*** value

- Commit to $\vec{m} = (m_1, \ldots, m_n)$

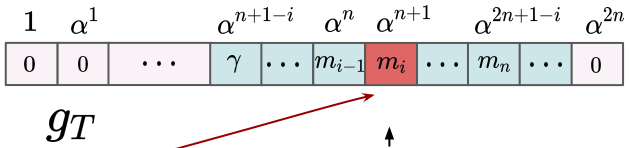
$$C = g^{\gamma} \cdot \prod_{j=1}^{n} g^{\alpha^j \cdot m_j}$$

$$= g$$



- Verify (C, i, π_i, m_i)

$$e\left(C, \hat{g}^{\alpha^{n+1-i}}\right) = e(\pi_i, \hat{g}) \cdot g_T^{m_i \cdot \alpha^{n+1}}$$



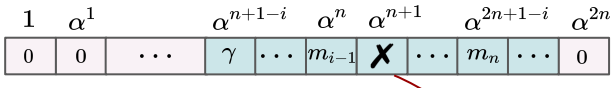
(LHS) Shifting the ***committed*** value

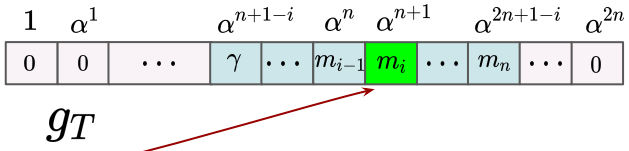
=

- Open the position i

$$\pi_i = \left(C / g^{\alpha^i \cdot m_i}\right)^{\alpha^{n+1-i}}$$

$$= g$$



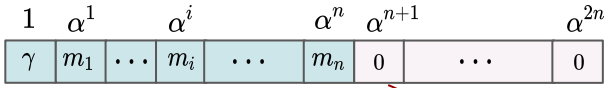


(RHS) Adding the ***opened*** value

- Commit to $\vec{m} = (m_1, \ldots, m_n)$

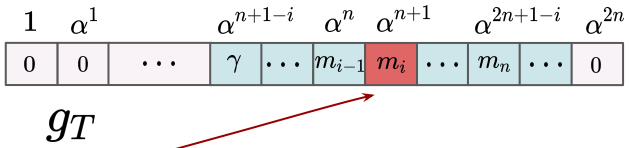
$$C = g^{\gamma} \cdot \prod_{j=1}^{n} g^{\alpha^j \cdot m_j}$$

$$= g$$



- Verify (C, i, π_i, m_i)

$$e\left(C, \hat{g}^{\alpha^{n+1-i}}\right) = e(\pi_i, \hat{g}) \cdot g_T^{m_i \cdot \alpha^{n+1}}$$



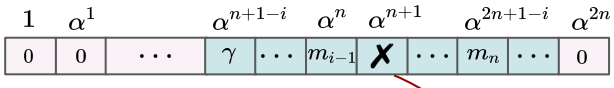
(LHS) Shifting the ***committed*** value

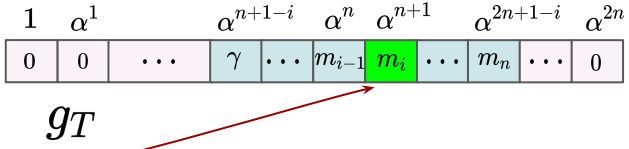
Correct ✓

- Open the position i

$$\pi_i = \left(C / g^{\alpha^i \cdot m_i}\right)^{\alpha^{n+1-i}}$$

$$= g$$





(RHS) Adding the ***opened*** value

- Public Parameters

$$\mathbb{G} = \langle g \rangle \text{ of prime order } p, \left(g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^n}, \cancel{g^{\alpha^{n+1}}}, g^{\alpha^{n+2}}, \ldots, g^{\alpha^{2n}}\right)$$

$$\left(\hat{g}, \hat{g}^{\alpha}, \ldots, \hat{g}^{\alpha^n}\right), \qquad \hat{\mathbb{G}} = \langle \hat{g} \rangle, \quad e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T, \quad g_T = e(g, \hat{g})$$

- Commit to $\vec{m} = (m_1, \ldots, m_n)$

$$C = g^{\gamma} \cdot \prod_{j=1}^{n} g^{\alpha^j \cdot m_j}$$

- Verify (C, i, π_i, m_i)

$$e\left(C, \hat{g}^{\alpha^{n+1-i}}\right) = e(\pi_i, \hat{g}) \cdot g_T^{m_i \cdot \alpha^{n+1}}$$

- Open the position i

$$\pi_i = \left(C / g^{\alpha^i \cdot m_i}\right)^{\alpha^{n+1-i}}$$

- Public Parameters

$\mathbb{G} = \langle g \rangle$ of prime order p, $\left(g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^n}, \cancel{g^{\alpha^{n+1}}}, g^{\alpha^{n+2}}, \ldots, g^{\alpha^{2n}}\right)$

$\left(\hat{g}, \hat{g}^{\alpha}, \ldots, \hat{g}^{\alpha^n}\right)$, $\hat{\mathbb{G}} = \langle \hat{g} \rangle$, $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$, $g_T = e(g, \hat{g})$

n-DHE assumption:
it is hard to compute $g^{\alpha^{n+1}}$
given these elements

- Commit to $\vec{m} = (m_1, \ldots, m_n)$

$$C = g^{\gamma} \cdot \prod_{j=1}^{n} g^{\alpha^j \cdot m_j}$$

- Verify (C, i, π_i, m_i)

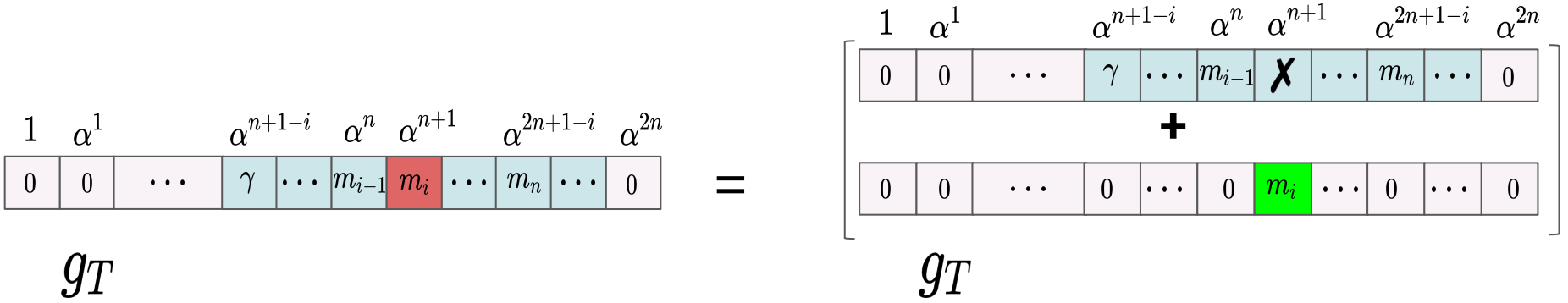
$$e\left(C, \hat{g}^{\alpha^{n+1-i}}\right) = e(\pi_i, \hat{g}) \cdot g_T^{m_i \cdot \alpha^{n+1}}$$

- Open the position i

$$\pi_i = \left(C / g^{\alpha^i \cdot m_i}\right)^{\alpha^{n+1-i}}$$

- Verify (C, i, π_i, m_i)

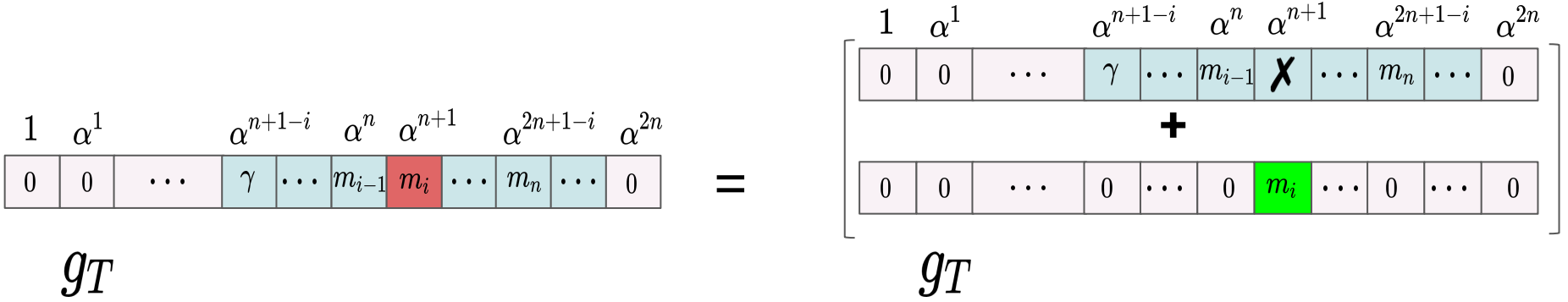
$$e\left(C, \hat{g}^{\alpha^{n+1-i}}\right) = e(\pi_i, \hat{g}) \cdot g_T^{m_i \cdot \alpha^{n+1}}$$



Binding ?

- Verify (C, i, π_i, m_i)

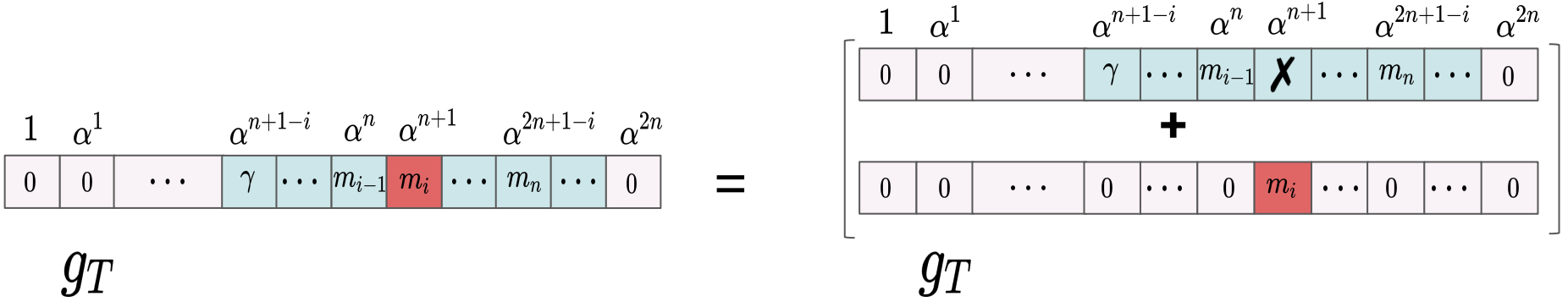
$$e\left(C, \hat{g}^{\alpha^{n+1-i}}\right) = e(\pi_i, \hat{g}) \cdot g_T^{m_i \cdot \alpha^{n+1}}$$



$g^{\alpha^{n+1}}$ is unknown $\Longrightarrow$
Any proof (even adversarial) should have 0 as the coeff of α^{n+1} $\Longrightarrow$
Only the same value of m_i as in the LHS (commitment) gets accepted.

- Verify (C, i, π_i, m_i)

$$e\left(C, \hat{g}^{\alpha^{n+1-i}}\right) = e(\pi_i, \hat{g}) \cdot g_T^{m_i \cdot \alpha^{n+1}}$$



Binding ✓

$$(C, \pi_i) \mapsto m_i$$

PointProofs [GRWZ'20]

Same-Commitment Proof Aggregation

- Commit to $\vec{m} = (m_1, \ldots, m_n)$

$$C = g^{\gamma} \cdot \prod_{j=1}^{n} g^{\alpha^j \cdot m_j}$$

- Open the position i

$$\pi_i = \left(C / g^{\alpha^i \cdot m_i} \right)^{\alpha^{n+1-i}}$$

- Aggregate $(\pi_i)_{i \in S}$

$$\pi_S = \prod_{i \in S} {\pi_i}^{h_i}; \quad h_i = H(i, C, S, m[S])$$

$$(H : \mathbb{G} \to \mathbb{Z}_p)$$

- Commit to $\vec{m} = (m_1, \ldots, m_n)$

$$C = g^{\gamma} \cdot \prod_{j=1}^{n} g^{\alpha^j \cdot m_j}$$

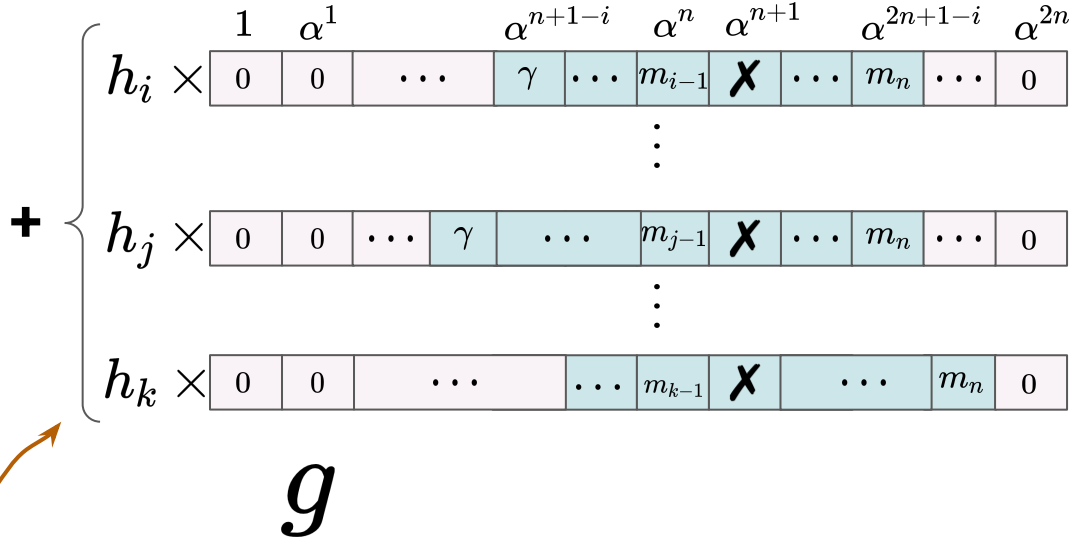
- Open the position i

$$\pi_i = \left(C / g^{\alpha^i \cdot m_i}\right)^{\alpha^{n+1-i}}$$

- Aggregate $(\pi_i)_{i \in S}$

$$\pi_S = \prod_{i \in S} \pi_i{}^{h_i}; \quad h_i = H(i, C, S, m[S])$$

$(H : \mathbb{G} \to \mathbb{Z}_p)$



Missing Part of π_S (coeff of α^{n+1}): $\sum_{i \in S} m_i \cdot h_i$

- Commit to $\vec{m} = (m_1, \ldots, m_n)$

$$C = g^{\gamma} \cdot \prod_{j=1}^{n} g^{\alpha^j \cdot m_j}$$

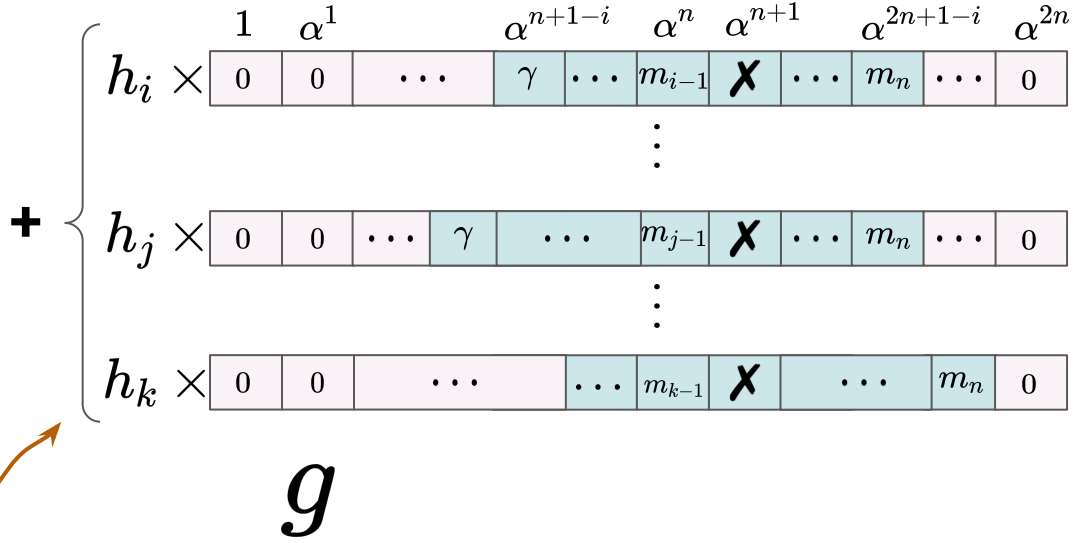
- Open the position i

$$\pi_i = \left(C / g^{\alpha^i \cdot m_i}\right)^{\alpha^{n+1-i}}$$

- Aggregate $(\pi_i)_{i \in S}$

$$\pi_S = \prod_{i \in S} \pi_i^{h_i}; \quad h_i = H(i, C, S, m[S])$$

$(H : \mathbb{G} \to \mathbb{Z}_p)$



$$(C, \pi_S) \mapsto \sum_{i \in S} m_i \cdot h_i$$

- Commit to $\vec{m} = (m_1, \ldots, m_n)$

$$C = g^{\gamma} \cdot \prod_{j=1}^{n} g^{\alpha^j \cdot m_j}$$

- Open the position i

$$\pi_i = \left(C / g^{\alpha^i \cdot m_i}\right)^{\alpha^{n+1-i}}$$

- Aggregate $(\pi_i)_{i \in S}$

$$\pi_S = \prod_{i \in S} {\pi_i}^{h_i}; \quad h_i = H(i, C, S, m[S])$$

$$(H : \mathbb{G} \to \mathbb{Z}_p)$$

- Verify $(C, S, \{\pi_i\}_{i \in S}, m[S])$

$$\underbrace{\prod_{i \in S} e\left(C, \hat{g}^{\alpha^{n+1-i}}\right)^{h_i}}_{\text{Shift and sum all of the committed values}} = \underbrace{e(\pi_S, \hat{g}) \cdot \prod_{i \in S} g_T^{m_i \cdot h_i \cdot \alpha^{n+1}}}_{\text{Adding the missing values to the proofs } \sum_{i \in S} m_i \cdot h_i}$$

- Commit to $\vec{m} = (m_1, \ldots, m_n)$

$$C = g^{\gamma} \cdot \prod_{j=1}^{n} g^{\alpha^j \cdot m_j}$$

- Open the position i

$$\pi_i = \left(C / g^{\alpha^i \cdot m_i}\right)^{\alpha^{n+1-i}}$$

- Aggregate $(\pi_i)_{i \in S}$

$$\pi_S = \prod_{i \in S} \pi_i^{h_i}; \quad h_i = H(i, C, S, m[S])$$

$$(H : \mathbb{G} \to \mathbb{Z}_p)$$

- Verify $(C, S, \{\pi_i\}_{i \in S}, m[S])$

$$\underbrace{\prod_{i \in S} e\left(C, \hat{g}^{\alpha^{n+1-i}}\right)^{h_i}}_{\text{Shift and sum all of the committed values}} = \underbrace{e(\pi_S, \hat{g}) \cdot \prod_{i \in S} g_T^{m_i \cdot h_i \cdot \alpha^{n+1}}}_{\text{Adding the missing values to the proofs } \sum_{i \in S} m_i \cdot h_i}$$

Correct ✓

Batch Binding

$$\mathcal{A} \to (C, S_0, S_1, \vec{m}_0[S_0], \vec{m}_1[S_1], \pi_0, \pi_1)$$

such that

$$\text{Verify}(C, S_0, \vec{m}_0[S_0], \pi_0) = 1 \qquad (C, \pi_0) \mapsto \sum_{i \in S_0} m_0[i] \cdot h_i^{(0)}$$

$$\wedge$$

$$\text{Verify}(C, S_1, \vec{m}_1[S_1], \pi_1) = 1 \qquad (C, \pi_1) \mapsto \sum_{i \in S_1} m_1[i] \cdot h_i^{(1)}$$

$$\wedge$$

$$\exists i^\star \in S_0 \cap S_1 \;\; s.t. \;\; \vec{m}_0[i^\star] \neq \vec{m}_1[i^\star].$$

Batch Binding

$$\mathcal{A} \to (C, S_0, S_1, \vec{m}_0[S_0], \vec{m}_1[S_1], \pi_0, \pi_1)$$

such that

$$\text{Verify}(C, S_0, \vec{m}_0[S_0], \pi_0) = 1$$

$$\wedge$$

$$\text{Verify}(C, S_1, \vec{m}_1[S_1], \pi_1) = 1$$

$$\wedge$$

$$\exists i^\star \in S_0 \cap S_1 \;\; s.t. \;\; \vec{m}_0[i^\star] \neq \vec{m}_1[i^\star].$$

$$(C, \pi_0) \mapsto \sum_{i \in S_0} m_0[i] \cdot h_i^{(0)}$$

$$m_0[i^\star] \cdot h_{i^\star}^{(0)}$$

$$(C, \pi_1) \mapsto \sum_{i \in S_1} m_1[i] \cdot h_i^{(1)}$$

$$m_1[i^\star] \cdot h_{i^\star}^{(1)}$$

Batch Binding

$$\mathcal{A} \to (C, S_0, S_1, \vec{m}_0[S_0], \vec{m}_1[S_1], \pi_0, \pi_1)$$

such that

$$\text{Verify}(C, S_0, \vec{m}_0[S_0], \pi_0) = 1 \qquad (C, \pi_0) \mapsto \sum_{i \in S_0} m_0[i] \cdot h_i^{(0)}$$

$$\wedge$$

$$\text{Verify}(C, S_1, \vec{m}_1[S_1], \pi_1) = 1 \qquad (C, \pi_1) \mapsto \sum_{i \in S_1} m_1[i] \cdot h_i^{(1)}$$

$$\wedge$$

Get rid of other elements to find conflicting proofs for the single position $i^\star$

$$\exists i^\star \in S_0 \cap S_1 \;\; s.t. \;\; \vec{m}_0[i^\star] \neq \vec{m}_1[i^\star].$$

$m_0[i^\star] \cdot h_{i^\star}^{(0)}$

$m_1[i^\star] \cdot h_{i^\star}^{(1)}$

Batch Binding

$$\mathcal{A} \to (C, S_0, S_1, \vec{m}_0[S_0], \vec{m}_1[S_1], \pi_0, \pi_1)$$

such that

$$\text{Verify}(C, S_0, \vec{m}_0[S_0], \pi_0) = 1 \qquad (C, \pi_0) \mapsto \sum_{i \in S_0} m_0[i] \cdot h_i^{(0)}$$

$$\wedge$$

$$\text{Verify}(C, S_1, \vec{m}_1[S_1], \pi_1) = 1 \qquad (C, \pi_1) \mapsto \sum_{i \in S_1} m_1[i] \cdot h_i^{(1)}$$

$$\wedge$$

$$\exists i^\star \in S_0 \cap S_1 \;\; s.t. \;\; \vec{m}_0[i^\star] \neq \vec{m}_1[i^\star].$$

Get rid of other elements to find conflicting proofs for the single position $i^\star$

$m_0[i^\star] \cdot h_{i^\star}^{(0)}$

$m_1[i^\star] \cdot h_{i^\star}^{(1)}$

Control the randomness

PointProofs [GRWZ'20]

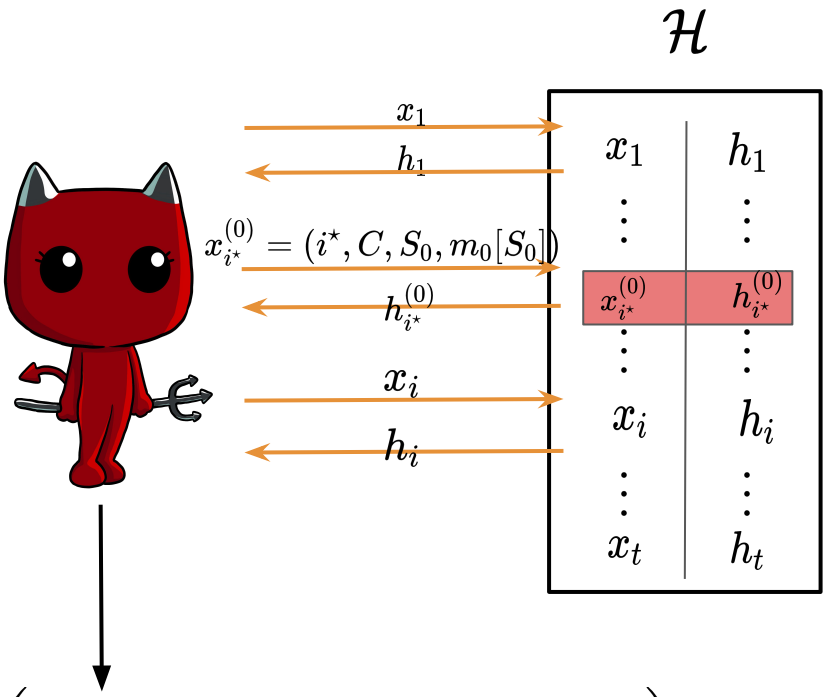
Improved Proof of Same-Commitment Aggregation

Using

Local Forking Lemma

Bellare, Dai, Li (Asiacrypt'19)

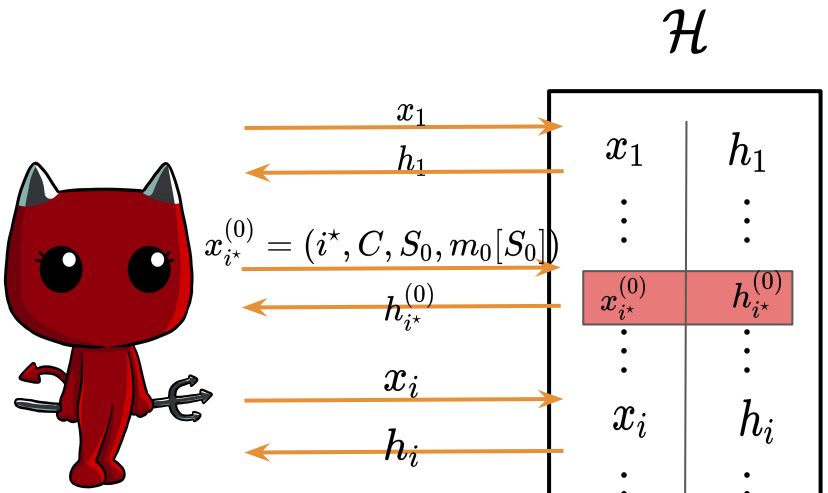
First Run



$$\left(C, S_0, S_1, \overrightarrow{m_0}[S_0], \overrightarrow{m_1}[S_1], \pi_0, \pi_1\right)$$

$$(C, \pi_0) \mapsto \sum_{i \in S_0} m_0[i] \cdot h_i^{(0)}$$

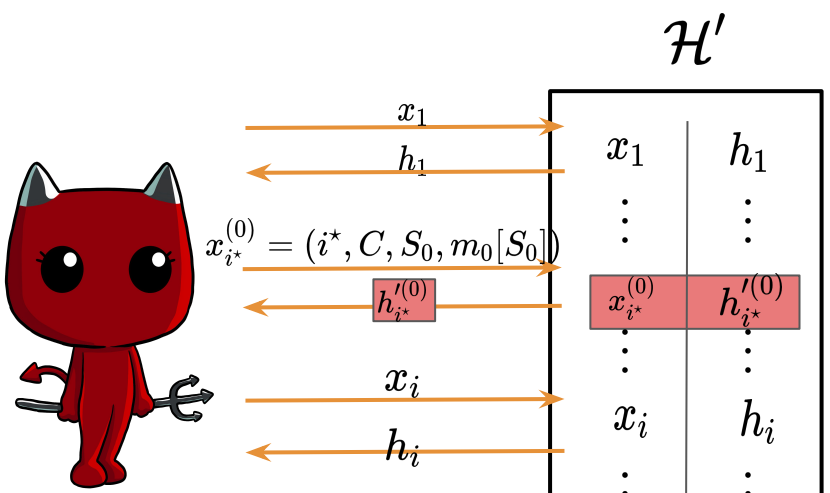
First Run



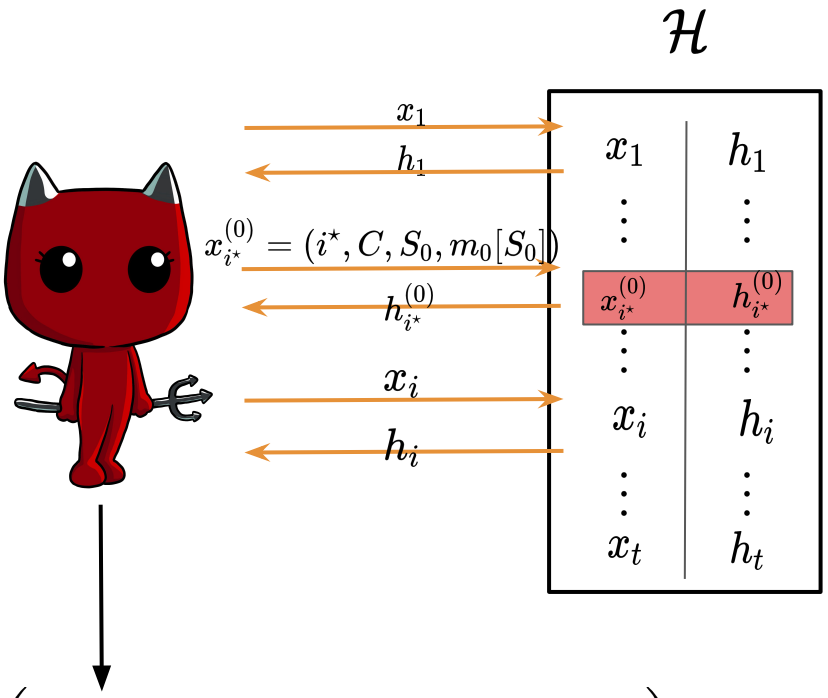
$$\left(C, S_0, S_1, \overrightarrow{m_0}[S_0], \overrightarrow{m_1}[S_1], \pi_0, \pi_1\right)$$

$$(C, \pi_0) \mapsto \sum_{i \in S_0} m_0[i] \cdot h_i^{(0)}$$

Second Run



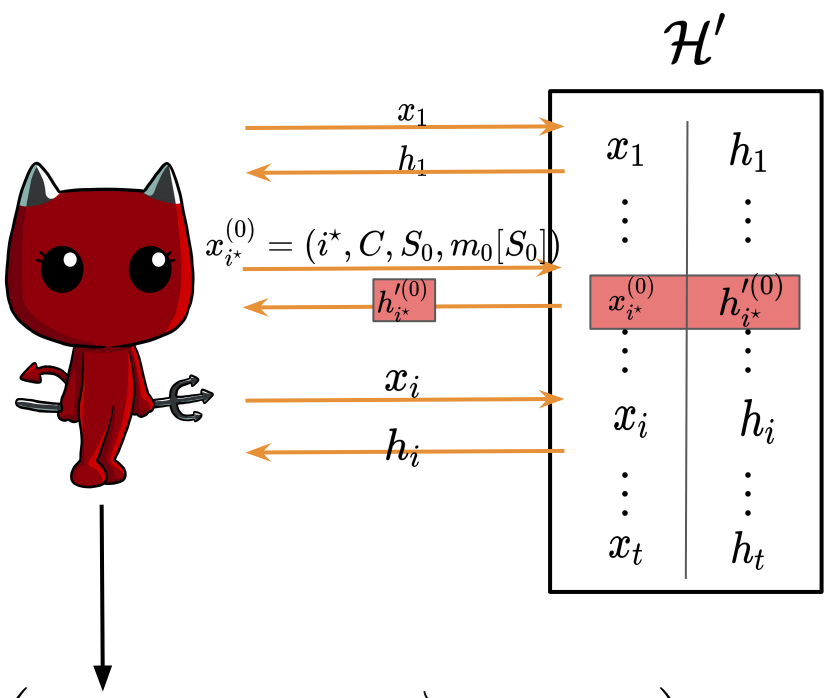
First Run



$$\left(C, S_0, S_1, \overrightarrow{m_0}[S_0], \overrightarrow{m_1}[S_1], \pi_0, \pi_1\right)$$

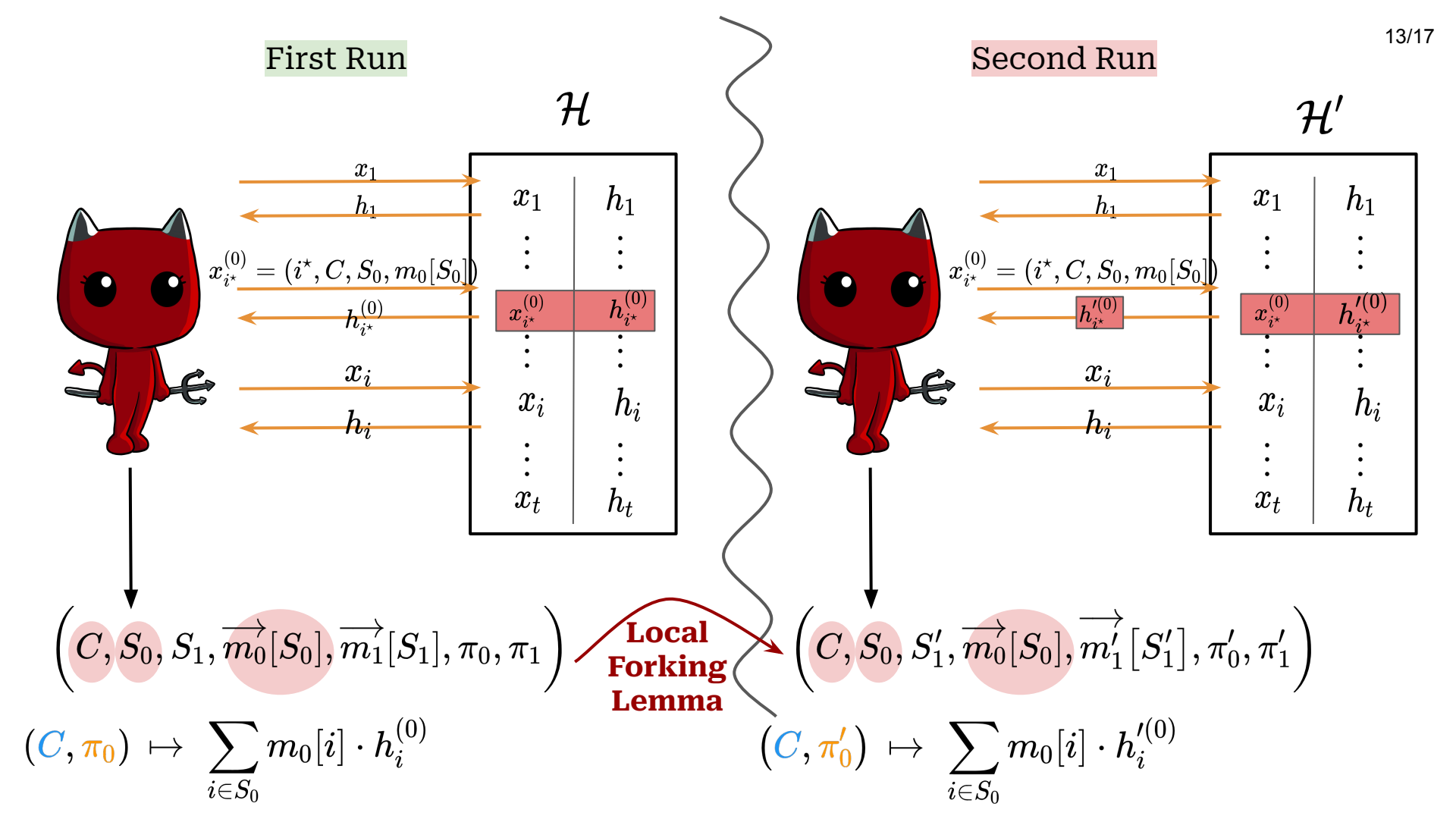
$$(C, \pi_0) \mapsto \sum_{i \in S_0} m_0[i] \cdot h_i^{(0)}$$

Second Run

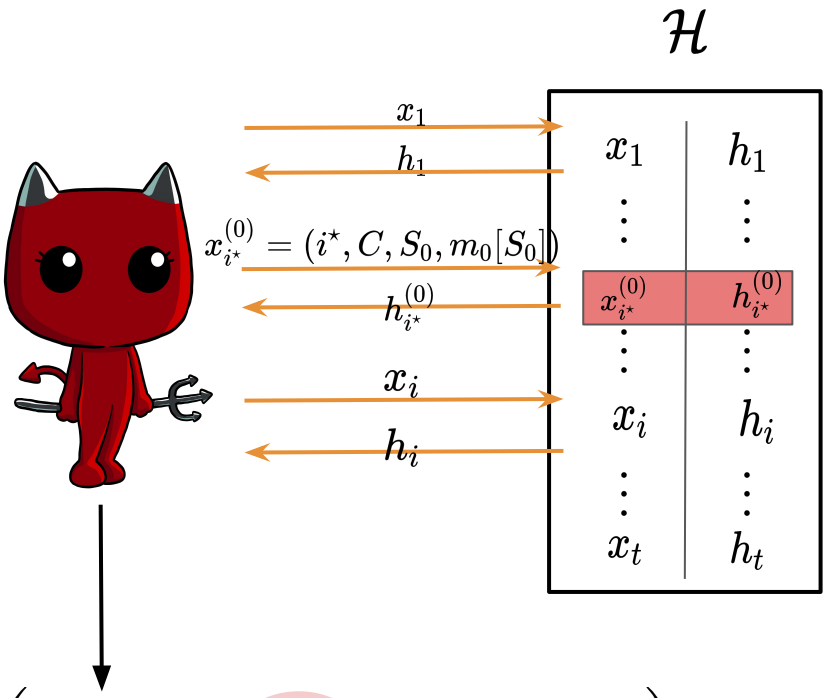


$$\left(C, S_0, S_1', \overrightarrow{m_0}[S_0], \overrightarrow{m_1'}[S_1'], \pi_0', \pi_1'\right)$$

$$(C, \pi_0') \mapsto \sum_{i \in S_0} m_0[i] \cdot h_i'^{(0)}$$


First Run
Second Run
$\mathcal{H}$
$\mathcal{H}'$
x_1
h_1
$x_{i^\star}^{(0)} = (i^\star, C, S_0, m_0[S_0])$
$h_{i^\star}^{(0)}$
$h_{i^\star}'^{(0)}$
x_i
h_i
x_t
h_t
$\left(C, S_0, S_1, \overrightarrow{m_0}[S_0], \overrightarrow{m_1}[S_1], \pi_0, \pi_1\right)$
$\left(C, S_0, S_1', \overrightarrow{m_0}[S_0], \overrightarrow{m_1'}[S_1'], \pi_0', \pi_1'\right)$
Local Forking Lemma
$(C, \pi_0) \mapsto \sum_{i \in S_0} m_0[i] \cdot h_i^{(0)}$
$(C, \pi_0') \mapsto \sum_{i \in S_0} m_0[i] \cdot h_i'^{(0)}$

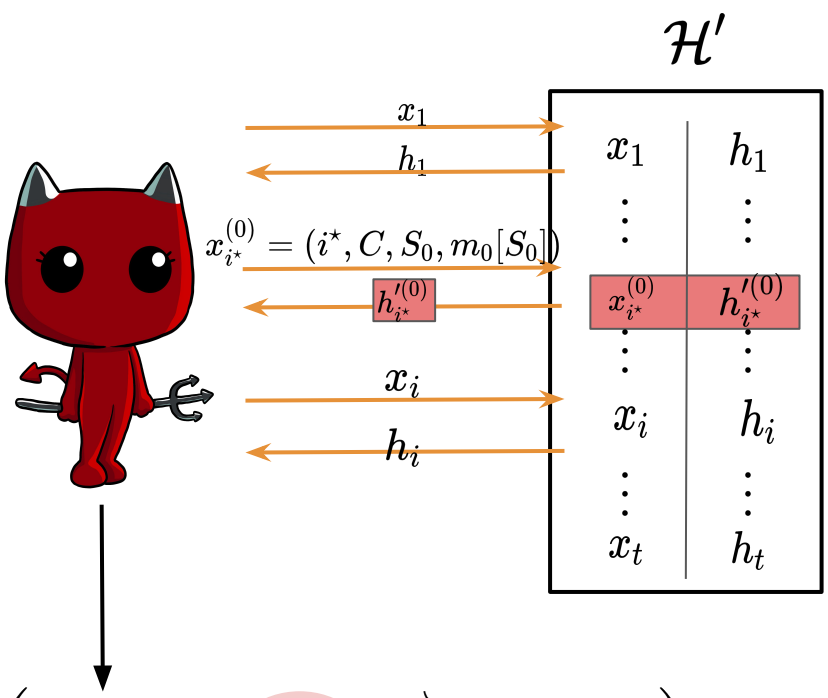
First Run



$$\left(C, S_0, S_1, \overrightarrow{m_0}[S_0], \overrightarrow{m_1}[S_1], \pi_0, \pi_1\right)$$

$$(C, \pi_0) \mapsto \sum_{i \in S_0 \setminus \{i^\star\}} m_0[i] \cdot h_i^{(0)} + m_0[i^\star] \cdot h_{i^\star}^{(0)}$$

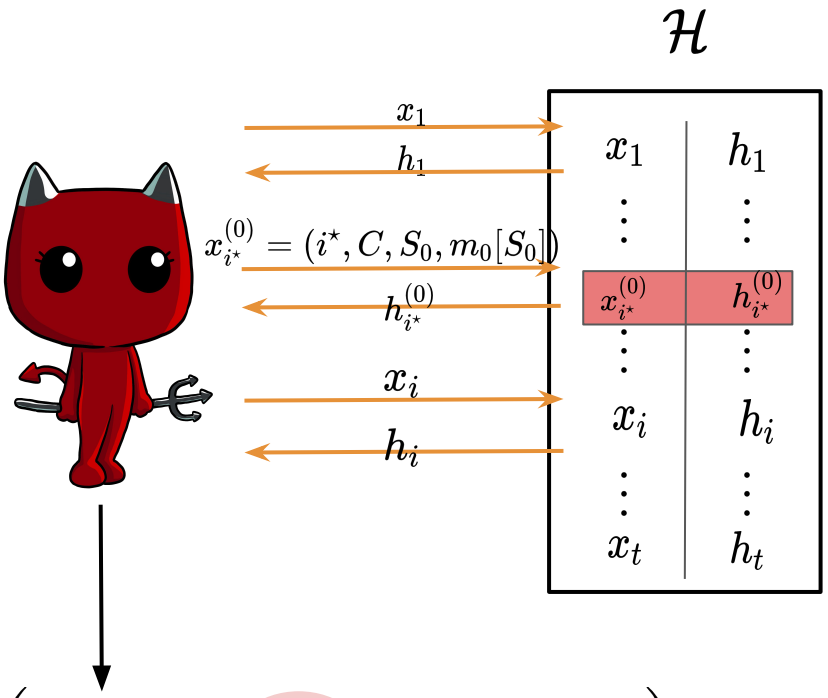
Second Run



$$\left(C, S_0, S_1', \overrightarrow{m_0}[S_0], \overrightarrow{m_1'}[S_1'], \pi_0', \pi_1'\right)$$

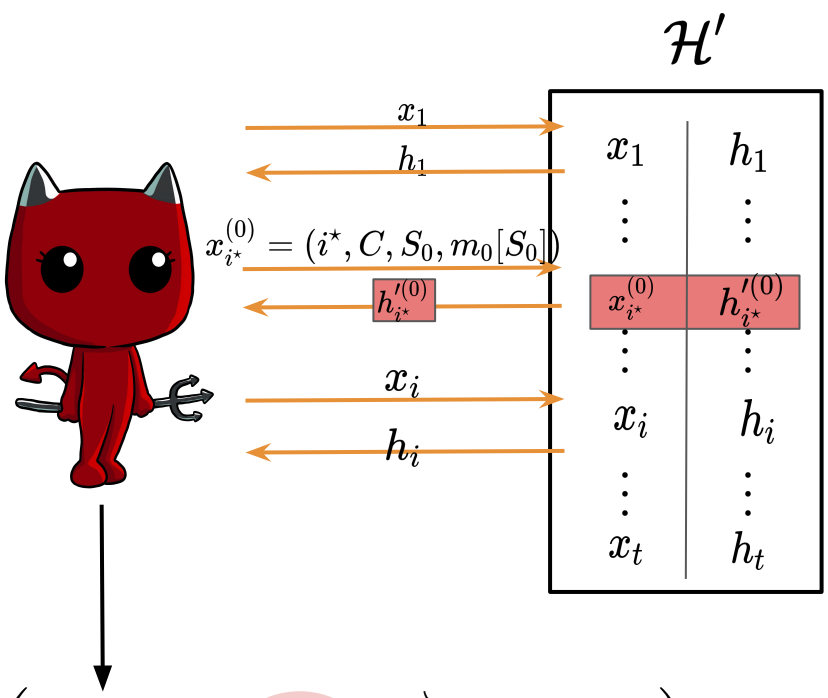
$$(C, \pi_0') \mapsto \sum_{i \in S_0 \setminus \{i^\star\}} m_0[i] \cdot h_i^{(0)} + m_0[i^\star] \cdot h_{i^\star}'^{(0)}$$

First Run



$$\left(C, S_0, S_1, \overrightarrow{m_0}[S_0], \overrightarrow{m_1}[S_1], \pi_0, \pi_1\right)$$

Second Run

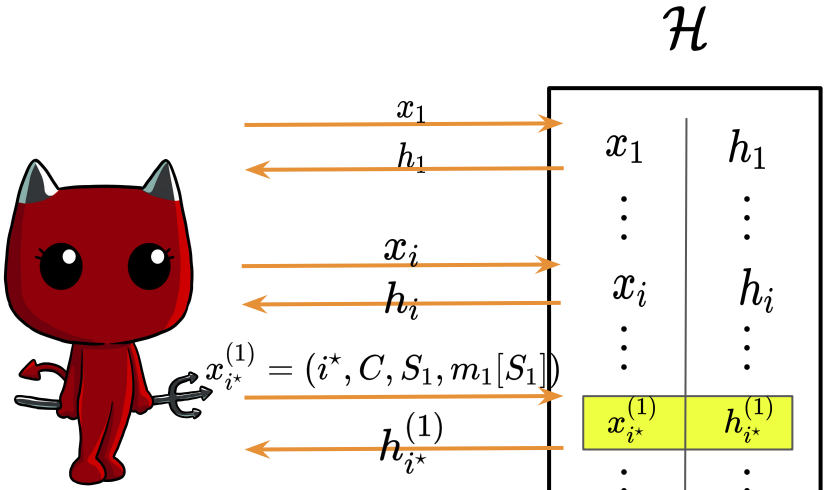


$$\left(C, S_0, S_1', \overrightarrow{m_0}[S_0], \overrightarrow{m_1'}[S_1'], \pi_0', \pi_1'\right)$$

Dividing the two verification equations of S_0

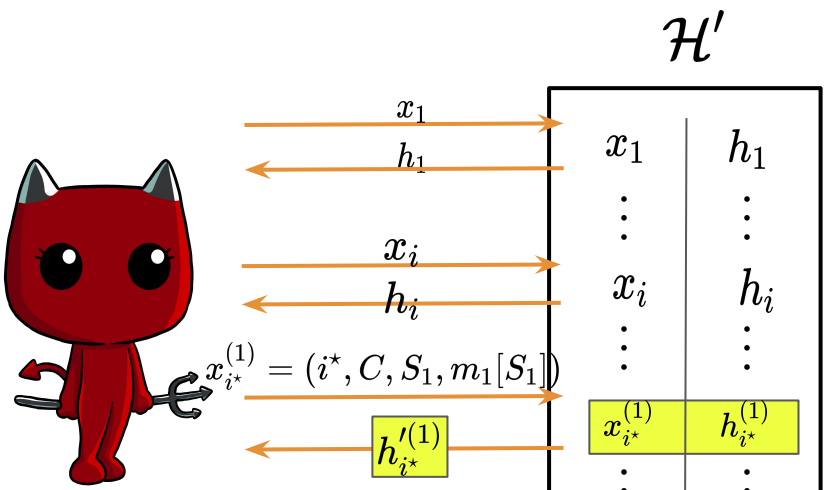
$$e\left(C, \hat{g}^{\alpha^{n+1-i^\star}}\right) = e\left(\left(\pi_0/\pi_0'\right)^{1/\Delta h_{i^\star}^{(0)}}, \hat{g}\right) \cdot g_T^{\alpha^{n+1} \cdot m_0[i^\star]}$$

First Run



$$\left(C, S_0, S_1, \overrightarrow{m_0}[S_0], \overrightarrow{m_1}[S_1], \pi_0, \pi_1\right)$$

Third Run



$$\left(C, S_0', S_1, \overrightarrow{m_0'}[S_0'], \overrightarrow{m_1}[S_1], \pi_0'', \pi_1''\right)$$

Dividing the two verification equations (For S_1)

$$e\left(C, \hat{g}^{\alpha^{n+1-i^\star}}\right) = e\left(\left(\pi_1/\pi_1''\right)^{1/\Delta h_{i^\star}^{(1)}}, \hat{g}\right) \cdot g_T^{\alpha^{n+1} \cdot m_1[i^\star]}$$

- Extracting two conflicting proofs for position $i^\star$ from two conflicting batches:

$$e\left(C, \hat{g}^{\alpha^{n+1-i^\star}}\right) = e\left(\left(\pi_0/\pi_0'\right)^{1/\Delta h_{i^\star}^{(0)}}, \hat{g}\right) \cdot g_T^{\alpha^{n+1} \cdot m_0[i^\star]}$$

$$e\left(C, \hat{g}^{\alpha^{n+1-i^\star}}\right) = e\left(\left(\pi_1/\pi_1''\right)^{1/\Delta h_{i^\star}^{(1)}}, \hat{g}\right) \cdot g_T^{\alpha^{n+1} \cdot m_1[i^\star]}$$

- Contradicting position binding.

- Solving n-DHE problem. ✗

- Extracting two conflicting proofs for position $i^\star$ from two conflicting batches:

$$e\left(C, \hat{g}^{\alpha^{n+1-i^\star}}\right) = e\left(\left(\pi_0/\pi_0'\right)^{1/\Delta h_{i^\star}^{(0)}}, \hat{g}\right) \cdot g_T^{\alpha^{n+1} \cdot m_0[i^\star]}$$

$$e\left(C, \hat{g}^{\alpha^{n+1-i^\star}}\right) = e\left(\left(\pi_1/\pi_1''\right)^{1/\Delta h_{i^\star}^{(1)}}, \hat{g}\right) \cdot g_T^{\alpha^{n+1} \cdot m_1[i^\star]}$$

- Contradicting position binding.

- Solving n-DHE problem. ✗

Batch Binding ✓

Conclusion

- A new security proof for the PointProofs in the ROM (without AGM) without changing the scheme.
  - Same-commitment (generalized or local forking lemma)
  - Cross-commitment (local forking lemma)

- Proposing the first perfectly-hiding polynomial commitment with optimal batch openings (1 group element) inspired by PointProofs techniques.

  Based on inner-product functional commitment of Libert, Ramanna, and Yung (ICALP'16), under the n-DHE assumption.

Conclusion

- A new security proof for the PointProofs in the ROM (without AGM) without changing the scheme.
  - Same-commitment (generalized or local forking lemma)
  - Cross-commitment (local forking lemma)

- Proposing the first perfectly-hiding polynomial commitment with optimal batch openings (1 group element) inspired by PointProofs techniques.

  Based on inner-product functional commitment of Libert, Ramanna, and Yung (ICALP'16), under the n-DHE assumption.

Thank You!