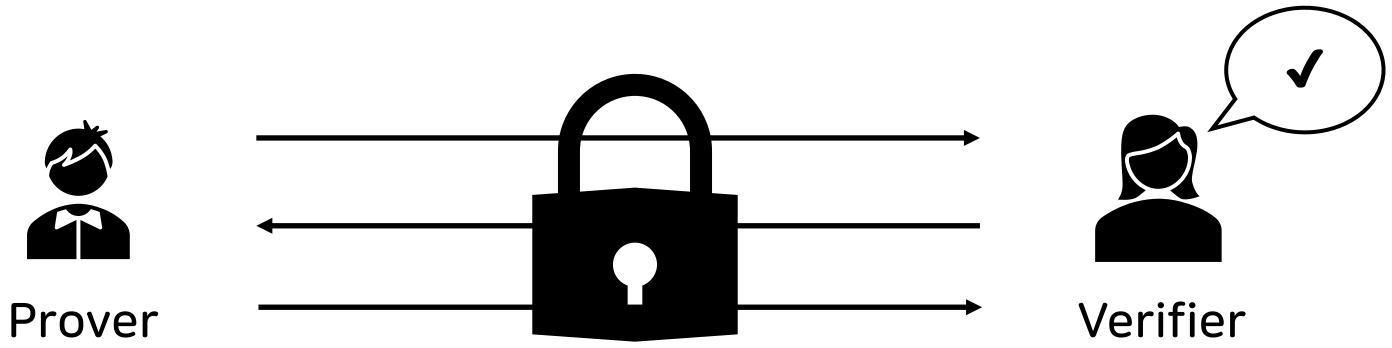
Polocolo: A ZK-Friendly Hash Function Based on S-boxes Using Power Residues

Eurocrypt 2025

Jincheol Ha, Seongha Hwang, Jooyoung Lee,
Seungmin Park, and Mincheol Son
KAIST

Background

Zero-knowledge Proof



- A two-party cryptographic protocol between a prover and a verifier that allows the prover to convince the verifier of their knowledge without revealing itself
- "I know the input x, which satisfies $SHA256(x) = 000\dots0$"

ZK-Unfriendly Operations

- Some operations are highly inefficient in ZKP (e.g., Compare, AND, XOR)
  - Common ZKP supports only addition/multiplication in $\mathbb{F}_p$ (256-bit prime field in ours)
- Constraints of XORing two 8-bit integers ($c = a \oplus b$) in ZKP:

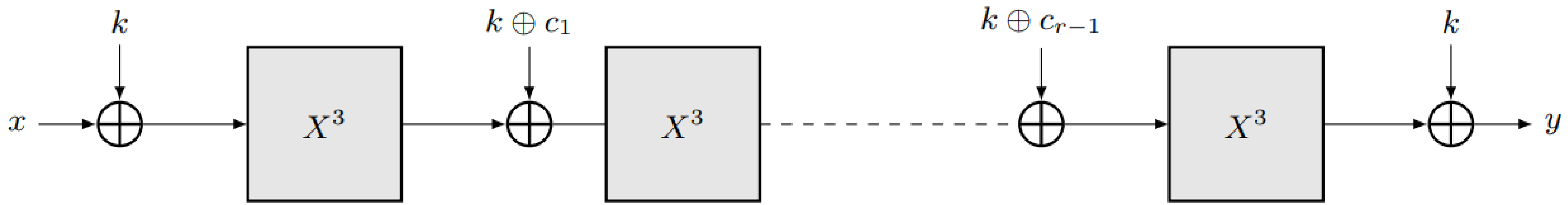
1. Bit decomposition of a, b, c: 21 add gates
2. $a_i, b_i \in \{0,1\}$: 16 mul gates
3. $c_i = a_i \oplus b_i$: 8 add gates & 4 mul gates

$$
\begin{array}{lcccccccc}
 & a_7 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & a_0 \\
a = & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
 & b_7 & b_6 & b_5 & b_4 & b_3 & b_2 & b_1 & b_0 \\
b = & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\
\hline
 & & & & \text{XOR} & & & & \\
 & c_7 & c_6 & c_5 & c_4 & c_3 & c_2 & c_1 & c_0 \\
c = & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1
\end{array}
$$

29 add gates & 20 mult gates → 49 gates in total

ZK-Friendly Hash Functions

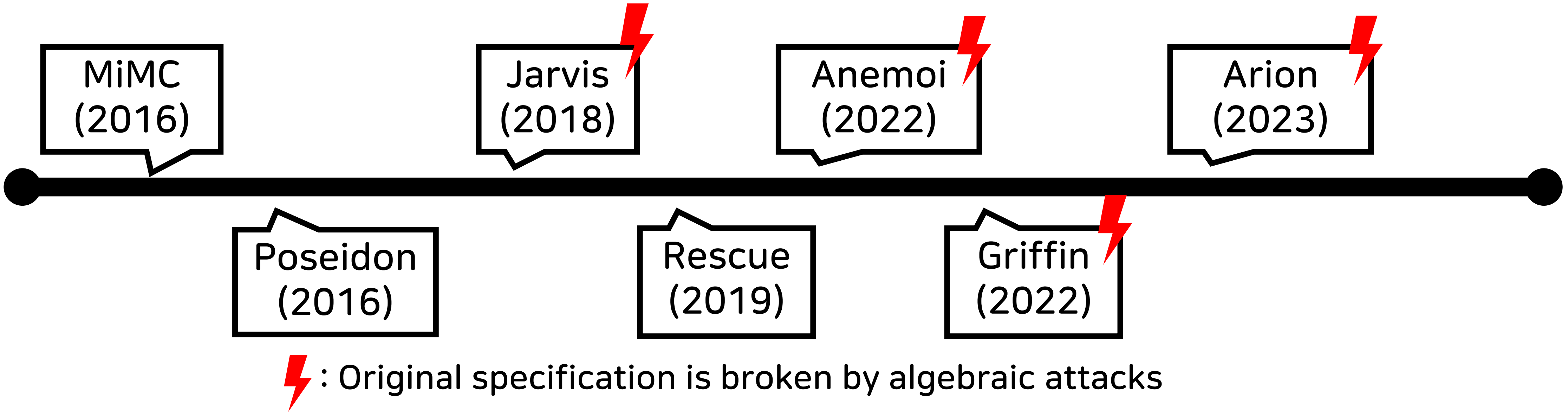
- Standardized hash functions (e.g., SHA2, SHA3) are inefficient in ZKP
- Lead to the invention of ZK-friendly hash functions
- ZK-friendly hash functions consist of add/mul in $\mathbb{F}_p$
- Compared to SHA2 and SHA3, ZK-friendly hash functions use 50-100x fewer gates in ZKP



MiMC illustration. Images from the original paper.

ZK-Friendly Hash Functions

- Due to their simple structures over $\mathbb{F}_p$, they are often vulnerable to algebraic attacks



Lookup Arguments

- Using lookup arguments, a prover can demonstrate a witness e belongs to a public table T
- Lookup operations allow "ZK-unfriendly" operations to be handled more efficiently
- Constraints of XORing two 8-bit integers ($c = a \oplus b$) in ZKP w/ lookup:

1. Define a public table $T = \{(x, y, z) \mid x, y \in \{0, \ldots, 15\}, z = x \oplus y\}$
2. 4-bit decomposition of a, b, c: 3 add gates
3. $a_i, b_i \in \{0, \ldots, 15\}$ and $c_i = a_i \oplus b_i$: 2 lookup gates

$a = 0100\ 1001$

$b = 1101\ 0010$

— — – XOR — — -

$c = 1001\ 1011$

3 add gates & 2 lookup gates → 5 gates in total (w/o lookup : 49 gates)

Plonkup

- Polocolo mainly focuses on Plonkup, an extension of Plonk
- Supported gates in Plonkup:
  - add/mul in $\mathbb{F}_p$ (natively supported in Plonk)
  - table lookup
- In Plonkup, the prover's complexity is proportional to

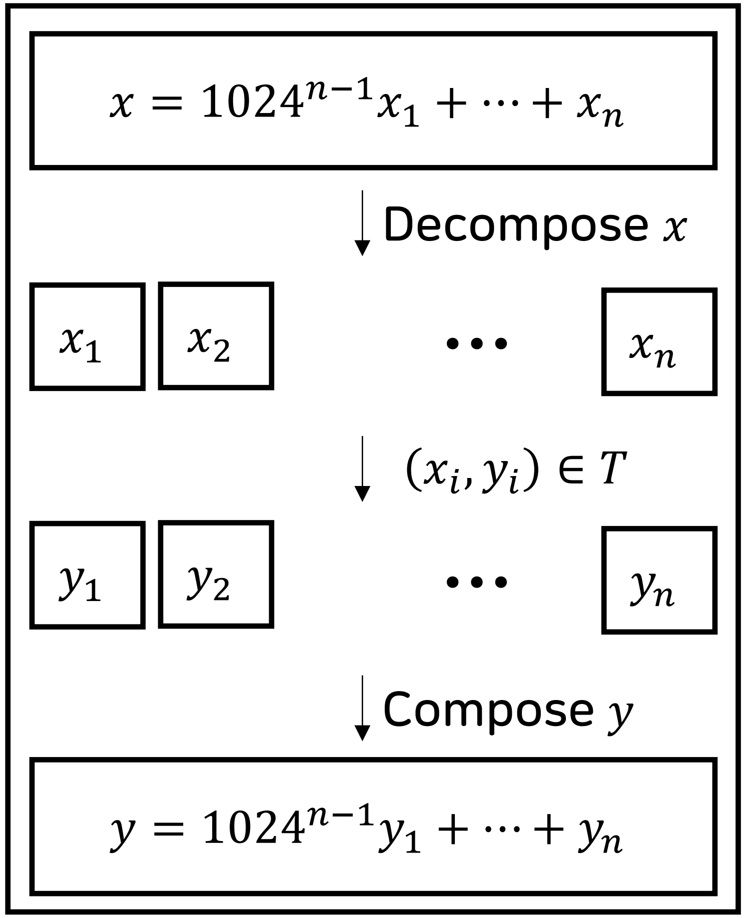
$$\max\{number\ of\ gates, table\ size\}$$

- The efficiency metric is the total number of gates in the Plonkup (= Plonk gates)

Motivation & Our Contribution

Motivation

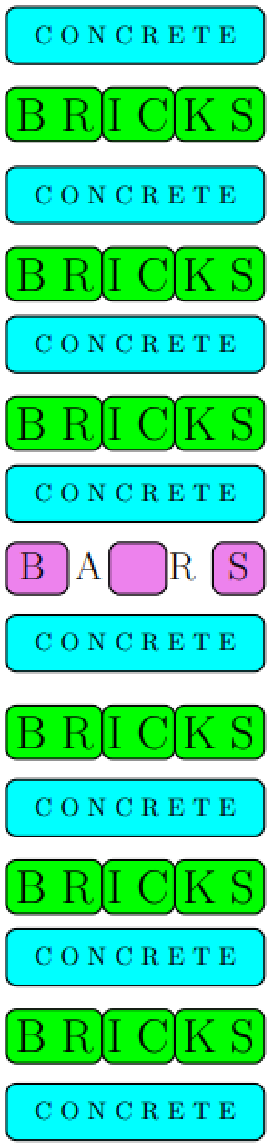
- Algebraically complex operation can be implemented by table lookups, provide resistance to algebraic attacks
- Reinforced Concrete (CCS'22) hash function uses the "base expansion method" to apply table lookup
  - An input $x \in \mathbb{F}_p$ is decomposed into an n-tuple $(x_1, \dots, x_n) \in \mathbb{Z}_{1024} \times \cdots \times \mathbb{Z}_{1024}$
  - Each component is fed to the underlying S-box $\mathbb{Z}_{1024} \to \mathbb{Z}_{1024}$ using lookup table



Bar function $bar : \mathbb{F}_p \to \mathbb{F}_p$
in Reinforced Concrete

Motivation

- Layers in Reinforced Concrete:
  - Linear layer (Concrete): 5 gates
  - Nonlinear layer (Bricks): 8 gates
  - Table lookup layer (Bars): 282 gates
- Only 1 layer out of the 15 layers uses lookup operation, it accounts 75% of Plonk gates
- Reducing the cost of applying a lookup table and iterating it over multiple rounds can enhance security and efficiency



Reinforced Concrete illustration.
Images from the original paper.

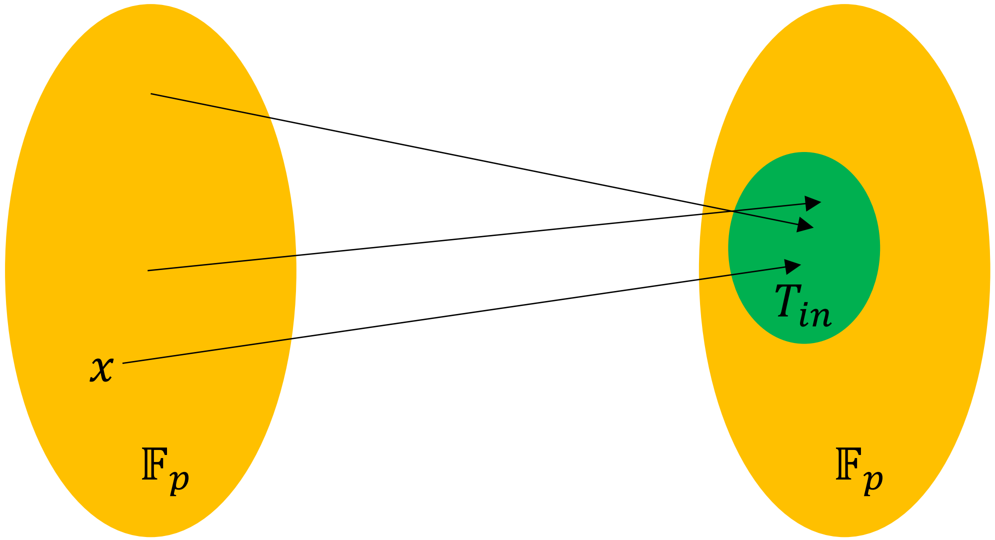
Our Contribution

- We propose Polocolo (Power residue for lower cost table lookup), a new lookup-based ZK-friendly hash function
  - An S-box is constructed using the "power residue method", which allows one to efficiently apply a lookup table to the elements of $\mathbb{F}_p$
  - A linear layer uses a new MDS matrix, optimized for Plonk circuits
- Polocolo requires fewer Plonk gates compared to the state-of-the-art ZK-friendly hash functions:
  - 21% fewer Plonk gates compared to Anemoi (when $\mathbb{F}_p^8 \to \mathbb{F}_p^8$)
  - 24% fewer Plonk gates compared to Reinforced Concrete (when $\mathbb{F}_p^3 \to \mathbb{F}_p^3$)

Polocolo Hash Function

Power Residue Method

- To insert an element $x \in \mathbb{F}_p$ into the table T, the function $f: \mathbb{F}_p \to T_{in}$ is required
  - In other words, the size of the range of f should be limited
  - Typically, $|T| \approx 1024$
  - In Reinforced Concrete, $f(x) = x\%1024$



Power Residue Method

- For a multiplicative generator g, an element $x \in \mathbb{F}_p \backslash \{0\}$ can be represented as $x = g^e$ for some exponent $0 \le e < p - 1$
- For an integer m such that $m \mid p - 1$, let $e = qm + r$ ($0 \le q < \frac{p-1}{m}, 0 \le r < m$)

x	e	q	r
$1(= g^0)$	0	0	0
$7(= g^1)$	1	0	1
$10(= g^2)$	2	0	2
…	…	…	…
$4(= g^{10})$	10	2	2
$2(= g^{11})$	11	2	3

Examples for $p = 13, g = 7, m = 4$

Power Residue Method

- For each x, the remainder r has m different values
- $f(x) = r$ works, but infeasible to compute function f efficiently
- Solution) $f(x) = x^{(p-1)/m} = g^{r(p-1)/m}$, determined by r

x	e	q	r	$f(x) = x^{(p-1)/m}$
$1(= g^0)$	0	0	0	1
$7(= g^1)$	1	0	1	5
$10(= g^2)$	2	0	2	12
…	…	…	…	…
$4(= g^{10})$	10	2	2	12
$2(= g^{11})$	11	2	3	8

Examples for $p = 13, g = 7, m = 4$

Power Residue Method

- $f(x) = x^{(p-1)/m}$ is a m-th power residue of x:

$$\left(\frac{x}{p}\right)_m = x^{(p-1)/m}$$

- Our new S-box $S : \mathbb{F}_p \to \mathbb{F}_p$ is defined as

$$S(x) = x^{-1} \cdot T\left[\left(\frac{x}{p}\right)_m\right]$$

A new S-Box from Power Residue

- Step 0:

$$g^0 \quad g^1 \quad g^2 \quad g^3 \quad g^4 \quad g^5 \quad g^6 \quad g^7 \quad g^8 \quad g^9 \quad g^{10} \quad g^{11}$$

$$\downarrow$$

$$g^0 \quad g^1 \quad g^2 \quad g^3 \quad g^4 \quad g^5 \quad g^6 \quad g^7 \quad g^8 \quad g^9 \quad g^{10} \quad g^{11}$$

$$S(g^{qm+r}) = g^{qm+r}$$

A new S-Box from Power Residue

- Step 1:

$$g^0 \quad g^1 \quad g^2 \quad g^3 \quad g^4 \quad g^5 \quad g^6 \quad g^7 \quad g^8 \quad g^9 \quad g^{10} \quad g^{11}$$

$$\downarrow$$

$$\underline{g^2 \quad g^0 \quad g^3 \quad g^1} \quad g^6 \quad g^4 \quad g^7 \quad g^5 \quad g^{10} \quad g^8 \quad g^{11} \quad g^9$$

$$S(g^{qm+r}) = g^{qm+\sigma(r)}$$

- σ is a permutation on $\{0, \dots, m-1\}$.

A new S-Box from Power Residue

- Step 2:

$$g^0 \quad g^1 \quad g^2 \quad g^3 \quad g^4 \quad g^5 \quad g^6 \quad g^7 \quad g^8 \quad g^9 \quad g^{10} \quad g^{11}$$

$$\downarrow$$

$$g^2 \quad g^0 \quad g^3 \quad g^1 \quad g^{10} \quad g^8 \quad g^{11} \quad g^9 \quad g^6 \quad g^4 \quad g^7 \quad g^5$$

$$S(g^{qm+r}) = g^{-qm+\sigma(r)}$$

- σ is a permutation on $\{0, \dots, m-1\}$.

A new S-Box from Power Residue

- Step 3:

$$g^0 \quad g^1 \quad g^2 \quad g^3 \quad g^4 \quad g^5 \quad g^6 \quad g^7 \quad g^8 \quad g^9 \quad g^{10} \quad g^{11}$$

$$\downarrow$$

$$g^2 \quad \underline{g^4} \quad g^{11} \quad g^1 \quad g^{10} \quad \underline{g^0} \quad g^7 \quad g^9 \quad g^6 \quad \underline{g^8} \quad g^3 \quad g^5$$

$$S(g^{qm+r}) = g^{-qm+rm+\sigma(r)}$$

- σ is a permutation on $\{0, \dots, m-1\}$.

A new S-Box from Power Residue

- The S-box in Polocolo:

$$S(x) = g^{-qm+rm+\sigma(r)}$$

  - $m \in \{32,64,128,256,512,1024\}$
  - σ is randomly chosen, with "weak" permutations being discarded
- $g^{-qm+rm+\sigma(r)} = g^{-qm-r} \cdot g^{r+rm+\sigma(r)}$
- $S(x) = x^{-1} \cdot T\left[\left(\frac{x}{p}\right)_m\right]$ where $T = \left\{\left(g^{r(p-1)/m}, g^{r+rm+\sigma(r)}\right) \mid 0 \leq r < m\right\} \cup \{(0,0)\}$

Plonk Constraints for a S-Box

$$x = \underbrace{(g^q)}_{w_1}{}^m \cdot \underbrace{g^r}_{w_2}$$

$$y = \underbrace{g^{-qm}}_{w_3} \cdot \underbrace{g^{rm+\sigma(r)}}_{w_4}$$

- Witnesses:
  - $w_1 = g^q$
  - $w_2 = g^r$
  - $w_3 = g^{-qm}$
  - $w_4 = g^{rm+\sigma(r)}$
- Constraints:
  - $x = w_1^m \times w_2$ ($\log m + 1$ mult gates)
  - $w_1^m \times w_3 = 1$ (1 mult gate)
  - $(w_2, w_4) \in T$ (1 lookup gate) where $T = \{(g^r, g^{rm+\sigma(r)}) \mid 0 \le r < m\} \cup \{(0,0)\}$
  - $w_3 \times w_4 = 1$ (1 mult gate)

$\log m + 3$ mul gates & 1 table lookup → $\log m + 4$ gates in total

Efficient Linear Layers

- In general, linear layers in ZK-friendly hash function are defined as follows:

$$LinLayer(\vec{x}) = M \times \vec{x} + \vec{c}$$

- The constant $\vec{c}$ is randomly chosen
- The matrix M is MDS (Maximum Distance Separable) matrix

Efficient Linear Layers

- Multiplying a matrix M requires $t(t-1)$ add gates:
- $\begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 2x_1 + x_2 + x_3 \\ x_1 + 2x_2 + x_3 \\ x_1 + x_2 + 2x_3 \end{pmatrix}$, 6 gates
- Depending on the matrix, multiplying a matrix M can be done efficiently:
- $\begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} s + x_1 \\ s + x_2 \\ s + x_3 \end{pmatrix}$, where $s = x_1 + x_2 + x_3$, 5 gates

Heuristic Approach

- Find a matrix M with fewer add gates in $M \times \vec{x}$

Coeffs	witnesses
$(1 \quad 0 \quad 0 \quad 0 \quad 0)$	x_1
$(0 \quad 1 \quad 0 \quad 0 \quad 0)$	x_2
$(0 \quad 0 \quad 1 \quad 0 \quad 0)$	x_3
$(0 \quad 0 \quad 0 \quad 1 \quad 0)$	x_4
$(0 \quad 0 \quad 0 \quad 0 \quad 1)$	x_5

Heuristic Approach

- New witness is chosen by adding two randomly chosen witnesses

Coeffs	witnesses
$(1 \quad 0 \quad 0 \quad 0 \quad 0)$	x_1
$(0 \quad 1 \quad 0 \quad 0 \quad 0)$	x_2
$(0 \quad 0 \quad 1 \quad 0 \quad 0)$	x_3
$(0 \quad 0 \quad 0 \quad 1 \quad 0)$	x_4
$(0 \quad 0 \quad 0 \quad 0 \quad 1)$	x_5
$(6 \quad 1 \quad 0 \quad 0 \quad 0)$	$w_1 = 6x_1 + x_2$

Heuristic Approach

- New witness is chosen by adding two randomly chosen witnesses

Coeffs	witnesses
$(1\ \ 0\ \ 0\ \ 0\ \ 0)$	x_1
$(0\ \ 1\ \ 0\ \ 0\ \ 0)$	x_2
$(0\ \ 0\ \ 1\ \ 0\ \ 0)$	x_3
$(0\ \ 0\ \ 0\ \ 1\ \ 0)$	x_4
$(0\ \ 0\ \ 0\ \ 0\ \ 1)$	x_5
$(6\ \ 1\ \ 0\ \ 0\ \ 0)$	$w_1 = 6x_1 + x_2$
$(0\ \ 0\ \ 2\ \ 4\ \ 0)$	$w_2 = 2x_3 + 4x_4$

Heuristic Approach

- New witness is chosen by adding two randomly chosen witnesses

Coeffs	witnesses
$(1\ \ 0\ \ 0\ \ 0\ \ 0)$	x_1
$(0\ \ 1\ \ 0\ \ 0\ \ 0)$	x_2
$(0\ \ 0\ \ 1\ \ 0\ \ 0)$	x_3
$(0\ \ 0\ \ 0\ \ 1\ \ 0)$	x_4
$(0\ \ 0\ \ 0\ \ 0\ \ 1)$	x_5
$(6\ \ 1\ \ 0\ \ 0\ \ 0)$	$w_1 = 6x_1 + x_2$
$(0\ \ 0\ \ 2\ \ 4\ \ 0)$	$w_2 = 2x_3 + 4x_4$
$(3\ \ 0\ \ 0\ \ 0\ \ 8)$	$w_3 = 3x_1 + 8x_5$

Heuristic Approach

- New witness is chosen by adding two randomly chosen witnesses

Coeffs	witnesses
$(1\ \ 0\ \ 0\ \ 0\ \ 0)$	x_1
$(0\ \ 1\ \ 0\ \ 0\ \ 0)$	x_2
$(0\ \ 0\ \ 1\ \ 0\ \ 0)$	x_3
$(0\ \ 0\ \ 0\ \ 1\ \ 0)$	x_4
$(0\ \ 0\ \ 0\ \ 0\ \ 1)$	x_5
$(6\ \ 1\ \ 0\ \ 0\ \ 0)$	$w_1 = 6x_1 + x_2$
$(0\ \ 0\ \ 2\ \ 4\ \ 0)$	$w_2 = 2x_3 + 4x_4$
$(3\ \ 0\ \ 0\ \ 0\ \ 8)$	$w_3 = 3x_1 + 8x_5$
$(48\ \ 8\ \ 6\ \ 12\ \ 0)$	$w_4 = 8w_1 + 3w_2$

Efficient Linear Layers

- Generate enough witnesses

Coeffs	witnesses
$(1\ \ 0\ \ 0\ \ 0\ \ 0)$	x_1
$(0\ \ 1\ \ 0\ \ 0\ \ 0)$	x_2
$(0\ \ 0\ \ 1\ \ 0\ \ 0)$	x_3
$(0\ \ 0\ \ 0\ \ 1\ \ 0)$	x_4
$(0\ \ 0\ \ 0\ \ 0\ \ 1)$	x_5
$(6\ \ 1\ \ 0\ \ 0\ \ 0)$	$w_1 = 6x_1 + x_2$
$(0\ \ 0\ \ 2\ \ 4\ \ 0)$	$w_2 = 2x_3 + 4x_4$
$(3\ \ 0\ \ 0\ \ 0\ \ 8)$	$w_3 = 3x_1 + 8x_5$
$(48\ \ 8\ \ 6\ \ 12\ \ 0)$	$w_4 = 8w_1 + 3w_2$
…	…
$(39\ \ 6\ \ 10\ \ 28\ \ 8)$	$w_9 = w_7 + w_5$
$(174\ \ 28\ \ 32\ \ 80\ \ 16)$	$w_{10} = 2w_4 + 2w_9$
$(348\ \ 58\ \ 42\ \ 84\ \ 2)$	$w_{11} = 2w_8 + 7w_4$
$(39\ \ 4\ \ 54\ \ 100\ \ 44)$	$w_{12} = w_6 + 2w_8$
$(204\ \ 20\ \ 300\ \ 560\ \ 244)$	$w_{13} = 3w_5 + 5w_{12}$

Efficient Linear Layers

- Check if the matrix created by the last t witnesses is an MDS

Coeffs	witnesses
$(1\ \ 0\ \ 0\ \ 0\ \ 0)$	x_1
$(0\ \ 1\ \ 0\ \ 0\ \ 0)$	x_2
$(0\ \ 0\ \ 1\ \ 0\ \ 0)$	x_3
$(0\ \ 0\ \ 0\ \ 1\ \ 0)$	x_4
$(0\ \ 0\ \ 0\ \ 0\ \ 1)$	x_5
$(6\ \ 1\ \ 0\ \ 0\ \ 0)$	$w_1 = 6x_1 + x_2$
$(0\ \ 0\ \ 2\ \ 4\ \ 0)$	$w_2 = 2x_3 + 4x_4$
$(3\ \ 0\ \ 0\ \ 0\ \ 8)$	$w_3 = 3x_1 + 8x_5$
$(48\ \ 8\ \ 6\ \ 12\ \ 0)$	$w_4 = 8w_1 + 3w_2$
…	…
$(39\ \ 6\ \ 10\ \ 28\ \ 8)$	$w_9 = w_7 + w_5$
$(174\ \ 28\ \ 32\ \ 80\ \ 16)$	$w_{10} = 2w_4 + 2w_9$
$(348\ \ 58\ \ 42\ \ 84\ \ 2)$	$w_{11} = 2w_8 + 7w_4$
$(39\ \ 4\ \ 54\ \ 100\ \ 44)$	$w_{12} = w_6 + 2w_8$
$(204\ \ 20\ \ 300\ \ 560\ \ 244)$	$w_{13} = 3w_5 + 5w_{12}$

$$\begin{pmatrix} 39 & 6 & 10 & 28 & 8 \\ 174 & 28 & 32 & 80 & 16 \\ 348 & 58 & 42 & 84 & 2 \\ 39 & 4 & 54 & 100 & 44 \\ 204 & 20 & 300 & 560 & 244 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \end{pmatrix}$$

13 addition gates (naive : 20 gates)

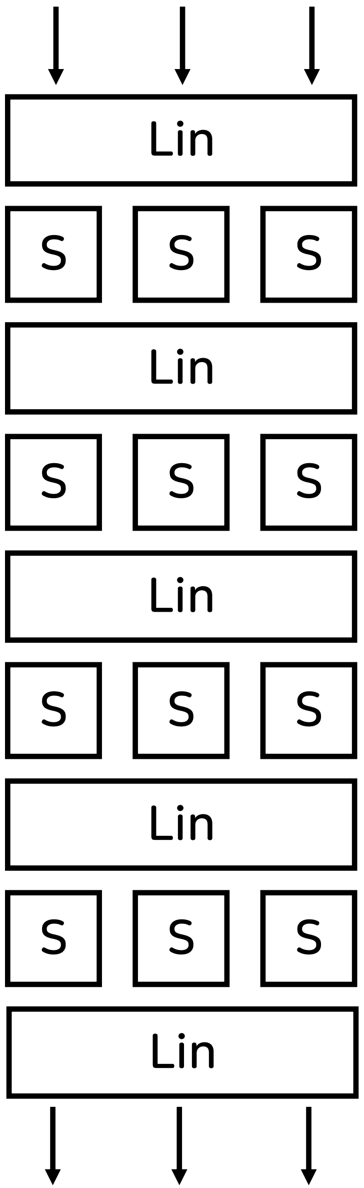
Plonk Gates of Linear Layers

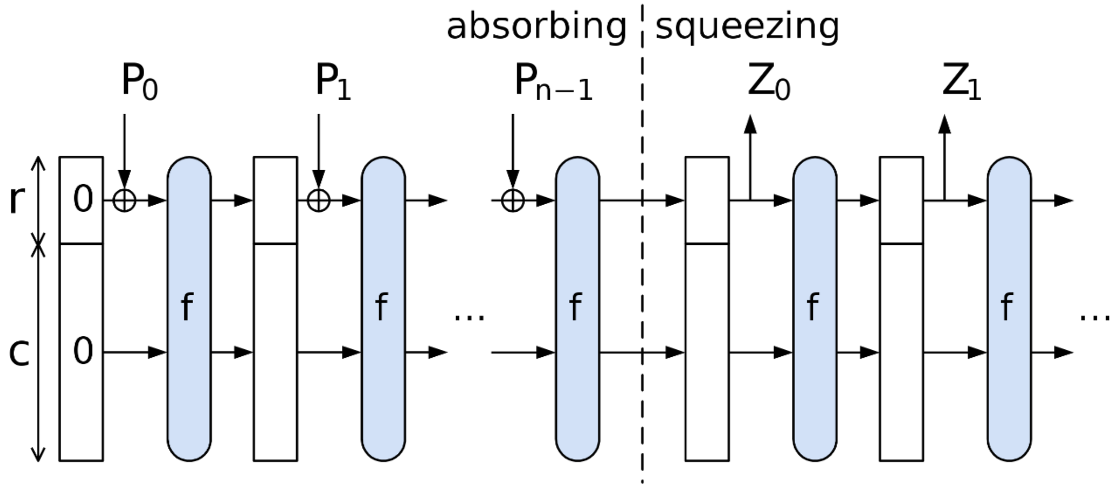
Hash functions	$t = 3$	$t = 4$	$t = 5$	$t = 6$	$t = 7$	$t = 8$
Polocolo	5	8	13	17	24	31
Reinforced Concrete	5	-	-	-	-	-
Rescue	6	12	20	30	42	56
Poseidon	6	12	20	30	42	56
Poseidon2	5	8	-	-	-	24
Griffin	5	8	-	-	-	24
Arion	6	12	16	20	24	28

"-" is used to indicate that the hash function is not defined for the given t
Red color is used to indicate that the corresponding matrix is not MDS

Polocolo Hash Function

- Designing a Polocolo$^{\pi}$ permutation using SPN structure:
  - ✓ Lin: A linear layer with a plonk-optimized matrix
  - ✓ S: S-box using the power residue method
- The permutation is converted to hash function using sponge construction





Security Analysis

- Statistical Attacks:
  - 4 rounds are sufficient to provide security
  - Include an additional 1 round as a security margin
- Algebraic Attacks:
  - Guessing power residues for each S-box is the most efficient
  - Our parameter selection ensures that the complexity of attack is at least 2^{160}

Performance

Plonk cost

Functions	$t = 3$	$t = 4$	$t = 6$	$t = 8$	Security margin
Polocolo	287	328	432	546	O
Reinforced Concrete	378	-	-	-	
Poseidon2	557	718	-	1416	
Rescue-Prime	420	528	768	1280	
Anemoi	-	340	490	688	
Polocolo (tight)	240	264	349	475	X
Griffin	243	348	-	960	
Arion	262	341	531	622	

"-" is used to indicate that the hash function is not defined for the given t

Conclusion

Conclusion

- We present a new ZK-friendly hash function Polocolo, based on the power residue method
- Polocolo achieves the smallest Plonk costs to existing schemes in this category
- We believe that the power residue method and the new linear layer optimized for Plonk can be applied to other ZK-friendly hash functions
- Third party analysis is always welcome: building an efficient system of equations against power residue method helps to understand the security of Polocolo accurately

Thank you!

Appendix

ZK-Unfriendly Operations

- Some operations are highly inefficient in ZKP (e.g., Compare, AND, XOR)
  - Common ZKP supports only addition/multiplication in $\mathbb{F}_p$ (256-bit prime field in ours)
- Constraints of XORing two 8-bit integers ($c = a \oplus b$) in ZKP:

1. Bit decomposition of a, b, c: 21 add gates

$$a = 2^0 a_0 + 2^1 a_1 + \cdots + 2^7 a_7$$
$$b = 2^0 b_0 + 2^1 b_1 + \cdots + 2^7 b_7$$
$$c = 2^0 c_0 + 2^1 c_1 + \cdots + 2^7 c_7$$

$$\begin{array}{rcccccccc}
 & a_7 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & a_0 \\
a = & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
 & b_7 & b_6 & b_5 & b_4 & b_3 & b_2 & b_1 & b_0 \\
b = & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0
\end{array}$$

— — — — XOR — — — —

$$\begin{array}{rcccccccc}
 & c_7 & c_6 & c_5 & c_4 & c_3 & c_2 & c_1 & c_0 \\
c = & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1
\end{array}$$

ZK-Unfriendly Operations

- Some operations are highly inefficient in ZKP (e.g., Compare, AND, XOR)
  - Common ZKP supports only addition/multiplication in $\mathbb{F}_p$ (256-bit prime field in ours)
- Constraints of XORing two 8-bit integers ($c = a \oplus b$) in ZKP:

1. Bit decomposition of a, b, c: 21 add gates
2. $a_i, b_i \in \{0,1\}$: 16 mul gates

$$a_i \times (1 - a_i) = 0 \text{ for } 0 \le i \le 7$$
$$b_i \times (1 - b_i) = 0 \text{ for } 0 \le i \le 7$$

$$\begin{array}{cccccccccc}
 & & a_7 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & a_0 \\
a & = & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
 & & b_7 & b_6 & b_5 & b_4 & b_3 & b_2 & b_1 & b_0 \\
b & = & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0
\end{array}$$

— — — — XOR — — — —

$$\begin{array}{cccccccccc}
 & & c_7 & c_6 & c_5 & c_4 & c_3 & c_2 & c_1 & c_0 \\
c & = & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1
\end{array}$$

ZK-Unfriendly Operations

- Some operations are highly inefficient in ZKP (e.g., Compare, AND, XOR)
  - Common ZKP supports only addition/multiplication in $\mathbb{F}_p$ (256-bit prime field in ours)
- Constraints of XORing two 8-bit integers ($c = a \oplus b$) in ZKP:

1. Bit decomposition of a, b, c: 21 add gates
2. $a_i, b_i \in \{0,1\}$: 16 mul gates
3. $c_i = a_i \oplus b_i$: 8 add gates & 4 mul gates

$$c_i = a_i + b_i - 2a_i \times b_i \text{ for } 0 \leq i \leq 3$$

$$\begin{array}{llllllllll}
 & a_7 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & a_0 \\
a = & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
 & b_7 & b_6 & b_5 & b_4 & b_3 & b_2 & b_1 & b_0 \\
b = & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0
\end{array}$$

— — — — XOR — — — —

$$\begin{array}{llllllllll}
 & c_7 & c_6 & c_5 & c_4 & c_3 & c_2 & c_1 & c_0 \\
c = & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1
\end{array}$$

29 add gates & 20 mult gates → 49 gates in total

Lookup Arguments

- Constraints of XORing two 8-bit integers ($c = a \oplus b$) in ZKP w/ lookup:
    1. Define a public table $T = \{(x, y, z) \mid x, y \in \{0, \ldots, 15\}, z = x \oplus y\}$

x	0000	0000	0000	0000	…	1111
y	0000	0001	0010	0011	…	1111
z	0000	0001	0010	0011	…	0000

$a = 0100\ 1001$

$b = 1101\ 0010$

— — – XOR — — -

$c = 1001\ 1011$

Lookup Arguments

- Constraints of XORing two 8-bit integers ($c = a \oplus b$) in ZKP w/ lookup:

    1. Define a public table $T = \{(x, y, z) \mid x, y \in \{0, \dots, 15\}, z = x \oplus y\}$
    2. 4-bit decomposition of a, b, c: 3 add gates

$$a = 16^0 a_0 + 16^1 a_1$$
$$b = 16^0 b_0 + 16^1 b_1$$
$$c = 16^0 c_0 + 16^1 c_1$$

$$a = 0100\ 1001$$
$$b = 1101\ 0010$$

— — – XOR — — -

$$c = 1001\ 1011$$

Lookup Arguments

- Constraints of XORing two 8-bit integers ($c = a \oplus b$) in ZKP w/ lookup:

  1. Define a public table $T = \{(x, y, z) \mid x, y \in \{0, \ldots, 15\}, z = x \oplus y\}$
  2. 4-bit decomposition of a, b, c: 3 add gates
  3. $a_i, b_i \in \{0, \ldots, 15\}$ and $c_i = a_i \oplus b_i$: 2 lookup gates

     $(a_i, b_i, c_i) \in T$ for $0 \leq i \leq 1$

$a = 0100\ 1001$

$b = 1101\ 0010$

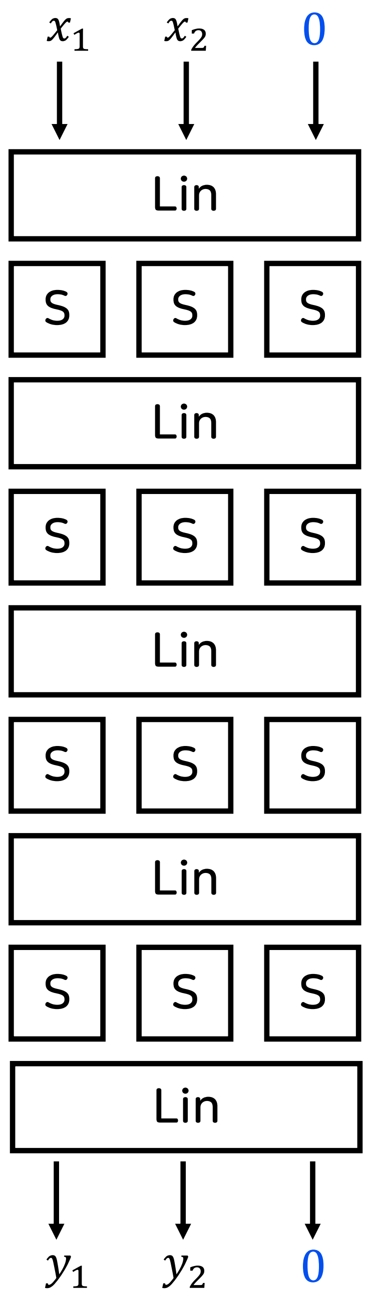
— — – XOR — — -

$c = 1001\ 1011$

3 add gates & 2 lookup gates → 5 gates in total (w/o lookup : 49 gates)

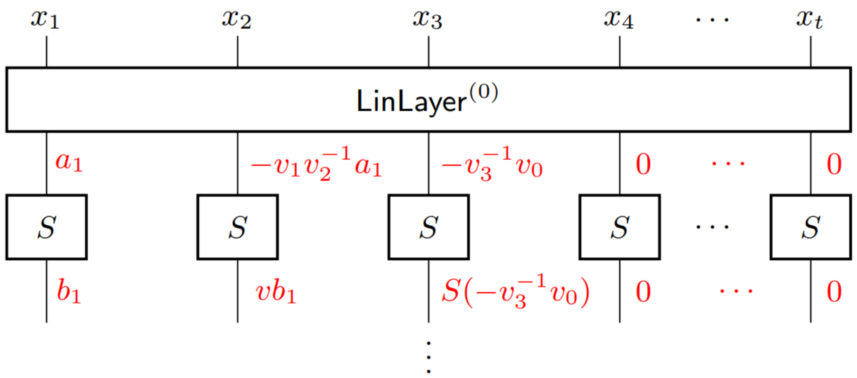
Security Analysis

- CICO (Constrained-input constrained-output) problem:
  - Find $x_1, \ldots, x_{t-1}, y_1, \ldots, y_{t-1}$ such that $P(x_1, \ldots, x_t) = (y_1, \ldots, y_t)$ for given permutation P
  - Once an attacker find those $x_1, \ldots, x_{t-1}, y_1, \ldots, y_{t-1}$, attacker can mount preimage/collision attack against the hash function derived from the permutation



Security Analysis

- Bypassing the First Round
  - Generic attack on hash functions constructed with sponge construction and SPN networks
  - By properly setting the output of the first S-box layer, the condition $x_t = 0$ is automatically satisfied
  - The number of S-boxes the attacker has to guess is reduced from tR to $t(R-1)$
  - Algebraic complexity is $m^{t(R-1)} \times C$, where C is the complexity of solving equation ($C \approx 2^{20}$)



Conditions of Permutation σ

- 1. Interpolating polynomial of $T = \left\{\left(g^{r(p-1)/m}, g^{r+rm+\sigma(r)}\right) \mid 0 \le r < m\right\} \cup \{(0,0)\}$ is dense and has max degree m
- 2. Interpolating polynomial of $T = \left\{\left(g^{r}, g^{rm+\sigma(r)}\right) \middle| \, 0 \le r < m\right\} \cup \{(0,0)\}$ is dense and has max degree m