Line Point Zero Knowledge and its Improvement

ITC 2021 & CCS 2022 Submission

Samuel Dittmer and Yuval Ishai and Rafail Ostrovsky

May 4, 2022· Presented by Hongrui Cui

* Some acknowledgments?

Introduction

- Landscape of Efficient Zero Knowledge

	zk-SNARK, GKR, etc.	GCZK	LPZK
Prover Computation	$\Omega(\lvert C\rvert \log(\lvert C\rvert))$	$O(\lvert C\rvert)$	$O(\lvert C\rvert)$
Prover Memory	$\Omega(\lvert C\rvert)$	$O(1)$	$O(1)$
Proof Size	$O(\log(\lvert C\rvert))$	$O(\kappa \cdot \lvert C\rvert)$	$O(\lvert C\rvert)$
Verifier Type	Universal	Designated	Designated
Advantage	Low-Bandwidth Small Circuit	High-Bandwidth Large Circuit	High-Bandwidth Large Circuit

Main techniques:

- Random (subfield) VOLE
- Low-Degree Test

Benchmark on 1024×1024 Matrix Multiplication

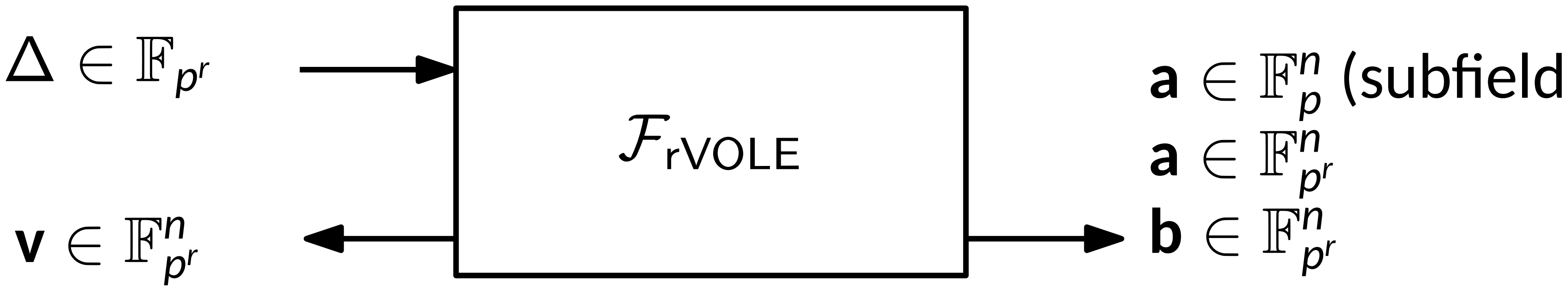
Protocol	Execution Time	Communication
Spartan [Set20]	≥ 5000 s	≤ 100 KB
Virgo [ZXZS20]	357 s	221 KB
Wolverine [WYKW21]	1627 s	34 GB
Mac′n′Cheese [BMRS21a]	2684 s	25.8 GB
QuickSilver (Circuit)	316 s	8.6 GB
QuickSilver (Polynomial)	10 s	25.2 MB

Table 5: **Performance of proving matrix multiplication using various protocols.** All numbers are based on proving knowledge of two 1024×1024 matrices over a 61-bit field, whose product is a public matrix. The execution time for Wolverine and Mac′n′Cheese is based on local-host, while our protocols and Virgo are based on a 500 Mbps network. Spartan consumed 600 GB memory before crash, and thus we extrapolate the execution time based on a smaller proving instance. Our protocols use just 1 GB of memory, but Virgo needs 148 GB of memory.

Random VOLE

- (subfield) Vector Oblivious Linear Evaluation

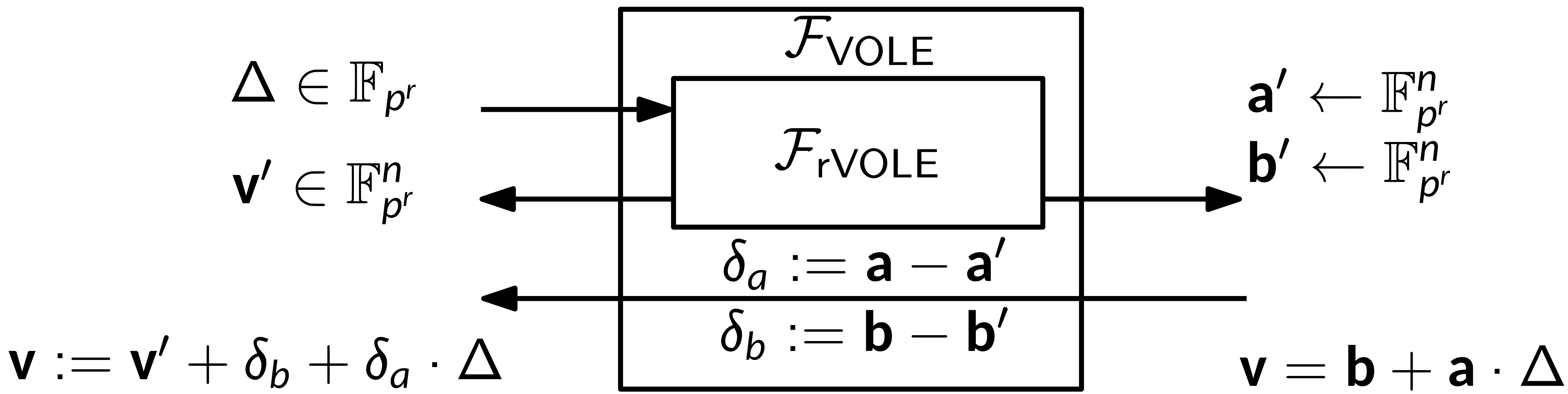
Receiver/Verifier — Sender/Prover



Constraint: $\mathbf{v} = \mathbf{b} + \mathbf{a} \cdot \Delta$

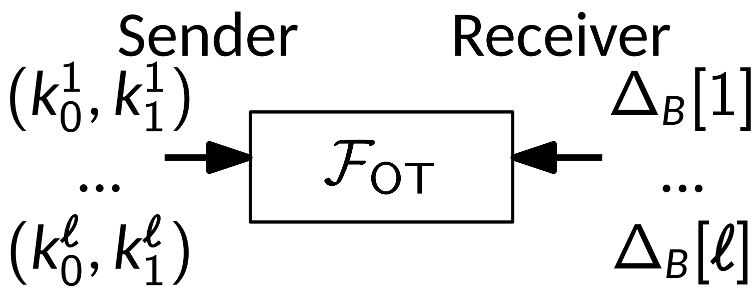
- Random VOLE $\rightarrow$ Chosen Input VOLE

Receiver/Verifier — Sender/Prover



Random VOLE

- State-of-the-art Random-VOLE Construction based on LPN*



Random VOLE

- State-of-the-art Random-VOLE Construction based on LPN*

Sender Receiver

(k_0^1, k_1^1) $\Delta_B[1]$

... $\rightarrow \boxed{\mathcal{F}_{\mathsf{OT}}} \leftarrow$ ...

(k_0^ℓ, k_1^ℓ) $\Delta_B[\ell]$

Extend (u)

$$m_1 := \mathsf{PRF}(k_0^1, j) + \mathsf{PRF}(k_1^1, j) + u$$

...

$$m_\ell := \mathsf{PRF}(k_0^\ell, j) + \mathsf{PRF}(k_1^\ell, j) + u$$

$\longrightarrow$

$$u\Delta_B[i] = \underbrace{\mathsf{PRF}(k_0^i, j)}_{\text{Sender}} + \underbrace{\mathsf{PRF}(k^i_{\Delta_B[i]}, j) + m_i\Delta_B[i]}_{\text{Receiver}}$$

Use LHL to remove selective failure leackage on Δ

Random VOLE

- State-of-the-art Random-VOLE Construction based on LPN*

Sender Receiver

$$(k_0^1, k_1^1) \rightarrow \boxed{\mathcal{F}_{\mathrm{OT}}} \leftarrow \Delta_B[1]$$

$$\dots \qquad\qquad \dots$$

$$(k_0^\ell, k_1^\ell) \qquad\qquad \Delta_B[\ell]$$

Extend (u)

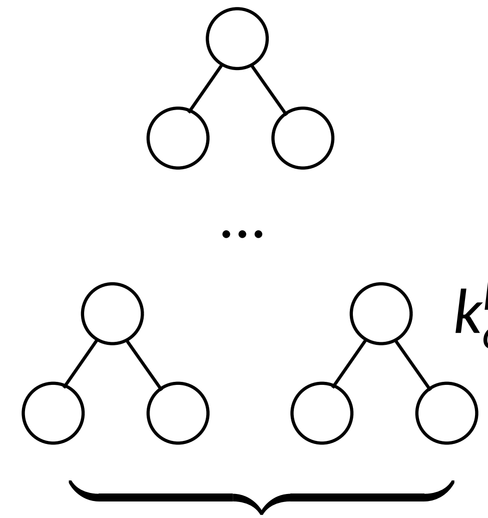
$$m_1 := \mathrm{PRF}(k_0^1, j) + \mathrm{PRF}(k_1^1, j) + u$$

$$\dots$$

$$m_\ell := \mathrm{PRF}(k_0^\ell, j) + \mathrm{PRF}(k_1^\ell, j) + u$$

$$u\Delta_B[i] = \underbrace{\mathrm{PRF}(k_0^i, j)}_{\text{Sender}} + \underbrace{\mathrm{PRF}(k_{\Delta_B[i]}^i, j) + m_i\Delta_B[i]}_{\text{Receiver}}$$

Receiver



$\dots$

$\mathbf{v}$

Sender

$$e \in \mathbb{F}_p^{2^h} \text{ s.t. } \begin{cases} e_\alpha = u \\ e_{\bar{\alpha}} = 0 \end{cases}$$

$$k_o^1, k_e^1$$

$$k_o^{h-1}, k_e^{h-1} \quad \boxed{\mathcal{F}_{\mathrm{OT}}} \quad \alpha_1, \dots, \alpha_h$$

$$k_o^h, k_e^h \qquad m = \sum \mathbf{v}_i + v_u \longrightarrow$$

Use LHL to remove selective failure leackage on Δ

Random VOLE

- State-of-the-art Random-VOLE Construction based on LPN*

Sender | Receiver

$$(k_0^1, k_1^1) \;\dots\; (k_0^\ell, k_1^\ell) \longrightarrow \boxed{\mathcal{F}_{\mathrm{OT}}} \longleftarrow \Delta_B[1] \;\dots\; \Delta_B[\ell]$$

Extend (u)

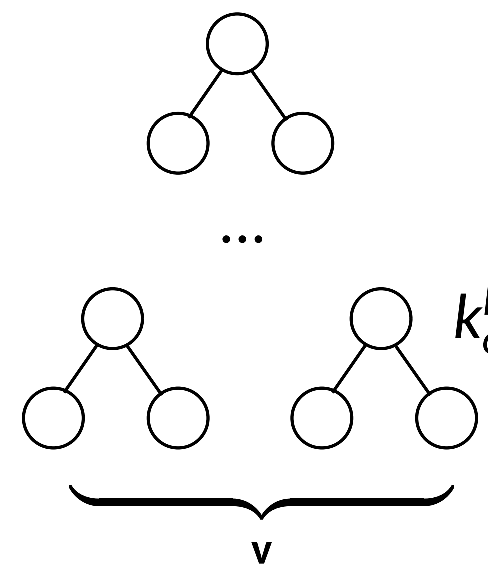
$$m_1 := \mathrm{PRF}(k_0^1, j) + \mathrm{PRF}(k_1^1, j) + u$$

$$\dots$$

$$m_\ell := \mathrm{PRF}(k_0^\ell, j) + \mathrm{PRF}(k_1^\ell, j) + u \longrightarrow$$

$$u\Delta_B[i] = \underbrace{\mathrm{PRF}(k_0^i, j)}_{\text{Sender}} + \underbrace{\mathrm{PRF}(k^i_{\Delta_B[i]}, j) + m_i\Delta_B[i]}_{\text{Receiver}}$$

Receiver



Sender

$$e \in \mathbb{F}_p^{2^h} \text{ s.t. } \begin{cases} e_\alpha = u \\ e_{\bar{\alpha}} = 0 \end{cases}$$

k_o^1, k_e^1

k_o^{h-1}, k_e^{h-1}

k_o^h, k_e^h

$$\boxed{\mathcal{F}_{\mathrm{OT}}} \; \alpha_1, \dots, \alpha_h$$

$$m = \sum \mathbf{v}_i + v_u \longrightarrow$$

Recover $\mathbf{b}_i, i \neq \alpha$

Recover $\mathbf{b}_\alpha$

$$\boxed{\text{Use LHL to remove selective failure leackage on } \Delta}$$

Random VOLE

- State-of-the-art Random-VOLE Construction based on LPN*

Sender — Receiver

(k_0^1, k_1^1) ... (k_0^ℓ, k_1^ℓ) $\rightarrow \boxed{\mathcal{F}_{\mathrm{OT}}} \leftarrow$ $\Delta_B[1]$ ... $\Delta_B[\ell]$

Extend (u)

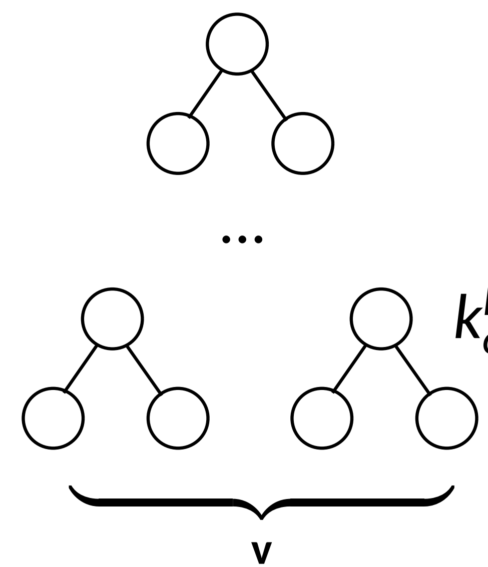
$$m_1 := \mathsf{PRF}(k_0^1, j) + \mathsf{PRF}(k_1^1, j) + u$$

...

$$m_\ell := \mathsf{PRF}(k_0^\ell, j) + \mathsf{PRF}(k_1^\ell, j) + u$$

$$u\Delta_B[i] = \underbrace{\mathsf{PRF}(k_0^i, j)}_{\text{Sender}} + \underbrace{\mathsf{PRF}(k^i_{\Delta_B[i]}, j) + m_i\Delta_B[i]}_{\text{Receiver}}$$

Receiver



k_o^1, k_e^1

k_o^{h-1}, k_e^{h-1}

k_o^h, k_e^h

Sender

$$e \in \mathbb{F}_p^{2^h} \text{ s.t. } \begin{cases} e_\alpha = u \\ e_{\bar{\alpha}} = 0 \end{cases}$$

$\boxed{\mathcal{F}_{\mathrm{OT}}}$ $\alpha_1, \ldots, \alpha_h$

$$m = \sum \mathbf{v}_i + v_u$$

Recover $\mathbf{b}_i, i \neq \alpha$

Recover $\mathbf{b}_\alpha$

- Use Multiple $\mathcal{F}_{\mathrm{spVOLE}}$ to get sparse $\mathbf{e}$
- Use LPN* to expand to pseudorandom $\mathbf{u}$

Use LHL to remove selective failure leackage on Δ

$\mathcal{F}_{\mathrm{OT}}$ implies selective failure on α (i.e., LPN noise)

VOLE as IT-MAC

Receiver/Verifier | Sender/Prover

$\Delta \in \mathbb{F}_{p^r}$ $\longrightarrow$ $\boxed{\mathcal{F}_{\mathrm{rVOLE}}}$ $\longrightarrow$ $\mathbf{a} \in \mathbb{F}_{p^r}^n$, $\mathbf{b} \in \mathbb{F}_{p^r}^n$

$\mathbf{v} \in \mathbb{F}_{p^r}^n$ $\longleftarrow$

Sender commits to $\mathbf{x}$ by sending $\delta_x := \mathbf{x} - \mathbf{a}$ $\longleftarrow$

IT-MAC $[\mathbf{x}] := (\mathbf{x}, \mathbf{v}, \mathbf{b})$ subject to $\mathbf{v} = \mathbf{b} + \mathbf{x} \cdot \Delta$

- Linear Homomorphism: $[\mathbf{x}] + [\mathbf{y}] \mapsto [\mathbf{x} + \mathbf{y}]$
- Open($[x]$): $P \to V : (x, b)$, V checks $v = b + x \cdot \Delta$
- Batched Open:

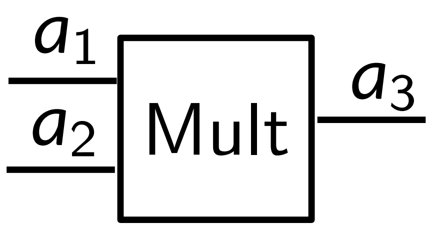
Receiver/Verifier | Sender/Prover

$\xrightarrow{\{\chi_i\}}$

$\xleftarrow{\mathbf{x},\ \mathrm{Open}([\sum_i \chi_i \cdot \mathbf{x}_i])}$

Verifying Multiplication Relation

- Checking $a_1 \times a_2 = a_3$ for $a_1, a_2, a_3 \in \mathbb{F}_p$



Prover samples $b_1, b_2, b_3, b_4 \leftarrow \mathbb{F}_p$,

$a_4 = a_1 b_2 + a_2 b_1 - b_3$

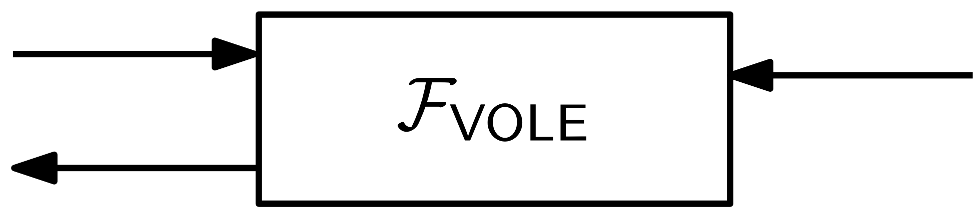
$(a_5, b_5) = (0, b_1 b_2 - b_4)$

Receiver/Verifier

$\Delta \in \mathbb{F}_p$

$\mathbf{v} \in \mathbb{F}_p^5$

Point



Sender/Prover

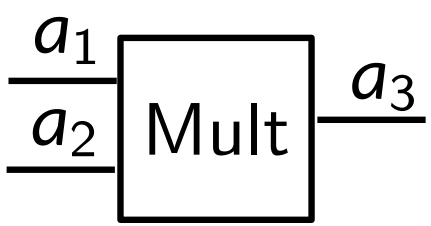
$\mathbf{a} = (a_1, a_2, a_3, a_4, a_5)$

$\mathbf{b} = (b_1, b_2, b_3, b_4, b_5)$

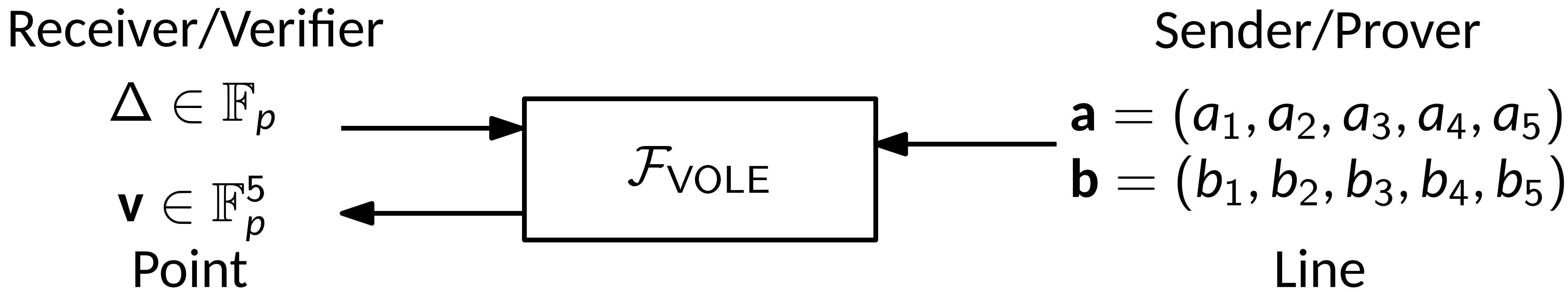
Line

Verifying Multiplication Relation

- Checking $a_1 \times a_2 = a_3$ for $a_1, a_2, a_3 \in \mathbb{F}_p$



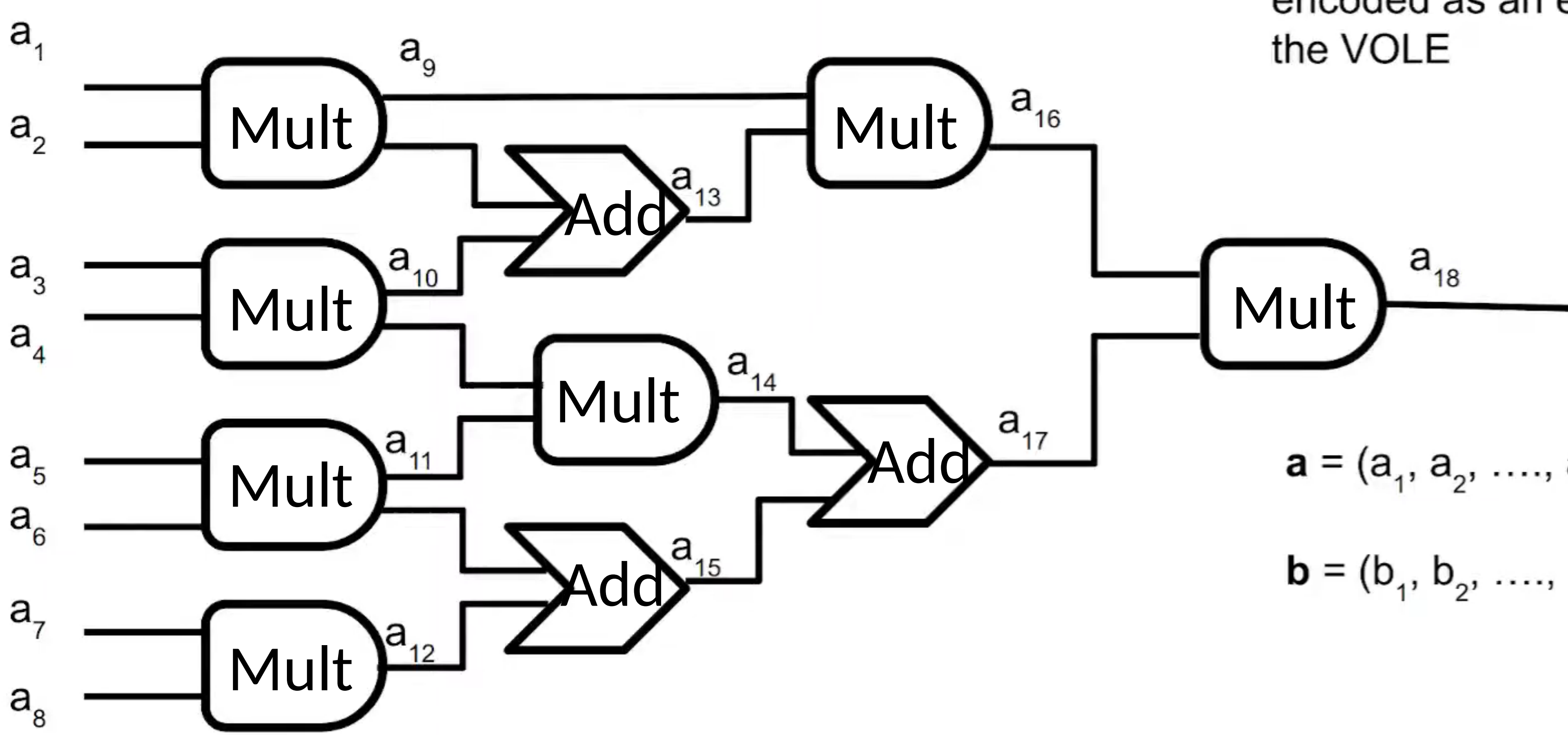
Prover samples $b_1, b_2, b_3, b_4 \leftarrow \mathbb{F}_p$,
$a_4 = a_1 b_2 + a_2 b_1 - b_3$
$(a_5, b_5) = (0, b_1 b_2 - b_4)$



- Completeness: Verifier checks $v(\Delta) = v_1 v_2 - v_3 \Delta - v_4 = v_5$
- Soundness error: $\frac{2}{p}$

  $v(\Delta) = (a_1 a_2 - a_3)\Delta^2 + (a_1 b_2 + a_2 b_1 - b_3 - a_4 - a_5)\Delta + (b_1 b_2 - b_4 - b_5)$
- Zero-Knowledge: $v_1, v_2, v_3, v_4 \leftarrow \mathbb{F}_p$, $v_5 := v_1 v_2 - v_3 \Delta - v_4$

IT-LPZK for Circuit Satisfiablity



Each wire in the circuit is encoded as an entry of the VOLE

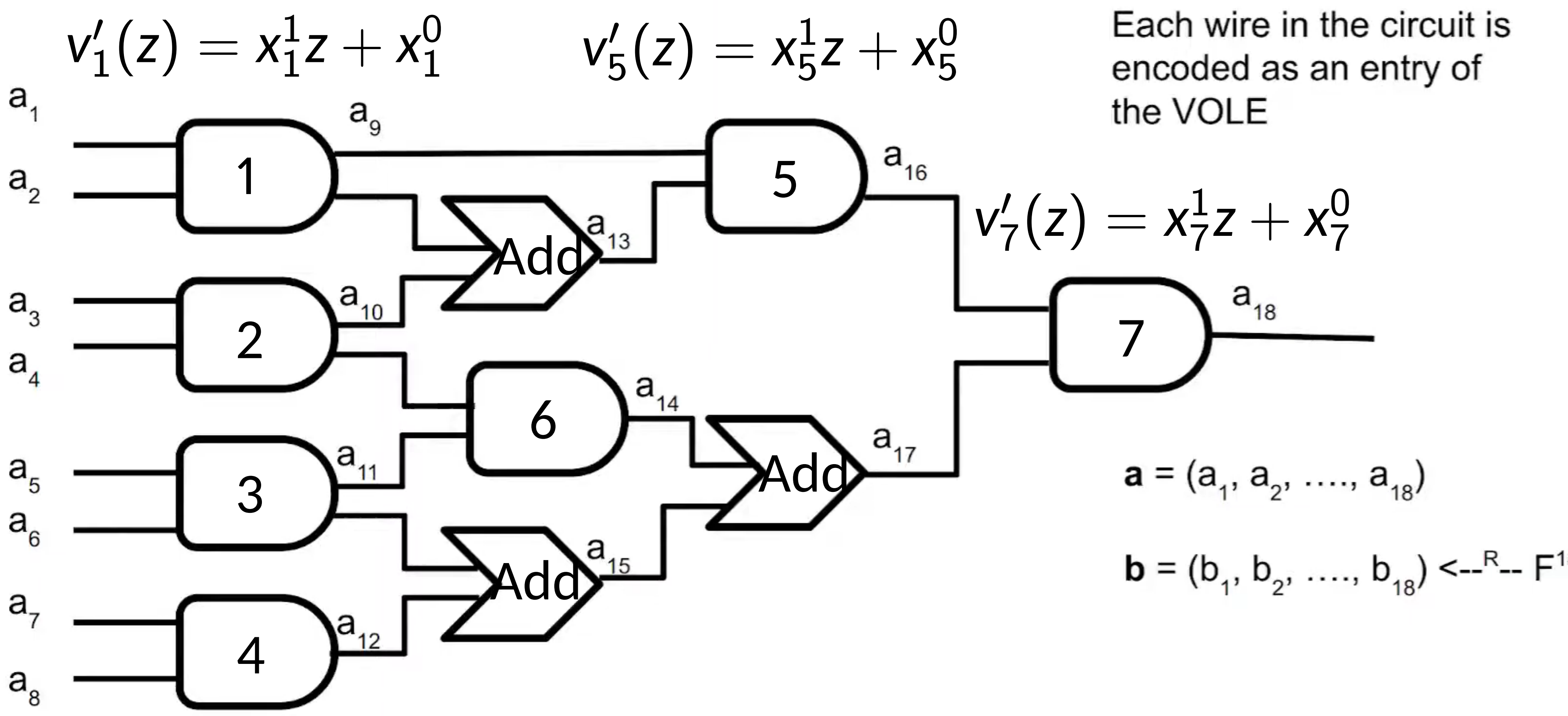
$\mathbf{a} = (a_1, a_2, \ldots, a_{18})$

$\mathbf{b} = (b_1, b_2, \ldots, b_{18}) \xleftarrow{R} F^{18}$

- For each (i, j, k), proves $v_i v_j - v_k$ is a linear function in Δ
- Prover prepares $a' := a_i b_j + a_j b_i - b_k$ for each AND gate
- Batch t gates by sending $\prod(b_i b_j - b')$

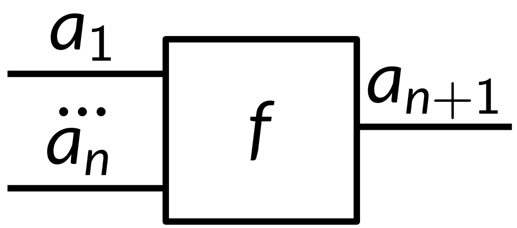
ROM-LPZK for Circuit Satisfiability

- Fiat-Shamir collapses interaction and fits LPZK model



- Prover sends $x_1 := \sum \chi_i x_i^1 + a_{19}$, $x_0 := \sum \chi_i x_i^0 + b_{19}$
- Verifier checks $x_1 \Delta + x_0 = \sum \chi_i v'_i(\Delta) + v_{19}$

ROM-LPZK for Polynomial Satisfiability

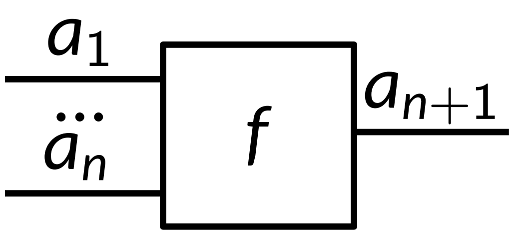


$$\mathbf{v} = \mathbf{a}\Delta + \mathbf{b}$$

$$f(\mathbf{a}) = f_d(\mathbf{a}) + f_{d-1}(\mathbf{a})... + f_0$$

$$\begin{aligned} f(\mathbf{v}) &= f_d(\mathbf{v}) + f_{d-1}(\mathbf{v}) + ... + f_0 \\ &= f_d(\mathbf{a})\Delta^d + f_{d-1}(\mathbf{a})\Delta^{d-1} + ... + f_0 + f_r(\mathbf{a}, \mathbf{b}) \end{aligned}$$

ROM-LPZK for Polynomial Satisfiability



$$\mathbf{v} = \mathbf{a}\Delta + \mathbf{b}$$

$$f(\mathbf{a}) = f_d(\mathbf{a}) + f_{d-1}(\mathbf{a}) ... + f_0$$

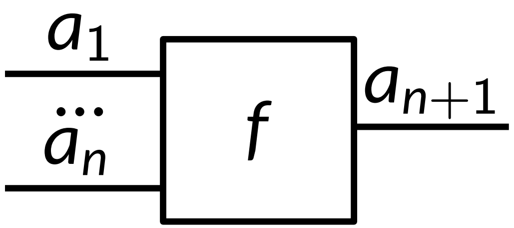
$$f(\mathbf{v}) = f_d(\mathbf{v}) + f_{d-1}(\mathbf{v}) + ... + f_0$$

$$= f_d(\mathbf{a})\Delta^d + f_{d-1}(\mathbf{a})\Delta^{d-1} + ... + f_0 + f_r(\mathbf{a}, \mathbf{b})$$

$$g(\mathbf{v}) := f_d(\mathbf{v}) + \Delta f_{d-1}(\mathbf{v}) + ... + \Delta^{d-1} f_1(\mathbf{v}) + \Delta^d f_0 - \Delta^{d-1} v_{n+1}$$

$$= (f_d(\mathbf{a}) + ... + f_0 - a_{n+1})\Delta^d + \underbrace{f'_{r,\mathbf{a},\mathbf{b}}(\Delta)}_{\deg(\Delta) < d}$$

ROM-LPZK for Polynomial Satisfiability



$$\mathbf{v} = \mathbf{a}\Delta + \mathbf{b}$$

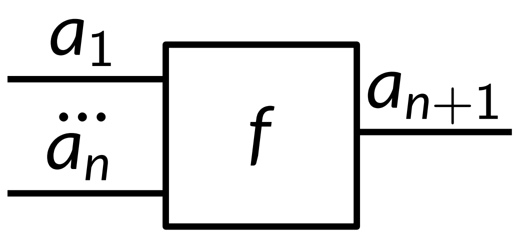
$$\begin{aligned} f(\mathbf{a}) &= f_d(\mathbf{a}) + f_{d-1}(\mathbf{a})... + f_0 \\ f(\mathbf{v}) &= f_d(\mathbf{v}) + f_{d-1}(\mathbf{v}) + ... + f_0 \\ &= f_d(\mathbf{a})\Delta^d + f_{d-1}(\mathbf{a})\Delta^{d-1} + ... + f_0 + f_r(\mathbf{a}, \mathbf{b}) \end{aligned}$$

$$\begin{aligned} g(\mathbf{v}) &:= f_d(\mathbf{v}) + \Delta f_{d-1}(\mathbf{v}) + ... + \Delta^{d-1} f_1(\mathbf{v}) + \Delta^d f_0 - \Delta^{d-1} v_{n+1} \\ &= (f_d(\mathbf{a}) + ... + f_0 - a_{n+1})\Delta^d + \underbrace{f'_{r,\mathbf{a},\mathbf{b}}(\Delta)}_{\deg(\Delta) < d} \end{aligned}$$

Π_{gen}^{d-1}

$$\left.\begin{aligned} v_1 &= a_1\Delta + b_1 \\ v_2\Delta &= a_2\Delta^2 + b_2\Delta \\ &\vdots \\ v_{d-1}\Delta^{d-2} &= a_{d-1}\Delta^{d-1} + b_{d-1}\Delta^{d-2} \end{aligned}\right\} \sum \Rightarrow g^*(\Delta)$$

ROM-LPZK for Polynomial Satisfiability



$$\mathbf{v} = \mathbf{a}\Delta + \mathbf{b}$$

$$\begin{aligned} f(\mathbf{a}) &= f_d(\mathbf{a}) + f_{d-1}(\mathbf{a}) ... + f_0 \\ f(\mathbf{v}) &= f_d(\mathbf{v}) + f_{d-1}(\mathbf{v}) + ... + f_0 \\ &= f_d(\mathbf{a})\Delta^d + f_{d-1}(\mathbf{a})\Delta^{d-1} + ... + f_0 + f_r(\mathbf{a}, \mathbf{b}) \end{aligned}$$

$$\begin{aligned} g(\mathbf{v}) &:= f_d(\mathbf{v}) + \Delta f_{d-1}(\mathbf{v}) + ... + \Delta^{d-1} f_1(\mathbf{v}) + \Delta^d f_0 - \Delta^{d-1} v_{n+1} \\ &= (f_d(\mathbf{a}) + ... + f_0 - a_{n+1})\Delta^d + \underbrace{f'_{r,\mathbf{a},\mathbf{b}}(\Delta)}_{\deg(\Delta) < d} \end{aligned}$$

Π^{d-1}_{gen}

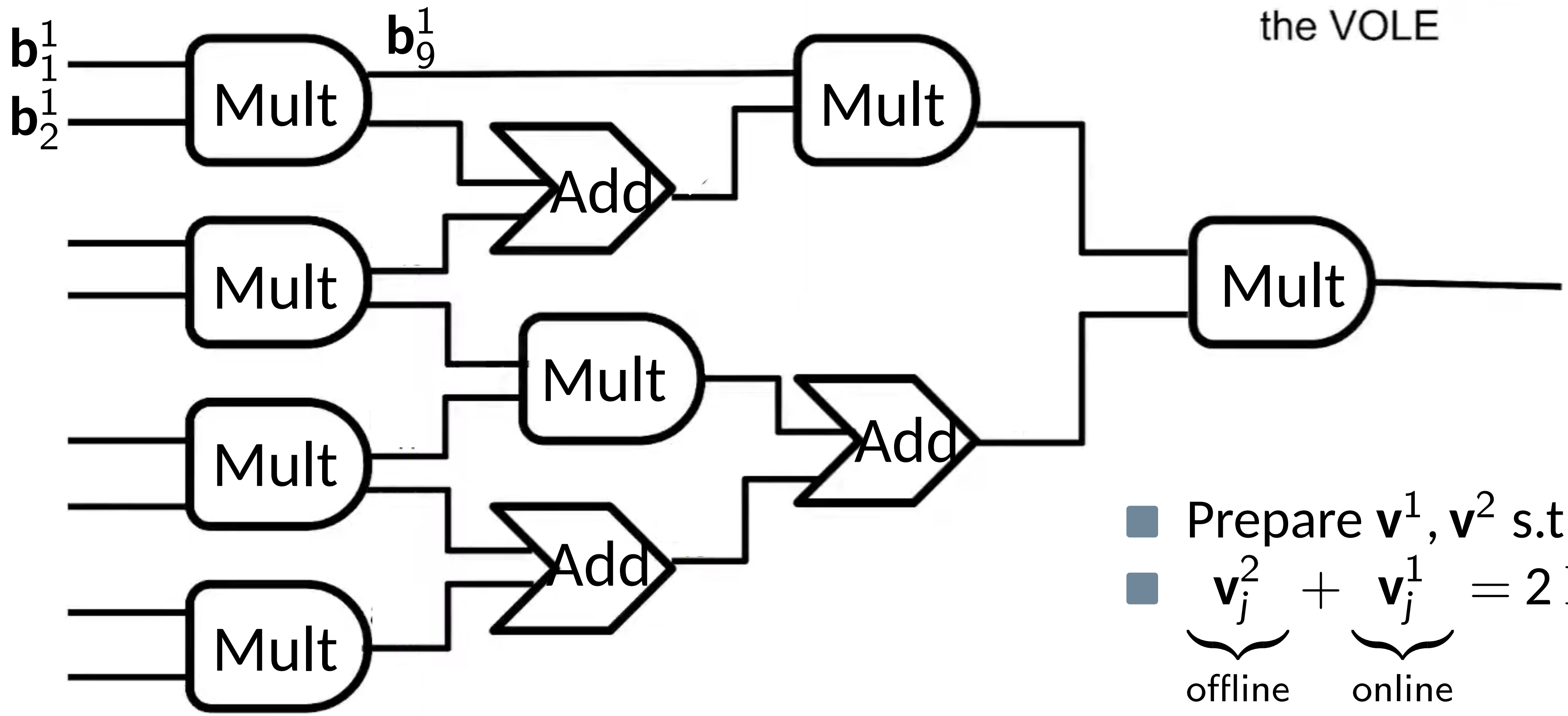
$$\left.\begin{aligned} v_1 &= a_1\Delta + b_1 \\ v_2\Delta &= a_2\Delta^2 + b_2\Delta \\ &\vdots \\ v_{d-1}\Delta^{d-2} &= a_{d-1}\Delta^{d-1} + b_{d-1}\Delta^{d-2} \end{aligned}\right\} \sum \Rightarrow g^*(\Delta)$$

- Sends collapsed, masked coeff. of $g(\mathbf{v})$
- Soundness: $\frac{d}{p} + \varepsilon_{RO}$

IT-LPZKv2 Optimizations

- Use $\mathbf{b}$ to encode wire values ($\mathbf{v} = \mathbf{a}\Delta + \mathbf{b}$)
- Preprocess quadratic terms in $\mathbf{a}$

Each wire in the circuit is encoded as an entry of the VOLE



- Prepare $\mathbf{v}^1, \mathbf{v}^2$ s.t. $\mathbf{a}_9^2 = \mathbf{a}_1^1\mathbf{a}_2^1$
- $\underbrace{\mathbf{v}_j^2}_{\text{offline}} + \underbrace{\mathbf{v}_j^1}_{\text{online}} = 2\ \mathbb{F}_p$ element per gate j

- Verify for zero-constant

$$\mathbf{v}_9^1 - \mathbf{v}_1^1\mathbf{v}_2^1 - \Delta\mathbf{v}_9^2 = (\mathbf{a}_1^1\mathbf{a}_2^1 - \mathbf{a}_9^2)\Delta^2 + (\mathbf{a}_9^1 + \mathbf{a}_1^1\mathbf{b}_2^1 + \mathbf{b}_1^1\mathbf{a}_2^1 + \mathbf{b}_9^2)\Delta + (\mathbf{b}_1^1\mathbf{b}_2^1 - \mathbf{b}_9^1)$$

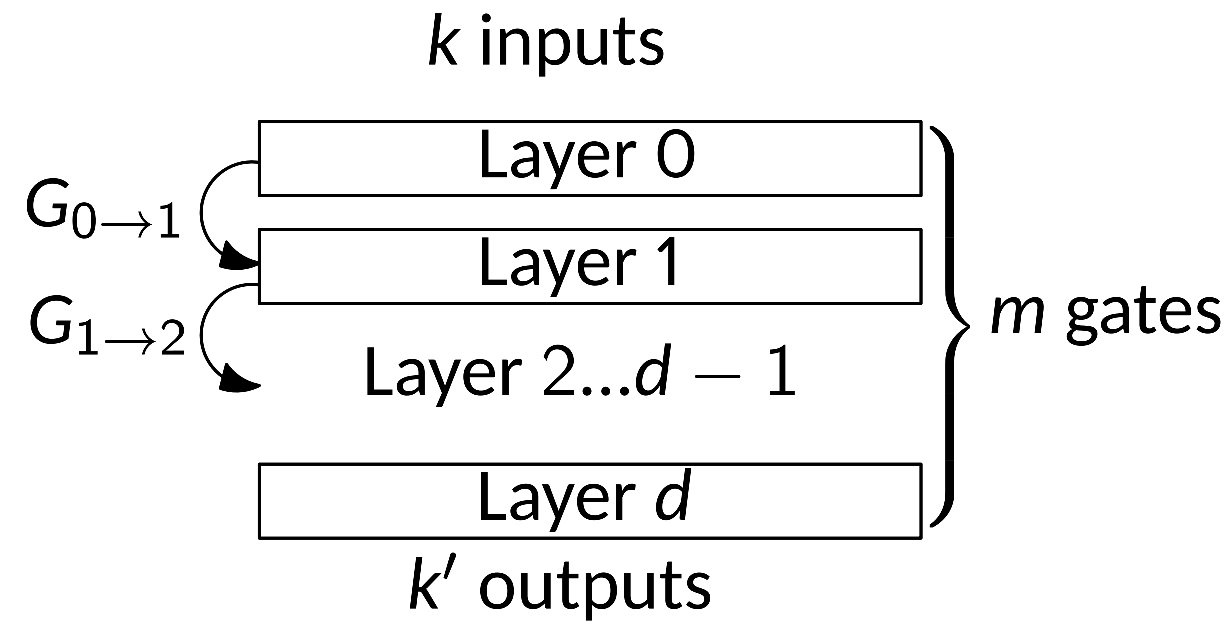
ROM-LPZKv2

- Let $\mathcal{I}_0, \mathcal{I}_1$ be even/odd layers
- $\mathcal{I}_\tau = \min(\mathcal{I}_0, \mathcal{I}_1), m' := |\mathcal{I}_\tau|$

Prover provides

$$
\begin{aligned}
\mathbf{v}^1 &= \mathbf{a}^1\Delta + \mathbf{b}_\tau^1 & |\mathbf{v}^1| = k + m' \\
\mathbf{v}^2 &= \mathbf{a}^2\Delta + \mathbf{b}^2 & |\mathbf{v}^2| = m - m'
\end{aligned}
$$

- $\mathbf{b}_\tau^1 = k$ inputs + m' wires in $\mathcal{I}_\tau$
- $\mathbf{a}^2 = G(\mathbf{a}^1)$: Input-Independent



QuickSilver:
Gate function G can be deg-2 polynomials

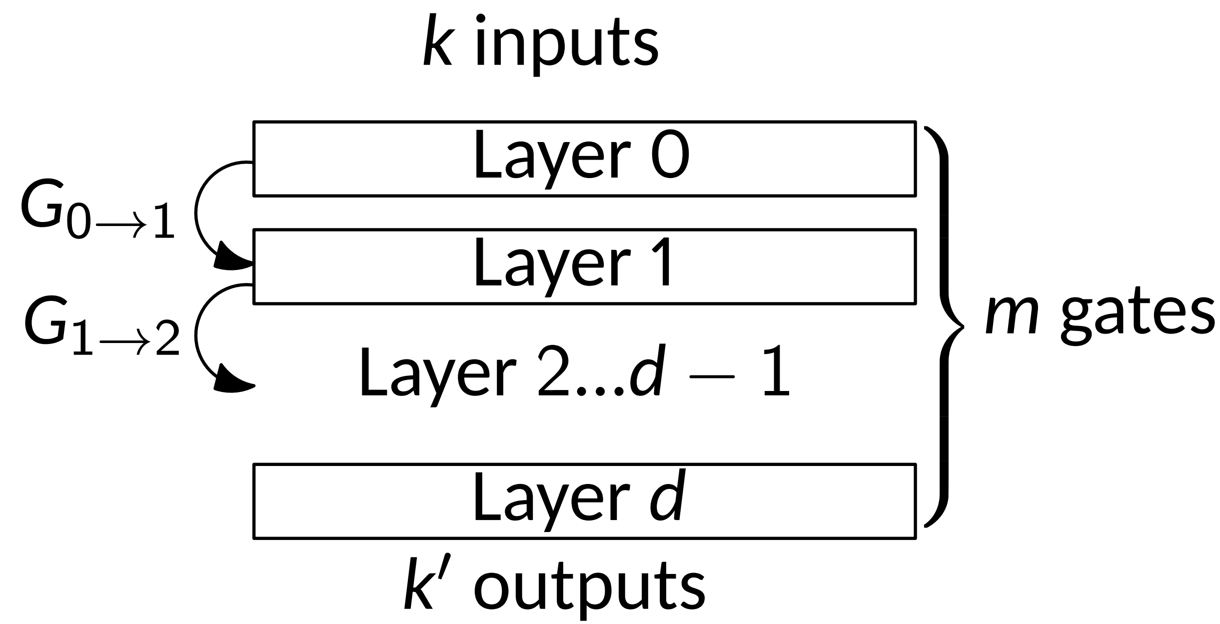
ROM-LPZKv2

- Let $\mathcal{I}_0, \mathcal{I}_1$ be even/odd layers
- $\mathcal{I}_\tau = \min(\mathcal{I}_0, \mathcal{I}_1), m' := |\mathcal{I}_\tau|$

Prover provides

$$\begin{aligned} \mathbf{v}^1 &= \mathbf{a}^1\Delta + \mathbf{b}^1_\tau & |\mathbf{v}^1| &= k + m' \\ \mathbf{v}^2 &= \mathbf{a}^2\Delta + \mathbf{b}^2 & |\mathbf{v}^2| &= m - m' \end{aligned}$$

- $\mathbf{b}^1_\tau = k$ inputs + m' wires in $\mathcal{I}_\tau$
- $\mathbf{a}^2 = G(\mathbf{a}^1)$: Input-Independent



QuickSilver:
Gate function G can be
deg-2 polynomials

- $\mathbf{v}^4 := G(\mathbf{v}^1) - \Delta\mathbf{v}^2 \Rightarrow$ Enc. of $\mathcal{I}_{1-\tau}$ wires
- Verify $G(\mathbf{v}^4) - \mathbf{v}^1 = \mathbf{x}_1\Delta^2 + \mathbf{x}_0\Delta$
- Soundness: $\frac{2}{p} + \varepsilon_{RO}$
- Communication: $\frac{m'}{m}\ \mathbb{F}_p$ per gate

Certified VOLE

- Equality Constraint: $\mathbf{a}_1[i] = \mathbf{a}_2[j]$

$$\begin{array}{c} \\ \boxed{\Delta_1} \\ \boxed{\Delta_2} \\ \\ \end{array} \times \begin{array}{c|c} i & \\ \hline \mathbf{a}_1 & \mathbf{b}_2[j] \\ \mathbf{a}_2 & \mathbf{b}_1[i] \\ \hline j & \end{array} + \begin{array}{c|c} i & \\ \hline \mathbf{b}_1 & b'_1 \\ \mathbf{b}_2 & b'_2 \\ \hline j & \end{array} = \begin{array}{c|c} i & \\ \hline \mathbf{v}_1 & v'_1 \\ \mathbf{v}_2 & v'_2 \\ \hline j & \end{array}$$

Prover sends $\delta_b = b'_1 - b'_2$

- Verifier checks

$$\begin{aligned} &\Delta_2 \mathbf{v}_1[i] - \Delta_1 \mathbf{v}_2[j] + v'_1 - v'_2 \\ &= \Delta_1 \Delta_2 (\mathbf{a}_1[i] - \mathbf{a}_2[j]) + \Delta_2(\mathbf{b}_1[i] - a'_2) - \Delta_1(\mathbf{b}_2[j] - a'_1) + b'_1 - b'_2 \\ &= \delta_b \end{aligned}$$

Certified VOLE

- Equality Constraint: $\mathbf{a}_1[i] = \mathbf{a}_2[j]$

$$\begin{array}{c} \\ \boxed{\Delta_1} \\ \boxed{\Delta_2} \\ \\ \end{array} \times \begin{array}{c} i \\ \boxed{\mathbf{a}_1 \mid \mathbf{b}_2[j]} \\ \boxed{\mathbf{a}_2 \mid \mathbf{b}_1[i]} \\ j \end{array} + \begin{array}{c} i \\ \boxed{\mathbf{b}_1 \mid b'_1} \\ \boxed{\mathbf{b}_2 \mid b'_2} \\ j \end{array} = \begin{array}{c} i \\ \boxed{\mathbf{v}_1 \mid v'_1} \\ \boxed{\mathbf{v}_2 \mid v'_2} \\ j \end{array}$$

Prover sends $\delta_b = b'_1 - b'_2$

- Verifier checks

$$\begin{aligned} &\Delta_2 \mathbf{v}_1[i] - \Delta_1 \mathbf{v}_2[j] + v'_1 - v'_2 \\ &\quad = \Delta_1\Delta_2(\mathbf{a}_1[i] - \mathbf{a}_2[j]) + \Delta_2(\mathbf{b}_1[i] - a'_2) - \Delta_1(\mathbf{b}_2[j] - a'_1) + b'_1 - b'_2 \\ &\quad = \delta_b \end{aligned}$$

- **b**-constraints: $\boxed{\begin{array}{c} v = a\Delta + b \\ \Delta\text{-MAC'ed} \end{array}} \quad \Leftrightarrow \quad \boxed{\begin{array}{c} v\Delta^{-1} = b\Delta^{-1} + a \\ \Delta^{-1}\text{-MAC'ed} \end{array}}$

Certified VOLE

Sender/Prover

$\hat{\mathbf{a}}_1, \ldots, \hat{\mathbf{a}}_L$
$\hat{\mathbf{b}}_1, \ldots, \hat{\mathbf{b}}_L$

$\mathcal{F}_{\mathsf{cVOLE}}$

Receiver/Verifier

$\Delta_1, \ldots, \Delta_L \in \mathbb{F}_p$

$$\begin{cases} \hat{\mathbf{v}}_1, \ldots, \hat{\mathbf{v}}_L & C(\hat{\mathbf{a}}_1, \ldots, \hat{\mathbf{b}}_L) = 0 \\ \perp & \textit{otherwise} \end{cases}$$

Certified VOLE

Sender/Prover

$\hat{\mathbf{a}}_1, \ldots, \hat{\mathbf{a}}_L$
$\hat{\mathbf{b}}_1, \ldots, \hat{\mathbf{b}}_L$

$\mathcal{F}_{\text{cVOLE}}$

Receiver/Verifier

$\Delta_1, \ldots, \Delta_L \in \mathbb{F}_p$

$$\begin{cases} \hat{\mathbf{v}}_1, \ldots, \hat{\mathbf{v}}_L & C(\hat{\mathbf{a}}_1, \ldots, \hat{\mathbf{b}}_L) = 0 \\ \perp & \textit{otherwise} \end{cases}$$

$\Delta_1 + \alpha, \ldots, \Delta_L + \alpha, \alpha, \beta$

$\mathcal{F}_{\text{VOLE}}$

$\mathbf{a}_1, \ldots, \mathbf{a}_L, \mathbf{a}_{L+1}, \mathbf{a}_{L+2}$
$\mathbf{b}_1, \ldots, \mathbf{b}_L, \mathbf{b}_{L+1}, \mathbf{b}_{L+2}$

Certified VOLE

Sender/Prover

Receiver/Verifier

$\hat{\mathbf{a}}_1, \ldots, \hat{\mathbf{a}}_L$

$\hat{\mathbf{b}}_1, \ldots, \hat{\mathbf{b}}_L$

$\mathcal{F}_{\mathsf{cVOLE}}$

$\Delta_1, \ldots, \Delta_L \in \mathbb{F}_p$

$$\begin{cases} \hat{\mathbf{v}}_1, \ldots, \hat{\mathbf{v}}_L & C(\hat{\mathbf{a}}_1, \ldots, \hat{\mathbf{b}}_L) = 0 \\ \perp & \textit{otherwise} \end{cases}$$

$\Delta_1 + \alpha, \ldots, \Delta_L + \alpha, \alpha, \beta$

$\mathcal{F}_{\mathsf{VOLE}}$

$\mathbf{a}_1, \ldots, \mathbf{a}_L, \mathbf{a}_{L+1}, \mathbf{a}_{L+2}$

$\mathbf{b}_1, \ldots, \mathbf{b}_L, \mathbf{b}_{L+1}, \mathbf{b}_{L+2}$

$\hat{\mathbf{a}}_1[i] =$

$\hat{\mathbf{b}}_1[i] + \mathbf{b}_{L+1}[i_2] =$

$$\begin{bmatrix} \Delta_1 + \alpha \\ \alpha \\ \beta \end{bmatrix} \times \begin{bmatrix} \mathbf{a}_1 \quad \mathbf{a}_1[i] \\ \mathbf{a}_{L+1} \quad \mathbf{a}_{L+1}[i_2] \\ \mathbf{a}_{L+2} \quad \mathbf{a}_{L+2}[i_3] \end{bmatrix} + \begin{bmatrix} \mathbf{b}_1 \quad \mathbf{b}_1[i] \\ \mathbf{b}_{L+1} \quad \mathbf{b}_{L+1}[i_2] \\ \mathbf{b}_{L+2} \quad \mathbf{b}_{L+2}[i_4] \quad \mathbf{b}_{L+2}[i_5] \end{bmatrix} = \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_{L+1} \\ \mathbf{v}_{L+2} \end{bmatrix}$$

Certified VOLE

Sender/Prover

$\hat{\mathbf{a}}_1, \ldots, \hat{\mathbf{a}}_L$

$\hat{\mathbf{b}}_1, \ldots, \hat{\mathbf{b}}_L$

$\mathcal{F}_{\text{cVOLE}}$

Receiver/Verifier

$\Delta_1, \ldots, \Delta_L \in \mathbb{F}_p$

$$\begin{cases} \hat{\mathbf{v}}_1, \ldots, \hat{\mathbf{v}}_L & C(\hat{\mathbf{a}}_1, \ldots, \hat{\mathbf{b}}_L) = 0 \\ \perp & \textit{otherwise} \end{cases}$$

$\Delta_1 + \alpha, \ldots, \Delta_L + \alpha, \alpha, \beta$

$\mathcal{F}_{\text{VOLE}}$

$\mathbf{a}_1, \ldots, \mathbf{a}_L, \mathbf{a}_{L+1}, \mathbf{a}_{L+2}$

$\mathbf{b}_1, \ldots, \mathbf{b}_L, \mathbf{b}_{L+1}, \mathbf{b}_{L+2}$

$\hat{\mathbf{a}}_1[i] =$

$\hat{\mathbf{b}}_1[i] + \mathbf{b}_{L+1}[i_2] =$

$$\begin{bmatrix} \Delta_1 + \alpha \\ \alpha \\ \beta \end{bmatrix} \times \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_1[i] \\ \mathbf{a}_{L+1} & \mathbf{a}_{L+1}[i_2] \\ \mathbf{a}_{L+2} & \mathbf{a}_{L+2}[i_3] \end{bmatrix} + \begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_1[i] & \\ \mathbf{b}_{L+1} & & \mathbf{b}_{L+1}[i_2] \\ \mathbf{b}_{L+2} & \mathbf{b}_{L+2}[i_4] & \mathbf{b}_{L+2}[i_5] \end{bmatrix} = \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_{L+1} \\ \mathbf{v}_{L+2} \end{bmatrix}$$

Prove Equality

Certified VOLE

Sender/Prover

$\hat{\mathbf{a}}_1, \ldots, \hat{\mathbf{a}}_L$
$\hat{\mathbf{b}}_1, \ldots, \hat{\mathbf{b}}_L$

$\mathcal{F}_{\text{cVOLE}}$

Receiver/Verifier

$\Delta_1, \ldots, \Delta_L \in \mathbb{F}_p$

$$\begin{cases} \hat{\mathbf{v}}_1, \ldots, \hat{\mathbf{v}}_L & C(\hat{\mathbf{a}}_1, \ldots, \hat{\mathbf{b}}_L) = 0 \\ \perp & \textit{otherwise} \end{cases}$$

$\Delta_1 + \alpha, \ldots, \Delta_L + \alpha, \alpha, \beta$

$\mathcal{F}_{\text{VOLE}}$

$\mathbf{a}_1, \ldots, \mathbf{a}_L, \mathbf{a}_{L+1}, \mathbf{a}_{L+2}$
$\mathbf{b}_1, \ldots, \mathbf{b}_L, \mathbf{b}_{L+1}, \mathbf{b}_{L+2}$

$\hat{\mathbf{a}}_1[i]$ $=$

$\hat{\mathbf{b}}_1[i] + \mathbf{b}_{L+1}[i_2]$ $=$

$$\begin{bmatrix} \Delta_1 + \alpha \\ \alpha \\ \beta \end{bmatrix} \times \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_1[i] \\ \mathbf{a}_{L+1} & \mathbf{a}_{L+1}[i_2] \\ \mathbf{a}_{L+2} & \mathbf{a}_{L+2}[i_3] \end{bmatrix} + \begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_1[i] & \\ \mathbf{b}_{L+1} & & \mathbf{b}_{L+1}[i_2] \\ \mathbf{b}_{L+2} & \mathbf{b}_{L+2}[i_4] & \mathbf{b}_{L+2}[i_5] \end{bmatrix} = \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_{L+1} \\ \mathbf{v}_{L+2} \end{bmatrix}$$

Prove Equality

Prove Equality

Prove Equality

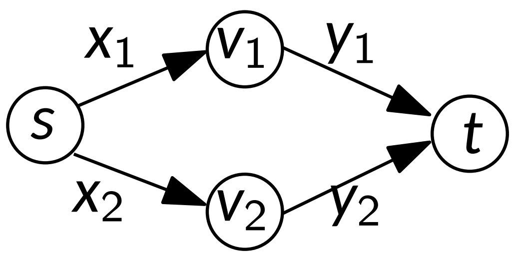
Run LPZK on $\mathbf{v}_{L+2}$ to check C-consistency

Reusable NISC over VOLE

- Theorem (Crypto19): Reusable NISC for arithmetic BP in VOLE-hybrid model
- This work: same result simplified and efficiency boosted

Reusable NISC over VOLE

- Theorem (Crypto19): Reusable NISC for arithmetic BP in VOLE-hybrid model
- This work: same result simplified and efficiency boosted



$$f(\mathbf{x}, \mathbf{y}) = x_1 y_1 + x_2 y_2$$

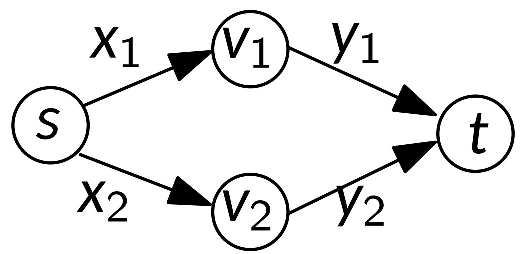
$$A_G = \begin{bmatrix} 0 & x_1 & x_2 & 0 \\ 0 & 0 & 0 & y_1 \\ 0 & 0 & 0 & y_2 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{aligned} f(\mathbf{x}, \mathbf{y}) &= (I + A_G + A_G^2 + \ldots)_{1,4} \\ &= ((I - A_G)^{-1})_{1,4} \\ &= \det(A^*) \end{aligned}$$

- Lemma (IK02): $\mathrm{Sim}(\det(A^*)) \equiv \det\left(\begin{bmatrix} 1 & r_1 & r_2 \\ 0 & 1 & r_3 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 & x_2 & 0 \\ -1 & 0 & y_1 \\ 0 & -1 & y_2 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & r_4 \\ 0 & 1 & r_5 \\ 0 & 0 & 1 \end{bmatrix} \right)$

Reusable NISC over VOLE

- Theorem (Crypto19): Reusable NISC for arithmetic BP in VOLE-hybrid model
- This work: same result simplified and efficiency boosted



$$A_G = \begin{bmatrix} 0 & x_1 & x_2 & 0 \\ 0 & 0 & 0 & y_1 \\ 0 & 0 & 0 & y_2 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

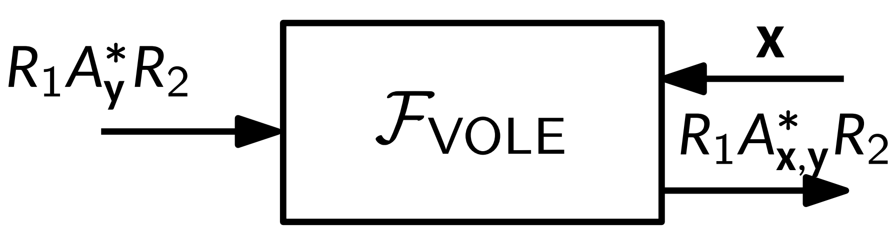
$$\begin{aligned} f(\mathbf{x}, \mathbf{y}) &= (I + A_G + A_G^2 + ...)_{1,4} \\ &= ((I - A_G)^{-1})_{1,4} \\ &= \det(A^*) \end{aligned}$$

$$f(\mathbf{x}, \mathbf{y}) = x_1 y_1 + x_2 y_2$$

- Lemma (IK02): $\mathrm{Sim}(\det(A^*)) \equiv \det\left(\begin{bmatrix} 1 & r_1 & r_2 \\ 0 & 1 & r_3 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 & x_2 & 0 \\ -1 & 0 & y_1 \\ 0 & -1 & y_2 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & r_4 \\ 0 & 1 & r_5 \\ 0 & 0 & 1 \end{bmatrix} \right)$

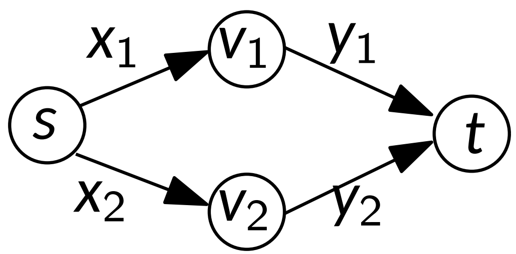
Semi-Honest Sender

Malicious Receiver



Reusable NISC over VOLE

- Theorem (Crypto19): Reusable NISC for arithmetic BP in VOLE-hybrid model
- This work: same result simplified and efficiency boosted



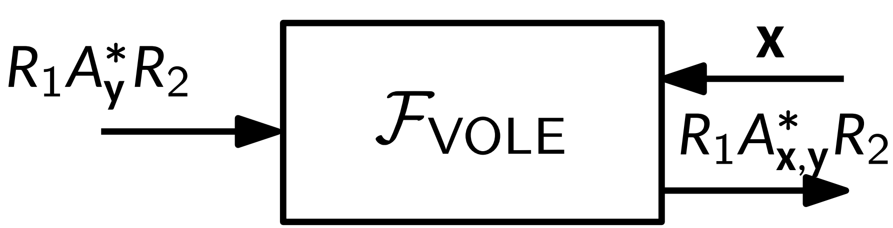
$f(\mathbf{x}, \mathbf{y}) = x_1 y_1 + x_2 y_2$

$$A_G = \begin{bmatrix} 0 & x_1 & x_2 & 0 \\ 0 & 0 & 0 & y_1 \\ 0 & 0 & 0 & y_2 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{aligned} f(\mathbf{x}, \mathbf{y}) &= (I + A_G + A_G^2 + \ldots)_{1,4} \\ &= ((I - A_G)^{-1})_{1,4} \\ &= \det(A^*) \end{aligned}$$

- Lemma (IK02): $\mathrm{Sim}(\det(A^*)) \equiv \det\left(\begin{bmatrix} 1 & r_1 & r_2 \\ 0 & 1 & r_3 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 & x_2 & 0 \\ -1 & 0 & y_1 \\ 0 & -1 & y_2 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & r_4 \\ 0 & 1 & r_5 \\ 0 & 0 & 1 \end{bmatrix} \right)$

Semi-Honest Sender | Malicious Receiver



Malicious Sender | Malicious Receiver