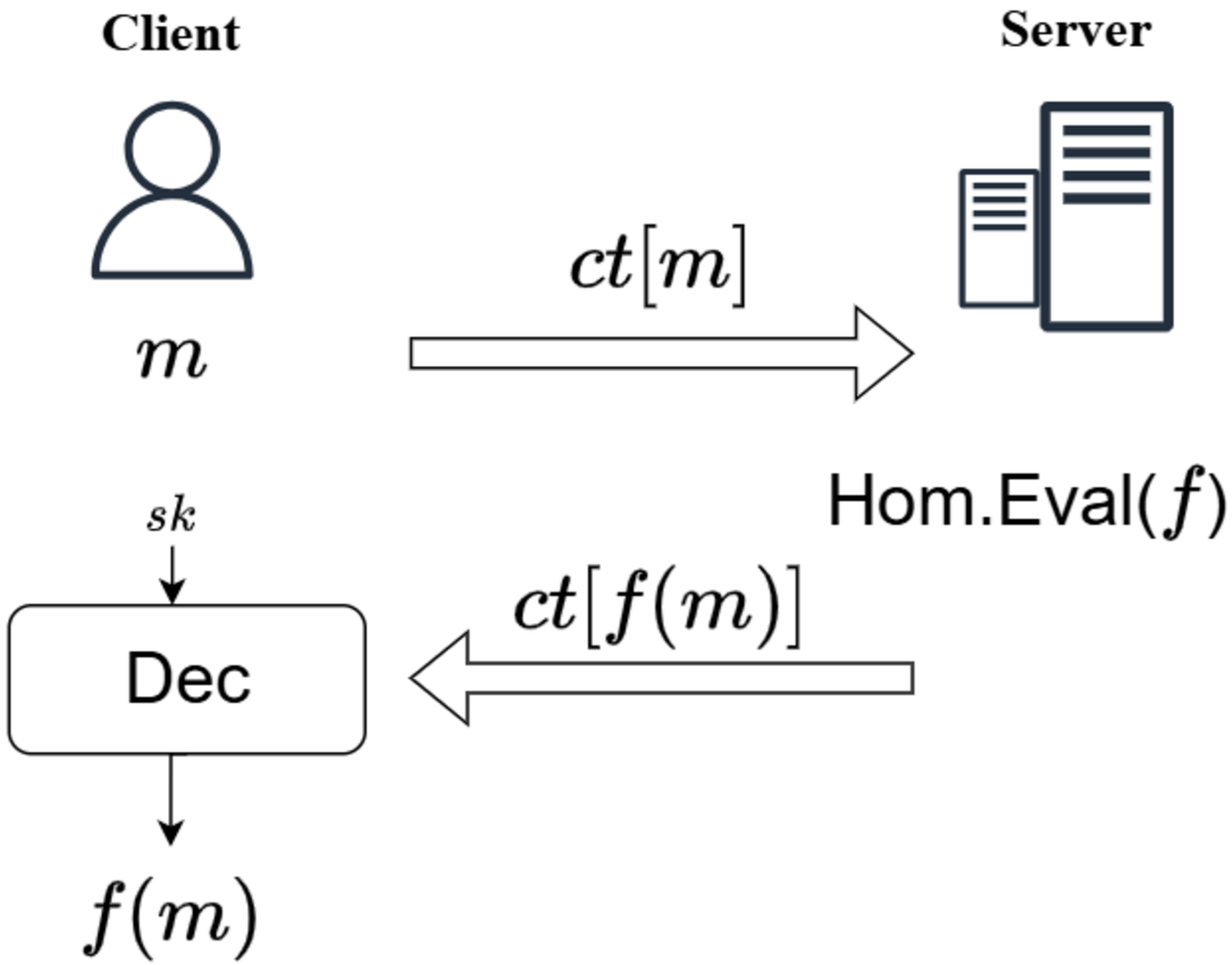
Blind zkSNARKs

for Private Proof Delegation and Verifiable Computation over Encrypted Data

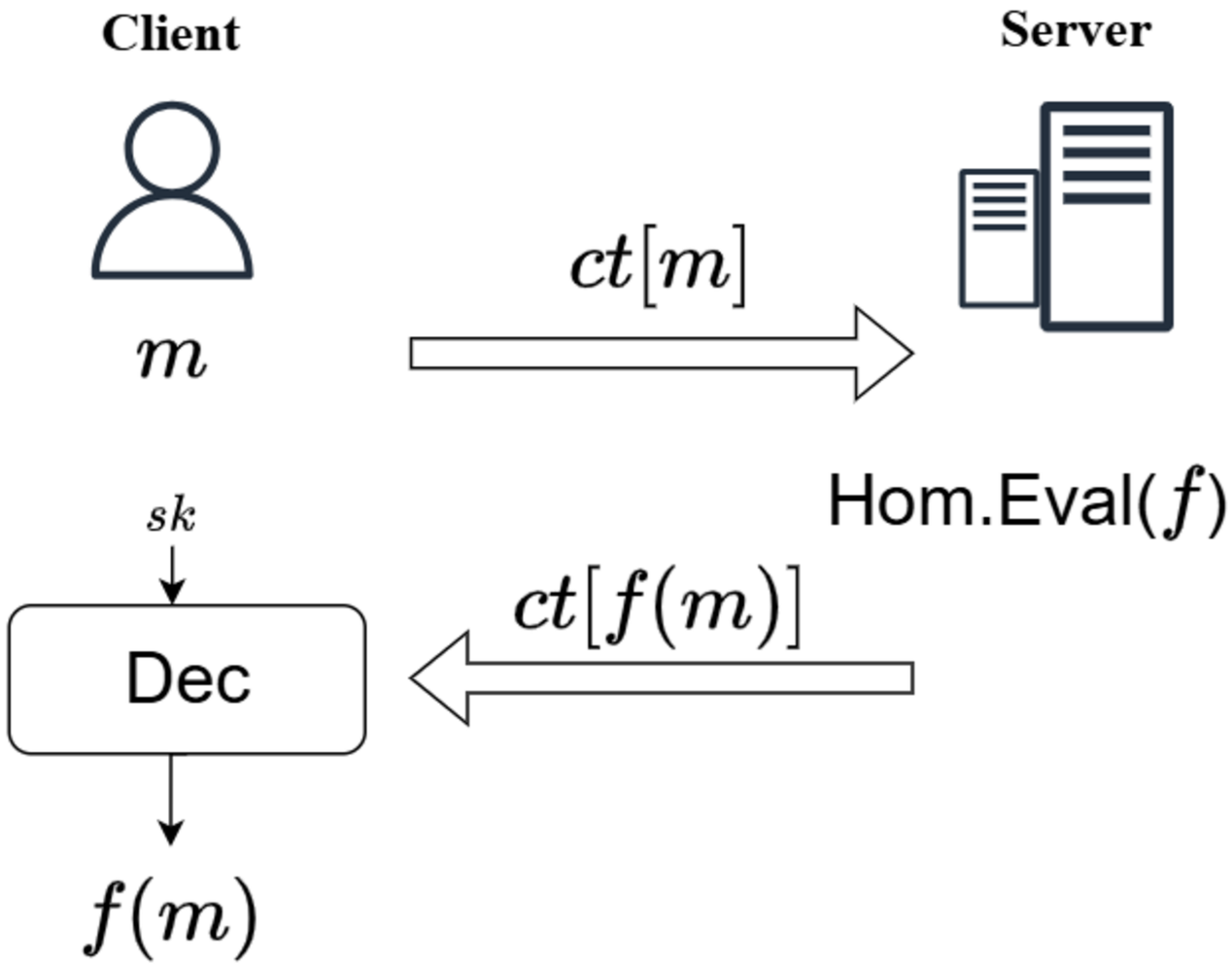
Mariana Gama[1], Emad Heydari Beni[1,2], *Jiayi Kang*[1]
Jannik Spiessens[1], Frederik Vercauteren[1]

[1]COSIC, KU Leuven [2]Nokia Bell Labs

FHE enables computation over encrypted data (COED)



FHE enables computation over encrypted data (COED)



How to achieve verifiable COED (vCOED)?

KU LEUVEN

Two vCOED approaches

vFHE

- [VKH23, GNS23, ABP+24, TW24, CCC+25, LLZ+25, WWX+25, ...]
- prove **ciphertext** relations

$$\mathrm{Prove}_{\mathrm{Hom.Eval}(\mathrm{F})}$$

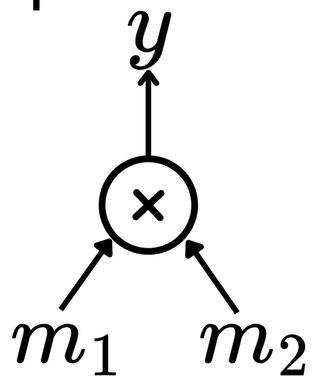
Two vCOED approaches

vFHE

- [VKH23, GNS23, ABP+24, TW24, CCC+25, LLZ+25, WWX+25, ...]
- prove **ciphertext** relations

$$\text{Prove}_{\text{Hom.Eval(F)}}$$

Simple example



Prove

$$ct[y] = \text{Hom. Mult}(ct[m_1], ct[m_2])$$

Two vCOED approaches

vFHE

- [VKH23, GNS23, ABP+24, TW24, CCC+25, LLZ+25, WWX+25, ...]
- prove **ciphertext** relations

$$\mathrm{Prove}_{\mathrm{Hom.Eval}(\mathrm{F})}$$

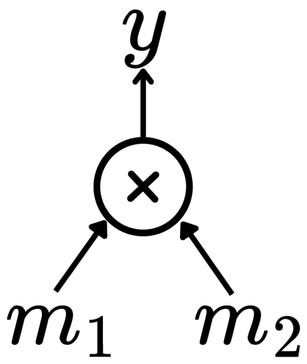
Blind zkSNARK

- [GGW24, ACG+24, GH**K**+24, ZWL+25, ...]
- prove **plaintext** relations

$$\mathrm{Hom.Eval}(\mathrm{Prove}_{\mathrm{F}})$$

Blind zkSNARK proves plaintext relations

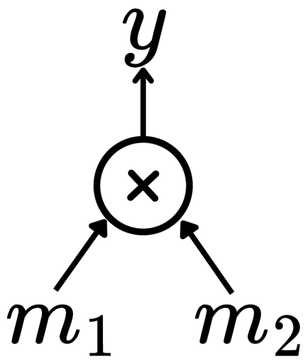
Simple example



- $$[0 \quad 1 \quad 0 \quad 0] \begin{bmatrix} 1 \\ m_1 \\ m_2 \\ y \end{bmatrix} \circ [0 \quad 0 \quad 1 \quad 0] \begin{bmatrix} 1 \\ m_1 \\ m_2 \\ y \end{bmatrix} = [0 \quad 0 \quad 0 \quad 1] \begin{bmatrix} 1 \\ m_1 \\ m_2 \\ y \end{bmatrix} \quad (1)$$

Blind zkSNARK proves plaintext relations

Simple example

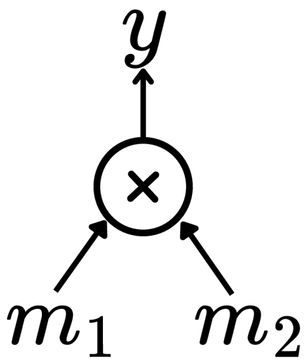


zkSNARK proves the following R1CS relation

- $$\begin{bmatrix}0 & 1 & 0 & 0\end{bmatrix}\begin{bmatrix}1\\ m_1\\ m_2\\ y\end{bmatrix} \circ \begin{bmatrix}0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}1\\ m_1\\ m_2\\ y\end{bmatrix} = \begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1\\ m_1\\ m_2\\ y\end{bmatrix} \quad (1)$$
- $\mathrm{Prove}_{(1)}$

Blind zkSNARK proves plaintext relations

Simple example



zkSNARK proves the following R1CS relation

- $$[0 \quad 1 \quad 0 \quad 0] \begin{bmatrix} 1 \\ m_1 \\ m_2 \\ y \end{bmatrix} \circ [0 \quad 0 \quad 1 \quad 0] \begin{bmatrix} 1 \\ m_1 \\ m_2 \\ y \end{bmatrix} = [0 \quad 0 \quad 0 \quad 1] \begin{bmatrix} 1 \\ m_1 \\ m_2 \\ y \end{bmatrix} \quad (1)$$
- $\mathrm{Prove}_{(1)}$

Blind zkSNARK proves the underlying plaintext relation from ciphertexts

- Given $ct[m_1]$, $ct[m_2]$ and $ct[y]$,

  perform $\mathrm{Hom.Eval}(\mathrm{Prove}_{(1)})$

Warmup Protocol for R1CS [BCR+19]

Given public (sparse) matrices $A, B, C \in \mathbb{F}^{N \times N}$,

the prover wants to prove the knowledge of $z, a, b, c \in \mathbb{F}^{N}$ such that:

$$Az = a$$

$$Bz = b$$

$$Cz = c$$

$$a \circ b = c$$

Warmup Protocol for R1CS [BCR+19]

Given public (sparse) matrices $A, B, C \in \mathbb{F}^{N \times N}$,

the prover wants to prove the knowledge of $z, a, b, c \in \mathbb{F}^{N}$ such that:

$$\left.\begin{aligned} Az &= a \\ Bz &= b \\ Cz &= c \end{aligned}\right\} \text{Lincheck}$$

$$a \circ b = c \quad \text{Rowcheck}$$

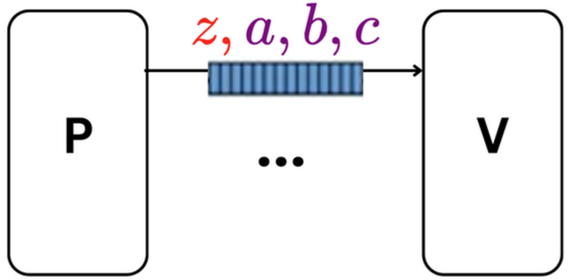
Warmup Protocol for R1CS [BCR+19]

Given public (sparse) matrices $A, B, C \in \mathbb{F}^{N \times N}$,

the prover wants to prove the knowledge of $z, a, b, c \in \mathbb{F}^{N}$ such that:

$$\left.\begin{aligned} Az &= a \\ Bz &= b \\ Cz &= c \end{aligned}\right\} \text{Lincheck}$$

$$a \circ b = c \quad \text{Rowcheck}$$



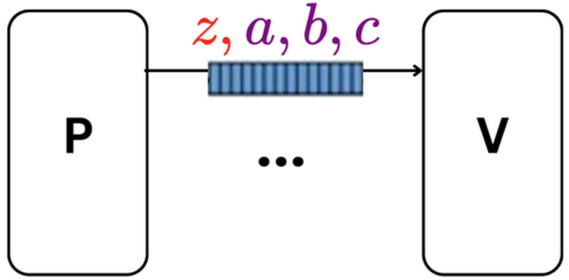
Warmup Protocol for R1CS [BCR+19]

Given public (sparse) matrices $A, B, C \in \mathbb{F}^{N \times N}$,

the prover wants to prove the knowledge of $z, a, b, c \in \mathbb{F}^N$ such that:

$$\left.\begin{aligned} Az &= a \\ Bz &= b \\ Cz &= c \end{aligned}\right\} \text{Lincheck}$$

$$a \circ b = c \quad \text{Rowcheck}$$



- To obtain sublinear query complexity, Aurora/Fractal uses Reed-Solomon codes

e.g. $z \longrightarrow f_z \in \mathsf{RS}[L, \rho]$

$\| \qquad\qquad \|$

$\hat{f}_z|_H \qquad \hat{f}_z|_L$

Prove R1CS with Aurora/Fractal [BCR+19, COS20]

Given $H \subset \mathbb{F}$ and public (sparse) matrices $A, B, C \in \mathbb{F}^{|H| \times |H|}$,

the prover wants to prove the knowledge of $f_z, f_a, f_b, f_c \in \mathrm{RS}[L, \rho]$ such that:

$$A \cdot \hat{f}_z|_H = \hat{f}_a|_H$$

$$B \cdot \hat{f}_z|_H = \hat{f}_b|_H$$

$$C \cdot \hat{f}_z|_H = \hat{f}_c|_H$$

$$\hat{f}_a|_H \circ \hat{f}_b|_H = \hat{f}_c|_H$$

Prove R1CS with Aurora/Fractal [BCR+19, COS20]

Given $H \subset \mathbb{F}$ and public (sparse) matrices $A, B, C \in \mathbb{F}^{|H| \times |H|}$,

the prover wants to prove the knowledge of $f_z, f_a, f_b, f_c \in \mathrm{RS}[L, \rho]$ such that:

$$A \cdot \hat{f}_z|_H = \hat{f}_a|_H$$
$$B \cdot \hat{f}_z|_H = \hat{f}_b|_H$$
$$C \cdot \hat{f}_z|_H = \hat{f}_c|_H$$

Univariate Lincheck

$$\hat{f}_a|_H \circ \hat{f}_b|_H = \hat{f}_c|_H$$

Univariate Rowcheck

Prove R1CS with Aurora/Fractal [BCR+19, COS20]

Given $H \subset \mathbb{F}$ and public (sparse) matrices $A, B, C \in \mathbb{F}^{|H| \times |H|}$,

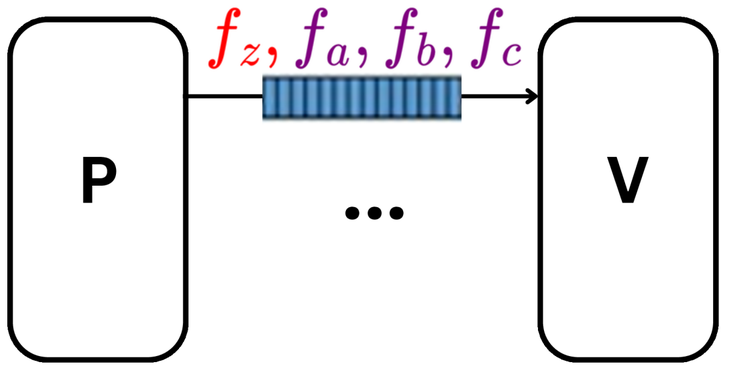
the prover wants to prove the knowledge of $f_z, f_a, f_b, f_c \in \mathrm{RS}[L, \rho]$ such that:

$$A \cdot \hat{f}_z|_H = \hat{f}_a|_H$$
$$B \cdot \hat{f}_z|_H = \hat{f}_b|_H$$
$$C \cdot \hat{f}_z|_H = \hat{f}_c|_H$$

Univariate Lincheck

$$\hat{f}_a|_H \circ \hat{f}_b|_H = \hat{f}_c|_H$$

Univariate Rowcheck



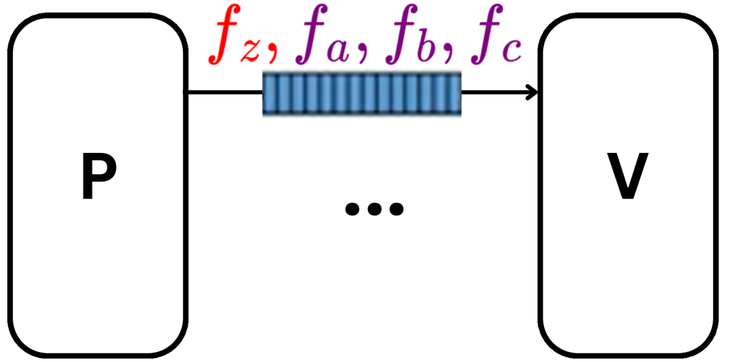
Prove R1CS with Aurora/Fractal [BCR+19, COS20]

Given $H \subset \mathbb{F}$ and public (sparse) matrices $A, B, C \in \mathbb{F}^{|H| \times |H|}$,

the prover wants to prove the knowledge of $f_z, f_a, f_b, f_c \in \mathrm{RS}[L, \rho]$ such that:

$$A \cdot \hat{f}_z|_H = \hat{f}_a|_H$$
$$B \cdot \hat{f}_z|_H = \hat{f}_b|_H$$
$$C \cdot \hat{f}_z|_H = \hat{f}_c|_H$$

Univariate Lincheck



$$\hat{f}_a|_H \circ \hat{f}_b|_H = \hat{f}_c|_H$$

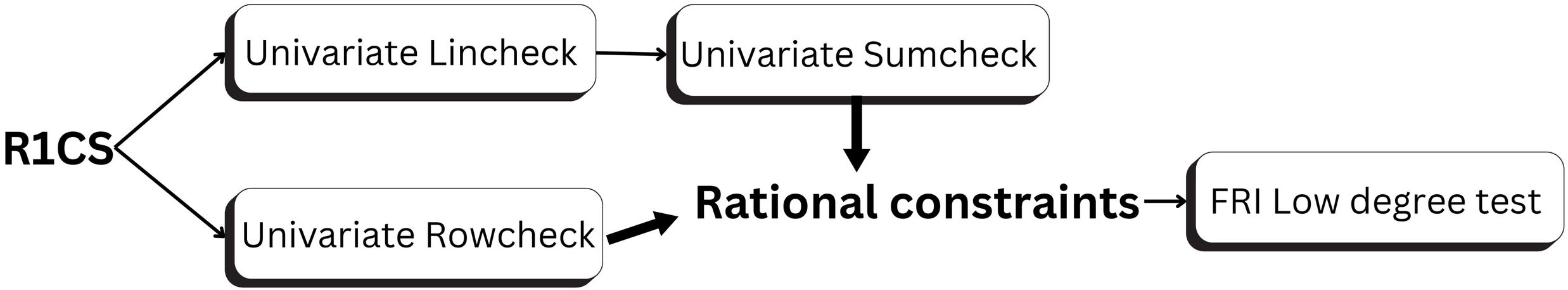
Univariate Rowcheck

Low-degree test of the rational constraint

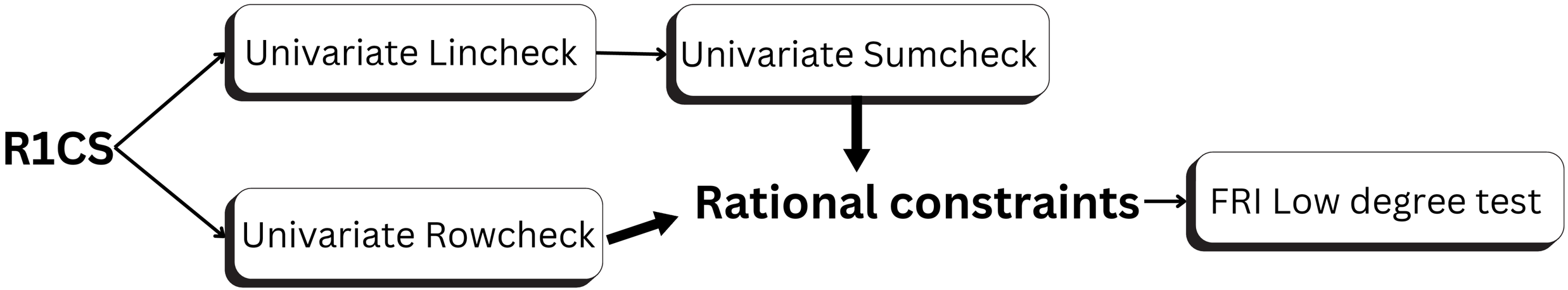
$$Q = \frac{\hat{f}_a \cdot \hat{f}_b - \hat{f}_c}{Z_H}$$

KU LEUVEN

Prove R1CS with Aurora/Fractal [BCR+19, COS20]



Prove R1CS with Aurora/Fractal [BCR+19, COS20]



- Blind FRI has been shown to be practical in [ACG+24].
- Our work further demonstrates the practicality of the whole blind Fractal.

Our instantiation of blind zkSNARKs: GBFV for the large-field arithmetic

- We use the GBFV scheme [GV24] to enable native arithmetics of a large field (without emulation)
- thanks to its flexibility in balancing packing capacity and noise growth

Our instantiation of blind zkSNARKs: GBFV for the large-field arithmetic

- We use the GBFV scheme [GV24] to enable native arithmetics of a large field (without emulation)
- thanks to its flexibility in balancing packing capacity and noise growth

Our parameter

	GBFV	BFV
(n,q)	**(12288, 382 bits)**	
slot	$\mathbb{F}_{p^2}$, $p = 2^{64} - 2^{32} + 1$	
#slots	**384**	6144
(Nptct, Nctct)	**(9, 13) bits**	(69, 73) bits

Our instantiation of blind zkSNARKs: Efficient packing-friendly 2D-NTT

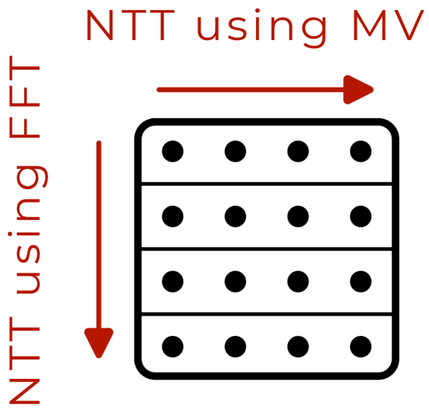
- NTT is an important component in Fractal-style zkSNARKs
  - e.g. RS-encoding: $f_z = \mathrm{NTT}\left(\mathrm{iNTT}(\hat{f}_z|_H)\right)$

Our instantiation of blind zkSNARKs: Efficient packing-friendly 2D-NTT

- NTT is an important component in Fractal-style zkSNARKs
  - e.g. RS-encoding: $f_z = \mathrm{NTT}\left(\mathrm{iNTT}(\hat{f}_z|_H)\right)$
- Different NTT algorithms
  - FFT-style algorithms give min. #(operations), but it necessitates expensive permutations within ciphertexts
  - NTT via matrix-vector multiplication is packing-friendly, but it requires more operations

Our instantiation of blind zkSNARKs: Efficient packing-friendly 2D-NTT

- NTT is an important component in Fractal-style zkSNARKs
  - e.g. RS-encoding: $f_z = \mathrm{NTT}\left(\mathrm{iNTT}(\hat{f}_z|_H)\right)$
- Different NTT algorithms
  - FFT-style algorithms give min. #(operations), but it necessitates expensive permutations within ciphertexts
  - NTT via matrix-vector multiplication is packing-friendly, but it requires more operations
- Our solution: 2D-NTT [CG99]



Our instantiation of blind zkSNARKs: other techniques and microbenchmarking

- Other techniques include
    - (sparse) matrix-vector multiplication
    - modulus switching and ring switching
    - …

Our instantiation of blind zkSNARKs: other techniques and microbenchmarking

- Other techniques include
  - (sparse) matrix-vector multiplication
  - modulus switching and ring switching
  - …
- microbenchmarking for blind Fractal with 2^{20} constraints
  - operation counts and noise estimation are publicly available
  - time estimation: 9.26 hours sequentially

    < 20min assuming 32x speedup from parallelization

Two vCOED approaches

vFHE

- [VKH23, GNS23, ABP+24, TW24, CCC+25, LLZ+25, WWX+25, ...]
- prove **ciphertext** relations

$$\mathrm{Prove}_{\mathrm{Hom.Eval}(\mathrm{F})}$$

Blind zkSNARK

- [GGW24, ACG+24, GH**K**+24, ZWL+25, ...]
- prove **plaintext** relations

$$\mathrm{Hom.Eval}(\mathrm{Prove}_{\mathrm{F}})$$

- ~ 20 mins to prove 2^{20} R1CS plaintext constraints

Two vCOED approaches

vFHE

- [VKH23, GNS23, ABP+24, TW24, CCC+25, LLZ+25, WWX+25, ...]
- prove **ciphertext** relations

$$\mathrm{Prove}_{\mathrm{Hom.Eval}(\mathrm{F})}$$

- ~ 20 mins to prove $< 2^{10}$ R1CS plaintext constraints

Blind zkSNARK

- [GGW24, ACG+24, GH**K**+24, ZWL+25, ...]
- prove **plaintext** relations

$$\mathrm{Hom.Eval}(\mathrm{Prove}_{\mathrm{F}})$$

- ~ 20 mins to prove 2^{20} R1CS plaintext constraints

Two vCOED approaches

vFHE

- [VKH23, GNS23, ABP+24, TW24, CCC+25, LLZ+25, WWX+25, ...]
- prove **ciphertext** relations

$$\mathrm{Prove}_{\mathrm{Hom.Eval}(\mathrm{F})}$$

- ~ 20 mins to prove $< 2^{10}$ R1CS plaintext constraints

Blind zkSNARK

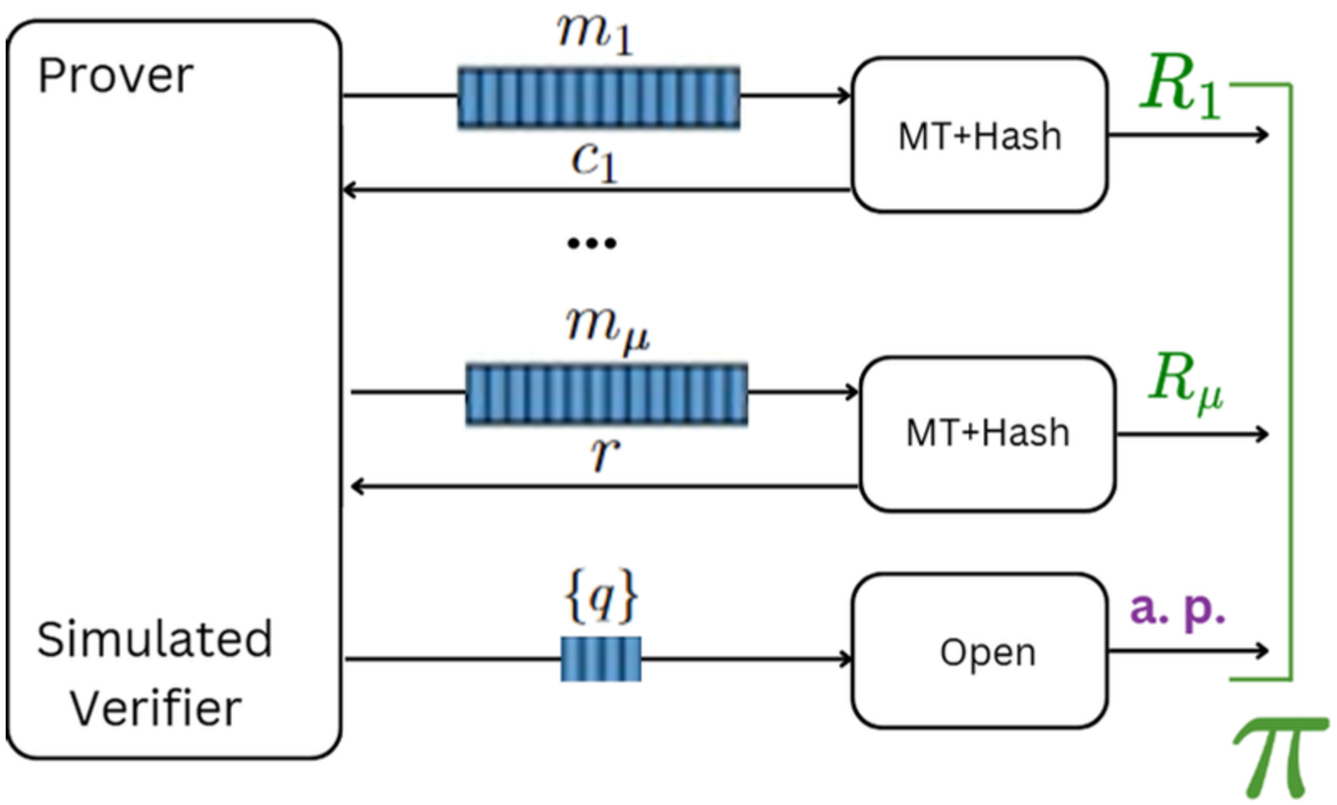
- [GGW24, ACG+24, GH**K**+24, ZWL+25, ...]
- prove **plaintext** relations

$$\mathrm{Hom.Eval}(\mathrm{Prove}_{\mathrm{F}})$$

- ~ 20 mins to prove 2^{20} R1CS plaintext constraints

What should the client do for verification?

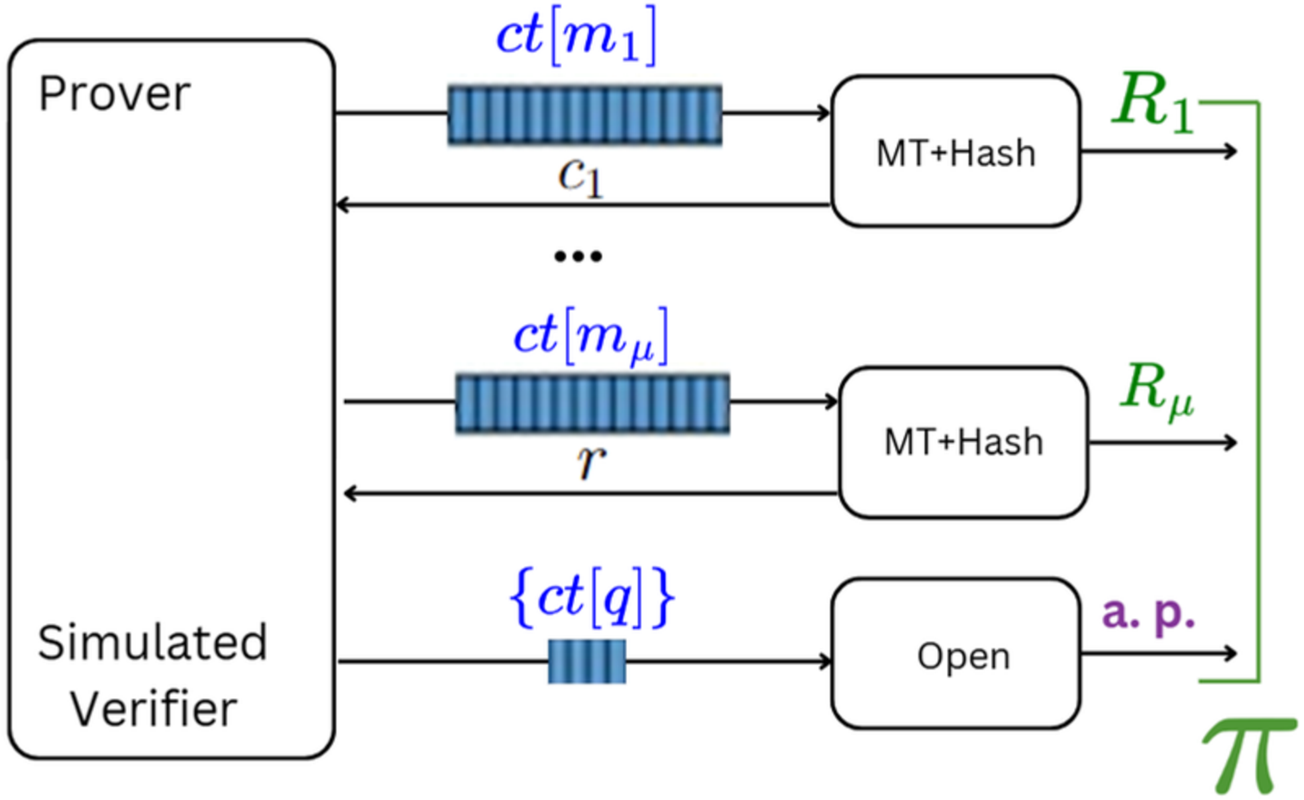
Non-interactive zero-knowledge proof [BCS16]



Anyone can Verify:

- $f(x, \{q\}, \tau) = 0$
- other consistency checks e.g. the derivation of τ

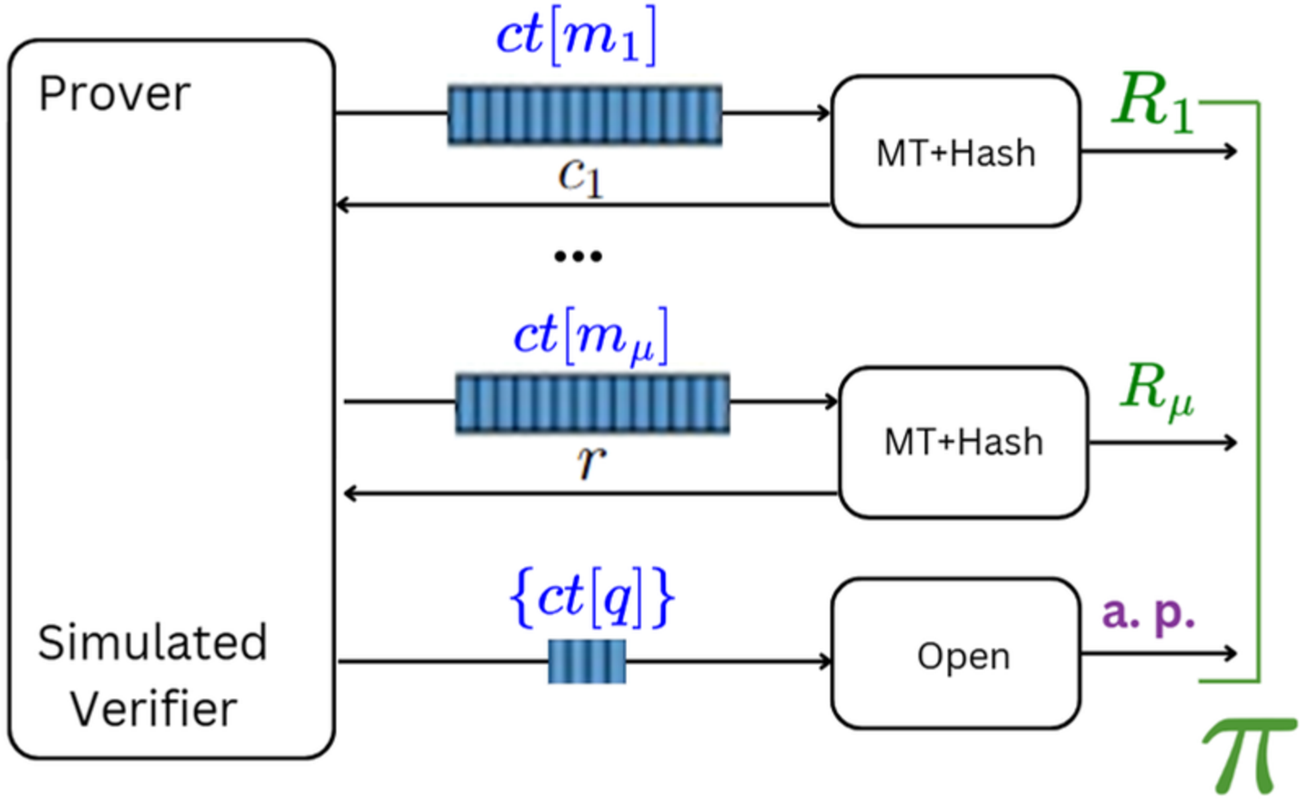
Blind zkSNARKs



Designated Verifier(DV)

- decrypt $\{ct[q]\}$
- $f(x, \{q\}, \tau) = 0$
- other consistency checks e.g. the derivation of τ

Blind zkSNARKs

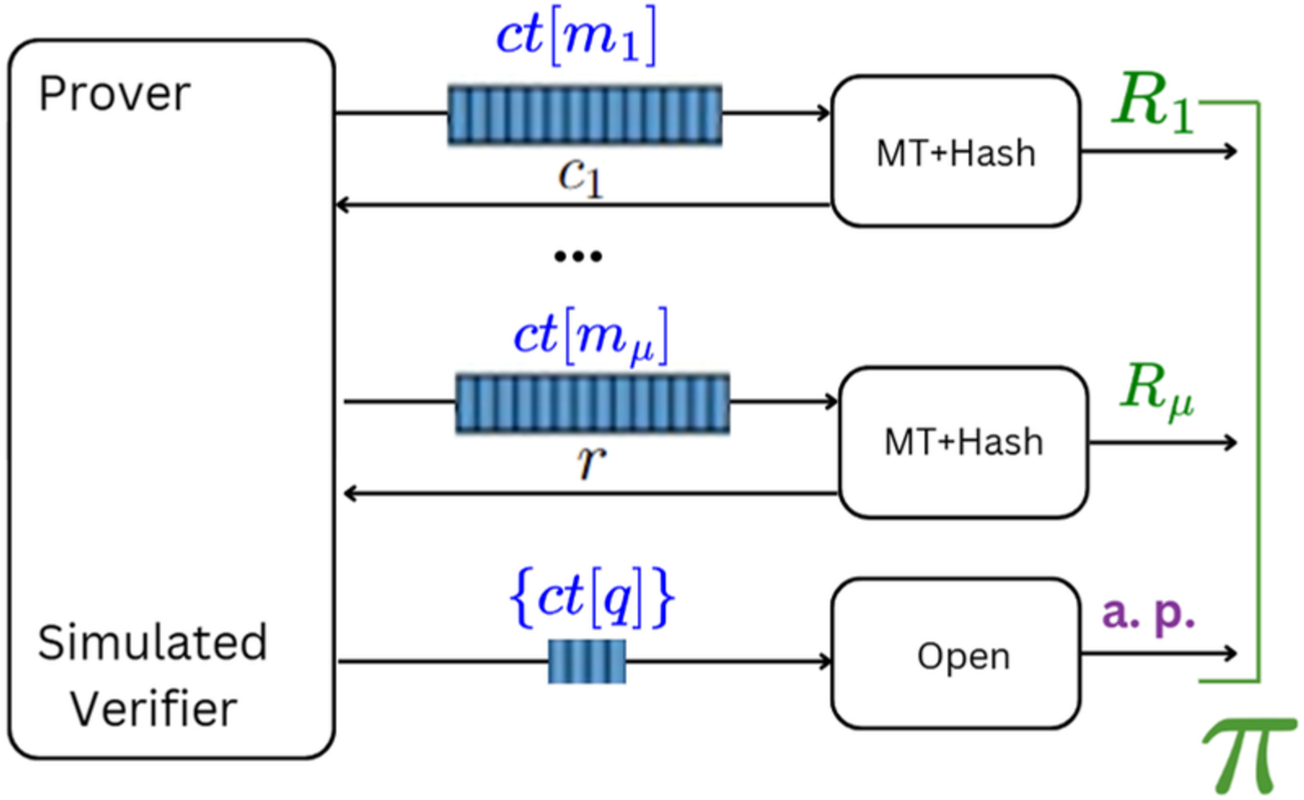


Designated Verifier(DV)

- decrypt $\{ct[q]\}$
- $f(x, \{q\}, \tau) = 0$
- other consistency checks e.g. the derivation of τ

The client (DV) revealing ver. result to the server(P) can leak 1-bit secret [ACG+24,ZWL+25]

Blind zkSNARKs



Designated Verifier(DV)

- decrypt $\{ct[q]\}$
- $f(x, \{q\}, \tau) = 0$
- other consistency checks e.g. the derivation of τ

The client (DV) revealing ver. result to the server(P) can leak 1-bit secret [ACG+24,ZWL+25]

Never re-use sk in delegation!

Two vCOED approaches

vFHE

- [VKH23, GNS23, ABP+24, TW24, CCC+25, LLZ+25, WWX+25, ...]
- prove **ciphertext** relations

$$\mathrm{Prove}_{\mathrm{Hom.Eval}(\mathrm{F})}$$

- ~ 20 mins to prove $< 2^{10}$ R1CS plaintext constraints
- can reuse sk in delegation

Blind zkSNARK

- [GGW24, ACG+24, GH**K**+24, ZWL+25, ...]
- prove **plaintext** relations

$$\mathrm{Hom.Eval}(\mathrm{Prove}_{\mathrm{F}})$$

- ~ 20 mins to prove 2^{20} R1CS plaintext constraints
- never reuse sk in delegation

Blind zkSNARKs with public verifiability

Blind zkSNARKs with public verifiability

Application I
vCOED

Client

Goal: to compute F, and convince the public audience of the correct computation

Blind zkSNARKs with public verifiability

Application I
vCOED

Client

Server

Goal: to compute F, and convince the public audience of the correct computation

$\mathrm{Hom.Eval}(F)$

$\mathrm{Hom.Eval}(\mathrm{Prove_F})$

Blind zkSNARKs with public verifiability

Application I
vCOED

Client

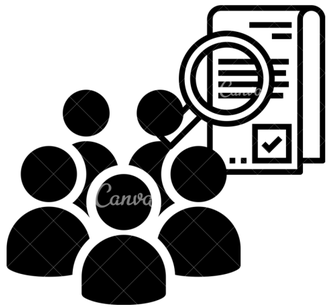
Server

Goal: to compute F, and convince the public audience of the correct computation

$\mathrm{Hom.Eval}(F)$

$\mathrm{Hom.Eval}(\mathrm{Prove_F})$

Public Verifier



Verify without knowledge of sk

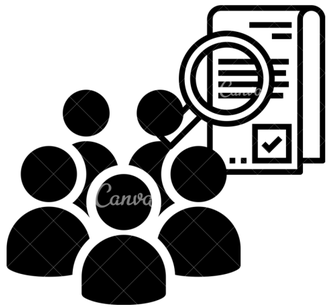
Blind zkSNARKs with public verifiability

Application II
zkDel

Client

Goal: to prove F to a public verifier

Public Verifier



Blind zkSNARKs with public verifiability

Application II
zkDel

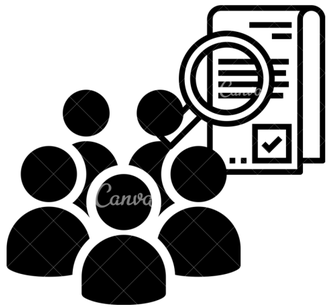
Client

Server

Goal: to prove F to a public verifier

$\mathrm{Hom.Eval}(\mathrm{Prove}_{\mathrm{F}})$

Public Verifier



Blind zkSNARKs with public verifiability

Application II
zkDel

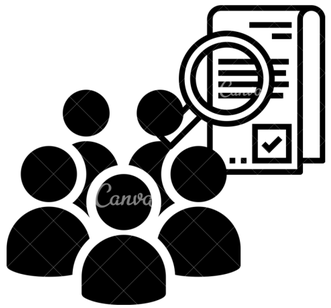
Client

Server

Goal: to prove F to a public verifier

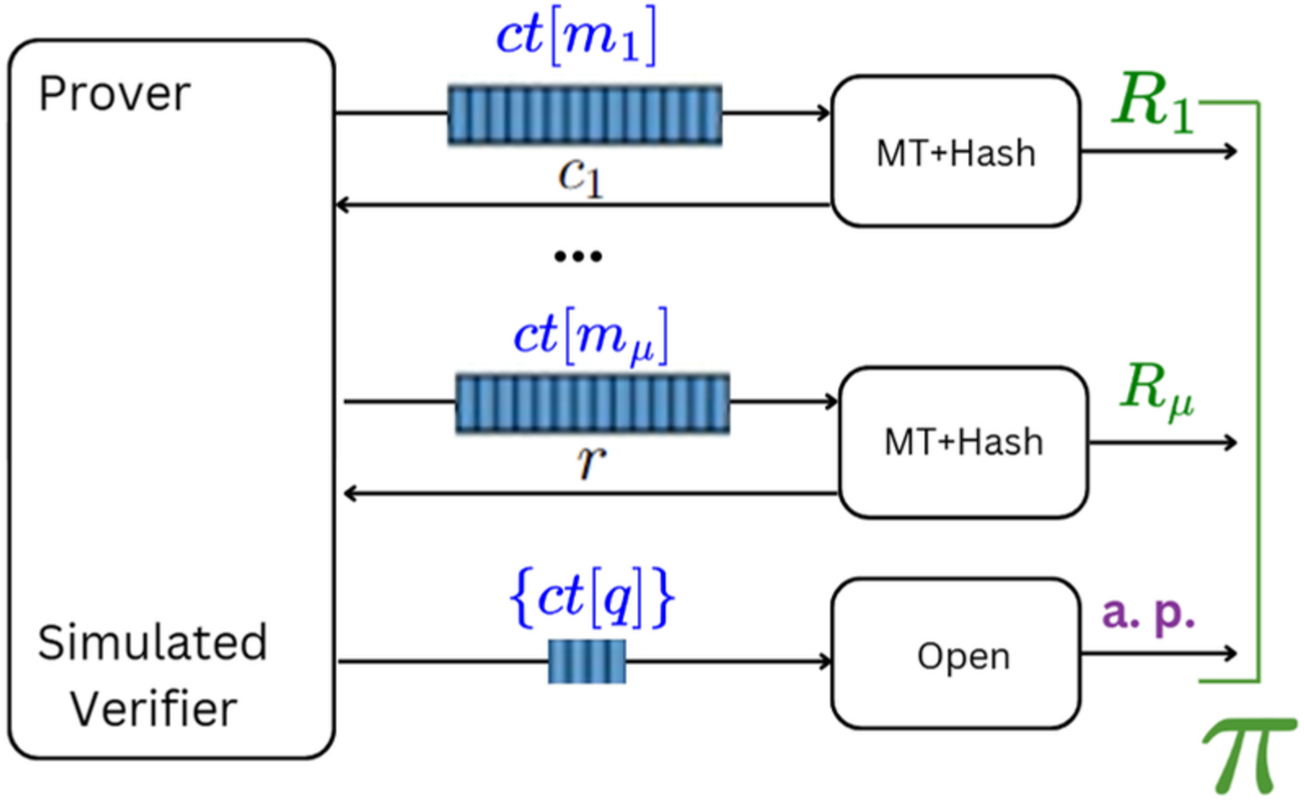
$\mathrm{Hom.Eval}(\mathrm{Prove}_{\mathrm{F}})$

Public Verifier



Verify without knowledge of sk

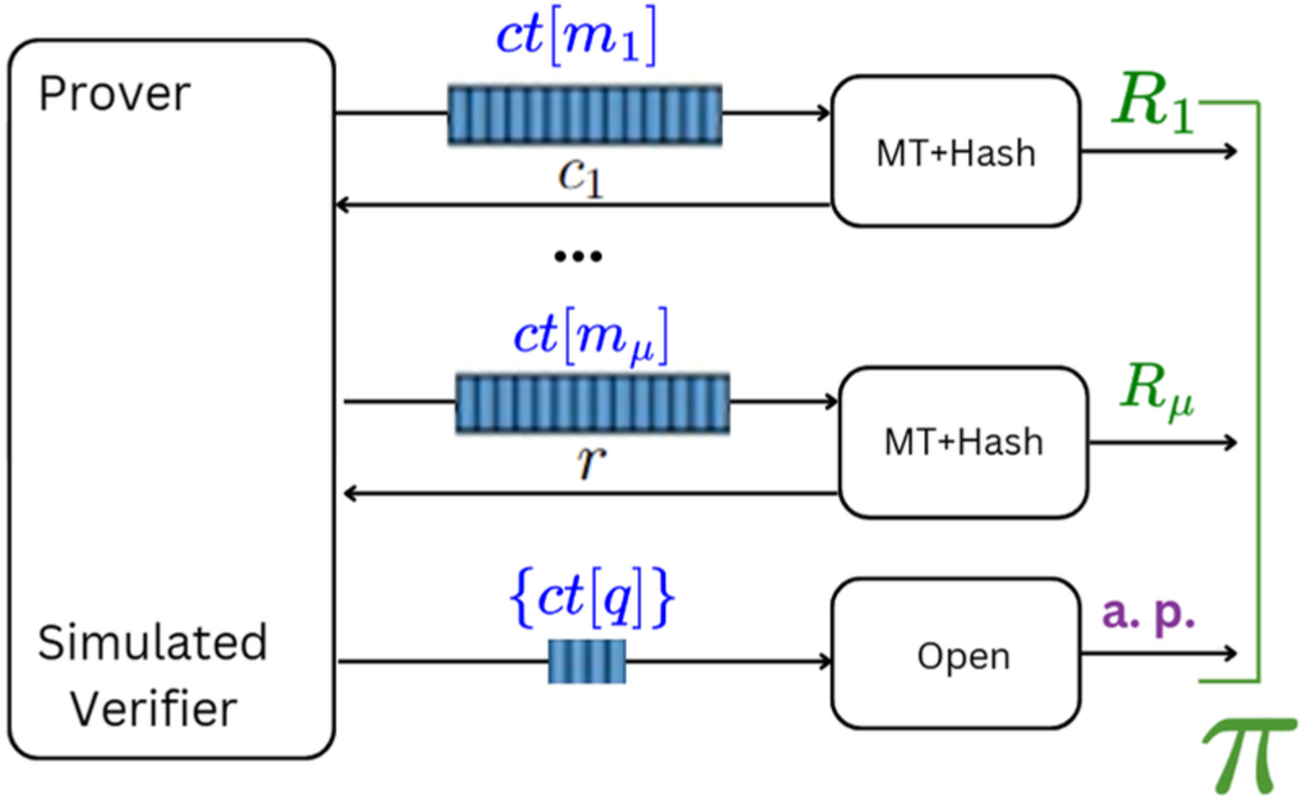
Blind zkSNARKs with public verifiability



Designated Verifier(DV)

- decrypt $\{ct[q]\}$
- $f(x, \{q\}, \tau) = 0$
- other consistency checks e.g. the derivation of τ

Blind zkSNARKs with public verifiability



Designated Verifier(DV)

- decrypt $\{ct[q]\}$
- $f(x, \{q\}, \tau) = 0$
- other consistency checks e.g. the derivation of τ

Make proof publicly verifiable

- append plaintext $\{q\}$
- append Proof of decryption (PoD) for $\{(q, ct[q])\}$

Proof of decryption (PoD)

Proof of decryption (PoD)

- Let $R_{q,n}$ denote the FHE ring with ct modulus q and degree n

$$c_0 + c_1 \cdot s = \Delta \cdot m + e \in R_{q,n}$$

Proof of decryption (PoD)

- Let $R_{q,n}$ denote the FHE ring with ct modulus q and degree n

$$c_0 + c_1 \cdot s = \Delta \cdot m + e \in R_{q,n}$$

- PoD $\iff$ the noise $e \in R_{q,n}$ is bounded

Proof of decryption (PoD)

- Let $R_{q,n}$ denote the FHE ring with ct modulus q and degree n

$$c_0 + c_1 \cdot s = \Delta \cdot m + e \in R_{q,n}$$

- PoD $\iff$ the noise $e \in R_{q,n}$ is bounded
- Our parameter: $(n, q) = (1536, 48 \text{ bits})$

Proof of decryption (PoD)

- Let $R_{q,n}$ denote the FHE ring with ct modulus q and degree n

$$c_0 + c_1 \cdot s = \Delta \cdot m + e \in R_{q,n}$$

- PoD $\iff$ the noise $e \in R_{q,n}$ is bounded
- Our parameter: $(n, q) = (1536, 48 \text{ bits})$

[LNP22] proof system

- Commitment ring is $R_{q',n'}$ where
  - n' is a power-of-two (typically 64 or 128)
  - q' is chosen s.t. $X^{n'} + 1$ has two irred. fact. mod q'

Proof of decryption: our techniques

- ModSwitch$_{q \to q'}$ to resolve the discrepancy between q and q'

Proof of decryption: our techniques

- $\mathsf{ModSwitch}_{q \to q'}$ to resolve the discrepancy between q and q'
- Commit to coefficient vectors
  - $\overrightarrow{e} = \mathsf{Rot}(c_1) \cdot \overrightarrow{s} + \overrightarrow{c_0} - \overrightarrow{\Delta \cdot m} \in \mathbb{Z}_q^n$
  - extend [LNP22] to the vectorized description

Proof of decryption: our techniques

- ModSwitch$_{q \to q'}$ to resolve the discrepancy between q and q'
- Commit to coefficient vectors
  - $\overrightarrow{e} = \mathsf{Rot}(c_1) \cdot \overrightarrow{s} + \overrightarrow{c_0} - \overrightarrow{\Delta \cdot m} \in \mathbb{Z}_q^n$
  - extend [LNP22] to the vectorized description
- Schwartz-Zippel Lemma to batch multiple PoDs

Proof of decryption: our techniques

- ModSwitch$_{q \to q'}$ to resolve the discrepancy between q and q'
- Commit to coefficient vectors
  - $\overrightarrow{e} = \mathsf{Rot}(c_1) \cdot \overrightarrow{s} + \overrightarrow{c_0} - \overrightarrow{\Delta \cdot m} \in \mathbb{Z}_q^n$
  - extend [LNP22] to the vectorized description
- Schwartz-Zippel Lemma to batch multiple PoDs

Performance

- Implementation of vec-LNP is publicly available
- ~2 seconds to prove the decryption of ~2500 cts (single thread)

vFHE

- [VKH23, GNS23, ABP+24, TW24, CCC+25, LLZ+25, WWX+25, ...]
- prove **ciphertext** relations

$$\mathrm{Prove}_{\mathrm{Hom.Eval(F)}}$$

- ~ 20 mins to prove $< 2^{10}$ R1CS plaintext constraints
- can reuse sk in delegation

Blind zkSNARK

- [GGW24, ACG+24, GH**K**+24, ZWL+25, ...]
- prove **plaintext** relations

$$\mathrm{Hom.Eval}(\mathrm{Prove}_{\mathrm{F}})$$

- ~ 20 mins to prove 2^{20} R1CS plaintext constraints
- never reuse sk in delegation

vFHE

- [VKH23, GNS23, ABP+24, TW24, CCC+25, LLZ+25, WWX+25, ...]
- prove **ciphertext** relations

$$\mathrm{Prove}_{\mathrm{Hom.Eval(F)}}$$

- ~ 20 mins to prove $< 2^{10}$ R1CS plaintext constraints
- can reuse sk in delegation

Blind zkSNARK

- [GGW24, ACG+24, GH**K**+24, ZWL+25, ...]
- prove **plaintext** relations

$$\mathrm{Hom.Eval}(\mathrm{Prove}_{\mathrm{F}})$$

- ~ 20 mins to prove 2^{20} R1CS plaintext constraints
- never reuse sk in delegation

Thanks for listening!

ia.cr/2024/1684