Low-Bandwidth Mixed Arithmetic in VOLE-Based ZK from Low-Degree PRGs

Amit Agarwal[1] Carsten Baum[2] Lennart Braun[3] Peter Scholl[4]

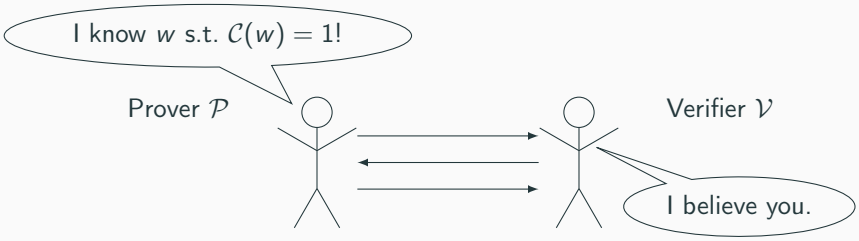
[1]University of Illinois Urbana-Champaign
[2]Technical University of Denmark
[3]Université Paris Cité, CNRS, IRIF
[4]Aarhus University

May 8, 2025 @ Eurocrypt 2025

INSTITUT DE RECHERCHE EN INFORMATIQUE FONDAMENTALE

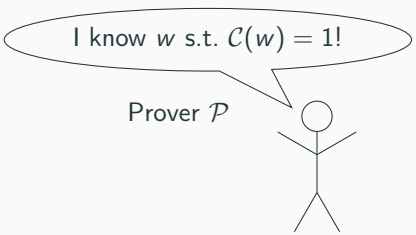
AARHUS UNIVERSITY

Zero-Knowledge Proofs (of Knowledge)

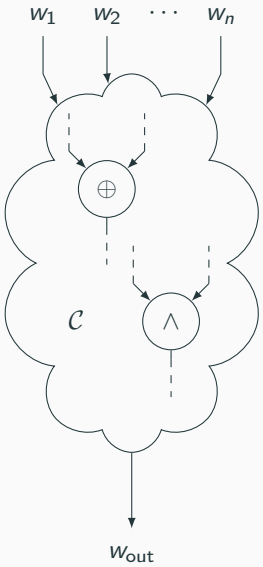


- Security Properties:
  - Soundness
  - Zero-Knowledge

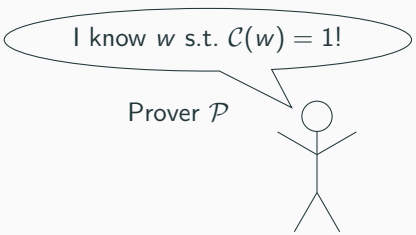
Zero-Knowledge Proofs (of Knowledge) for Circuits



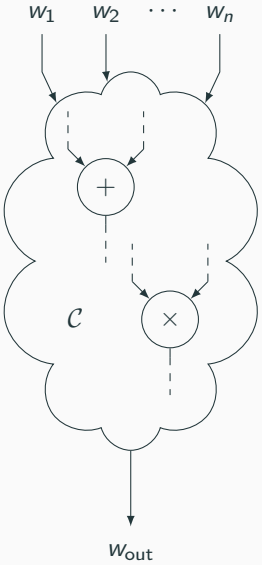
- Security Properties:
  - Soundness
  - Zero-Knowledge
- $\mathcal{C}$ is a circuit
  - Boolean over $\mathbb{F}_2 = \{0, 1\}$



Zero-Knowledge Proofs (of Knowledge) for Circuits



- Security Properties:
  - Soundness
  - Zero-Knowledge
- $\mathcal{C}$ is a circuit
  - Boolean over $\mathbb{F}_2 = \{0, 1\}$
  - arithmetic over a larger ring or field



Hybrid Circuits and Mixed Arithmetic

Boolean and arithmetic circuits have different strengths:

- Boolean over $\mathbb{F}_2$: e.g., simple comparisons
- arithmetic over $\mathbb{F}_p$: cheap addition/multiplication

$\Longrightarrow$ combine both into a hybrid circuit

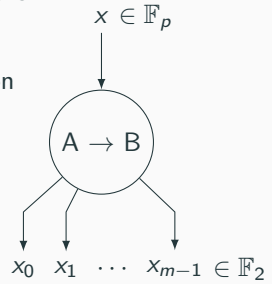
Hybrid Circuits and Mixed Arithmetic

Boolean and arithmetic circuits have different strengths:

- Boolean over $\mathbb{F}_2$: e.g., simple comparisons
- arithmetic over $\mathbb{F}_p$: cheap addition/multiplication

$\Longrightarrow$ combine both into a hybrid circuit

Conversion Gates



such that $x = \sum_{i=0}^{m-1} 2^i \cdot x_i$

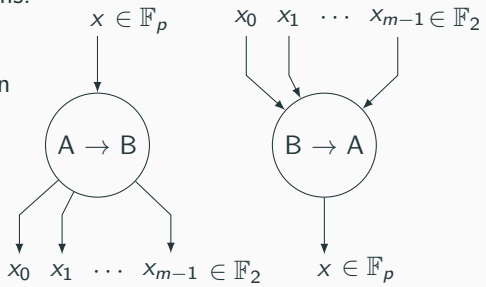
Hybrid Circuits and Mixed Arithmetic

Boolean and arithmetic circuits have different strengths:

- Boolean over $\mathbb{F}_2$: e.g., simple comparisons
- arithmetic over $\mathbb{F}_p$: cheap addition/multiplication

$\Longrightarrow$ combine both into a hybrid circuit

Conversion Gates



such that $x = \sum_{i=0}^{m-1} 2^i \cdot x_i$

Hybrid Circuits and Mixed Arithmetic

Boolean and arithmetic circuits have different strengths:

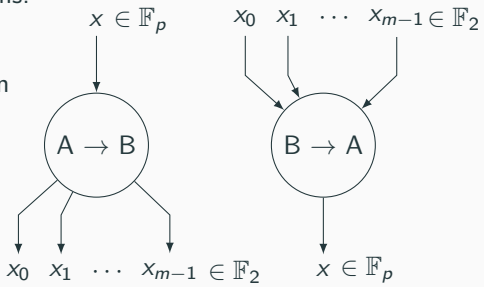
- Boolean over $\mathbb{F}_2$: e.g., simple comparisons
- arithmetic over $\mathbb{F}_p$: cheap addition/multiplication

$\Longrightarrow$ combine both into a hybrid circuit

More Gadgets:

- (fixed-point) truncation: $x \mapsto \lfloor x/2^k \rfloor$
- MSB extraction: $x \mapsto x_{m-1}$
- ReLU: $x \mapsto (1 - x_{m-1}) \cdot x$

Conversion Gates



such that $x = \sum_{i=0}^{m-1} 2^i \cdot x_i$

1. Introduction to VOLE-based Zero-Knowledge

This Talk

1. Introduction to VOLE-based Zero-Knowledge
2. Proving Conversions

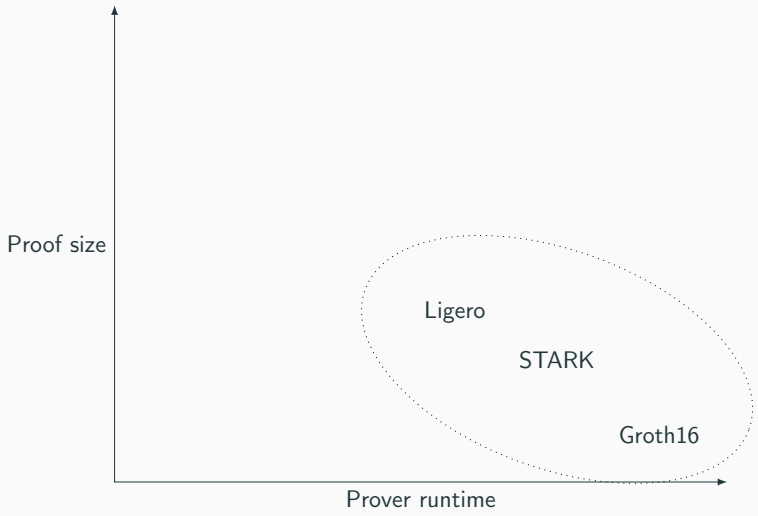
This Talk

1. Introduction to VOLE-based Zero-Knowledge
2. Proving Conversions
3. Committed Bits from Low-Degree PRGs

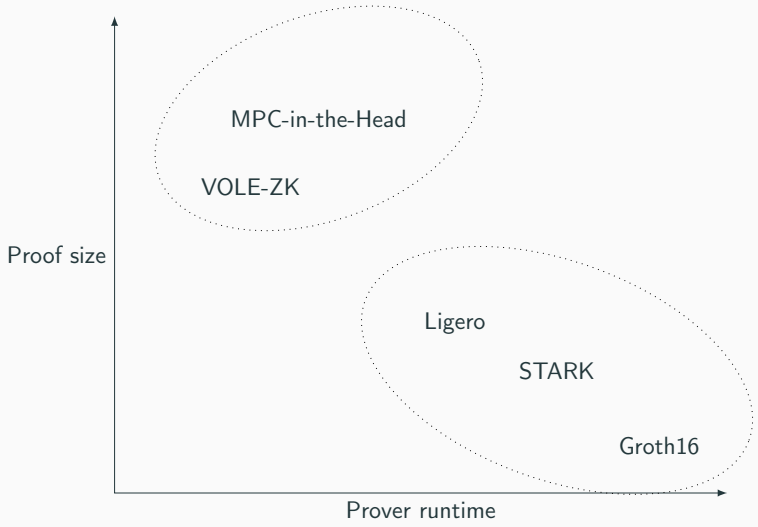
VOLE-based Zero-Knowledge

Space of Zero-Knowledge Proofs

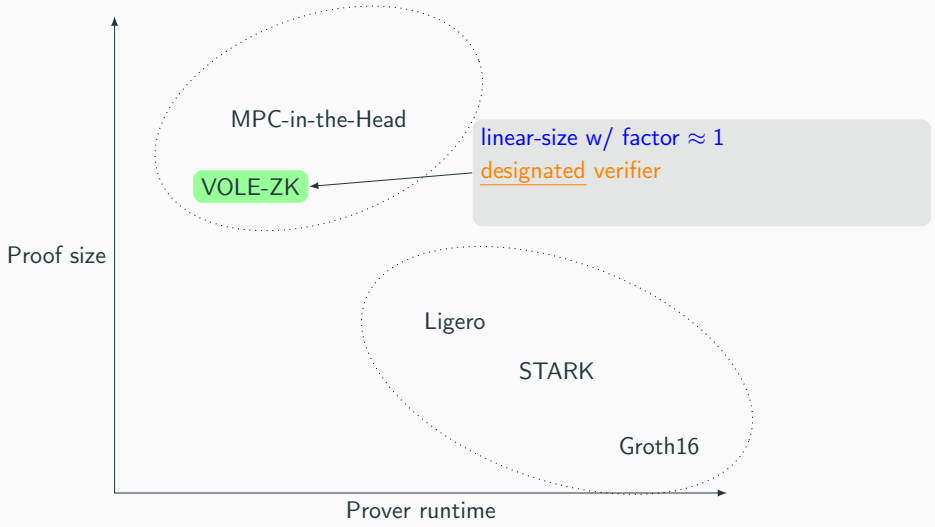
Space of Zero-Knowledge Proofs



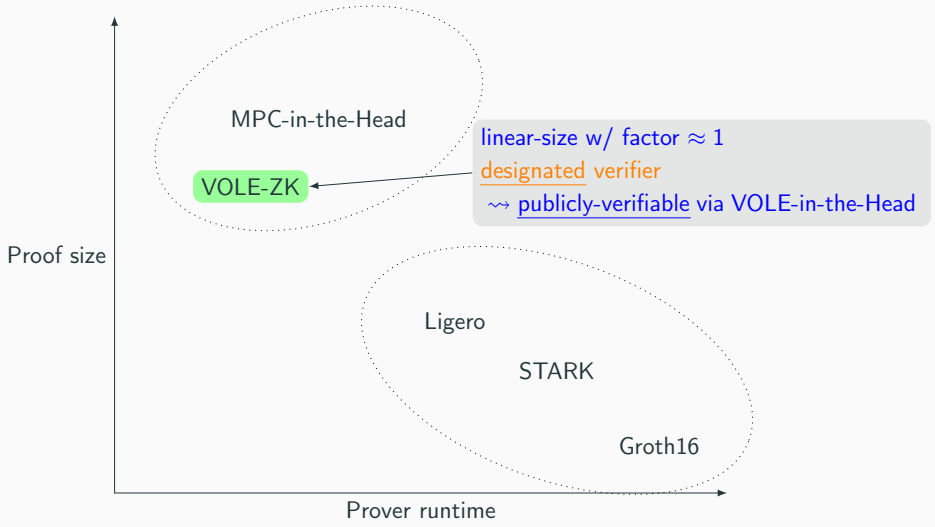
Space of Zero-Knowledge Proofs



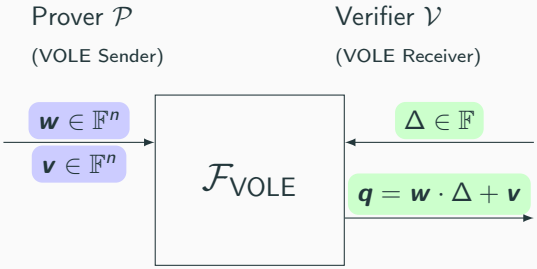
Space of Zero-Knowledge Proofs

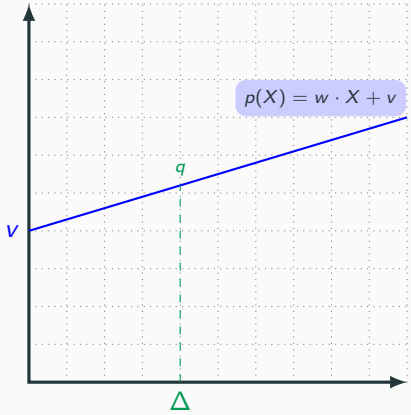


Space of Zero-Knowledge Proofs



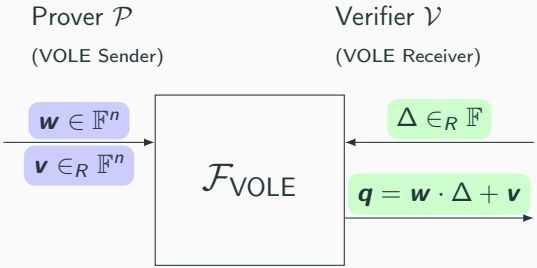
Vector Oblivious Linear Evaluation (VOLE)

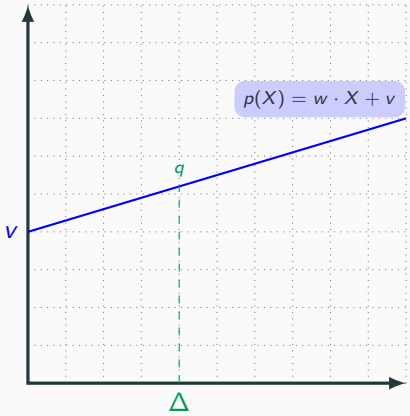




[CF13] Catalano and Fiore (2013), "Practical Homomorphic MACs for Arithmetic Circuits".
[BDOZ11] Bendlin et al. (2011), "Semi-homomorphic Encryption and Multiparty Computation".

Vector Oblivious Linear Evaluation (VOLE) as Homomorphic Commitments



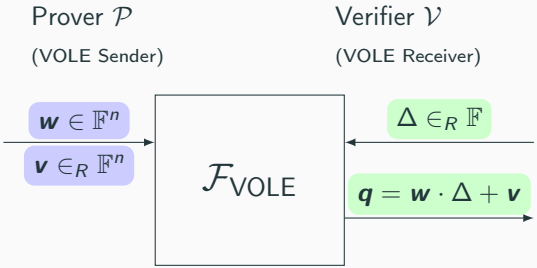


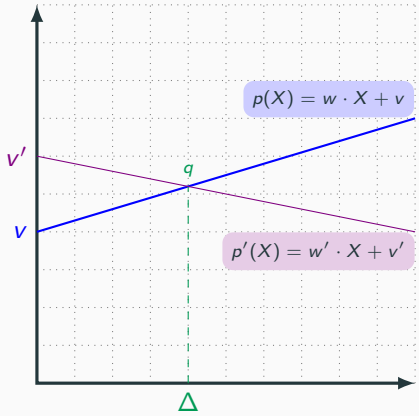
Linearly Homomorphic Commitments (cf. [CF13]; [BDOZ11])
use $q_i = w_i \cdot \Delta + v_i$ as information-theoretic MAC on w_i

- hiding since v_i is random
- breaking binding $\implies$ guessing Δ $\implies$ prob. $1/|\mathbb{F}|$

[CF13] Catalano and Fiore (2013), "Practical Homomorphic MACs for Arithmetic Circuits".
[BDOZ11] Bendlin et al. (2011), "Semi-homomorphic Encryption and Multiparty Computation".

Vector Oblivious Linear Evaluation (VOLE) as Homomorphic Commitments



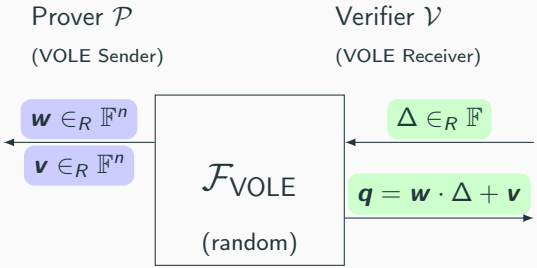


Linearly Homomorphic Commitments (cf. [CF13]; [BDOZ11])
use $q_i = w_i \cdot \Delta + v_i$ as information-theoretic MAC on w_i

- hiding since v_i is random
- breaking binding $\implies$ guessing Δ $\implies$ prob. $1/|\mathbb{F}|$

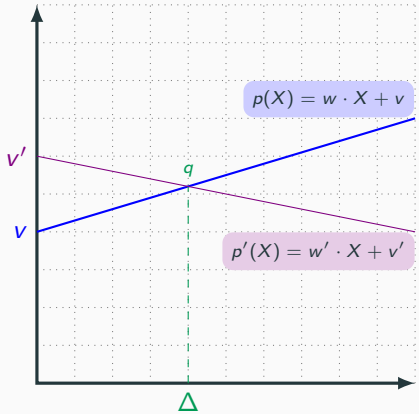
[CF13] Catalano and Fiore (2013), "Practical Homomorphic MACs for Arithmetic Circuits".
[BDOZ11] Bendlin et al. (2011), "Semi-homomorphic Encryption and Multiparty Computation".

Vector Oblivious Linear Evaluation (VOLE) as Homomorphic Commitments



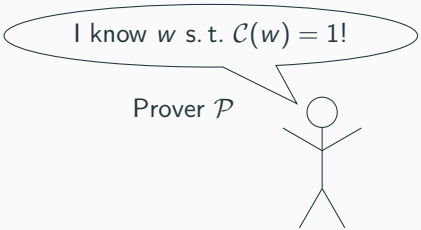
Linearly Homomorphic Commitments (cf. [CF13]; [BDOZ11])
use $q_i = w_i \cdot \Delta + v_i$ as information-theoretic MAC on w_i

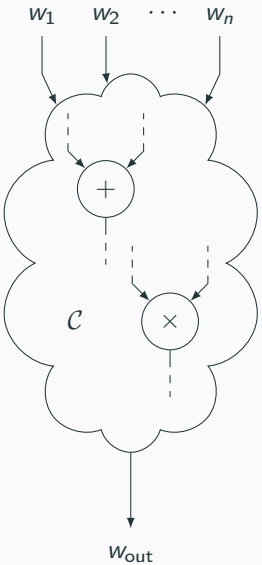
- hiding since v_i is random
- breaking binding $\implies$ guessing Δ $\implies$ prob. $1/|\mathbb{F}|$



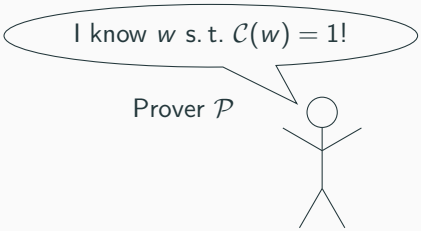
[CF13] Catalano and Fiore (2013), "Practical Homomorphic MACs for Arithmetic Circuits".
[BDOZ11] Bendlin et al. (2011), "Semi-homomorphic Encryption and Multiparty Computation".

Commit & Prove Zero-Knowledge



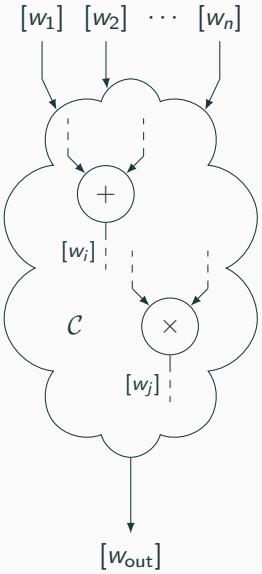


Commit & Prove Zero-Knowledge

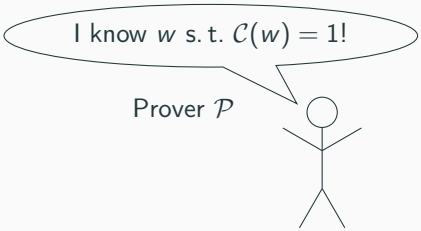


Ingredients:

1. linearly homomorphic commitments $[\cdot]$
   - can compute $[z] \leftarrow a \cdot [x] + [y] + b$ ✅

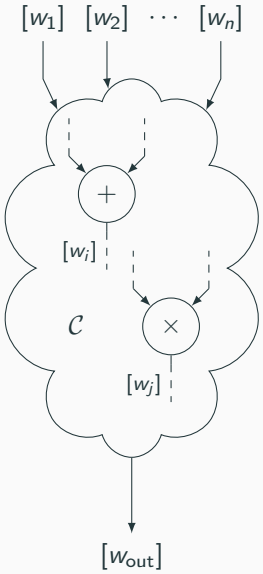


Commit & Prove Zero-Knowledge



Ingredients:

1. linearly homomorphic commitments $[\cdot]$
   - can compute $[z] \leftarrow a \cdot [x] + [y] + b$ ✅
2. multiplication check
   - given $([a], [b], [c])$, verify $a \cdot b \stackrel{?}{=} c$



Verifying Multiplications – LPZK [DIO21], QuickSilver [YSWW21]

Goal: Given $([a], [b], [c])$, verify that $a \cdot b = c$ in $\mathbb{F}$

Verifying Multiplications – LPZK [DIO21], QuickSilver [YSWW21]

Goal: Given $([a], [b], [c])$, verify that $a \cdot b = c$ in $\mathbb{F}$

QuickSilver Check: Convert the three MAC equations $q_x = x \cdot \Delta + v_x$ for $x \in \{a, b, c\}$ into a polynomial in Δ:

$$\underbrace{\Delta \cdot q_c - q_a \cdot q_b}_{\text{known by } \mathcal{V}} = \underbrace{(c - a \cdot b)}_{=0 \text{ if } \mathcal{P} \text{ honest}} \cdot \Delta^2 + \underbrace{(v_c - a \cdot v_b - b \cdot v_a)}_{\text{known by } \mathcal{P}} \cdot \Delta + \underbrace{(-v_a \cdot v_b)}_{\text{known by } \mathcal{P}}$$

[DIO21] Dittmer, Ishai, and Ostrovsky (2021), "Line-Point Zero Knowledge and Its Applications".

[YSWW21] Yang et al. (2021), "QuickSilver: Efficient and Affordable Zero-Knowledge Proofs for Circuits and Polynomials over Any Field".

Verifying Multiplications – LPZK [DIO21], QuickSilver [YSWW21]

Goal: Given $([a], [b], [c])$, verify that $a \cdot b = c$ in $\mathbb{F}$

QuickSilver Check: Convert the three MAC equations $q_x = x \cdot \Delta + v_x$ for $x \in \{a, b, c\}$ into a polynomial in Δ:

$$\underbrace{\Delta \cdot q_c - q_a \cdot q_b}_{\text{known by } \mathcal{V}} = \underbrace{(c - a \cdot b)}_{=0 \text{ if } \mathcal{P} \text{ honest}} \cdot \Delta^2 + \underbrace{(v_c - a \cdot v_b - b \cdot v_a)}_{\text{known by } \mathcal{P}} \cdot \Delta + \underbrace{(-v_a \cdot v_b)}_{\text{known by } \mathcal{P}}$$

use a random linear combination to verify many multiplications

[DIO21] Dittmer, Ishai, and Ostrovsky (2021), "Line-Point Zero Knowledge and Its Applications".

[YSWW21] Yang et al. (2021), "QuickSilver: Efficient and Affordable Zero-Knowledge Proofs for Circuits and Polynomials over Any Field".

Verifying Multiplications – LPZK [DIO21], QuickSilver [YSWW21]

Goal: Given $([a], [b], [c])$, verify that $a \cdot b = c$ in $\mathbb{F}$

QuickSilver Check: Convert the three MAC equations $q_x = x \cdot \Delta + v_x$ for $x \in \{a, b, c\}$ into a polynomial in Δ:

$$\underbrace{\Delta \cdot q_c - q_a \cdot q_b}_{\text{known by } \mathcal{V}} = \underbrace{(c - a \cdot b)}_{=0 \text{ if } \mathcal{P} \text{ honest}} \cdot \Delta^2 + \underbrace{(v_c - a \cdot v_b - b \cdot v_a)}_{\text{known by } \mathcal{P}} \cdot \Delta + \underbrace{(-v_a \cdot v_b)}_{\text{known by } \mathcal{P}}$$

Soundness: cheating $\mathcal{P}$ needs to come up with $p(X) = e_2 \cdot X^2 + e_1 \cdot X + e_0$ such that

[DIO21] Dittmer, Ishai, and Ostrovsky (2021), "Line-Point Zero Knowledge and Its Applications".

[YSWW21] Yang et al. (2021), "QuickSilver: Efficient and Affordable Zero-Knowledge Proofs for Circuits and Polynomials over Any Field".

Verifying Multiplications – LPZK [DIO21], QuickSilver [YSWW21]

Goal: Given $([a], [b], [c])$, verify that $a \cdot b = c$ in $\mathbb{F}$

QuickSilver Check: Convert the three MAC equations $q_x = x \cdot \Delta + v_x$ for $x \in \{a, b, c\}$ into a polynomial in Δ:

$$\underbrace{\Delta \cdot q_c - q_a \cdot q_b}_{\text{known by } \mathcal{V}} = \underbrace{(c - a \cdot b)}_{=0 \text{ if } \mathcal{P} \text{ honest}} \cdot \Delta^2 + \underbrace{(v_c - a \cdot v_b - b \cdot v_a)}_{\text{known by } \mathcal{P}} \cdot \Delta + \underbrace{(-v_a \cdot v_b)}_{\text{known by } \mathcal{P}}$$

Soundness: cheating $\mathcal{P}$ needs to come up with $p(X) = e_2 \cdot X^2 + e_1 \cdot X + e_0$ such that

- $p(\Delta) = 0$, and

[DIO21] Dittmer, Ishai, and Ostrovsky (2021), "Line-Point Zero Knowledge and Its Applications".

[YSWW21] Yang et al. (2021), "QuickSilver: Efficient and Affordable Zero-Knowledge Proofs for Circuits and Polynomials over Any Field".

Verifying Multiplications – LPZK [DIO21], QuickSilver [YSWW21]

Goal: Given $([a], [b], [c])$, verify that $a \cdot b = c$ in $\mathbb{F}$

QuickSilver Check: Convert the three MAC equations $q_x = x \cdot \Delta + v_x$ for $x \in \{a, b, c\}$ into a polynomial in Δ:

$$\underbrace{\Delta \cdot q_c - q_a \cdot q_b}_{\text{known by } \mathcal{V}} = \underbrace{(c - a \cdot b)}_{=0 \text{ if } \mathcal{P} \text{ honest}} \cdot \Delta^2 + \underbrace{(v_c - a \cdot v_b - b \cdot v_a)}_{\text{known by } \mathcal{P}} \cdot \Delta + \underbrace{(-v_a \cdot v_b)}_{\text{known by } \mathcal{P}}$$

Soundness: cheating $\mathcal{P}$ needs to come up with $p(X) = e_2 \cdot X^2 + e_1 \cdot X + e_0$ such that

- $p(\Delta) = 0$, and
- $e_2 := c - a \cdot b \neq 0$

[DIO21] Dittmer, Ishai, and Ostrovsky (2021), "Line-Point Zero Knowledge and Its Applications".

[YSWW21] Yang et al. (2021), "QuickSilver: Efficient and Affordable Zero-Knowledge Proofs for Circuits and Polynomials over Any Field".

Verifying Multiplications – LPZK [DIO21], QuickSilver [YSWW21]

Goal: Given $([a], [b], [c])$, verify that $a \cdot b = c$ in $\mathbb{F}$

QuickSilver Check: Convert the three MAC equations $q_x = x \cdot \Delta + v_x$ for $x \in \{a, b, c\}$ into a polynomial in Δ:

$$\underbrace{\Delta \cdot q_c - q_a \cdot q_b}_{\text{known by } \mathcal{V}} = \underbrace{(c - a \cdot b)}_{=0 \text{ if } \mathcal{P} \text{ honest}} \cdot \Delta^2 + \underbrace{(v_c - a \cdot v_b - b \cdot v_a)}_{\text{known by } \mathcal{P}} \cdot \Delta + \underbrace{(-v_a \cdot v_b)}_{\text{known by } \mathcal{P}}$$

Soundness: cheating $\mathcal{P}$ needs to come up with $p(X) = e_2 \cdot X^2 + e_1 \cdot X + e_0$ such that

- $p(\Delta) = 0$, and
- $e_2 := c - a \cdot b \neq 0$

$\implies$ p has degree 2 $\implies$ p has at most 2 roots $\implies$ soundness error $2/|\mathbb{F}|$

[DIO21] Dittmer, Ishai, and Ostrovsky (2021), "Line-Point Zero Knowledge and Its Applications".

[YSWW21] Yang et al. (2021), "QuickSilver: Efficient and Affordable Zero-Knowledge Proofs for Circuits and Polynomials over Any Field".

Verifying Multiplications – LPZK [DIO21], QuickSilver [YSWW21]

Goal: Given $([a], [b], [c])$, verify that $a \cdot b = c$ in $\mathbb{F}$

QuickSilver Check: Convert the three MAC equations $q_x = x \cdot \Delta + v_x$ for $x \in \{a, b, c\}$ into a polynomial in Δ:

$$\underbrace{\Delta \cdot q_c - q_a \cdot q_b}_{\text{known by } \mathcal{V}} = \underbrace{(c - a \cdot b)}_{=0 \text{ if } \mathcal{P} \text{ honest}} \cdot \Delta^2 + \underbrace{(v_c - a \cdot v_b - b \cdot v_a)}_{\text{known by } \mathcal{P}} \cdot \Delta + \underbrace{(-v_a \cdot v_b)}_{\text{known by } \mathcal{P}}$$

Generalizable to higher-degree constraints:

- $[a + b]^{\max(d_a, d_b)} \leftarrow [a]^{d_a} + [b]^{d_b}$ and $[a \cdot b]^{d_a + d_b} \leftarrow [a]^{d_a} \cdot [b]^{d_b}$
- for degree-d circuit $\mathcal{C}$: $[y]^d \leftarrow \mathcal{C}([\boldsymbol{x}]^1)$

$$[y]^d\text{:} \quad q_y = y \cdot \Delta^d + \sum_{k=0}^{d-1} p_k \cdot \Delta^k \qquad \rightsquigarrow \text{proof size } d \text{ field elements}$$

[DIO21] Dittmer, Ishai, and Ostrovsky (2021), "Line-Point Zero Knowledge and Its Applications".

[YSWW21] Yang et al. (2021), "QuickSilver: Efficient and Affordable Zero-Knowledge Proofs for Circuits and Polynomials over Any Field".

Conversions

Emulating $\mathbb{F}_2$ Arithmetic over $\mathbb{F}_p$

Bit decomposition

$$[x]_p = \sum_{k=0}^{m-1} 2^k \cdot [x_i]_p$$

Emulating $\mathbb{F}_2$ Arithmetic over $\mathbb{F}_p$

Bit decomposition

$$[x]_p = \sum_{k=0}^{m-1} 2^k \cdot [x_i]_p \qquad \wedge \qquad \bigwedge_{k=0}^{m-1} [x_i]_p \cdot (1 - [x_i]_p) = 0$$

Emulating $\mathbb{F}_2$ Arithmetic over $\mathbb{F}_p$

Bit decomposition

$$[x]_p = \sum_{k=0}^{m-1} 2^k \cdot [x_i]_p \qquad \wedge \qquad \bigwedge_{k=0}^{m-1} [x_i]_p \cdot (1 - [x_i]_p) = 0$$

Arithmetizing Boolean operations:

$$x \wedge y = x \cdot y$$
$$x \oplus y = x + y - 2 \cdot x \cdot y \qquad \text{for } x, y \in \{0, 1\}$$

Emulating $\mathbb{F}_2$ Arithmetic over $\mathbb{F}_p$

Bit decomposition

$$[x]_p = \sum_{k=0}^{m-1} 2^k \cdot [x_i]_p \qquad \wedge \qquad \bigwedge_{k=0}^{m-1} [x_i]_p \cdot (1 - [x_i]_p) = 0$$

Arithmetizing Boolean operations:

$$\begin{aligned} x \wedge y &= x \cdot y \\ x \oplus y &= x + y - 2 \cdot x \cdot y \end{aligned} \qquad \text{for } x, y \in \{0, 1\}$$

- ✗ Committing to $[x_0]_p, \ldots, [x_{m-1}]_p$ costs $m^2 = (\log p)^2$ bits of communication
- ✗ XOR not $\mathbb{F}_p$-linear $\rightsquigarrow$ no longer free

Conversions Based on (Extended) Doubly Authenticated Bits

Given $[x]_p, [\boldsymbol{x}]_2$, verify $x = \sum_i 2^i \cdot x_i$ with $m := \log p$

[RW19] Rotaru and Wood (2019), "MArBled Circuits: Mixing Arithmetic and Boolean Circuits with Active Security".

[EGKRS20] Escudero et al. (2020), "Improved Primitives for MPC over Mixed Arithmetic-Binary Circuits".

[BBMRS21] Baum et al. (2021), "Appenzeller to Brie: Efficient Zero-Knowledge Proofs for Mixed-Mode Arithmetic and Z2k".

[WYXKW21] Weng et al. (2021), "Mystique: Efficient Conversions for Zero-Knowledge Proofs with Applications to Machine Learning".

Conversions Based on (Extended) Doubly Authenticated Bits

Given $[x]_p, [\boldsymbol{x}]_2$, verify $x = \sum_i 2^i \cdot x_i$ with $m := \log p$

Preprocessing	Creation	Usage
daBits [RW19] $([\boldsymbol{r}]_p, [\boldsymbol{r}]_2)$ $\boldsymbol{r} \in_R \{0,1\}^m$		

[RW19] Rotaru and Wood (2019), "MArBled Circuits: Mixing Arithmetic and Boolean Circuits with Active Security".

[EGKRS20] Escudero et al. (2020), "Improved Primitives for MPC over Mixed Arithmetic-Binary Circuits".

[BBMRS21] Baum et al. (2021), "Appenzeller to Brie: Efficient Zero-Knowledge Proofs for Mixed-Mode Arithmetic and Z2k".

[WYXKW21] Weng et al. (2021), "Mystique: Efficient Conversions for Zero-Knowledge Proofs with Applications to Machine Learning".

Conversions Based on (Extended) Doubly Authenticated Bits

Given $[x]_p, [\boldsymbol{x}]_2$, verify $x = \sum_i 2^i \cdot x_i$ with $m := \log p$

Preprocessing	Creation	Usage
daBits [RW19]	m^2 bits ✗	
$([\boldsymbol{r}]_p, [\boldsymbol{r}]_2)$	+ consistency check	
$\boldsymbol{r} \in_R \{0,1\}^m$		

[RW19] Rotaru and Wood (2019), "MArBled Circuits: Mixing Arithmetic and Boolean Circuits with Active Security".

[EGKRS20] Escudero et al. (2020), "Improved Primitives for MPC over Mixed Arithmetic-Binary Circuits".

[BBMRS21] Baum et al. (2021), "Appenzeller to Brie: Efficient Zero-Knowledge Proofs for Mixed-Mode Arithmetic and Z2k".

[WYXKW21] Weng et al. (2021), "Mystique: Efficient Conversions for Zero-Knowledge Proofs with Applications to Machine Learning".

Conversions Based on (Extended) Doubly Authenticated Bits

Given $[x]_p, [\boldsymbol{x}]_2$, verify $x = \sum_i 2^i \cdot x_i$ with $m := \log p$

Preprocessing	Creation	Usage
daBits [RW19] $([\boldsymbol{r}]_p, [\boldsymbol{r}]_2)$ $\boldsymbol{r} \in_R \{0,1\}^m$	m^2 bits ✗ + consistency check	correction: $\boldsymbol{d} := \boldsymbol{x} \oplus \boldsymbol{r} \in \{0,1\}^m$ $[\boldsymbol{x}]_2 = [\boldsymbol{r}]_2 \oplus \boldsymbol{d}$ $[x]_p = \sum_i 2^i \cdot ([r_i]_p + d_i - 2d_i[r_i]_p)$ ← linear ✅

[RW19] Rotaru and Wood (2019), "MArBled Circuits: Mixing Arithmetic and Boolean Circuits with Active Security".

[EGKRS20] Escudero et al. (2020), "Improved Primitives for MPC over Mixed Arithmetic-Binary Circuits".

[BBMRS21] Baum et al. (2021), "Appenzeller to Brie: Efficient Zero-Knowledge Proofs for Mixed-Mode Arithmetic and Z2k".

[WYXKW21] Weng et al. (2021), "Mystique: Efficient Conversions for Zero-Knowledge Proofs with Applications to Machine Learning".

Conversions Based on (Extended) Doubly Authenticated Bits

Given $[x]_p, [\boldsymbol{x}]_2$, verify $x = \sum_i 2^i \cdot x_i$ with $m := \log p$

Preprocessing	Creation	Usage
daBits [RW19] $([\boldsymbol{r}]_p, [\boldsymbol{r}]_2)$ $\boldsymbol{r} \in_R \{0,1\}^m$	m^2 bits ✗ + consistency check	correction: $\boldsymbol{d} := \boldsymbol{x} \oplus \boldsymbol{r} \in \{0,1\}^m$ $[\boldsymbol{x}]_2 = [\boldsymbol{r}]_2 \oplus \boldsymbol{d}$ $[x]_p = \sum_i 2^i \cdot ([r_i]_p + d_i - 2d_i[r_i]_p)$ } ← linear ✅
edaBits [EGKRS20] $([r]_p, [\boldsymbol{r}]_2)$ $r = \sum_i 2^i \cdot r_i \in_R \mathbb{F}_p$		

[RW19] Rotaru and Wood (2019), "MArBled Circuits: Mixing Arithmetic and Boolean Circuits with Active Security".

[EGKRS20] Escudero et al. (2020), "Improved Primitives for MPC over Mixed Arithmetic-Binary Circuits".

[BBMRS21] Baum et al. (2021), "Appenzeller to Brie: Efficient Zero-Knowledge Proofs for Mixed-Mode Arithmetic and Z2k".

[WYXKW21] Weng et al. (2021), "Mystique: Efficient Conversions for Zero-Knowledge Proofs with Applications to Machine Learning".

Conversions Based on (Extended) Doubly Authenticated Bits

Given $[x]_p, [\boldsymbol{x}]_2$, verify $x = \sum_i 2^i \cdot x_i$ with $m := \log p$

Preprocessing	Creation	Usage
daBits [RW19] $([\boldsymbol{r}]_p, [\boldsymbol{r}]_2)$ $\boldsymbol{r} \in_R \{0,1\}^m$	m^2 bits ✗ + consistency check	correction: $\boldsymbol{d} := \boldsymbol{x} \oplus \boldsymbol{r} \in \{0,1\}^m$ $[\boldsymbol{x}]_2 = [\boldsymbol{r}]_2 \oplus \boldsymbol{d}$ $[x]_p = \sum_i 2^i \cdot ([r_i]_p + d_i - 2d_i[r_i]_p)$ } ← linear ✓
edaBits [EGKRS20] $([r]_p, [\boldsymbol{r}]_2)$ $r = \sum_i 2^i \cdot r_i \in_R \mathbb{F}_p$	$B \cdot m$ bits ✓ $B \in \{3,4,5\}$ + cut'n'choose consistency check	

[RW19] Rotaru and Wood (2019), "MArBled Circuits: Mixing Arithmetic and Boolean Circuits with Active Security".

[EGKRS20] Escudero et al. (2020), "Improved Primitives for MPC over Mixed Arithmetic-Binary Circuits".

[BBMRS21] Baum et al. (2021), "Appenzeller to Brie: Efficient Zero-Knowledge Proofs for Mixed-Mode Arithmetic and Z2k".

[WYXKW21] Weng et al. (2021), "Mystique: Efficient Conversions for Zero-Knowledge Proofs with Applications to Machine Learning".

Conversions Based on (Extended) Doubly Authenticated Bits

Given $[x]_p, [\boldsymbol{x}]_2$, verify $x = \sum_i 2^i \cdot x_i$ with $m := \log p$

Preprocessing	Creation	Usage
daBits [RW19] $([\boldsymbol{r}]_p, [\boldsymbol{r}]_2)$ $\boldsymbol{r} \in_R \{0,1\}^m$	m^2 bits ✗ + consistency check	correction: $\boldsymbol{d} := \boldsymbol{x} \oplus \boldsymbol{r} \in \{0,1\}^m$ $[\boldsymbol{x}]_2 = [\boldsymbol{r}]_2 \oplus \boldsymbol{d}$ $[x]_p = \sum_i 2^i \cdot ([r_i]_p + d_i - 2d_i[r_i]_p)$ } ← linear ✓
edaBits [EGKRS20] $([r]_p, [\boldsymbol{r}]_2)$ $r = \sum_i 2^i \cdot r_i \in_R \mathbb{F}_p$	$B \cdot m$ bits ✓ $B \in \{3,4,5\}$ + cut'n'choose consistency check	correction: $d := x - r \in \mathbb{F}_p$ $[x]_p = [r]_p + d$ $[\boldsymbol{x}]_2 = [\boldsymbol{r}]_2 + d \leftarrow$ Boolean addition circuit ✗

- need large batches for $B = 3$ ⚠

⇝ VOLE-ZK: A2B/Mystique [B**B**MRS21]; [WYXKW21]

[RW19] Rotaru and Wood (2019), "MArBled Circuits: Mixing Arithmetic and Boolean Circuits with Active Security".

[EGKRS20] Escudero et al. (2020), "Improved Primitives for MPC over Mixed Arithmetic-Binary Circuits".

[BBMRS21] Baum et al. (2021), "Appenzeller to Brie: Efficient Zero-Knowledge Proofs for Mixed-Mode Arithmetic and Z2k".

[WYXKW21] Weng et al. (2021), "Mystique: Efficient Conversions for Zero-Knowledge Proofs with Applications to Machine Learning".

Can we create daBits with low communication?

daBits from Low-Degree PRGs

Creating daBits with Low Communication – Our Approach

1. Commit to short seed $\boldsymbol{x} \in \{0,1\}^{\ell}$ over both $\mathbb{F}_2$ and $\mathbb{F}_p \rightsquigarrow ([\boldsymbol{x}]_2, [\boldsymbol{x}]_p)$
   - commit to ℓ daBits and prove consistency
   - one-time cost

Creating daBits with Low Communication – Our Approach

1. Commit to short seed $\boldsymbol{x} \in \{0,1\}^{\ell}$ over both $\mathbb{F}_2$ and $\mathbb{F}_p$ $\rightsquigarrow$ $([\boldsymbol{x}]_2, [\boldsymbol{x}]_p)$
   - commit to ℓ daBits and prove consistency
   - one-time cost
2. Non-interactively apply a low-degree PRG to obtain long $\boldsymbol{y} \in \{0,1\}^{\ell^s}$
   - use higher-degree QuickSilver

   $\rightsquigarrow$ get $\boldsymbol{y}$ committed over both fields $\rightsquigarrow$ $([\boldsymbol{y}]_2^{d_2}, [\boldsymbol{y}]_p^{d_p})$

Creating daBits with Low Communication – Our Approach

1. Commit to short seed $\boldsymbol{x} \in \{0,1\}^{\ell}$ over both $\mathbb{F}_2$ and $\mathbb{F}_p$ $\rightsquigarrow$ $([\boldsymbol{x}]_2, [\boldsymbol{x}]_p)$
   - commit to ℓ daBits and prove consistency
   - one-time cost
2. Non-interactively apply a low-degree PRG to obtain long $\boldsymbol{y} \in \{0,1\}^{\ell^s}$
   - use higher-degree QuickSilver

   $\rightsquigarrow$ get $\boldsymbol{y}$ committed over both fields $\rightsquigarrow$ $([\boldsymbol{y}]_2^{d_2}, [\boldsymbol{y}]_p^{d_p})$
3. Use daBits $([y_i]_2^{d_2}, [y_i]_p^{d_p})$ for conversions and other gadgets 🎉

Creating daBits with Low Communication – Our Approach

1. Commit to short seed $\boldsymbol{x} \in \{0,1\}^{\ell}$ over both $\mathbb{F}_2$ and $\mathbb{F}_p$ $\rightsquigarrow$ $([\boldsymbol{x}]_2, [\boldsymbol{x}]_p)$
   - commit to ℓ daBits and prove consistency
   - one-time cost
2. Non-interactively apply a low-degree PRG to obtain long $\boldsymbol{y} \in \{0,1\}^{\ell^s}$
   - use higher-degree QuickSilver
   - $\rightsquigarrow$ get $\boldsymbol{y}$ committed over both fields $\rightsquigarrow$ $([\boldsymbol{y}]_2^{d_2}, [\boldsymbol{y}]_p^{d_p})$
3. Use daBits $([y_i]_2^{d_2}, [y_i]_p^{d_p})$ for conversions and other gadgets 🎉
4. Reuse ℓ bits of the output as seed for next iteration

(reduce degree)

Mixed Low-Degree PRGs

Instantiation from Goldreich-style Random Local PRGs [Gol00]

$$k\text{-local PRG}\colon \{0,1\}^{\ell} \to \{0,1\}^{\ell^s} \text{ with stretch } s$$

$$\mathrm{PRG}(\boldsymbol{x})_i = P(x_{i_1}, \ldots, x_{i_k}) \text{ for } P\colon \{0,1\}^k \to \{0,1\} \text{ and } \{i_1, \ldots, i_k\} \subset_R [\ell]$$

[Gol00] Goldreich (2000), Candidate One-Way Functions Based on Expander Graphs.

Mixed Low-Degree PRGs

Instantiation from Goldreich-style Random Local PRGs [Gol00]

$$k\text{-local PRG}\colon \{0,1\}^{\ell} \to \{0,1\}^{\ell^s} \text{ with stretch } s$$
$$\mathrm{PRG}(\boldsymbol{x})_i = P(x_{i_1}, \ldots, x_{i_k}) \text{ for } P\colon \{0,1\}^k \to \{0,1\} \text{ and } \{i_1, \ldots, i_k\} \subset_R [\ell]$$

TSPA

$$P(x_1, \ldots, x_5) = x_1 \oplus x_2 \oplus x_3 \oplus (x_2 \oplus x_4) \wedge (x_3 \oplus x_5) \qquad \begin{cases} \mathbb{F}_2\text{-degree } d_2 = 2 \\ \mathbb{F}_p\text{-degree } d_p = 3 \end{cases}$$

[Gol00] Goldreich (2000), Candidate One-Way Functions Based on Expander Graphs.

Mixed Low-Degree PRGs

Instantiation from Goldreich-style Random Local PRGs [Gol00]

$$k\text{-local PRG}\colon \{0,1\}^{\ell} \to \{0,1\}^{\ell^s} \text{ with stretch } s$$
$$\mathrm{PRG}(\boldsymbol{x})_i = P(x_{i_1}, \ldots, x_{i_k}) \text{ for } P\colon \{0,1\}^k \to \{0,1\} \text{ and } \{i_1, \ldots, i_k\} \subset_R [\ell]$$

TSPA

$$P(x_1, \ldots, x_5) = x_1 \oplus x_2 \oplus x_3 \oplus (x_2 \oplus x_4) \wedge (x_3 \oplus x_5) \qquad \begin{cases} \mathbb{F}_2\text{-degree } d_2 = 2 \\ \mathbb{F}_p\text{-degree } d_p = 3 \end{cases}$$

XOR_4-$\mathrm{Majority}_7$ (better stretch)

$$P(x_1, \ldots, x_{11}) = x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus \mathrm{Majority}(x_5, \ldots, x_{11}) \qquad \begin{cases} \mathbb{F}_2\text{-degree } d_2 \leq 7 \\ \mathbb{F}_p\text{-degree } d_p \leq 11 \end{cases}$$

[Gol00] Goldreich (2000), Candidate One-Way Functions Based on Expander Graphs.

Mixed Low-Degree PRGs

Instantiation from Goldreich-style Random Local PRGs [Gol00]

$$k\text{-local PRG}\colon \{0,1\}^{\ell} \to \{0,1\}^{\ell^s} \text{ with stretch } s$$
$$\mathrm{PRG}(\boldsymbol{x})_i = P(x_{i_1}, \ldots, x_{i_k}) \text{ for } P\colon \{0,1\}^k \to \{0,1\} \text{ and } \{i_1, \ldots, i_k\} \subset_R [\ell]$$

TSPA

$$P(x_1, \ldots, x_5) = x_1 \oplus x_2 \oplus x_3 \oplus (x_2 \oplus x_4) \wedge (x_3 \oplus x_5) \qquad \begin{cases} \mathbb{F}_2\text{-degree } d_2 = 2 \\ \mathbb{F}_p\text{-degree } d_p = 3 \end{cases}$$

XOR_4-$\mathrm{Majority}_7$ (better stretch)

$$P(x_1, \ldots, x_{11}) = x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus \mathrm{Majority}(x_5, \ldots, x_{11}) \qquad \begin{cases} \mathbb{F}_2\text{-degree } d_2 \leq 7 \\ \mathbb{F}_p\text{-degree } d_p \leq 11 \end{cases}$$

$$[r_i]_2^{d_2} \leftarrow P_2([x_{i_1}]_2, \ldots, [x_{i_k}]_2) \qquad [r_i]_p^{d_p} \leftarrow P_p([x_{i_1}]_p, \ldots, [x_{i_k}]_p)$$

[Gol00] Goldreich (2000), Candidate One-Way Functions Based on Expander Graphs.

Performance and Summary

Performance

Rust Implementation: `https://github.com/AarhusCrypto/vole-zk-conversions/`

[BBMRS21] Baum et al. (2021), "Appenzeller to Brie: Efficient Zero-Knowledge Proofs for Mixed-Mode Arithmetic and Z2k".

[BBDKORS23] Baum et al. (2023), "Publicly Verifiable Zero-Knowledge and Post-Quantum Signatures from VOLE-in-the-Head".

Performance

Rust Implementation: `https://github.com/AarhusCrypto/vole-zk-conversions/`

Interactive VOLE-ZK with $p = 2^{61} - 1$ – Comparison with A2B [B**B**MRS21] (edaBits)

Improvements	LAN Time	WAN Time	Communication	#VOLEs
2^{10} conversions	2–3×	2–3×	2×	
2^{20} conversions	0.3–0.5×	≈	10×	
fixed-point multiplication	5×	13×	10×	

- amortized ≈ 0 VOLEs per conversion for $\text{XOR}_4\text{-Maj}_7$

[BBMRS21] Baum et al. (2021), "Appenzeller to Brie: Efficient Zero-Knowledge Proofs for Mixed-Mode Arithmetic and Z2k".

[BBDKORS23] Baum et al. (2023), "Publicly Verifiable Zero-Knowledge and Post-Quantum Signatures from VOLE-in-the-Head".

Performance

Rust Implementation: `https://github.com/AarhusCrypto/vole-zk-conversions/`

Interactive VOLE-ZK with $p = 2^{61} - 1$ – Comparison with A2B [B**B**MRS21] (edaBits)

Improvements	LAN Time	WAN Time	Communication	#VOLEs
2^{10} conversions	2–3×	2–3×	2×	
2^{20} conversions	0.3–0.5×	≈	10×	
fixed-point multiplication	5×	13×	10×	

- amortized ≈ 0 VOLEs per conversion for $\text{XOR}_4\text{-Maj}_7$

VOLE-in-the-Head [B**B**DKORS23] with $p \approx 2^{128}$

- VOLEs are more expensive ⇝ estimated 20–150× smaller proofs

[BBMRS21] Baum et al. (2021), "Appenzeller to Brie: Efficient Zero-Knowledge Proofs for Mixed-Mode Arithmetic and Z2k".

[BBDKORS23] Baum et al. (2023), "Publicly Verifiable Zero-Knowledge and Post-Quantum Signatures from VOLE-in-the-Head".

Summary

VOLE-based Zero-Knowledge

- lightweight, fast, linear size
- non-interactive with VOLE-in-the-Head

Summary

VOLE-based Zero-Knowledge

- lightweight, fast, linear size
- non-interactive with VOLE-in-the-Head

Low-Bandwidth Conversions from Low-Degree PRGs

- Up to 10× less communication, a bit more computation
- Vastly fewer VOLEs needed $\rightsquigarrow$ great for VOLE-in-the-Head
- Instantiations from Goldreich PRGs and sparse LPN
- More: Fixed-point arithmetic, comparisons, bit extraction, ReLU

Summary

VOLE-based Zero-Knowledge

- lightweight, fast, linear size
- non-interactive with VOLE-in-the-Head

Low-Bandwidth Conversions from Low-Degree PRGs

- Up to 10× less communication, a bit more computation
- Vastly fewer VOLEs needed $\rightsquigarrow$ great for VOLE-in-the-Head
- Instantiations from Goldreich PRGs and sparse LPN
- More: Fixed-point arithmetic, comparisons, bit extraction, ReLU

Open Question

- How to efficiently commit to even fewer bits over a large field?

Thank you!

References i

[B**B**DKORS23] C. Baum, Lennart Braun, C. Delpech de Saint Guilhem, M. Klooß, E. Orsini, L. Roy, and P. Scholl. **"Publicly Verifiable Zero-Knowledge and Post-Quantum Signatures from VOLE-in-the-Head".** In: CRYPTO 2023, Part V. Aug. 2023.

[B**B**MRS21] C. Baum, Lennart Braun, A. Munch-Hansen, B. Razet, and P. Scholl. **"Appenzeller to Brie: Efficient Zero-Knowledge Proofs for Mixed-Mode Arithmetic and Z2k".** In: ACM CCS 2021. Nov. 2021.

[BBMORRRS24] C. Baum, W. Beullens, S. Mukherjee, E. Orsini, S. Ramacher, C. Rechberger, L. Roy, and P. Scholl. **"One Tree to Rule Them All: Optimizing GGM Trees and OWFs for Post-Quantum Signatures".** In: ASIACRYPT 2024, Part I. Dec. 2024.

[BDOZ11] R. Bendlin, I. Damgård, C. Orlandi, and S. Zakarias. **"Semi-homomorphic Encryption and Multiparty Computation".** In: EUROCRYPT 2011. May 2011.

References ii

[CF13] D. Catalano and D. Fiore. **"Practical Homomorphic MACs for Arithmetic Circuits".** In: EUROCRYPT 2013. May 2013.

[DIO21] S. Dittmer, Y. Ishai, and R. Ostrovsky. **"Line-Point Zero Knowledge and Its Applications".** In: ITC 2021. July 2021.

[EGKRS20] D. Escudero, S. Ghosh, M. Keller, R. Rachuri, and P. Scholl. **"Improved Primitives for MPC over Mixed Arithmetic-Binary Circuits".** In: CRYPTO 2020, Part II. Aug. 2020.

[Gol00] O. Goldreich. **Candidate One-Way Functions Based on Expander Graphs.** Cryptology ePrint Archive, Report 2000/063. 2000. URL: https://eprint.iacr.org/2000/063.

[RW19] D. Rotaru and T. Wood. **"MArBled Circuits: Mixing Arithmetic and Boolean Circuits with Active Security".** In: INDOCRYPT 2019. Dec. 2019.

References iii

[WYXKW21] C. Weng, K. Yang, X. Xie, J. Katz, and X. Wang. **"Mystique: Efficient Conversions for Zero-Knowledge Proofs with Applications to Machine Learning".** In: USENIX Security 2021. Aug. 2021.

[YSWW21] K. Yang, P. Sarkar, C. Weng, and X. Wang. **"QuickSilver: Efficient and Affordable Zero-Knowledge Proofs for Circuits and Polynomials over Any Field".** In: ACM CCS 2021. Nov. 2021.

PRG Parameters

Predicate	(d_2, d_p)	ℓ	s	ℓ^s
TSPA	$(2, 3)$	4096	1.19	19 893
XOR_4-MAJ_7	$(4, 11)$	1024	2	2^{20}

High(er)-Degree QuickSilver

Recall: QuickSilver [YSWW21] can be used to prove degree-d constraints

[YSWW21] Yang et al. (2021), "QuickSilver: Efficient and Affordable Zero-Knowledge Proofs for Circuits and Polynomials over Any Field".

[BBMORRRS24] Baum et al. (2024), "One Tree to Rule Them All: Optimizing GGM Trees and OWFs for Post-Quantum Signatures".

High(er)-Degree QuickSilver

Recall: QuickSilver [YSWW21] can be used to prove degree-d constraints

Another view [BBMORRRS24]**:**

– View $[a]$ as degree-1 commitment: $[a]^1$: $q_a = a \cdot \Delta + v_a$

[YSWW21] Yang et al. (2021), "QuickSilver: Efficient and Affordable Zero-Knowledge Proofs for Circuits and Polynomials over Any Field".

[BBMORRRS24] Baum et al. (2024), "One Tree to Rule Them All: Optimizing GGM Trees and OWFs for Post-Quantum Signatures".

High(er)-Degree QuickSilver

Recall: QuickSilver [YSWW21] can be used to prove degree-d constraints

Another view [BBMORRRS24]**:**

- View $[a]$ as degree-1 commitment: $[a]^1$: $q_a = a \cdot \Delta + v_a$
- Multiplying $[a]^1$ and $[b]^1$ results in a degree-2 commitment

$$[a \cdot b]^2: \quad q_a \cdot q_b = a \cdot b \cdot \Delta^2 + (a \cdot v_b + b \cdot v_a) \cdot \Delta + (v_a \cdot v_b)$$

[YSWW21] Yang et al. (2021), "QuickSilver: Efficient and Affordable Zero-Knowledge Proofs for Circuits and Polynomials over Any Field".

[BBMORRRS24] Baum et al. (2024), "One Tree to Rule Them All: Optimizing GGM Trees and OWFs for Post-Quantum Signatures".

High(er)-Degree QuickSilver

Recall: QuickSilver [YSWW21] can be used to prove degree-d constraints

Another view [BBMORRRS24]**:**

- View $[a]$ as degree-1 commitment: $[a]^1$: $q_a = a \cdot \Delta + v_a$
- Multiplying $[a]^1$ and $[b]^1$ results in a degree-2 commitment

$$[a \cdot b]^2: \quad q_a \cdot q_b = a \cdot b \cdot \Delta^2 + (a \cdot v_b + b \cdot v_a) \cdot \Delta + (v_a \cdot v_b)$$

- ... adding $[c]^1$ needs shift by Δ:

$$[a \cdot b + c]^2: \quad q_a \cdot q_b + \Delta \cdot q_c = (a \cdot b) + c) \cdot \Delta^2 + ((a \cdot v_b + b \cdot v_a) + v_c) \cdot \Delta + (v_a \cdot v_b)$$

[YSWW21] Yang et al. (2021), "QuickSilver: Efficient and Affordable Zero-Knowledge Proofs for Circuits and Polynomials over Any Field".

[BBMORRRS24] Baum et al. (2024), "One Tree to Rule Them All: Optimizing GGM Trees and OWFs for Post-Quantum Signatures".

High(er)-Degree QuickSilver

Recall: QuickSilver [YSWW21] can be used to prove degree-d constraints

Another view [BBMORRRS24]**:**

- View $[a]$ as degree-1 commitment: $[a]^1$: $q_a = a \cdot \Delta + v_a$
- Multiplying $[a]^1$ and $[b]^1$ results in a degree-2 commitment

$$[a \cdot b]^2\colon \quad q_a \cdot q_b = a \cdot b \cdot \Delta^2 + (a \cdot v_b + b \cdot v_a) \cdot \Delta + (v_a \cdot v_b)$$

- ... adding $[c]^1$ needs shift by Δ:

$$[a{\cdot}b{+}c]^2\colon \quad q_a \cdot q_b + \Delta \cdot q_c = (a \cdot b) + c) \cdot \Delta^2 + ((a \cdot v_b + b \cdot v_a) + v_c) \cdot \Delta + (v_a \cdot v_b)$$

$\Longrightarrow$ **In general:**

- $[a+b]^{\max(d_a,d_b)} \leftarrow [a]^{d_a} + [b]^{d_b}$ and $[a \cdot b]^{d_a+d_b} \leftarrow [a]^{d_a} \cdot [b]^{d_b}$

[YSWW21] Yang et al. (2021), "QuickSilver: Efficient and Affordable Zero-Knowledge Proofs for Circuits and Polynomials over Any Field".

[BBMORRRS24] Baum et al. (2024), "One Tree to Rule Them All: Optimizing GGM Trees and OWFs for Post-Quantum Signatures".

High(er)-Degree QuickSilver

Recall: QuickSilver [YSWW21] can be used to prove degree-d constraints

Another view [BBMORRRS24]**:**

- View $[a]$ as degree-1 commitment: $[a]^1$: $q_a = a \cdot \Delta + v_a$
- Multiplying $[a]^1$ and $[b]^1$ results in a degree-2 commitment

$$[a \cdot b]^2\text{:} \quad q_a \cdot q_b = a \cdot b \cdot \Delta^2 + (a \cdot v_b + b \cdot v_a) \cdot \Delta + (v_a \cdot v_b)$$

- ... adding $[c]^1$ needs shift by Δ:

$$[a \cdot b + c]^2\text{:} \quad q_a \cdot q_b + \Delta \cdot q_c = (a \cdot b) + c) \cdot \Delta^2 + ((a \cdot v_b + b \cdot v_a) + v_c) \cdot \Delta + (v_a \cdot v_b)$$

$\Longrightarrow$ **In general:**

- $[a + b]^{\max(d_a, d_b)} \leftarrow [a]^{d_a} + [b]^{d_b}$ and $[a \cdot b]^{d_a + d_b} \leftarrow [a]^{d_a} \cdot [b]^{d_b}$
- for degree-d circuit $\mathcal{C}$: $[y]^d \leftarrow \mathcal{C}([x]^1)$

$$[y]^d\text{:} \quad q_y = y \cdot \Delta^d + \sum_{k=0}^{d-1} p_k \cdot \Delta^k \quad \rightsquigarrow \quad \begin{array}{l}\text{proof size } d \text{ field elements} \\ \text{soundness error } d/|\mathbb{F}|\end{array}$$

[YSWW21] Yang et al. (2021), "QuickSilver: Efficient and Affordable Zero-Knowledge Proofs for Circuits and Polynomials over Any Field".

[BBMORRRS24] Baum et al. (2024), "One Tree to Rule Them All: Optimizing GGM Trees and OWFs for Post-Quantum Signatures".

Fixed-Point Arithmetic Even Cheaper

We do not are about $\mathbb{F}_2$! Generate committed bits over $\mathbb{F}_p$ only:

- Commit to seed $[\boldsymbol{x}]_p$ and prove all $x_i \in \{0, 1\}$
- Generate committed bits $[r_i]_p^{d_p} \leftarrow P_p([x_{i_1}]_p, \ldots, [x_{i_k}]_p)$

Fixed-Point Arithmetic Even Cheaper

We do not are about $\mathbb{F}_2$! Generate committed bits over $\mathbb{F}_p$ only:

- Commit to seed $[\boldsymbol{x}]_p$ and prove all $x_i \in \{0, 1\}$
- Generate committed bits $[r_i]_p^{d_p} \leftarrow P_p([x_{i_1}]_p, \ldots, [x_{i_k}]_p)$

Fixed-point Multiplication: Given $([a]_p, [b]_p, [c]_p)$ prove $c = \lfloor (a \cdot b)/2^f \rfloor$:

- Use the $[r_i]_p^{d_p}$ to commit to bit decomposition of $c' := a \cdot b$:

$$([c'_0]_p^{d_p}, \ldots, [c'_{m-1}]_p^{d_p})$$

- Prove

$$\left([a \cdot b]_p^2 = \sum_{k=0}^{m-1} 2^k \cdot [c'_k]_p^{d_p}\right) \quad \bigwedge \quad \left([c]_p = \sum_{k=f}^{m-1} 2^{k-f} \cdot [c'_k]_p^{d_p}\right)$$