Post-Quantum Signatures via Publicly Verifiable LPZK

CCS'23 Submission 1342

Anonymous Authors

June 1, 2023 · Presented by Hongrui Cui

Motivations

- Efficient VOLE-based DVZK
- How to transform DVZK to (NI)ZK?

- P.S. Landscape of Efficient Zero Knowledge

	zk-SNARK, GKR, etc.	GCZK	DVZK
Prover Computation	$\Omega(\lvert C \rvert)$	$O(\lvert C \rvert)$	$O(\lvert C \rvert)$
Prover Memory	$\Omega(\lvert C \rvert)$	$O(1)$	$O(1)$
Proof Size	$O(\log(\lvert C \rvert))$	$O(\kappa \cdot \lvert C \rvert)$	$O(\lvert C \rvert^{\{1, \frac{3}{4}, \frac{1}{2}\}})$
Verifier Type	Universal	Designated	Designated
Advantage (Scalability)	Low-Bandwidth Small Circuit	High-Bandwidth Large Circuit	High-Bandwidth Large Circuit Polynomials

Main techniques (of DVZK):

- Random (subfield) VOLE
- Low-Degree Test

Preliminary: VOLE as IT-MAC (Linear Commitment)

Receiver/$\mathcal{V}$ | Sender/$\mathcal{P}$

$\Delta \in \mathbb{F}_{p^r}$ (global key)
$\mathbf{v} \in \mathbb{F}_{p^r}^n$ (MAC Key)

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

$\mathbf{a} \in \mathbb{F}_{p^r}^n$ (message)
$\mathbf{b} \in \mathbb{F}_{p^r}^n$ (MAC Tag)

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

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

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

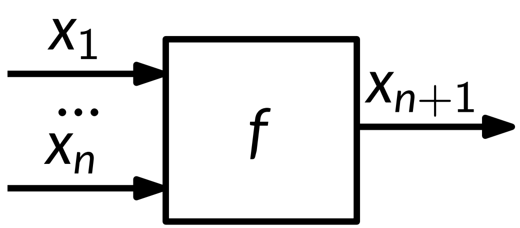
Receiver/$\mathcal{V}$ | Sender/$\mathcal{P}$

$\{\chi_i\}$ $\longrightarrow$

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

- $\mathcal{P}$ opens a different value $\to$ $\mathcal{P}$ guesses Δ
- Soundness error $= \frac{1}{p^r}$

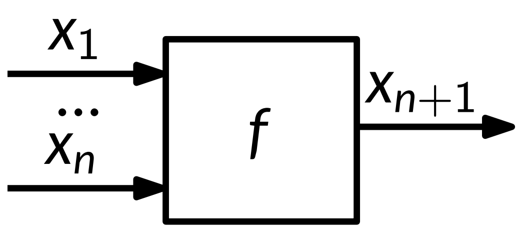
Starting Point: Designated Verifier Zero Knowledge



$$\underbrace{\mathbf{v} = \Delta}_{\mathcal{V}} \cdot \underbrace{\mathbf{x} + \mathbf{b}}_{\mathcal{P}}$$

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

Starting Point: Designated Verifier Zero Knowledge



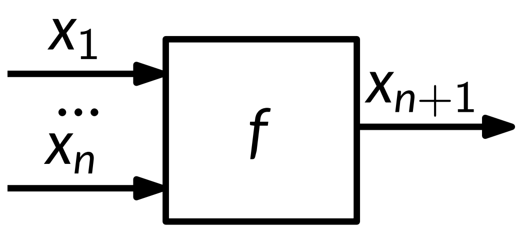
$$\underbrace{\mathbf{v} = \Delta}_{\mathcal{V}} \cdot \underbrace{\mathbf{x} + \mathbf{b}}_{\mathcal{P}}$$

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

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

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

Starting Point: Designated Verifier Zero Knowledge



$$\underbrace{\mathbf{v} = \Delta}_{\mathcal{V}} \cdot \underbrace{\mathbf{x} + \mathbf{b}}_{\mathcal{P}}$$

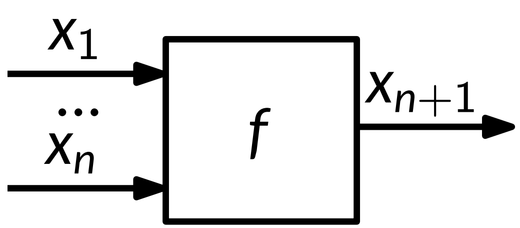
$$\begin{aligned} f(\mathbf{x}) &= f_d(\mathbf{x}) + f_{d-1}(\mathbf{x})... + f_0 \\ f(\mathbf{v}) &= f_d(\mathbf{v}) + f_{d-1}(\mathbf{v}) + ... + f_0 \\ &= f_d(\mathbf{x})\Delta^d + f_{d-1}(\mathbf{x})\Delta^{d-1} + ... + f_0 + f_r(\mathbf{x}, \mathbf{b}) \end{aligned}$$

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

Π^{d-1}_{Setup}

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

Starting Point: Designated Verifier Zero Knowledge



$$\underbrace{\mathbf{v} = \Delta}_{\mathcal{V}} \cdot \underbrace{\mathbf{x} + \mathbf{b}}_{\mathcal{P}}$$

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

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

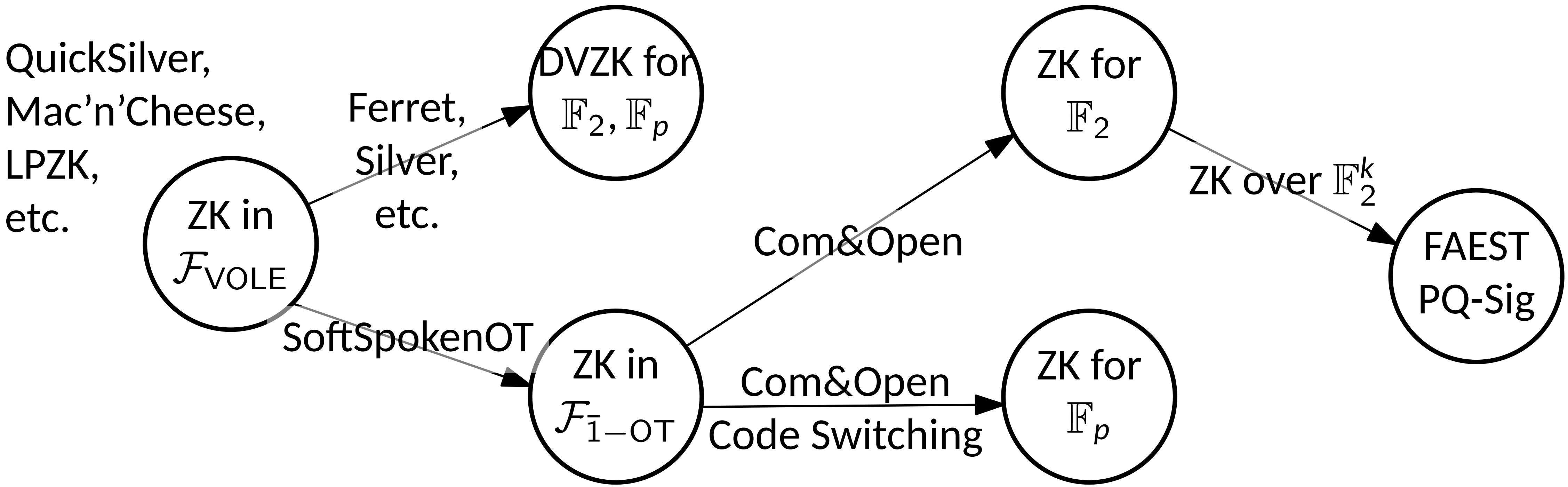
$$\begin{aligned} g(\mathbf{v}) &:= f_d(\mathbf{v}) + \Delta f_{d-1}(\mathbf{v}) + ... + \Delta^{d-1} f_1(\mathbf{v}) + \Delta^d f_0 - \Delta^{d-1} v_{n+1} \\ &= (f_d(\mathbf{x}) + ... + f_0 - x_{n+1})\Delta^d + \underbrace{f'_{r,\mathbf{x},\mathbf{b}}(\Delta)}_{\deg(\Delta) < d} \end{aligned}$$

Π^{d-1}_{Setup}

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

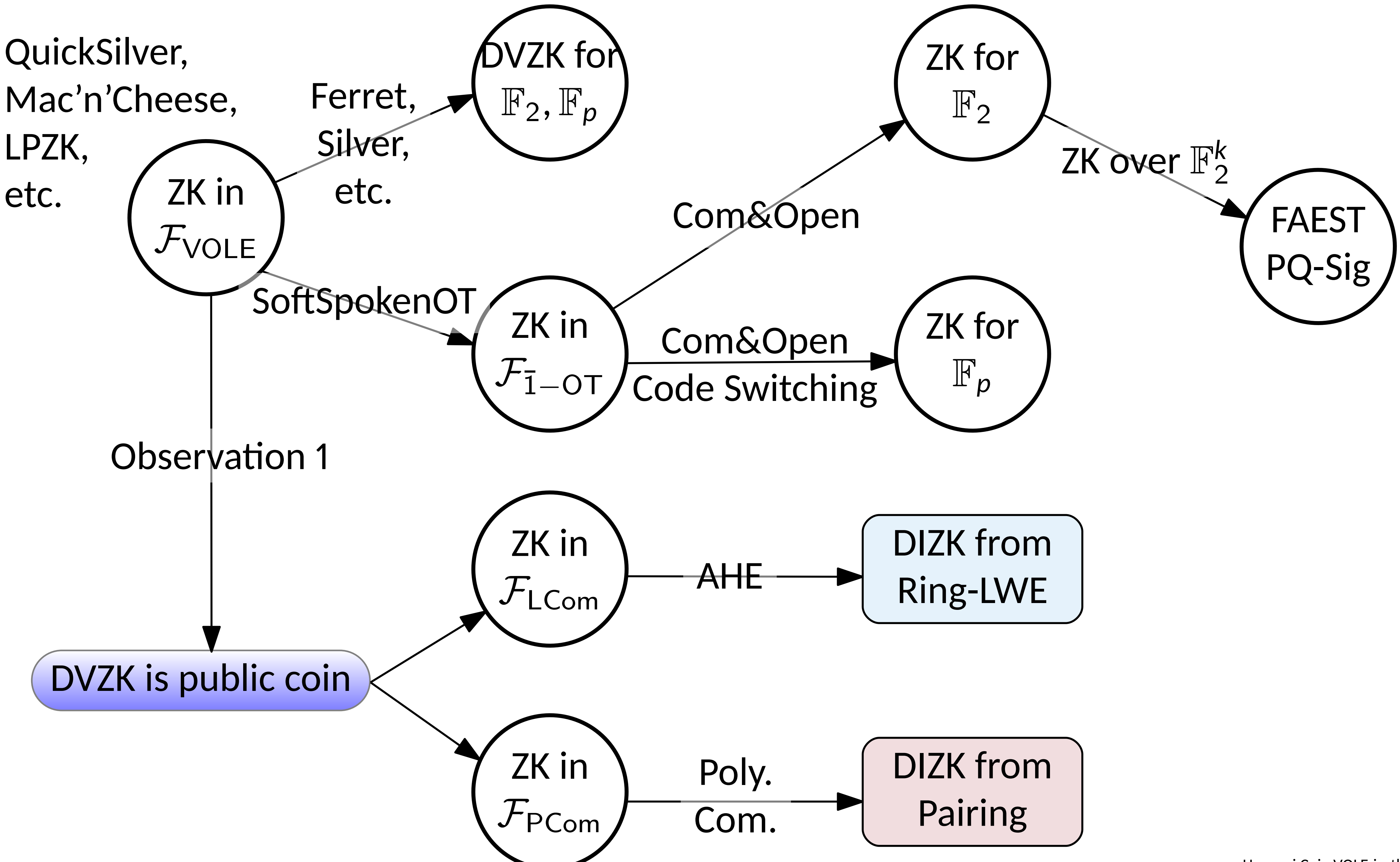
- Sends collapsed, masked coeff. of $g(\mathbf{v})$
- Soundness: $\frac{d}{p}$

Contributions (of VOLEitH)

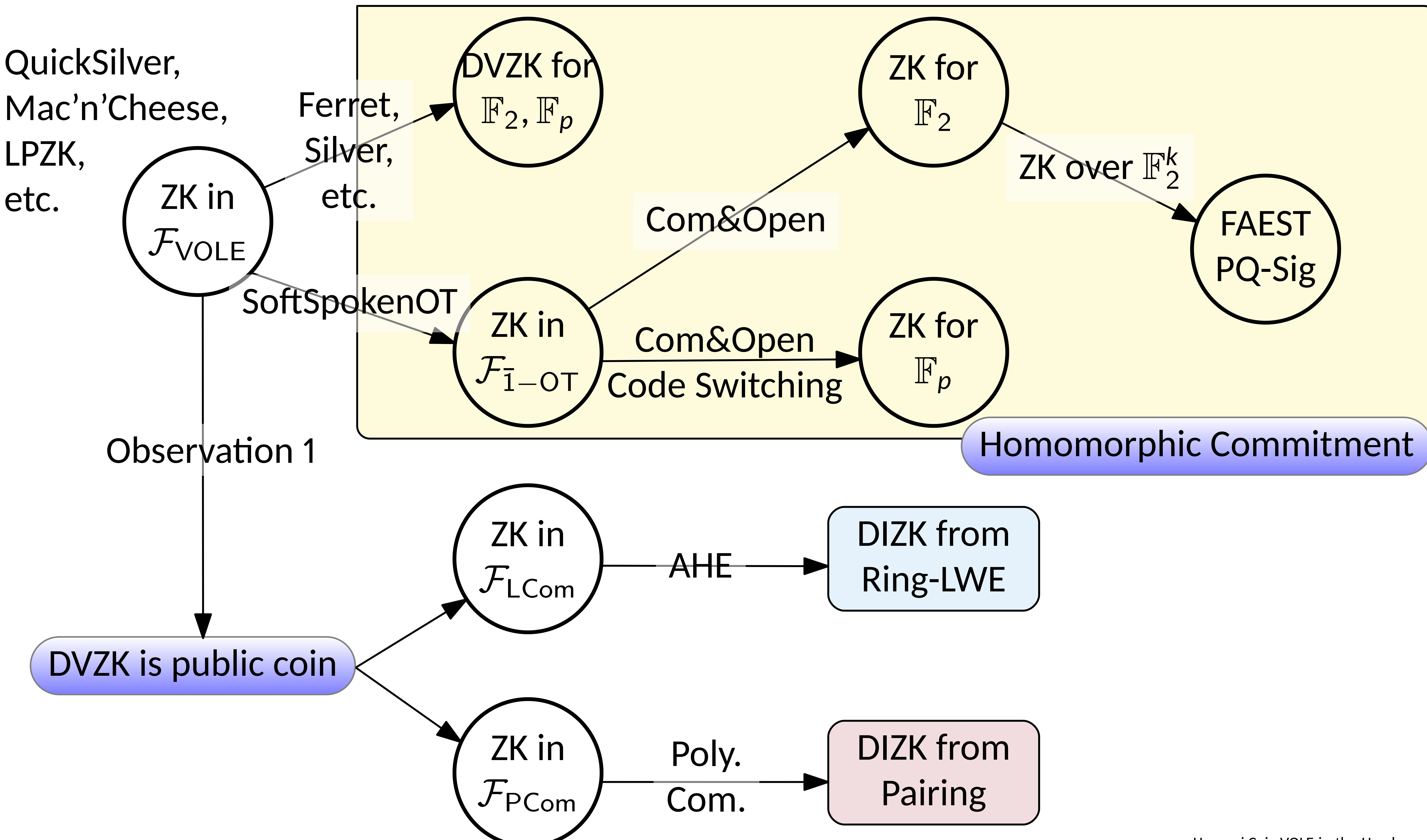


- Observation 1: In DVZK, Verifier is **public coin** and VOLE output can be delayed to the very end after all communications
- Observation 2: Subspace VOLE (SoftSpokenOT) allows reduction to OT
- Observation 3: OT can be replaced with com-and-open

Contributions (of LPZKitH)



Contributions (of LPZKitH)



Performance of the ZK Compilers

Scheme	Key size	Sig. size	Gen	Sign	Verify
Picnic2 [24]	64 B	45.9 kB	0.01 ms	28 ms	28 ms
Banquet [3]	-	19.78 kB	-	6.36 ms	4.86 ms
PorcRoast [7]	-	7.2 kB	-	2.8 ms	-
RSD-S [12]	0.09 kB	8.55 kB	-	31 ms	-
Falcon[21]	897 B	0.67 kB	8.64 ms	168 μs	35 μs
Dilithium3-AES	1.95 kB	3.3 kB	30μs	93μs	30μs
SPHINCS+	1.06 kB	41 kB	0.82 ms	13 ms	0.58 ms
LPZK PAL	68.3 kB	69.2 kB	$\approx$ 15 ms	$\approx$ 1 ms	$\approx$ 1 ms
LPZK SHIELD	6.2 kB	7.4 kB	-	-	-

Table 1: Metrics for post-quantum signature schemes. LPZK PALISADE numbers estimated from pv-LPZK PALISADE performance, LPZK SHIELD estimated analytically.

Performance of the ZK Compilers

Scheme	$t_{\mathcal{P}}$ (ms)	$t_{\mathcal{V}}$ (ms)	|sign| (B)	Assumption
SDitH [FJR22b] (fast)	13.40	12.70	17 866	SD $\mathbb{F}_2$
SDitH [FJR22b] (short)	64.20	60.70	12 102	SD $\mathbb{F}_2$
SDitH [FJR22b] (fast)	6.40	5.90	12 115	SD $\mathbb{F}_{256}$
SDitH [FJR22b] (short)	29.50	27.10	8 481	SD $\mathbb{F}_{256}$
Rainier$_3$ [DKR+22]	2.96	2.92	6 176	RAIN$_3$
Rainier$_4$ [DKR+22]	3.47	3.42	6 816	RAIN$_4$
Limbo [dOT21] (fast)	2.61	2.25	23 264	Hash
Limbo [dOT21] (short)	24.51	21.82	13 316	Hash
SPHINCS+-SHA2 [HBD+22] (fast)	4.40	0.40	17 088	Hash
SPHINCS+-SHA2 [HBD+22] (short)	88.21	0.15	7 856	Hash
Falcon-512 [PFH+22]	0.11	0.02	666	Lattice
Dilithium2 [LDK+22]	0.07	0.03	2 420	Lattice
FAEST (**this work**, fast, $q = 2^8$)	2.28	2.11	6 583	Hash
FAEST (**this work**, short, $q = 2^{11}$)	11.05	10.18	5 559	Hash

LPZK PAL	68.3 kB	69.2 kB	$\approx$ 15 ms	$\approx$ 1 ms	$\approx$ 1 ms
LPZK SHIELD	6.2 kB	7.4 kB	-	-	-

Table 1: Metrics for post-quantum signature schemes. LPZK PALISADE numbers estimated from pv-LPZK PALISADE performance, LPZK SHIELD estimated analytically.

DIZK From AHE

Figure 3: Post-quantum publicly verifiable LPZK from add tive homorphic encryption using Ring-LWE.

Protocol $\Pi_{\text{PV-LPZK}}$: Post-quantum publicly verifiable LPZK from Ring-LWE.

Parametrized by a finite field $\mathbb{F}$, a circuit C, and a length n, with a randomized LPZK scheme as in §2.1 and an commitment scheme under AHE as in § 2.2.

(1) (preprocessing) P computes $(pk; sk) := \mathsf{Gen}(\kappa)$ under an AHE scheme.

(2) (preprocessing) P generates random vectors $(\mathbf{a}, \mathbf{b})$ of length n and generates the encryptions $\langle \mathbf{a} \rangle, \langle \mathbf{b} \rangle := \mathsf{Enc}(\mathbf{a}, pk), \mathsf{Enc}(\mathbf{b}, pk)$.

(3) P generates $\mathbf{m} := \mathsf{Prove}(\mathbf{a}, \mathbf{b}, C, \mathbf{w})$ under rLPZK.

(4) P computes $\alpha := H(\langle \mathbf{a} \rangle, \langle \mathbf{b} \rangle || \mathbf{m})$ and $\mathbf{v} := \mathbf{a}\alpha + \mathbf{b}$.

(5) P computes $\mathbf{q} := H(\alpha || \mathbf{v})$, $m_q := \sum q_i(a_i\alpha + b_i)$ and

$$\pi := \mathsf{Open}_{\text{AHE}}\Big(\sum_{i=1}^{n} (q_i\alpha \cdot \langle a_i \rangle + q_i \cdot \langle b_i \rangle), m_q, sk\Big).$$

(6) P sends $(pk, \langle \mathbf{a} \rangle, \langle \mathbf{b} \rangle, \mathbf{m}, \mathbf{v}, \pi)$ to V.

(7) V computes $\alpha = H(\langle \mathbf{a} \rangle || \langle \mathbf{b} \rangle || \mathbf{m})$ and $\mathbf{q} = H(\alpha || \mathbf{v})$, computes $m_q := \sum q_i v_i$, and invokes $\mathsf{Verify}_{\text{AHE}}(\sum_{i=1}^{n} (q_i\alpha \cdot \langle a_i \rangle + q_i \cdot \langle b_i \rangle), m_q, \pi)$.

(8) V runs $\mathsf{Verify}(C, \alpha, \mathbf{v}, \mathbf{m})$ and returns acc if all verification steps succeed, and rej otherwise.

DVZK from Polynomial Commitment

Figure 5: Publicly verifiable LPZK from a polynomial commitment scheme

Protocol $\Pi_{\text{PV-LPZK}'}$: pv-LPZK from a polynomial commitment scheme.

Parametrized by a finite field $\mathbb{F}$ and a length n, with a polynomial commitment scheme $(\text{Commit}_{\text{POLY}}, \text{Open}, \text{Open-Verify})$ and an LPZK scheme $(\text{Prove}, \text{Verify})$.

(1) (preprocessing) P generates random vectors $(\mathbf{a}, \mathbf{b})$ of length $n+1$ (i.e. extending the usual vectors by one entry) and generates commitments $g_a := \text{Commit}_{\text{POLY}}(f_{\mathbf{a}}), g_b := \text{Commit}_{\text{POLY}}(f_{\mathbf{b}})$.

(2) P generates $(\mathbf{m}) := \text{Prove}(\mathbf{a}, \mathbf{b}, C, \mathbf{w})$ under LPZK.

(3) P computes $\alpha := H(g_a||g_b||\mathbf{m})$ and $\mathbf{v} := \mathbf{a}\alpha + \mathbf{b}$.

(4) P computes $q := H(\alpha||\mathbf{v})$ and $(w_a, \pi_a) := \text{Open}_{\text{POLY}}(f_{\mathbf{a}}(q))$ and $(w_b, \pi_b) := (\text{Open}_{\text{POLY}}(f_{\mathbf{a}}(q))$

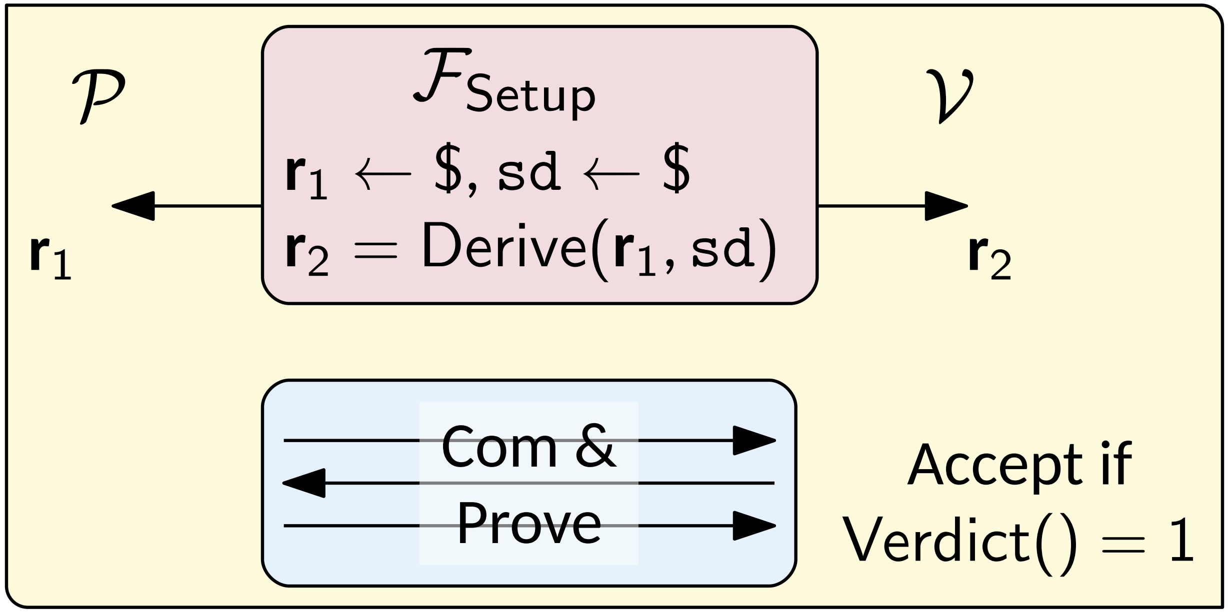
(5) P sends $(g_a, g_b, \mathbf{m}, \mathbf{v}, w_a, m_a, w_b, m_b)$ to V.

(6) V computes $\alpha = H(g_a||g_b||\mathbf{m})$ and $q = H(\alpha||\mathbf{v})$ and invokes $\text{Verify}_{\text{POLY}}(g_a, , q, \pi_a, g_a)$ and $\text{Verify}_{\text{POLY}}(g_b, q, w_b, \pi_b)$.

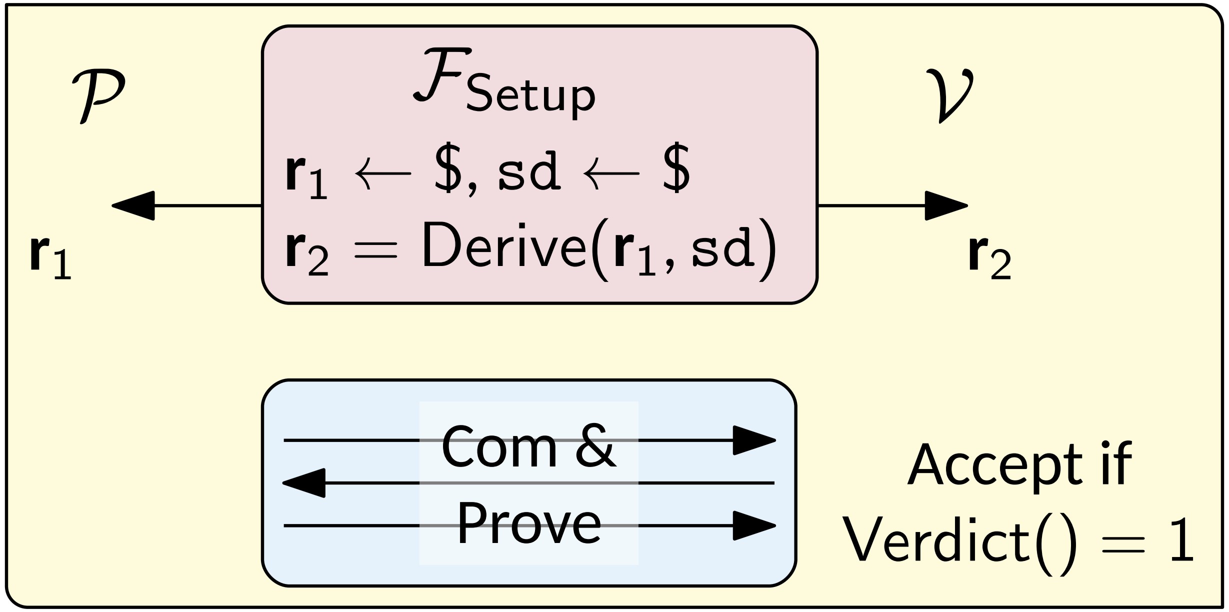
(7) V verifies that $f_{\mathbf{v}}(q) = w_a\alpha + w_b$.

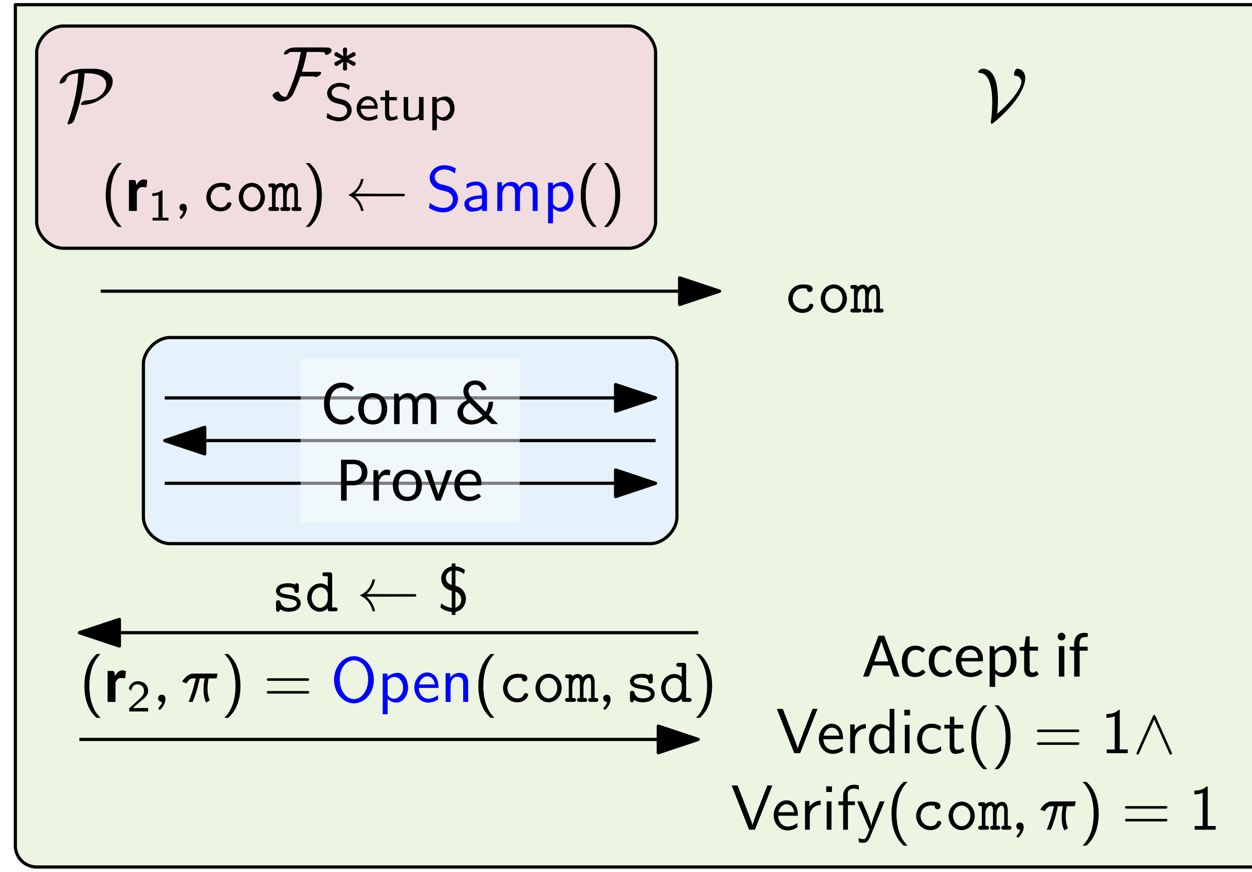
(8) V runs $\text{Verify}(C, \alpha, \mathbf{v})$ and returns the result.

Insights of LPZK-in-the-Head

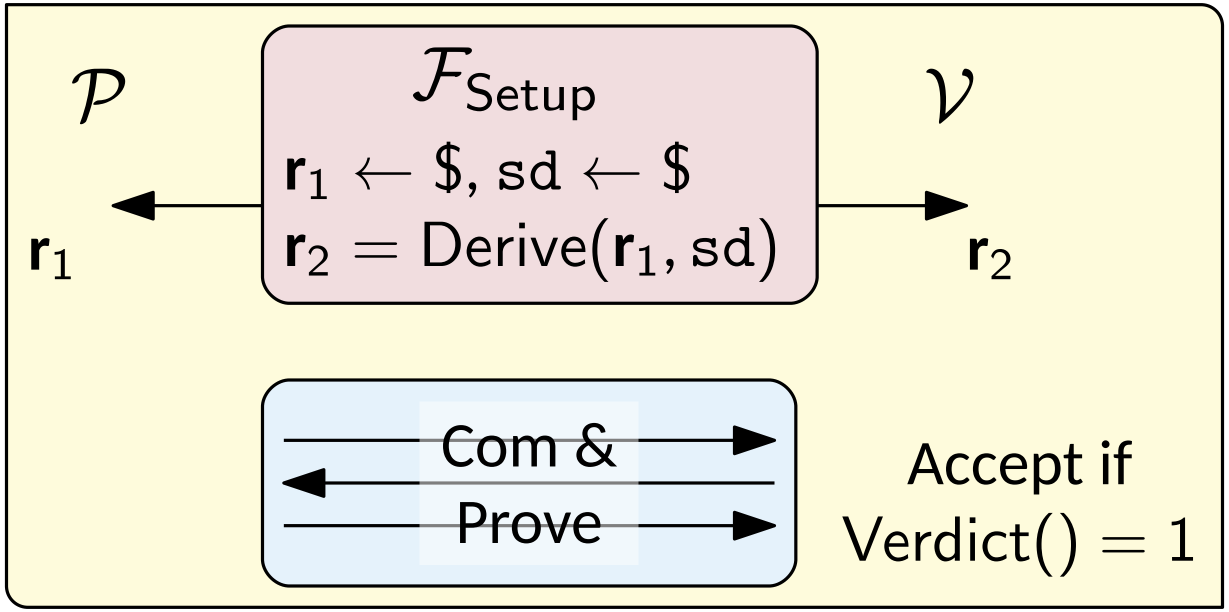


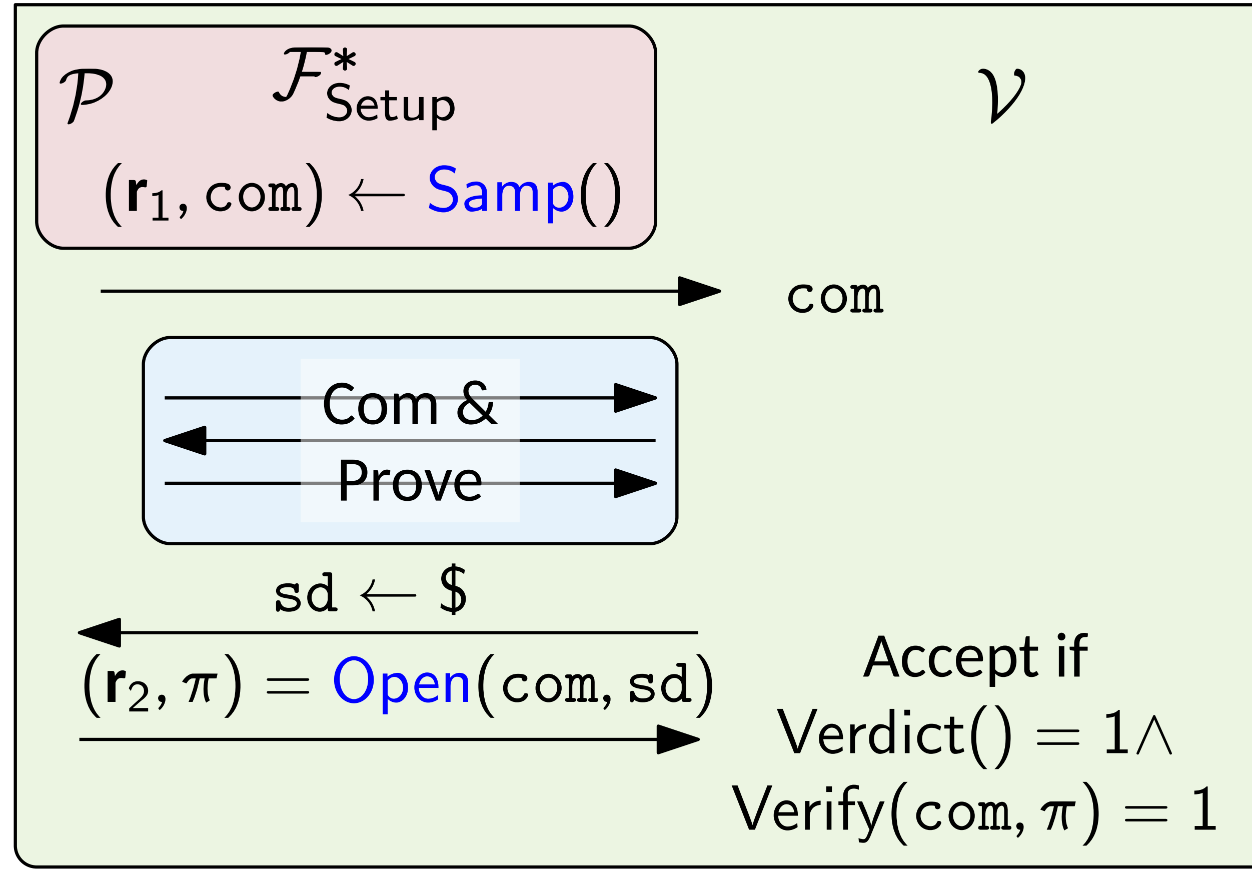
Insights of LPZK-in-the-Head

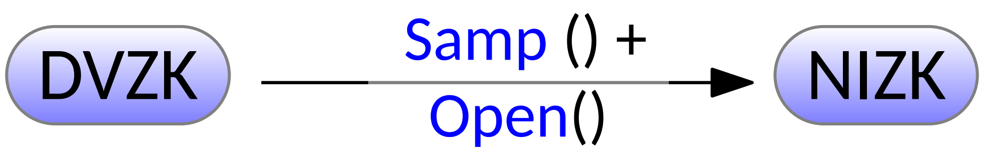




Insights of LPZK-in-the-Head

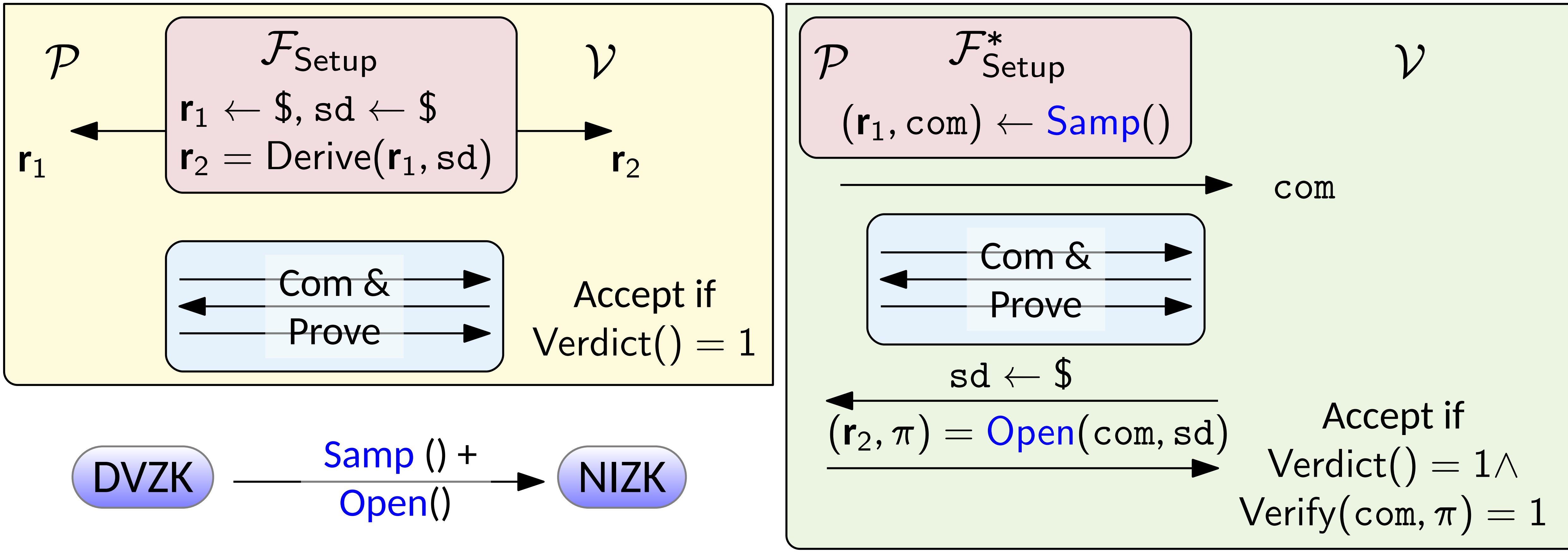






- AHE: Samp$() \mapsto (\mathbf{r}_1, \text{Enc}(\mathbf{r}_1; \mathtt{coin}))$,
  Open$() \mapsto (\mathbf{r}_2 = \text{Lin}_{\mathtt{sd}}(\mathbf{r}_1), \mathtt{coin})$

Insights of LPZK-in-the-Head



- AHE: $\text{Samp}() \mapsto (\mathbf{r}_1, \text{Enc}(\mathbf{r}_1; \mathtt{coin}))$,
  $\text{Open}() \mapsto (\mathbf{r}_2 = \text{Lin}_{\mathtt{sd}}(\mathbf{r}_1), \mathtt{coin})$
- PC: $\text{Samp}() \mapsto (\mathbf{r}_1, \text{PC.Com}(f_{\mathbf{r}_1}))$,
  $\text{Open}() \mapsto (\mathbf{r}_2 = \text{Lin}_{\mathtt{sd}}(\mathbf{r}_1), \text{PC.Open}(f_{\mathbf{r}_1}(q)))$

Why should we care about this?

- For PQ-Sig, witness length is small!
- For AES-128, witness $\approx 200\ \mathbb{F}_{2^8}$

Why should we care about this?

- For PQ-Sig, witness length is small!
- For AES-128, witness $\approx 200\ \mathbb{F}_{2^8}$
- For Syndrome Decoding/LPN

Scheme	SD Parameters					MPC Parameters			
	q	m	k	w	d	$\lvert\mathbb{F}_{\text{poly}}\rvert$	$\lvert\mathbb{F}_{\text{points}}\rvert$	t	p
Variant 1	2	1280	640	132	1	2^{11}	2^{22}	6	$\approx 2^{-69}$
Variant 2	2	1536	888	120	6	2^{8}	2^{24}	5	$\approx 2^{-79}$
Variant 3	2^8	256	128	80	1	2^{8}	2^{24}	5	$\approx 2^{-78}$

Table 3: SD and MPC parameters.

Why should we care about this?

- For PQ-Sig, witness length is small!
- For AES-128, witness $\approx 200\ \mathbb{F}_{2^8}$
- For Syndrome Decoding/LPN

Scheme	SD Parameters					MPC Parameters			
	q	m	k	w	d	$\lvert\mathbb{F}_{\text{poly}}\rvert$	$\lvert\mathbb{F}_{\text{points}}\rvert$	t	p
Variant 1	2	1280	640	132	1	2^{11}	2^{22}	6	$\approx 2^{-69}$
Variant 2	2	1536	888	120	6	2^{8}	2^{24}	5	$\approx 2^{-79}$
Variant 3	2^8	256	128	80	1	2^{8}	2^{24}	5	$\approx 2^{-78}$

Table 3: SD and MPC parameters.

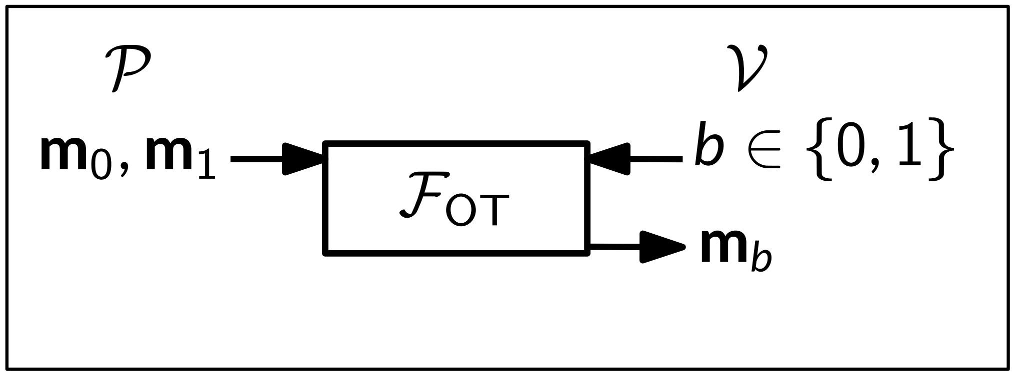
- Witness length $= 1500 \sim 1600$ bits

Warning: results are very crude

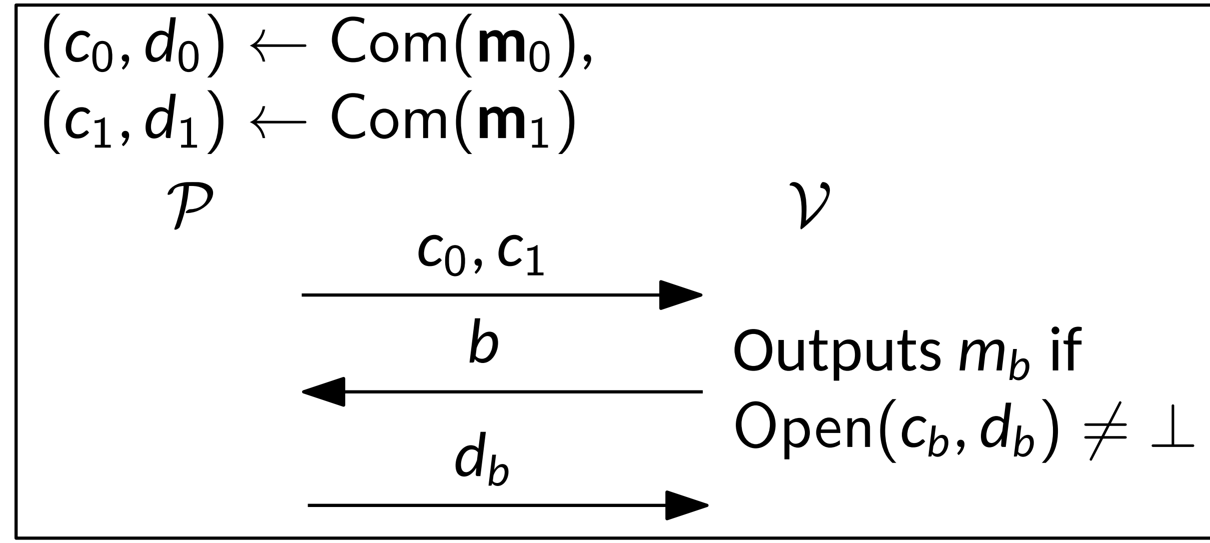
Ratios			
OT	OTe	QS Proof	Total Comm.
62.12%	31.72%	6.16%	3786.0
62.10%	31.70%	6.20%	3787.8
59.80%	34.45%	5.75%	4066.9
59.77%	34.43%	5.80%	4068.9
59.74%	34.41%	5.85%	4070.9
59.71%	34.40%	5.89%	4072.9
57.16%	37.38%	5.46%	4282.8
57.13%	37.36%	5.51%	4285.0
57.10%	37.34%	5.56%	4287.2
57.07%	37.32%	5.61%	4289.5

Starting Point: Public Coin $\mathcal{F}_{\mathrm{OT}}$ by Com&Open

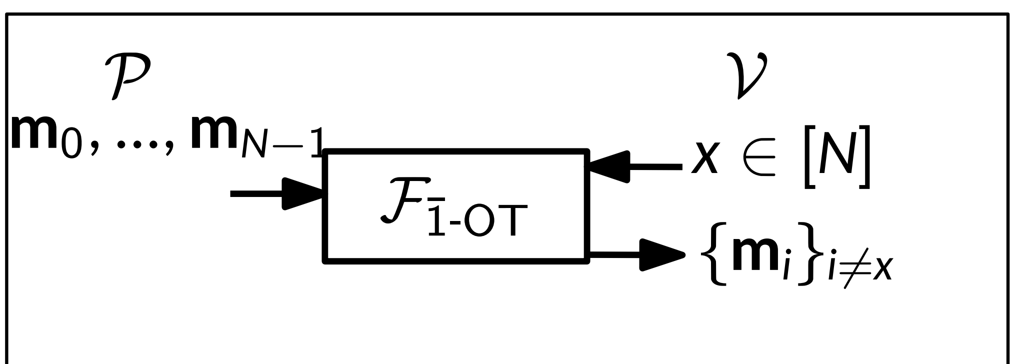
- For public-coin $\mathcal{V}$, we have public-coin $\binom{2}{1}$-OT



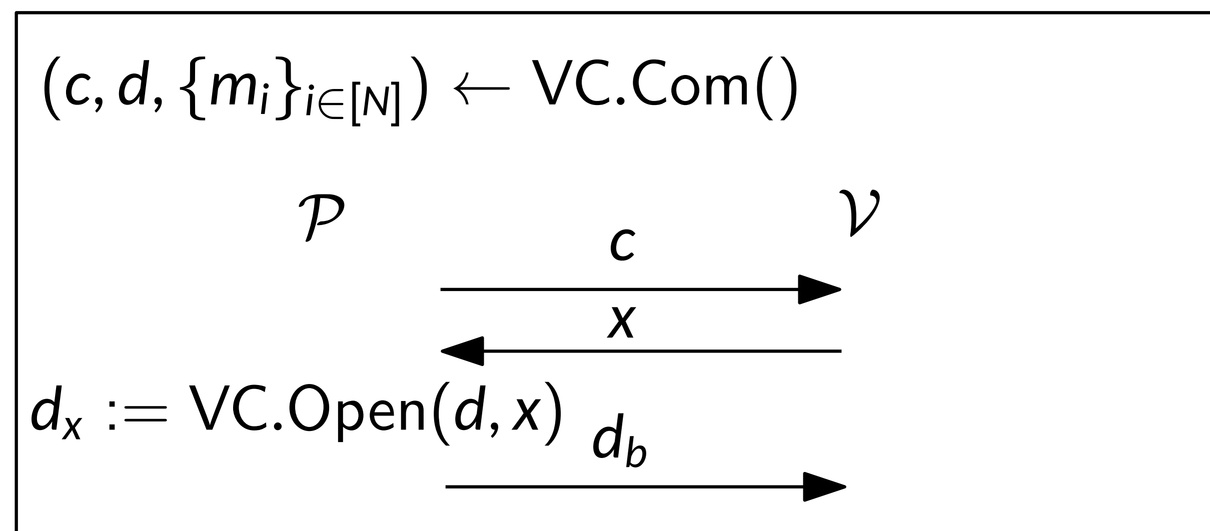
$\equiv$



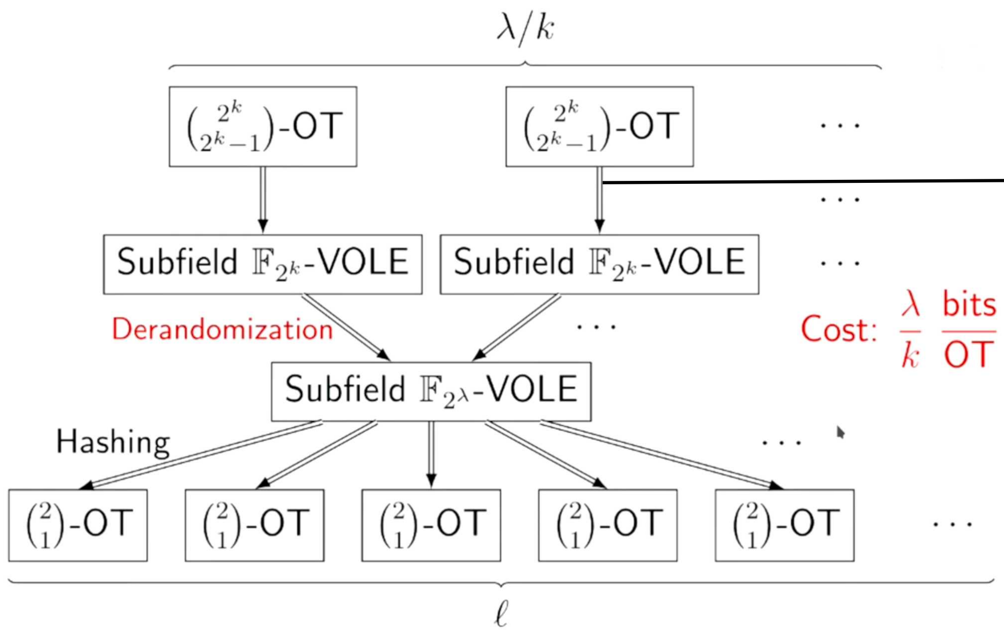
- In particular, we have public-coin $\binom{N}{N-1}$-OT with $O(\log N)$ comm.



$\equiv$



Next Step: From $\mathcal{F}_{\bar{1}\text{-OT}}$ to Subspace $\mathcal{F}_{\text{VOLE}}$ (SoftSpokenOT)



$\mathcal{P}$		$\mathcal{V}$
$\{m_i\}_{i\in[2^k]}$	$\longleftarrow \boxed{\mathcal{F}_{\bar{1}\text{-OT}}} \longrightarrow$	$\{m_i\}_{i\neq x}$
$\mathbf{a} := \sum_i m_i$		$\Delta := x$
$\mathbf{b} := \sum_i -i \cdot m_i$		$\mathbf{v} := \sum_i (\Delta - i) \cdot m_i$

$$\mathbf{b} = \mathbf{v} - \mathbf{a} \times \Delta \quad \in \mathbb{F}_{2^k}$$

Repeat $\ell = \lceil \kappa/k \rceil$ times

$$B = V' - A' \times \begin{pmatrix} \Delta_1 & & \\ & \ddots & \\ & & \Delta_\ell \end{pmatrix}$$

κ-bit entropy

From $\mathcal{F}_{\bar{1}\text{-OT}}$ to Subspace $\mathcal{F}_{\text{VOLE}}$ (SoftSpokenOT), Continued

- Goal: q^{-d}-sound IT-MAC

$$\ell\ \underbrace{\boxed{B}}_{n_{\mathcal{C}}} = \underbrace{\boxed{V}}_{n_{\mathcal{C}}} - \underbrace{\boxed{A}}_{k_{\mathcal{C}}} \times \underbrace{\boxed{G_{\mathcal{C}}}}_{n_{\mathcal{C}}} \times \begin{bmatrix} \Delta_1 & & \\ & \ddots & \\ & & \Delta_{n_{\mathcal{C}}} \end{bmatrix}$$

$$\mathbb{F}_{2^\kappa} \equiv \mathbb{F}_2^\kappa$$

$$\Delta \in \mathbb{F}_{2^\kappa} \equiv \boxed{Rep(\kappa)} \times \begin{bmatrix} \Delta_1 & & \\ & \ddots & \\ & & \Delta_{\kappa} \end{bmatrix}$$

- Send Syndrome

$$\boxed{B} = \boxed{V'} - \boxed{A'} \times \begin{bmatrix} \Delta_1 & & \\ & \ddots & \\ & & \Delta_{n_{\mathcal{C}}} \end{bmatrix}$$

$\mathcal{C}$-decompose

$$\boxed{A'} \xrightarrow{\mathcal{C}\text{-decompose}} \boxed{A} \times \boxed{G_{\mathcal{C}}} + \boxed{C} \times \boxed{H_{\mathcal{C}}}$$

From $\mathcal{F}_{\bar{1}\text{-OT}}$ to Subspace $\mathcal{F}_{\mathsf{VOLE}}$ (SoftSpokenOT), Continued

- $\mathcal{V}$ locally sets $V = V' - C \cdot H_{\mathcal{C}}$

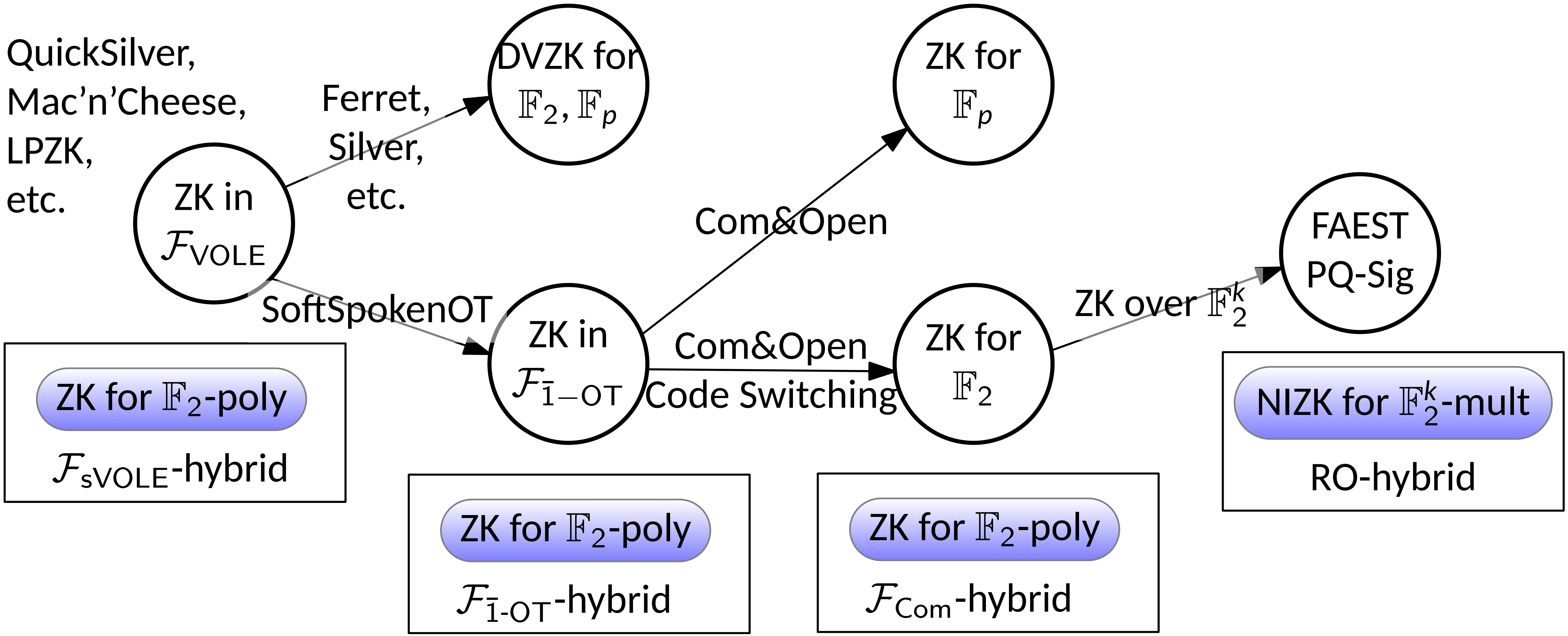
$$\boxed{\text{B}} = \boxed{\text{V'}} - \left(\boxed{\text{A}} \times \boxed{G_{\mathcal{C}}} + \boxed{\text{C}} \times \boxed{H_{\mathcal{C}}} \right) \times \begin{bmatrix} \Delta_1 & & \\ & \ddots & \\ & & \Delta_{n_{\mathcal{C}}} \end{bmatrix}$$

$$= \boxed{\text{V}} - \boxed{\text{A}} \times \boxed{G_{\mathcal{C}}} \times \begin{bmatrix} \Delta_1 & & \\ & \ddots & \\ & & \Delta_{n_{\mathcal{C}}} \end{bmatrix}$$

- Consistency Check: Use Linear-UHF to hash and reveal some rows to check $\mathcal{C}$-Δ-relations

Theorem 2. Protocol Π_{sVOLE} securely realizes $\mathcal{F}_{\mathsf{sVOLE}}$ with distinguishing advantage $\binom{n_{\mathcal{C}}}{k_{\mathcal{C}}+1} \cdot \varepsilon$

ZK for Polynomial Constraints Over **Small** Fields



The 3-Round Protocol

Protocol $\Pi^t_{\text{2D-Rep}}$

Parameters: Code $\mathcal{C}_{\text{Rep}} = [\tau, 1, \tau]_p$ with $\mathbf{G}_{\mathcal{C}} = (1 \dots 1) \in \mathbb{F}_p^{1\times\tau}$. VOLE size $q = p^r$.
Inputs: Polynomials $f_i \in \mathbb{F}_{p^k}[X_1, \dots, X_\ell]_{\leq 2}$, $i \in [t]$. The prover $\mathcal{P}$ also holds a witness $\mathbf{w} \in \mathbb{F}_p^\ell$ such that $f_i(\mathbf{w}) = 0$ for all $i \in [t]$.

Round 1. $\mathcal{P}$ does the following:

1. Call the functionality $\mathcal{F}_{\text{sVOLE}}^{p,q,S_\Delta,\mathcal{C}_{\text{Rep}},\ell+r\tau,\mathcal{L}}$ and receive $\mathbf{u} \in \mathbb{F}_p^{\ell+r\tau}, \mathbf{V} \in \mathbb{F}_q^{(\ell+r\tau)\times\tau}$.
   $\mathcal{V}$ receives done.
2. Compute $\mathbf{d} = \mathbf{w} - \mathbf{u}_{[1..\ell]} \in \mathbb{F}_p^\ell$ and send $\mathbf{d}$ to $\mathcal{V}$.
3. For $i \in [\ell + 1..\ell + r\tau]$, embed $u_i \hookrightarrow \mathbb{F}_{q^\tau}$.
   For $i \in [\ell + r\tau]$, lift $\mathbf{v}_i \in \mathbb{F}_q^\tau$ into $v_i \in \mathbb{F}_{q^\tau}$.
   For $i \in [\ell]$, also embed $w_i \hookrightarrow \mathbb{F}_{q^\tau}$.

Round 2. $\mathcal{V}$ sends challenges $\chi_i \in \mathbb{F}_{q^\tau}$, $i \in [t]$.

Round 3. $\mathcal{P}$ does the following:

1. For each $i \in [t]$, compute $A_{i,0}, A_{i,1} \in \mathbb{F}_{q^\tau}$ such that
$$c_i(Y) = \bar{f}_i(w_1, \dots, w_n) \cdot Y^2 + A_{i,1} \cdot Y + A_{i,0}.$$
2. Compute
$$u^* = \sum_{i\in[r\tau]} u_i X^{i-1} \qquad v^* = \sum_{i\in[r\tau]} v_i X^{i-1},$$
   where $\mathbb{F}_{q^\tau} \simeq \mathbb{F}_p[X]/F(X)$.
3. Compute $\tilde{b} = \sum_{i\in[t]} \chi_i \cdot A_{i,0} + v^* \in \mathbb{F}_{q^\tau}$ and $\tilde{a} = \sum_{i\in[t]} \chi_i \cdot A_{i,1} + u^* \in \mathbb{F}_{q^\tau}$ and send $(\tilde{a}, \tilde{b})$ to $\mathcal{V}$.

Verification. $\mathcal{V}$ runs the following check:

1. Call $\mathcal{F}_{\text{sVOLE}}^{p,q,S_\Delta,\mathcal{C}_{\text{Rep}},\ell+r\tau,\mathcal{L}}$ on input (get) and obtain $\boldsymbol{\Delta} \in \mathbb{F}_q^\tau$, $\mathbf{Q} \in \mathbb{F}_q^{(\ell+r\tau)\times\tau}$ such that $\mathbf{Q} = \mathbf{V} + \mathbf{u}^T \mathbf{G}_{\mathcal{C}} \mathsf{diag}(\boldsymbol{\Delta})$.
2. Compute $\mathbf{Q}' = \mathbf{Q}_{[1..\ell]} + \mathbf{d}^T \mathbf{G}_{\mathcal{C}} \mathsf{diag}(\boldsymbol{\Delta}) = \mathbf{V}_{[1..\ell]} + \mathbf{w}^T \mathbf{G}_{\mathcal{C}} \mathsf{diag}(\boldsymbol{\Delta})$.
3. Lift $\boldsymbol{\Delta}, \mathbf{q}'_1, \dots, \mathbf{q}'_\ell, \mathbf{q}_{\ell+1}, \dots, \mathbf{q}_{\ell+r\tau} \in \mathbb{F}_q^\tau$ into $\Delta, q'_1, \dots, q'_\ell, q_{\ell+1}, \dots, q_{\ell+r\tau} \in \mathbb{F}_{q^\tau}$.
4. For each $i \in [t]$, compute
$$c_i(\Delta) = \sum_{h\in[0,2]} \bar{f}_{i,h}(q'_1, \dots, q'_\ell) \cdot \Delta^{2-h}$$
5. Compute $q^* = \sum_{i\in[r\tau]} q_{\ell+i} \cdot X^{i-1}$ such that $q^* = v^* + u^*\Delta$.
6. Compute $\tilde{c} = \sum_{i\in[t]} \chi_i \cdot c_i(\Delta) + q^*$.
7. Check that $\tilde{c} \stackrel{?}{=} \tilde{a} \cdot \Delta + \tilde{b}$.

Theorem 4. The Protocol $\Pi^t_{\text{2D-Rep}}$ is a ZKPoK with soundness error $\frac{3}{p^{r\tau}}$.

How to Handle Arbitrary $\mathcal{C}$?

- For subspace VOLE with general code $[n_\mathcal{C}, k_\mathcal{C}, d_\mathcal{C}]$ and witness $\mathbf{w} = \mathbb{F}_p^{\ell \times k_\mathcal{C}}$
- The committed witness is as follows

$$\underbrace{\mathrm{B}}_{\ell \times n_\mathcal{C}} = \mathrm{V} - \underbrace{\mathrm{A}}_{k_\mathcal{C}} \times \underbrace{\mathrm{G}_\mathcal{C}}_{n_\mathcal{C}} \times \begin{pmatrix} \Delta_1 & & \\ & \ddots & \\ & & \Delta_{n_\mathcal{C}} \end{pmatrix}$$

Problem: Only row-wise linearity

In $Rep(\kappa)$, $k_\mathcal{C} = 1$

- Solution: Simulate VOLE in $\mathcal{P}$'s head once again

$\mathcal{P}$		$\mathcal{V}$
Prepares $(2\ell + 2)$ rows in A'		
$A' = \begin{bmatrix} A \\ B \end{bmatrix}$	DVZK π as if Δ' is the key $\longrightarrow$	
	$\longleftarrow$ Δ'	
	$V = B + \Delta' \cdot A$ $\longrightarrow$	

$\mathcal{P}$ and $\mathcal{V}$ continue the subspace VOLE simulation

$\mathcal{V}$ accepts if

- π is valid under Δ'
- The opening of V is correct under $\mathrm{diag}(\vec{\Delta})$

The Code-Switching Technique

Protocol $\Pi^t_{\mathsf{2D-LC}}$

The protocol is parameterized by an $[n_\mathcal{C}, k_\mathcal{C}, d_\mathcal{C}]_p$ linear code $\mathcal{C}$, set $S_\Delta \subset \mathbb{F}_p^{n_\mathcal{C}}$ and a leakage space $\mathcal{L}$ (used in $\mathcal{F}_{\mathsf{sVOLE}}$).

INPUTS: Both parties hold a set of polynomials $f_i \in \mathbb{F}_p[X_1, \ldots, X_\ell]_{\leq 2}$, $i \in [t]$. $\mathcal{P}$ also holds a witness $\mathbf{w} \in \mathbb{F}_p^{k_\mathcal{C}\ell}$ such that $f_i(\mathbf{w}) = 0$, for all $i \in [t]$.

Round 1. $\mathcal{P}$ does as follows:

1. $\mathcal{P}$ and $\mathcal{V}$ call $\mathcal{F}_{\mathsf{sVOLE}}^{p,p,S_\Delta,\mathcal{C},2\ell+1,\mathcal{L}}$, $\mathcal{P}$ receives $\mathbf{U} \in \mathbb{F}_p^{(2\ell+2)\times k_\mathcal{C}}$, $\mathbf{V} \in \mathbb{F}_p^{(2\ell+2)\times n_\mathcal{C}}$, while $\mathcal{V}$ gets the message done.
2. $\mathcal{P}$ sets $\mathbf{V}_1 = \mathbf{V}_{[1..\ell+1]}$, $\mathbf{V}_2 = \mathbf{V}_{[\ell+2..2\ell+2]}$ and $\mathbf{R} = \mathbf{U}_{[\ell+2..2\ell+2]}$
3. $\mathcal{P}$ commits to its witness by sending $\mathbf{D} = \mathbf{W} - \mathbf{U}_{[1..\ell]}$.

Round 2. $\mathcal{V}$ samples $\boldsymbol{\chi} \leftarrow \mathbb{F}_p^t$ and sends it to $\mathcal{P}$.

Round 3. $\mathcal{P}$ proceeds as follows.

1. For each $i \in [t]$, compute
$$g_i(Y) := \sum\nolimits_{h\in[0,2]} f_{i,h}(\mathbf{r}_1 + \mathbf{w}_1 \cdot Y, \ldots, \mathbf{r}_\ell + \mathbf{w}_\ell \cdot Y) \cdot Y^{2-h} = \sum\nolimits_{h\in[0,1]} A_{i,h} \cdot Y^h$$
2. Compute $\widetilde{\mathbf{b}} = \sum_{i\in[t]} \chi_i \cdot A_{i,0} + \mathbf{r}_{\ell+1}$ and $\widetilde{\mathbf{a}} = \sum_{i\in[t]} \chi_i \cdot A_{i,1} + \mathbf{u}_{1,\ell+1}$, where $\mathbf{u}_{1,i}$ is the ith row of $\mathbf{U}$.
3. Send $(\widetilde{\mathbf{b}}, \widetilde{\mathbf{a}})$ to $\mathcal{V}$.

Round 4. $\mathcal{V}$ samples $\Delta' \leftarrow \mathbb{F}_p$ and sends it to the prover.

Round 5. $\mathcal{P}$ sends $\mathbf{S} = \mathbf{R} + \mathbf{U}_{[1..\ell+1]} \cdot \Delta' \in \mathbb{F}_p^{(\ell+1)\times n_\mathcal{C}}$ to $\mathcal{V}$

Round 6. $\mathcal{V}$ samples $\boldsymbol{\eta} \leftarrow \mathbb{F}_p^{\ell+1}$ and sends it to $\mathcal{P}$

Round 7. $\mathcal{P}$ computes $\widetilde{\mathbf{v}} = \boldsymbol{\eta}^\top(\mathbf{V}_2 + \mathbf{V}_1 \cdot \Delta')$ and sends it to $\mathcal{V}$.

Verification. $\mathcal{V}$ runs the following checks.

1. *Check the constraints:*
   - Compute $\mathbf{S}' = \mathbf{S} + \begin{bmatrix}\mathbf{D}\\ 0\end{bmatrix} \cdot \Delta' = \mathbf{R} + \begin{bmatrix}\mathbf{W}\\ \mathbf{u}_{\ell+1}\end{bmatrix} \cdot \Delta'$.
   - For each $i \in [t]$, compute
   $$\mathbf{c}_i(Y) = \sum_{h\in[0,2]} f_{i,h}(\mathbf{s}'_1, \ldots, \mathbf{s}'_\ell) \cdot Y^{2-h}.$$
   - Let $\widetilde{\mathbf{s}} = \sum_{i\in[t]} \chi_i \cdot \mathbf{c}_i(\Delta') + \mathbf{s}'_{\ell+1}$.
   - Check that $\widetilde{\mathbf{s}} = \widetilde{\mathbf{b}} + \widetilde{\mathbf{a}} \cdot \Delta'$.
2. *Check the opening of* $\mathbf{S}$*:*
   - Call $\mathcal{F}_{\mathsf{sVOLE}}^{p,p,S_\Delta,\mathcal{C},2\ell+1,\mathcal{L}}$ on input (get) and obtain $\boldsymbol{\Delta} \in \mathbb{F}_p^{n_\mathcal{C}}$ and $\mathbf{Q} \in \mathbb{F}_p^{(2\ell+2)\times n_\mathcal{C}}$ such that $\mathbf{Q} = \mathbf{V} + \mathcal{C}(\mathbf{U}) \cdot \mathsf{diag}(\boldsymbol{\Delta})$
   - Set $\mathbf{Q}_1 = \mathbf{Q}_{[1..\ell+1]}$ and $\mathbf{Q}_2 = \mathbf{Q}_{[\ell+2..2\ell+2]}$.
   - Check that
   $$\boldsymbol{\eta}^\top(\mathbf{Q}_2 + \mathbf{Q}_1 \cdot \Delta') = \widetilde{\mathbf{v}} + \boldsymbol{\eta}^\top \cdot \mathcal{C}(\mathbf{S}) \cdot \mathsf{diag}(\boldsymbol{\Delta})$$

Theorem 3. The protocol $\Pi^t_{\text{2D-LC}}$ is a SHVZKPoK with soundness error $\frac{3}{p} + 2|S_\Delta|^{-d_\mathcal{C}}$ in the $\mathcal{F}_{\mathsf{sVOLE}}^{p,S_\Delta,\mathcal{C},2(\ell+2),\mathcal{L}}$-hybrid model

The Problem with LPN-based State-of-the-Art

Sender/$\mathcal{P}$		Receiver/$\mathcal{V}$
(k_0^1, k_1^1)		$\Delta_B[1]$
...	$\rightarrow \boxed{\mathcal{F}_{\mathsf{OT}}} \leftarrow$	...
(k_0^ℓ, k_1^ℓ)		$\Delta_B[\ell]$

The Problem with LPN-based State-of-the-Art

Sender/$\mathcal{P}$ Receiver/$\mathcal{V}$

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

$$\dots \longrightarrow \boxed{\mathcal{F}_{\mathsf{OT}}} \longleftarrow \dots$$

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

Extend (u)

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

$$\dots$$

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

$$u\Delta_B[i] =$$

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

Use LHL to remove selective failure leackage on Δ

The Problem with LPN-based State-of-the-Art

Sender/$\mathcal{P}$ Receiver/$\mathcal{V}$

$$\begin{array}{ccc} (k_0^1, k_1^1) & & \Delta_B[1] \\ \dots & \rightarrow \boxed{\mathcal{F}_{\mathsf{OT}}} \leftarrow & \dots \\ (k_0^\ell, k_1^\ell) & & \Delta_B[\ell] \end{array}$$

Extend (u)

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

$$\dots$$

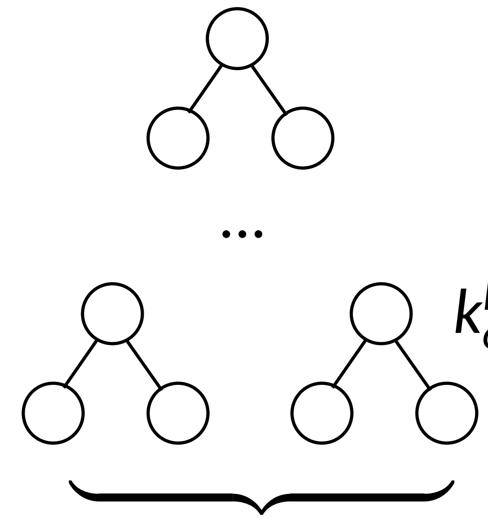
$$m_\ell := \mathsf{PRF}(k_0^\ell, j) + \mathsf{PRF}(k_1^\ell, j) + u$$

$$\longrightarrow$$

$$u\Delta_B[i] =$$

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

Receiver/$\mathcal{V}$ Sender/$\mathcal{P}$



$\mathbf{v}$

k_o^1, k_e^1

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

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

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

k_o^h, k_e^h

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

$$\longrightarrow$$

Use LHL to remove selective failure leackage on Δ

The Problem with LPN-based State-of-the-Art

Sender/$\mathcal{P}$ Receiver/$\mathcal{V}$

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

$$\ldots \qquad \ldots$$

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

Extend (u)

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

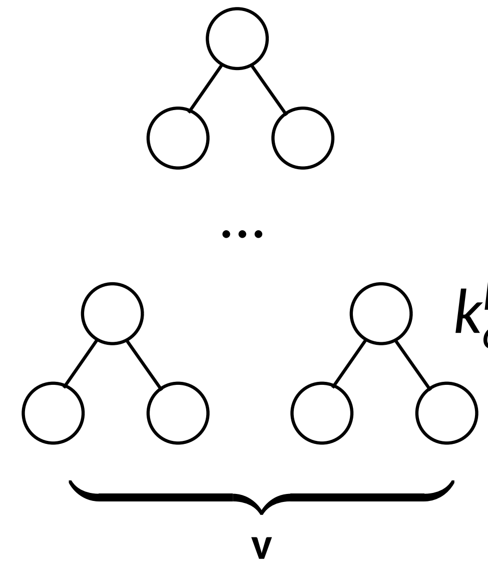
$$\ldots$$

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

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

Use LHL to remove selective failure leackage on Δ

Receiver/$\mathcal{V}$ Sender/$\mathcal{P}$



k_o^1, k_e^1

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

k_o^h, k_e^h

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

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

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

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

Recover $\mathbf{b}_\alpha$

The Problem with LPN-based State-of-the-Art

Sender/$\mathcal{P}$ Receiver/$\mathcal{V}$

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

Extend (u)

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

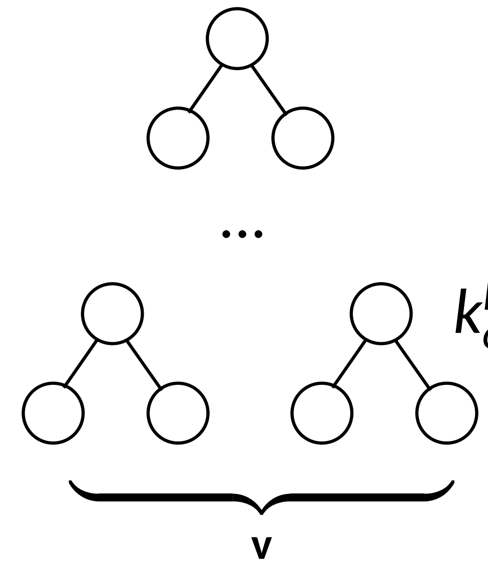
$$\cdots$$

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

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

Use LHL to remove selective failure leackage on Δ

Receiver/$\mathcal{V}$ Sender/$\mathcal{P}$



k_o^1, k_e^1

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

k_o^h, k_e^h

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

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

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

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

Recover $\mathbf{b}_\alpha$

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

Com&Open doesn't work when $\mathcal{P}$ is OT receiver

FAEST Signature

- Apply FS transform to $\Pi^{t}_{\text{2D-LC}}$ scheme
- $\text{pk} = x, y \in \mathbb{F}_2^{128}$, $\text{sk} = k \in \mathbb{F}_2^{128}$
- Relation: $y = \text{Enc}_k(x)$
- For AES128, S-box is $\mathbb{F}_{2^8}$ inversion, so we can use 2D polynomial to express it

Theorem 5. *The Π_{FAEST} protocol, defined as*

$$\Pi_{\mathsf{FAEST}} = \mathsf{FS}^{H_{\mathsf{FS}}}[\mathsf{O2C}^{H_{\mathsf{O2C}}}[\Pi_{\textit{2D-Rep-OT}}]],$$

is a zero-knowledge non-interactive proof system in the CRS+RO model with knowledge error

$$2 \cdot (Q_{\mathsf{FS}} + Q_{\mathsf{Verify}}) \cdot \frac{2}{p^{r\tau}} + M \cdot (Q_{\mathsf{FS}} + Q_{\mathsf{Verify}}) \cdot \mathsf{AdvEB}^{\mathsf{VC}}_{\mathcal{A}'}[Q_{H_{\mathsf{O2C}}}] + \mathsf{AdvDist}^{\mathsf{VC.Setup},\mathsf{VC.TSetup}}_{\mathcal{D}},$$

where M is an upper bound on the number of VC *commitments sent during a run of* $\mathsf{O2C}[\Pi_{\textit{2D-Rep-OT}}]$.

Claimed Performance of FAEST

Scheme	$t_{\mathcal{P}}$ (ms)	$t_{\mathcal{V}}$ (ms)	|sign| (B)	Assumption
SDitH [FJR22b] (fast)	13.40	12.70	17 866	SD $\mathbb{F}_2$
SDitH [FJR22b] (short)	64.20	60.70	12 102	SD $\mathbb{F}_2$
SDitH [FJR22b] (fast)	6.40	5.90	12 115	SD $\mathbb{F}_{256}$
SDitH [FJR22b] (short)	29.50	27.10	8 481	SD $\mathbb{F}_{256}$
Rainier$_3$ [DKR+22]	2.96	2.92	6 176	RAIN$_3$
Rainier$_4$ [DKR+22]	3.47	3.42	6 816	RAIN$_4$
Limbo [dOT21] (fast)	2.61	2.25	23 264	Hash
Limbo [dOT21] (short)	24.51	21.82	13 316	Hash
SPHINCS+-SHA2 [HBD+22] (fast)	4.40	0.40	17 088	Hash
SPHINCS+-SHA2 [HBD+22] (short)	88.21	0.15	7 856	Hash
Falcon-512 [PFH+22]	0.11	0.02	666	Lattice
Dilithium2 [LDK+22]	0.07	0.03	2 420	Lattice
FAEST (**this work**, fast, $q = 2^8$)	2.28	2.11	6 583	Hash
FAEST (**this work**, short, $q = 2^{11}$)	11.05	10.18	5 559	Hash