More Efficient Zero-Knowledge Protocols over $\mathbb{Z}_{2^k}$ via Galois Rings

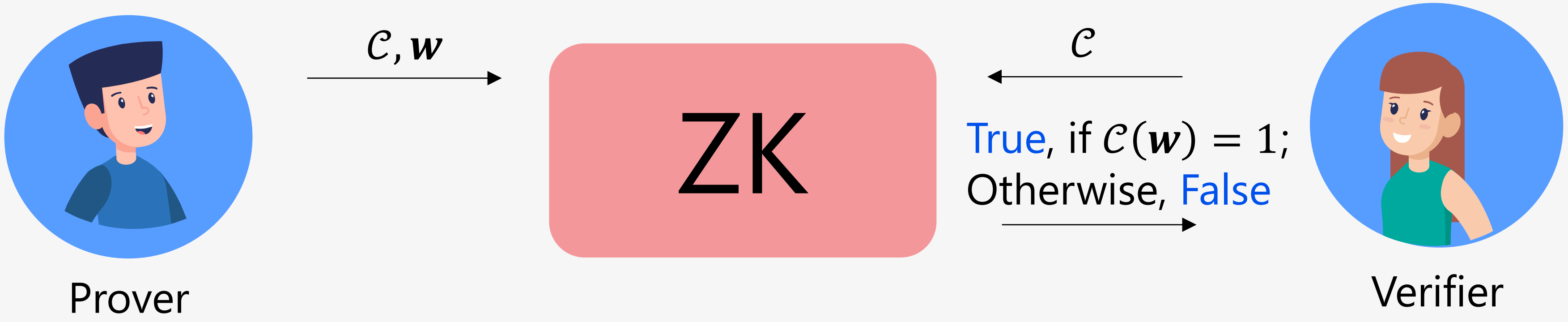
Fuchun Lin, Chaoping Xing, and Yizhou Yao

Shanghai Jiao Tong University

22/8/2024 – Crypto2024

饮 水 思 源 • 爱 国 荣 校

Zero-Knowledge Proofs for Circuit Satisfiability



Completeness: Verifier always accepts a valid proof.

Knowledge Soundness: if Verifier accepts a proof, then Prover must know a valid witness $\boldsymbol{w}$.

Zero-Knowledge: Verifier learns nothing about $\boldsymbol{w}$ except $\mathcal{C}(\boldsymbol{w}) = 1$.

In particular, we consider $\mathcal{C}$ is over rings $\mathbb{Z}_{2^k}$, e.g. $\mathbb{Z}_{2^{32}}$, $\mathbb{Z}_{2^{64}}$.

Computations over $\mathbb{Z}_{2^k}$ vs. $\mathbb{F}_{p^k}$

Typical advantages of ZK protocols for $\mathbb{Z}_{2^k}$

1. $\mathbb{Z}_{2^k}$ is more compatible with binary operations.

2. Easier to convert computer programs to circuits, avoiding the efficiency gap of emulating $\mathbb{Z}_{2^k}$ computations by large field operations.

But "Bad" for protocol designers!

1. A half of zero-divisors.

2. At most two points can be used for Lagrange interpolation.

3. More complicated security analysis.

Computations over $\mathbb{Z}_{2^k}$ vs. $\mathbb{F}_{p^k}$

Typical advantages of ZK protocols for $\mathbb{Z}_{2^k}$

1. $\mathbb{Z}_{2^k}$ is more compatible with binary operations.
2. Easier to convert computer programs to circuits, avoiding the efficiency gap of emulating $\mathbb{Z}_{2^k}$ computations by large field operations.

But "Bad" for protocol designers!

1. A half of zero-divisors.
2. At most two points can be used for Lagrange interpolation.
3. More complicated security analysis.

There are numerous efficient ZK protocols for fields, e.g., Pinocchio[PHGR16], Limbo[DOT21], Virgo[ZXZS20], Marlin[CHM+20], Breakdown[GLS+23], Quicksilver[YSWW21], AntMan[WYY+22]……

But current state of ZK protocols over rings $\mathbb{Z}_{2^k}$ leaves too much to be desired. Rinocchio[GNS23], ZK-for-Z2K[BDJ+23], A2B[BBM+21], MozZarella[BBMS22].

Our Contributions

1. A more efficient ZK protocols for $\mathbb{Z}_{2^k}$ compared to the state-of-the-art MozZarella [BBMS22]

-Optimal $O(1)$ computational overhead, communication of $1.5-3$ elements of $\mathbb{Z}_{2^k}$ per gate, constant-round, small memory (streaming Prover & Verifier), public-coin, UC-security.

-Compatible with the VOLEitH [BBD+23] technique, yielding publicly verifiable NIZK for $\mathbb{Z}_{2^k}$.

k	κ	Moz$\mathbb{Z}_{2^k}$arella		This work ($\Pi_{\mathrm{ZK}}^{m,n,t}$)	
		Comm.	$\mathbb{R}$	Comm.	$\mathbb{R}$
32	40	179	$\mathbb{Z}_{2^{130}}$	93	$\mathtt{GR}(2^{32}, 45)$
	80	302	$\mathbb{Z}_{2^{212}}$	104	$\mathtt{GR}(2^{32}, 85)$
64	40	211	$\mathbb{Z}_{2^{162}}$	183	$\mathtt{GR}(2^{64}, 45)$
	80	334	$\mathbb{Z}_{2^{244}}$	205	$\mathtt{GR}(2^{64}, 85)$

Our Contributions

1. A more efficient ZK protocols for $\mathbb{Z}_{2^k}$ compared to the state-of-the-art MozZarella [BBMS22]

-Optimal $O(1)$ computational overhead, communication of $1.5-3$ elements of $\mathbb{Z}_{2^k}$ per gate, constant-round, small memory (streaming Prover & Verifier), public-coin, UC-security.

-Compatible with the VOLEitH [BBD+23] technique, yielding publicly verifiable NIZK for $\mathbb{Z}_{2^k}$.

k	κ	Moz$\mathbb{Z}_{2^k}$arella		This work ($\Pi_{\mathrm{ZK}}^{m,n,t}$)	
		Comm.	$\mathbb{R}$	Comm.	$\mathbb{R}$
32	40	179	$\mathbb{Z}_{2^{130}}$	93	$\mathtt{GR}(2^{32}, 45)$
	80	302	$\mathbb{Z}_{2^{212}}$	104	$\mathtt{GR}(2^{32}, 85)$
64	40	211	$\mathbb{Z}_{2^{162}}$	183	$\mathtt{GR}(2^{64}, 45)$
	80	334	$\mathbb{Z}_{2^{244}}$	205	$\mathtt{GR}(2^{64}, 85)$

2. A designated verifier ZK for $\mathbb{Z}_{2^k}$ with *sublinear* communication and *quasilinear* computation.

Recap: VOLE-ZK

Vector-OLE based Zero-Knowledge Proof

Linearly homomorphic commitment from VOLE:

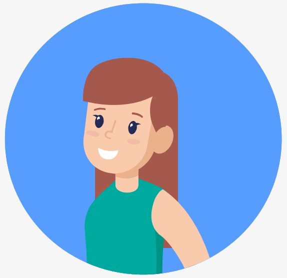
Sender

$\mathbf{M}_\mathrm{x}, \mathbf{x} \in R^n$ →

VOLE

← $\Delta \in R$

$\mathbf{K}_\mathrm{x} = \Delta\, \mathbf{x} + \mathbf{M}_\mathrm{x}$ →



Receiver

MAC tags $\mathbf{M}_\mathrm{x}$ and values $\mathbf{x}$

$[\mathbf{x}]$

MAC keys $\mathbf{K}_\mathrm{x}$ and global key Δ

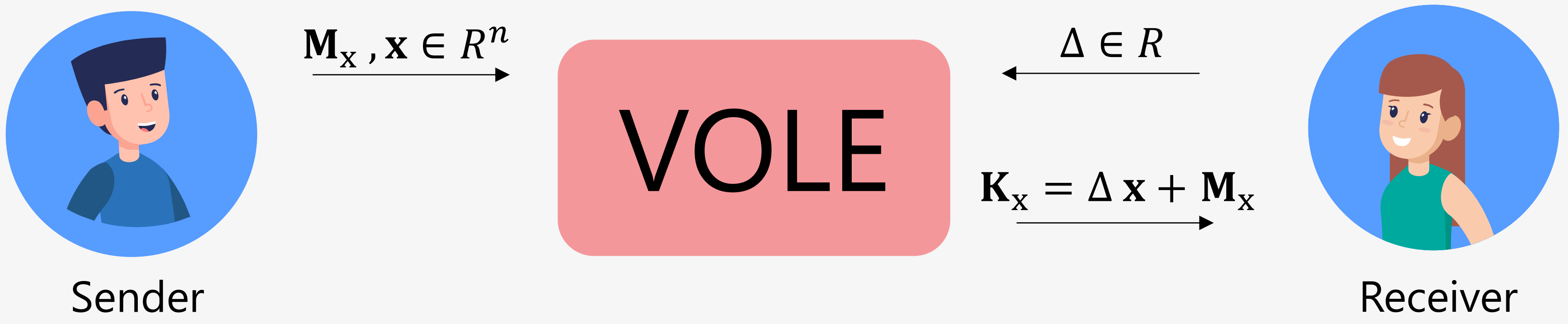
Linear homomorphism:

Given $[\mathbf{x}]$, $[\mathbf{y}]$, then $[\mathbf{z}] \coloneqq [\alpha\mathbf{x} + \mathbf{y}]$ is obtained by

$$\underbrace{(\alpha\mathbf{K}_\mathrm{x} + \mathbf{K}_\mathrm{y})}_{\mathbf{K}_\mathrm{z}} = \Delta \cdot \underbrace{(\alpha\mathbf{x} + \mathbf{y})}_{\mathbf{z}} + \underbrace{(\alpha\mathbf{M}_\mathrm{x} + \mathbf{M}_\mathrm{y})}_{\mathbf{M}_\mathrm{z}}$$

Vector-OLE based Zero-Knowledge Proof

Linearly homomorphic commitment from VOLE:



MAC tags $\mathbf{M}_{\mathrm{x}}$ and values $\mathbf{x}$ $[\mathbf{x}]$ MAC keys $\mathbf{K}_{\mathrm{x}}$ and global key Δ

"Commit-and-prove" paradigm:

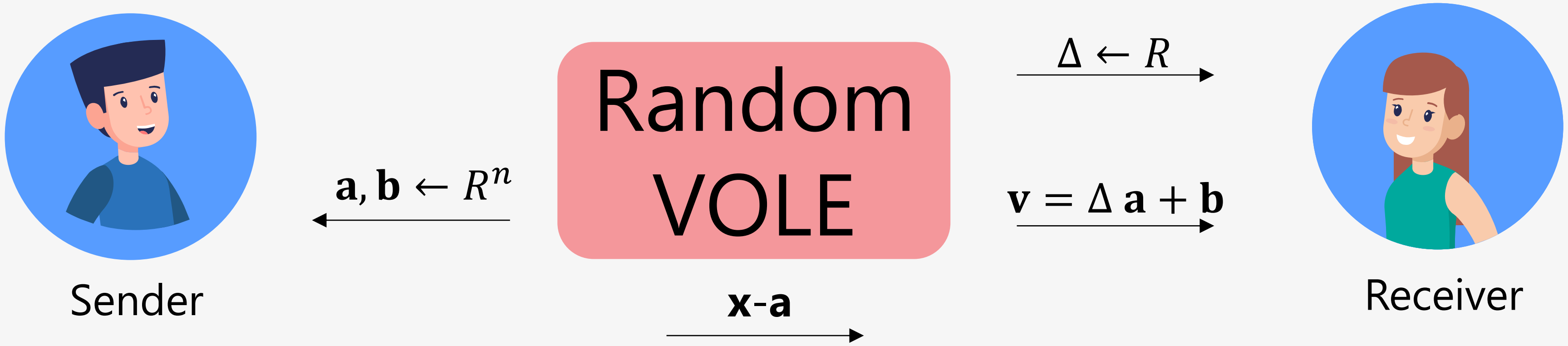
1. Prover first commits all intermediate wire values via VOLE.
2. Then proves to Verifier values underneath the commitments satisfy the circuit topology.

What we need:

Procedures for *Open*, *CheckZero*, *CheckMultiplication*.

Vector-OLE based Zero-Knowledge Proof

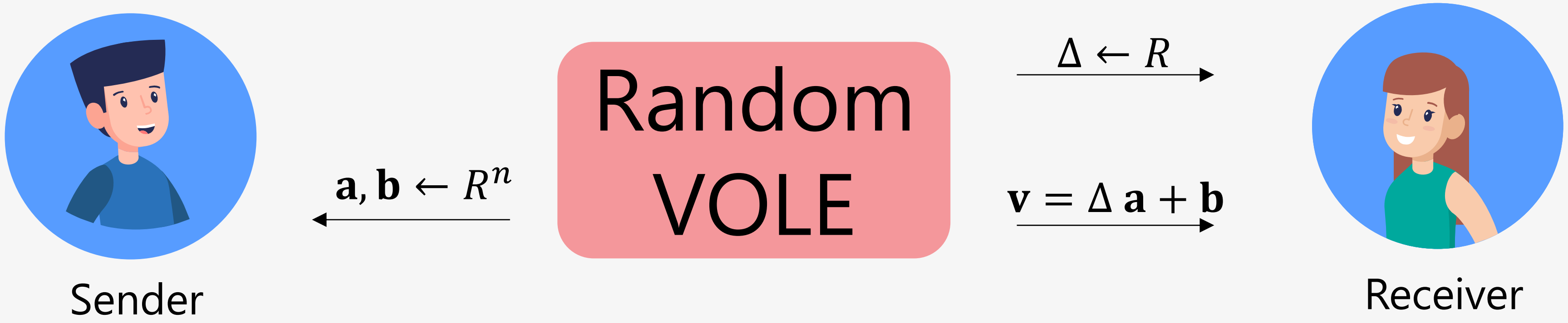
Reduction of chosen input VOLE to random VOLE:



$$\underbrace{\mathbf{v} + \Delta \cdot (\mathbf{x} - \mathbf{a})}_{\mathbf{K_x}} = \Delta \cdot \underbrace{(\mathbf{a} + \mathbf{x} - \mathbf{a})}_{\mathbf{x}} + \underbrace{(\mathbf{b})}_{\mathbf{M_x}}$$

VOLE-ZK: from MPC view

Offline-phase: generate random VOLE correlation.

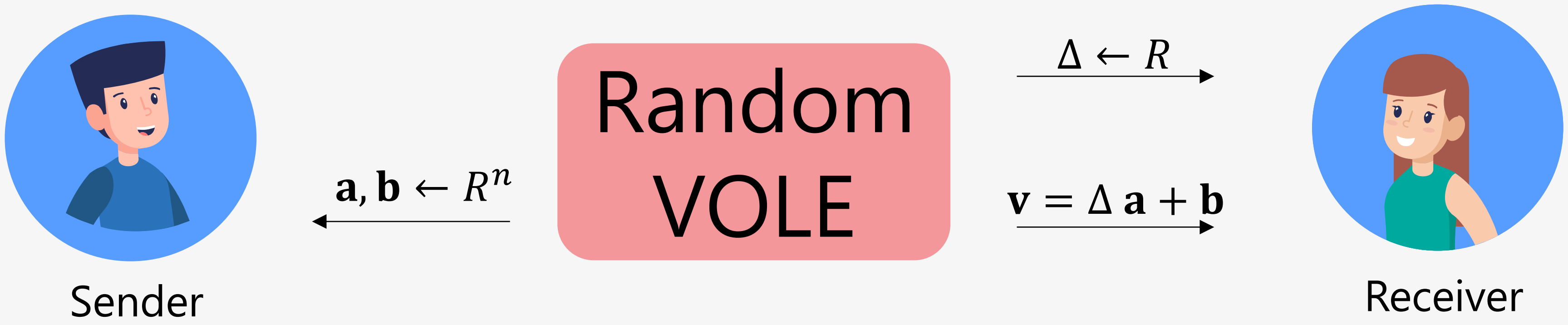


Online-phase:

They collectively evaluate the circuit in an authenticated way, consuming random VOLE.

VOLE-ZK: from MPC view

Offline-phase: generate random VOLE correlation.



Online-phase:

They collectively evaluate the circuit in an authenticated way, consuming random VOLE.

Things are easier compared to general MPC:

Additions are *free* to evaluate.

It suffices to *verify* multiplications, rather than to evaluate them.

Methodology of our Approach

Inspired by [EXY22], to construct MPC/ZK for $\mathbb{F}_p/\mathbb{Z}_{2^k}$,

1. Realize a MPC/ZK protocol over its extension (i.e., Galois field/ring).
2. Transform it into a protocol for $\mathbb{F}_p/\mathbb{Z}_{2^k}$ via *reverse multiplicative friendly embedding (RMFE)*.
3. Deal with the case that malicious parties can deviate from RMFE encoding.

	[EXY22]	This work	MozZarella
Setting	n-party MPC	2-party ZK	2-party ZK
Corruption	Dishonest majority	Dishonest majority	Dishonest majority
SS Scheme	SPDZ-like	VOLE	VOLE
Technique	Quintuple	Re-embedding pair	SPD$\mathbb{Z}_{2^k}$-VOLE
Ring	$\mathrm{GR}(2^k, d)$	$\mathrm{GR}(2^k, d)$	$\mathbb{Z}_{2^{k+s}}$

Technique Details

Galois Ring

Definition (Galois ring)

Let p be a prime, and $k, d \geq 1$ be integers. Let $f(X) \in \mathbb{Z}_{p^k}[X]$ be a monic polynomial of degree d such that $\overline{f(X)} := f(X) \mod p$ is irreducible over $\mathbb{F}_p$. A Galois ring over $\mathbb{Z}_{p^k}$ of degree d denoted by $\mathtt{GR}(p^k, d)$ is a ring extension $\mathbb{Z}_{p^k}[X]/(f(X))$ of $\mathbb{Z}_{p^k}$.

Basic algebraic properties:

1. if $\mathrm{d} = 1, \mathrm{GR}(p^k, \mathrm{d}) = \mathbb{Z}_{\mathrm{p^k}}$; if $\mathrm{k} = 1$, $\mathrm{GR}(p^k, \mathrm{d}) = \mathbb{F}_{\mathrm{p^d}}$

2. $\mathrm{GR}(p^k, \mathrm{d})/(\mathrm{p}) \cong \mathbb{F}_{\mathrm{p^d}}$

3. "Schwartz-Zipple" Lemma for Galois ring:

for any non-zero degree- r polynomial $f(x)$ over $\mathrm{GR}(p^k, \mathrm{d})$,

$$\Pr[f(\alpha) = 0 | \alpha \leftarrow \mathrm{GR}(p^k, \mathrm{d})] \leq rp^{-d}$$

Reverse Multiplicative Friendly Embedding

Definition (RMFE)

Let p be a prime, $k, r, m, d \geq 1$ be integers. A pair (ϕ, ψ) is called an (m, d)-RMFE over $\mathrm{GR}(p^k, r)$ if $\phi : \mathrm{GR}(p^k, r)^m \to \mathrm{GR}(p^k, rd)$ and $\psi : \mathrm{GR}(p^k, rd) \to \mathrm{GR}(p^k, r)^m$ are two $\mathrm{GR}(p^k, r)$-linear maps such that

$$\psi(\phi(\mathbf{x}_1) \cdot \phi(\mathbf{x}_2)) = \mathbf{x}_1 * \mathbf{x}_2 \quad (1)$$

for all $\mathbf{x}_1, \mathbf{x}_2 \in \mathrm{GR}(p^k, r)^m$, where $*$ denotes the entry-wise multiplication.

Nice properties [CCXY18,CRX21,EHL+23]:

1. We can assume $\phi(\mathbf{1}) = 1$.
2. Given an RMFE (ϕ, ψ) with $\phi(\mathbf{1}) = 1$, we have $\mathrm{GR}(p^k, \mathrm{rd}) = \mathrm{Ker}(\psi) \oplus \mathrm{Im}(\phi)$.
3. There exists a family of (m, d) RMFEs over $\mathbb{Z}_{2^\mathrm{k}}$ for all $\mathrm{k} \geq 1$ with $\lim_{m\to\infty} \frac{d}{m} = 4.92$.

Technical Overview

1. Construct a Galois ring analogue of Quicksilver [YSWW21].

2. Convert it to a ZK protocol for $\mathbb{Z}_{2^k}$ via RMFE:

   i). Suppose all MACed values are in $\mathrm{Im}(\phi)$.

   ii). It is reduced to a verification problem in the ZK setting, i.e., give $[\mathrm{x}], [\mathrm{y}], [z]$ related to a multiplication gate, check $\psi(\mathrm{x} \cdot \mathrm{y}) = \psi(\mathrm{z}) \Leftrightarrow \boldsymbol{x} * \boldsymbol{y} = \boldsymbol{z}$, where $\mathrm{x} = \phi(\boldsymbol{x})$, $\mathrm{y} = \phi(\boldsymbol{y})$, $\mathrm{z} = \phi(\boldsymbol{z})$.

   iii). Design an efficient approach to check the above relation. ★ **Multiplication Check**

3. Design an efficient approach that guarantees all MACed values are in Im(ϕ).

   ★ **Image Check**

Technical Overview

Main Obstacle: RMFE only preserves one time multiplication.

Concretely, $z = x \cdot y$ might not belong to $\mathrm{Im}(\phi)$, for $x, y \in \mathrm{Im}(\phi)$.

Our Observation: $\mathrm{GR}(2^{\mathrm{k}}, d) = \mathrm{Ker}(\psi) \oplus \mathrm{Im}(\phi)$, and ψ: $\mathrm{Im}(\phi) \to \mathbb{Z}_{2^k}^m$ is a bijection.

Given $[z]$, $z \in \mathrm{GR}(2^{\mathrm{k}}, d)$, we can

i). Re-embed $[z]$ to $[\tau(z)]$, where $\tau := \phi \circ \psi$, $[\tau(z)] \coloneqq [z] + \tau(z) - z$, by sending $\tau(z) - z$ to Verifier.

Main Obstacle: RMFE only preserves one time multiplication.

Concretely, $z = x \cdot y$ might not belong to $\mathrm{Im}(\phi)$, for $x, y \in \mathrm{Im}(\phi)$.

Our Observation: $\mathrm{GR}(2^{\mathrm{k}}, d) = \mathrm{Ker}(\psi) \oplus \mathrm{Im}(\phi)$, and $\psi\colon \mathrm{Im}(\phi) \rightarrow \mathbb{Z}_{2^k}^m$ is a bijection.

Given $[z]$, $z \in \mathrm{GR}(2^{\mathrm{k}}, d)$, we can

i). Re-embed $[z]$ to $[\tau(z)]$, where $\tau := \phi \circ \psi$, $[\tau(z)] \coloneqq [z] + \tau(z) - z$, by sending $\tau(z) - z$ to Verifier.

ii). However, this might leak information about x, y.

iii). To this end, we introduce re-embedding pairs: $([\mathrm{a}], [\tau(a)])$.

So that $[\tau(z)] \coloneqq [\tau(a)] + \tau(z - a)$.

Re-embedding pair: Image Check

Guarantee a MACed x belongs to $\mathrm{Im}(\phi)$, via re-embedding pair:

Goal: given re-embedding pairs $([a], [\tau(a)])$, obtain $[x]$ with $x \in \mathrm{Im}(\phi)$.

Our observation: let $\delta \coloneqq x - a$,

$$x \in \mathrm{Im}(\phi) \Leftrightarrow x = \tau(x) \Leftrightarrow a + \delta = \tau(a) + \tau(\delta)$$

Protocol specification:

1. Prover sends $\delta \coloneqq x - a$ to Verifier.
2. Prover and Verifier compute $[\tau(x)] \coloneqq [\tau(a)] + \tau(x - a)$.

Re-embedding pair: Multiplication Check

Verify Multiplication gates: our observation

Goal: given $[x], [y], [z]$, check $\psi(x) * \psi(y) = \psi(z)$, where $x, y, z \in \mathrm{Im}(\phi)$.

It is equivalent to check $\tau(x \cdot y) = z$.

Goal': given $[x], [y], [z']$, check $x \cdot y = z'$, and re-embed z' to $z \coloneqq \tau(z')$.

Re-embedding pair: Multiplication Check

Evaluate & Verify Multiplication gates via re-embedding pair:

Goal: given $[x], [y], [a], [\tau(a)]$, obtain $[\tau(z)]$, such that $z = x \cdot y$, where $x, y \in \mathrm{Im}(\phi)$.

We incorporate re-embedding pair with the check mechanism of QuickSilver [YSWW21].

Protocol specification:

1. Prover sends $\delta := x \cdot y - a$ to Verifier.
2. They compute $[\mathrm{z}] := [\mathrm{a}] + \delta$ and $[\tau(z)] := [\tau(a)] + \tau(\delta)$.
3. They check the following:

$$
\begin{aligned}
B &:= K_x \cdot K_y - K_z \cdot \Delta \\
&= (M_x + \Delta \cdot x)(M_y + \Delta \cdot y) - (M_z + z \cdot \Delta) \cdot \Delta \\
&= \underbrace{(M_x \cdot M_y)}_{A_0} + \underbrace{(x \cdot M_y + y \cdot M_x - M_z)}_{A_1} \cdot \Delta + \underbrace{(x \cdot y - z)}_{0} \cdot \Delta^2
\end{aligned}
$$

Soundness: $2/2^d$.

Offline phase: preparing re-embedding pairs

A construction via "construct and sacrifice".

-First construct $n+\kappa$ re-embedding pairs.

-Then sacrifice last κ pairs through masking random linear combinations of first n pairs.

Communication: $n+\kappa$ Galois ring elements in addition to preparing random VOLE.

Soundness: $\frac{1}{2^\kappa}+\frac{1}{2^{\mathrm{d}}}$.

Offline phase: VOLE instantiations

1. PCG instantiations from primal-LPN and dual-LPN.

-We adapt constructions from Wolverine [WYKW21] and LPZK [DIO21].

-We analyze security of LPN over Galois ring via approaches of [LWYY24].

2. A SoftSpokenOT [Roy22]-like instantiation from (N-1)-out-of-N OT.

-So that we can apply VOLEitH [BBD+23] to make it a publicly verifiable NIZK.

-A naïve adaption:

$$\underbrace{\sum_{y \in GR \backslash \{\Delta\}} \boldsymbol{s}_y \cdot (\Delta - y)}_{Verifier} = \sum_{y \in GR} \boldsymbol{s}_y \cdot (\Delta - y) = (\underbrace{\sum_{y \in GR} \boldsymbol{s}_y}_{Prover}) \cdot \Delta - \underbrace{\sum_{y \in GR} \boldsymbol{s}_y \cdot y}_{Prover}$$

Computation: $O\left(2^{kd}\right)$ Galois ring operations!

Offline phase: VOLE instantiations

2. A SoftSpokenOT [Roy22]-like instantiation from (N-1)-out-of-N OT.

-So that we can apply VOLEitH [BBD+23] to make it a publicly verifiable NIZK.

-Our optimization:

$$\mathbf{K} = \sum_{y \in \mathbb{F}_{2^d} \setminus \{\Delta\}} \boldsymbol{s}_y \cdot (\Delta - y) = \Big(\sum_{y \in \mathbb{F}_{2^d}} \boldsymbol{s}_y\Big) \cdot \Delta - \sum_{y \in \mathbb{F}_{2^d}} \boldsymbol{s}_y \cdot y = \mathbf{x} \cdot \Delta + \mathbf{M}$$

Computation: $O(2^d)$ cheaper Galois ring operations.

-Concrete parameter choice: For 80-bit security, we set $d = 15$, and repeat online phase protocol 6 times, where we can use a (6,15)-RMFE.

Open Problems & Future Work

1. How to achieve ZK with *linear prover* computation and *sublinear communication* for $\mathbb{Z}_{2^k}$?

2. How to achieve VOLE-based ZK with *sublinear communication* and *sublinear online verifier computation*?

Fast Implementation.

-Computations over Galois ring: lack of hardware/algorithm optimizations, e.g., inverse algorithms.

上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

Thank You

饮 水 思 源　爱 国 荣 校