Polocolo: A ZK-Friendly Hash Function Based on S-boxes Using Power Residues (Full Version)

Jincheol Ha, Seongha Hwang, Jooyoung Lee[⋆], Seungmin Park, and Mincheol Son

KAIST, Daejeon, Korea,
`{smilecjf,mathience98,hicalf,smpak,encrypted.def}@kaist.ac.kr`

Abstract. Conventional hash functions are often inefficient in zero-knowledge proof settings, leading to design of several ZK-friendly hash functions. On the other hand, lookup arguments have recently been incorporated into zero-knowledge protocols, allowing for more efficient handling of "ZK-unfriendly" operations, and hence ZK-friendly hash functions based on lookup tables.

In this paper, we propose a new ZK-friendly hash function, dubbed Polocolo, that employs an S-box constructed using power residues. Our approach reduces the numbers of gates required for table lookups, in particular, when combined with Plonk, allowing one to use such nonlinear layers over multiple rounds. We also propose a new MDS matrix for the linear layer of Polocolo. In this way, Polocolo requires fewer Plonk gates compared to the state-of-the-art ZK-friendly hash functions. For example, when $t = 8$, Polocolo requires 21% less Plonk gates compared to Anemoi, which is currently the most efficient ZK-friendly hash function, where t denotes the size of the underlying permutation in blocks of $\mathbb{F}_p$. For $t = 3$, Polocolo requires 24% less Plonk gates than Reinforced Concrete, which is one of the recent lookup-based ZK-friendly hash functions.

Keywords: Hash function, Zero-knowledge proof, Plonk, Lookup argument, Arithmetization-oriented cipher, ZK-friendly hash function, power residue, MDS matrix

1 Introduction

With increasing concern for privacy, privacy enhancing technologies such as multi-party computation, homomorphic encryption, and zero-knowledge proof (ZKP) are gaining attention in the cryptographic community, leading to active research in this area. ZKPs are cryptographic protocols that allow a prover to convince a verifier of the truth of a proposition without revealing any additional

⋆ This research was sponsored and supported by the collaboration project between LG Electronics, Inc. and KAIST(Korea Advanced Institute of Science and Technology).

information. Recently proposed ZK protocols enable the generation of *succint* proofs [41,9,32,16,10,64,63].

Hash functions are used in ZK proof systems for membership proof and commitment in a recursive proof. In these cases, hash function evaluations are checked in zero-knowledge settings, where the metrics to estimate efficiency are significantly different from traditional scenarios (e.g., rank-1 constraints system [41,10,17,19], algebraic intermediate representation [9], and Plonk arithmetization [32]). These metrics share a common feature that they highly depend on the complexity of the algebraic representation of the hash function in a large field $\mathbb{F}_p$. Standardized hash functions such as SHA-2 [52] and Keccak [13] are considered secure and fast in modern CPUs, while they are not "ZK-friendly" since they are not simply represented in a large field. This observation motivated research on a new design principle for ZK-friendly hash functions, and so far, numerous constructions have been proposed [1,3,36,33,15,6].

Most ZK-friendly hash functions are based on addition and multiplication operations that can be simply represented in a large field. Due to their simple algebraic structures, algebraic attacks become a major threat to ZK-friendly hash functions [8,44,2,5,24,62]. Most ZK-friendly hash functions proposed to date turned out to be vulnerable to certain algebraic attacks (and their variants) with their original specifications.

One new approach to constructing ZK-friendly hash functions that resist algebraic attacks is to use lookup tables: highly nonlinear operations can be easily implemented by table lookups, while they can be handled in certain ZKP settings such as Plonk. Reinforced Concrete [34], Tip5 [57], and Monolith [35] are examples of ZK-friendly hash functions using table lookup.

1.1 History of ZK-friendly Hash Functions

A common strategy for building ZK-friendly hash functions is to first design a permutation and then convert it to a hash function using the sponge construction [12]. The sponge construction, being provably secure, has been used to build many other ZK-friendly hash functions [1,3,36,33,15,6]. The Sponge construction will be reviewed in Section 2.2.

Beginning of ZK-friendly Hash Functions. MiMC [1] is the first cryptographic primitive over a large field targeted for ZK protocols. In MiMC, a permutation $\mathbb{F}_p^2 \to \mathbb{F}_p^2$ is defined as a Feistel cipher with a power map $x \mapsto x^\alpha$ as the underlying round function, where α is a small positive integer such that $\gcd(\alpha, p-1) = 1$[1]. Then it is turned into a hash function via the sponge construction. MiMC is the oldest in this line of research, rendering it widely used despite its relatively weak performance in terms of the number of multiplication gates in its circuit [61,45,51].

The same power map $S(x) = x^\alpha$ is also used in Poseidon [36] and its optimized version Poseidon2 [38]. One key difference is that Poseidon follows the

[1] The authors originally proposed only $\alpha = 3$, while their implementation also uses $\alpha \in \{5, 7\}$ according to the parameter p.

HADES design strategy [39], which can be seen as a variant of the substitution-permutation network (SPN). For the first and the last rounds (called the full rounds), S-boxes are applied to the entire state, while for the remaining rounds in between (called the partial rounds), only a single S-box is applied to a part of the state. Compared to the original SPN, the HADES design maintains its algebraic degree, and statistical attacks are also mitigated by the full rounds.

Legendre Symbol. Grendel [55] is based on an S-box

$$x \mapsto x^{\alpha} \times \left(\frac{x}{p}\right)$$

where the Legendre symbol is a high-degree power map (i.e., $\left(\frac{x}{p}\right) = x^{(p-1)/2}$), which is believed to be secure against algebraic attacks. However, Grendel turns out to be vulnerable to a preimage attack, where the Legendre symbol values are guessed and then the root-finding attack [37] is applied.

Inverse Power Map. Rescue [3] first uses a power map $x \mapsto x^{\alpha}$ and an inverse power map $x \mapsto x^{1/\alpha}$ alternately. By alternating these two S-boxes, the function achieves high algebraic degrees in both directions. Rescue-Prime [56] is a variant of Rescue, incorporating several simple modifications, including reduction of the security margin for the number of rounds from 100% to 50%.

Griffin [33] can be viewed as a combination of a SPN based on two invertible maps $x \mapsto x^{\alpha}$ and $x \mapsto x^{1/\alpha}$ and a Feistel-like structure, dubbed Horst; each round of Horst is defined as $(x, y) \mapsto (y \cdot G(x) + F(x), x)$ (where $G(x) \neq 0$) for certain round functions F and G. Compared to the Feistel structure, a half of the state is multiplied to the other half, increasing the overall algebraic degree while reducing the number of rounds required.

Anemoi [15] employs an S-box, called Flystel, which is based on the well-studied butterfly structure [50]. It is argued that the Flystel structure is secure against classical differential/linear cryptanalysis and also resistant to Gröbner basis attacks.

Arion [53] is based on the Generalized Triangular Dynamical System (GTDS). In this approach, S-boxes are aligned sequentially, increasing the overall algebraic degree. In particular, the rightmost branch in Arion employs an inverse power map.

Rescue, Griffin, Anemoi, and Arion were considered more efficient than MiMC and Poseidon (when they were first proposed) with fewer rounds. Later, Freelunch attacks [8] and resultant-based methods [62] have demonstrated that most of structures require more rounds than originally recommended. We will discuss these attacks further in Section 2.6.

Lookup Tables and the Base Expansion Method. Recently, some proof systems have incorporated table lookup [31,4,63,64,14,23]. The Reinforced Concrete hash function [34] targets proof systems that support lookup operations. Reinforced Concrete consists of three layers: Concrete, Bricks, and Bars. Among these, the Bars layer uses a lookup table. The Bars layer comprises three Bar functions arranged in parallel. In each Bar function, the input $x \in \mathbb{F}_p$ is converted to an element $(x_1, \dots, x_n) \in \mathbb{Z}_{s_1} \times \cdots \times \mathbb{Z}_{s_n}$ that satisfies

$$x = \sum_{i=1}^{n} \left(x_i \cdot \prod_{i<j} s_j \right),$$

and then a small S-box S is applied to each x_i, namely one has $y_i = S(x_i)$. Then the final output of the Bar function, denoted y, is defined as

$$y = \sum_{i=1}^{n} \left(y_i \cdot \prod_{i<j} s_j \right).$$

The Bars layer provides high resistance against algebraic attacks at a high cost. Specifically, the Bars layer requires 282 gates in total. Therefore, the designers of Reinforced Concrete opted to apply only a single Bars layer in the entire construction.

1.2 Motivation

Our motivation is to design an efficient ZK-friendly hash function using a lookup-based S-box. Without lookup tables in ZK-friendly hash functions, addition and multiplication are two main operations where only multiplication increases the overall algebraic degree, while it is also expensive in most ZK settings, so designers aim to optimize the use of multiplications to balance efficiency and security. Sometimes such attempts fail mainly due to algebraic attacks. For instance, Rescue [3], Griffin [33], and Anemoi [15] are based on a power map $x \mapsto x^{\alpha}$ and its inverse $x \mapsto x^{1/\alpha}$ for a small integer α. By alternating the two types of S-boxes, the entire function achieves a high algebraic degree in both directions. However, recent attacks [8,62] effectively exploit certain weaknesses of their algebraic structures.

Reinforced Concrete [34] is tailored for proof systems that support lookup operations, where lookup operations allow one to effectively mitigate algebraic attacks. In the Bars layer of Reinforced Concrete consists of three Bar functions arranged in parallel. In each Bar function, an element is first decomposed into smaller integers in order to be fed to a lookup table of a small size. The Bars layer provides robust resistance against algebraic attacks, and no known attacks have been found against Reinforced Concrete.

However, one drawback of this approach is the high cost required to evaluate the Bars layer in the ZK setting. Out of the 15 layers in Reinforced Concrete, only one is the Bars layer, while it still accounts for 75% of the constraints in the ZK setting. In general, it is desirable and natural to iterate a simple layer over multiple rounds. Reducing the cost of the Bars layer and iterating it over multiple rounds might enhance security and allow a trade-off between efficiency and security.

1.3 Our Contribution

In order to handle various "ZK-unfriendly" operations such as XOR, range check, and comparison, ZK protocols employ the *base expansion method*. The Bar function in Reinforced Concrete also utilizes the base expansion method to apply a lookup table to an element of $\mathbb{F}_p$; the input $x \in \mathbb{F}_p$ is converted into an n-tuple $(x_1, \ldots, x_n) \in \mathbb{Z}_{s_1} \times \cdots \times \mathbb{Z}_{s_n}$ in a similar way to base expansion. Then each component is fed to the underlying S-box (using table lookup). The outputs from the S-boxes are combined again to define the corresponding output of the Bar function. However, base expansion is significantly expensive in most ZK-settings.

In order to address this limitation, we propose an alternative approach, dubbed the *power residue method*, which allows one to efficiently apply a lookup table to the elements of $\mathbb{F}_p$ for a large prime $p(\approx 2^{256})$. A power residue can be seen as a generalization of the Legendre symbol; when a positive integer m divides $p-1$, the m-th power residue of x is defined as

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

The m-th power residue takes $m+1$ distinct values, so each possibility can be an input to a lookup table T of size $m+1$.

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

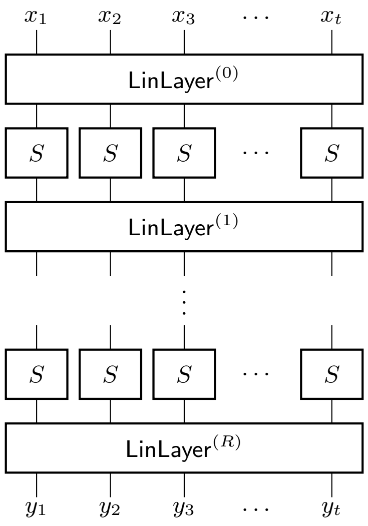
$$S(x) = x^{-1} \cdot T\left[\left(\frac{x}{p}\right)_m\right]. \tag{1}$$

By appropriately choosing T, this S-box can be made bijective. Moreover, this S-box is of a high degree that requires only 14 Plonk gates, for example, when $m = 1024$, which is significantly fewer than 94 gates required for the Bar function from Reinforced Concrete. Using this S-box, we propose a new permutation $\mathsf{Polocolo}^\pi$ (which represents $\underline{\text{Po}}$wer residue for $\underline{\text{lo}}$wer $\underline{\text{co}}$st table $\underline{\text{lo}}$okup) based on the paradigm of substitution-permutation networks (see Figure 1). This permutation is then transformed into a ZK-friendly hash function Polocolo based on the sponge construction [11].

When combined with Plonk, addition is not free, and multiplication of a matrix to the witness vector requires $t(t-1)$ Plonk gates, which is not negligible, where t denotes the size of the permutation in blocks of $\mathbb{F}_p$. With higher priority on efficiency of the linear layer, Griffin, Poseidon2, and Arion employ non-MDS matrices when $t > 4$. On the other hand, we employ a heuristic approach to find a new MDS matrix for Polocolo; the linear layer based on our matrix provides stronger security (against statistical attacks) without significantly degrading efficiency compared to the existing hash functions.

Based on comprehensive analysis on various attacks (mainly on algebraic attacks), we present recommended sets of parameters as given in Table 1. Table 2 compares the efficiency of Polocolo to existing constructions in terms of the number of Plonk gates in the Plonk setting. It turns out that Polocolo requires fewer Plonk gates compared to the state-of-the-art ZK-friendly hash functions.

For example, when $t = 8$, Polocolo requires 21% less Plonk gates compared to Anemoi [15], which is currently the most efficient ZK-friendly hash function, where t denotes the size of the underlying permutation in blocks of $\mathbb{F}_p$. For $t = 3$, Polocolo requires 24% less Plonk gates than Reinforced Concrete, which is one of the recent lookup-based ZK-friendly hash functions.



x_1 x_2 x_3 $\cdots$ x_t
$\mathsf{LinLayer}^{(0)}$
S S S $\cdots$ S
$\mathsf{LinLayer}^{(1)}$
S S S $\cdots$ S
$\mathsf{LinLayer}^{(R)}$
y_1 y_2 y_3 $\cdots$ y_t

Fig. 1: The $\mathsf{Polocolo}^{\pi}$ permutation over $\mathbb{F}_p^t$. Linear layers $\mathsf{LinLayer}^{(i)}$ are defined as affine transformations over $\mathbb{F}_p$, with distinct round constants for each layer.

Table 1: Recommended sets of parameters of Polocolo. t denotes the size of the permutation in blocks of $\mathbb{F}_p$, R denotes the round number, and m denotes the exponent in the power residue. The field $\mathbb{F}_p$ is defined as the scalar fields of the BLS12 and the BN254 elliptic curves.

t	3	4	5	6	7	8
R	6	5	5	5	5	5
m	1024	512	128	64	32	32

Table 2: The number of Plonk gates required for ZK-friendly hash functions where the field is the scalar fields of the BLS12 and the BN254 elliptic curves. "-" is used to indicate that the hash function is not defined for the given value of t. The Plonk gates for the hash functions above (resp. below) the dashed line are estimated using the original parameters (resp. modified parameters). The modified parameters are chosen to achieve 128-bit security against all the recent attacks with no security margin.

Hash functions	Plonk gates			
	$t = 3$	$t = 4$	$t = 6$	$t = 8$
Polocolo	287	308	402	546
Reinforced Concrete	378	-	-	-
Poseidon2	557	718	-	1416
Rescue-Prime	420	528	768	1280
Anemoi	-	340	490	688
Polocolo (tight)	240	264	325	443
Griffin	243	348	-	960
Arion	262	341	531	622

2 Preliminaries

Polocolo will be considered to be combined with Plonk, using lookup arguments and providing 128-bit security. Therefore the field $\mathbb{F}_p$ is defined as the scalar fields of the BLS12[2] and the BN254[3] elliptic curves.

2.1 Notation

For a positive integer n, we write $[n] = \{1, \ldots, n\}$. An n-dimensional vector over $\mathbb{F}_p$ is written as $\vec{x} = (x_1, \ldots, x_n)$, where x_i is the i-th element of $\vec{x}$. In particular, we write 0^n to denote the vector of all zeros. The concatenation of vectors $\vec{x}$ and $\vec{y}$ is written as $\vec{x} \parallel \vec{y}$. For a set S, we write $x \leftarrow_{\$} S$ to denote that x is chosen uniformly at random from S. For two integers a and b, we write $a \mid b$ to mean that a divides b.

2.2 Sponge Construction

The sponge construction [11], which was first proposed to construct Keccak [13], is a variable-input-variable-output function. The sponge function consists of the absorb phase and the squeeze phase. The state, which goes through the permutation, is separated into two parts: one is the inner part of c bits and the other is the outer part of r bits, where c and r are called the capacity and the rate of

[2] $p_{\mathrm{BLS12}} = \texttt{0x73eda753299d7d483339d80809a1d80553bda402fffe5bfeffffffff00000001}$

[3] $p_{\mathrm{BN254}} = \texttt{0x30644e72e131a029b85045b68181585d2833e84879b9709143e1f593f0000001}$

the construction, respectively. In the absorb phase, the input is separated into r-bit blocks with appropriate padding. When the state passes through the permutation, one input block is added to the outer part of the state. In the squeeze phase, when the state goes through the permutation, it extracts the outer part of r bits as the output block. Figure 2 illustrate how the sponge construction works with input $\vec{x} = (x_1, \ldots, x_m)$ and output $\vec{y} = (y_1, \ldots, y_l)$.

Most ZK-friendly hash functions [3,36,34,33] are built on the sponge construction, where the underlying permutation P is defined on $\mathbb{F}_p^t$ for a large prime p. In this case, rate r and capacity c are typically fixed in terms of the number of elements of $\mathbb{F}_p$. For $p \approx 2^{256}$, $c = 1$ is sufficient to guarantee 128-bit security.

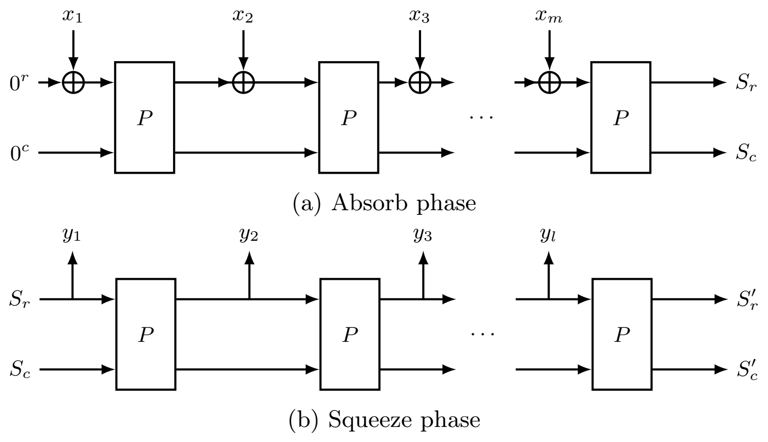


Fig. 2: Sponge construction based on a permutation P. S_r and S'_r are the leftmost r bits of state S. S_c and S'_c are the rightmost c bits of state S.

CICO problem. If P is modeled as a random permutation, then the sponge construction permits no generic attack for appropriate parameters r and c. On the other hand, the most effective attack against ZK-friendly hash functions exploiting certain characteristics of P is a preimage attack, and the security of P against the preimage attack can be estimated by the hardness of the constrained-input constrained-output (CICO) problem.

Problem 1 (CICO-1). Let $P : \mathbb{F}_p^t \to \mathbb{F}_p^t$ be a permutation, and let r and c be positive integers such that $r + c = t$. The CICO-1 problem is to find vectors $\vec{x} \in \mathbb{F}_p^r$ and $\vec{y} \in \mathbb{F}_p^c$ such that $P(\vec{x} \parallel 0^c) = 0^r \parallel \vec{y}$.

In Problem 1, $\vec{x}$ is a preimage of the output 0^r for the sponge construction of rate r. As an easier variant, Problem 2 is also considered.

Problem 2 (CICO-2). Let $P : \mathbb{F}_p^t \to \mathbb{F}_p^t$ be a permutation, and let r and c be positive integers such that $r + c = t$. The CICO-2 problem is to find $\vec{x} \in \mathbb{F}_p^r$ and $\vec{y} \in \mathbb{F}_p^r$ such that $P(\vec{x} \parallel 0^c) = \vec{y} \parallel 0^c$.

When we analyze the hardness of the CICO-2 problems on P, capacity c is typically set to 1 (and hence $r = t - 1$), in favor of an adversary. Solving the CICO-2 problems do not immediately lead to a preimage attack on the sponge construction. However it allows a partial preimage attack, and also leads a collision between inputs d and $(x, d - y)$ for any $d \in \mathbb{F}_p^r$, as in Figure 3. In Polocolo, the parameters are chosen to be secure against Problem 2.

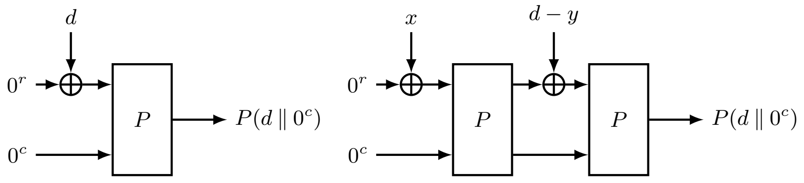


Fig. 3: Collision between two inputs d and $(x, d - y)$ when $P(\vec{x} \parallel 0^c) = \vec{y} \parallel 0^c$.

2.3 Power Residue

For a prime p and a positive integer m such that $m \mid p - 1$, the *m-th power residue* of $x \in \mathbb{F}_p$ is defined as

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

Let g be a generator of the multiplicative group $\mathbb{F}_p^*$. Any element $x \in \mathbb{F}_p^*$ can be uniquely represented as $x = g^{qm+r}$, where $0 \le q < \frac{p-1}{m}$ and $0 \le r < m$. Then the m-th power residue is determined by r since $\left(\frac{x}{p}\right)_m = x^{\frac{p-1}{m}} = g^{\frac{r(p-1)}{m}}$.

The quadratic residue is hard to predict when combined with addition in $\mathbb{F}_p$. For example, the Legendre PRF [20], defined as $F_K(x) = \left(\frac{x+K}{p}\right)_2$, has been widely studied [47,21,58,42]. Application of the Legendre symbol to ZK/MPC-friendly hash functions has also been explored [55,40]. In this line of research, we believe that power residues might also be useful in the design of new ZK-friendly hash functions.

2.4 Zero-Knowledge Proof

A zero-knowledge proof system is 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. For example, given the hash function h and the hash value y, the prover can convince the verifier that their knowledge of preimage x such that $h(x) = y$. Performance metrics for the protocol, including prover time, verifier time, and proof size, are influenced by the function h. Generally, the simpler the h can be expressed algebraically, the more efficient the protocol becomes. Among various protocols, we focus on Plonk, as Polocolo targets on the Plonk with a lookup argument.

Plonk arithmetization. Plonk [32] empolys its own arithmetization technique. For each addition and multiplication gate, let a_i, b_i represent the left and right inputs, respectively, and c_i represent the output. The gate is then represented by the following equation:

$$(q_{L_i})a_i + (q_{R_i})b_i + (q_{O_i})c_i + (q_{M_i})a_i b_i + q_{C_i} = 0. \quad (2)$$

The selectors $(q_{L_i}, q_{R_i}, q_{O_i}, q_{M_i}, q_{C_i})$ are determined by the type of the gate being used. The cost metric of Plonk is based on the number of gates. In particular, addition is not free unlike R1CS and AET. Plonk also has several variants, such as HyperPlonk [18] and Plonkup [49], which support custom gates and lookup gates.

2.5 Lookup Arguments in ZK Protocols

Originally, ZK protocols faced a common challenge: operations with high degrees in the native field $\mathbb{F}_p$ require significant overhead. For instance, the XOR operation between two elements $x, y \in \mathbb{F}_p$ for a large prime p cannot be easily represented as a low degree polynomial. To handle XOR in those protocols, the elements x and y must be decomposed into bit strings, and the XOR operation must be performed bit by bit. This approach introduces a substantial number of constraints, making such operations *ZK-unfriendly*.

To address these challenges, table lookup serves as an alternative approach. Using lookup arguments, a prover can demonstrate that an element e belongs to a public table T, with e being a private witness. After representing $x = \sum_i (16^i \cdot x_i)$ and $y = \sum_i (16^i \cdot y_i)$, The XOR operation $x \oplus y$ is then derived from $x \oplus y = \sum_i (16^i \cdot z_i)$ where $z_i = x_i \oplus y_i$. The correctness of z_i is ensured through a lookup of $(x_i, y_i, z_i) \in T$ in a pre-defined table $T = \{(a, b, c) \mid a, b \in \{0, \ldots, 15\}, c = a \oplus b\}$.

The concept of a lookup argument in zero-knowledge proofs was first introduced in Arya [14], a protocol designed to verify the correct execution of a program on public and private inputs with nearly linear time complexity for the prover. Since then, several protocols incorporating lookup arguments have been proposed [23,31,49,4,63,64].

In this paper, we focus primarily on Plonkup [49], which builds upon Plonk [32] by incorporating plookup [31]. Our efficiency metrics is the number of fan-in-two multiplication, addition, and lookup gates, with the scalar fields of the BLS12 and the BN254 elliptic curves. This metric is equivalent to the total number of gates in the Plonkup, and we will refer to this as *Plonk gates*.

Plonkup. In Plonkup, a new lookup selector, q_{K_i}, is introduced to indicate a lookup gate. Statements that the prover wants to prove are:

$$\begin{aligned}
&(q_{L_i})a_i + (q_{R_i})b_i + (q_{O_i})c_i + (q_{M_i})a_i b_i + q_{C_i} = 0,\\
&q_{K_i}(a_i + \zeta b_i + \zeta^2 c_i - f_i) = 0,\\
&f_i \in \{t_1, \ldots, t_n\},
\end{aligned}$$

where a_i, b_i, and c_i represent the left input, right input, and output, respectively; ζ is a randomly chosen compression factor, and $t_i = x_i + \zeta y_i + \zeta^2 z_i$, where (x_i, y_i, z_i) is the i-th element of the public table T.

If the i-th gate is an arithmetic gate, then $q_{K_i} = 0$, and the other selectors are chosen properly according to the gate type. Conversely, if the i-th gate is a lookup gate, then $q_{K_i} = 1$, and all the other selectors are zero.

The prover's complexity is proportional to max{number of gates, table size}. Since the number of gates in a circuit is the sum of arithmetic and lookup gates, the complexity remains unchanged regardless of whether a gate is arithmetic or a lookup gate as long as their total numbers are the same.

2.6 Algebraic Attacks

Since ZK-friendly hash functions are expressed simply in $\mathbb{F}_p$, they are particularly susceptible to algebraic attacks [8,44,2,5,24,62]. In this section, we discuss the Gröbner Basis Attack.

Gröbner Basis Attack. The Gröbner basis attack is to solve a system of equations by computing its Gröbner basis. The attack basically consists of the following steps.

1. Compute a Gröbner basis in the *grevlex* (graded reverse lexicographic) order.
2. Change the order of terms to obtain a Gröbner basis in the *lex* (lexicographic) order.
3. Find a univariate polynomial in this basis and solve it.
4. Substitute this solution into the Gröbner basis and repeat Step 3 for the next variable.

It is known that a univariate polynomial exists in the Gröbner basis under the *lex* order when the system of equations has only finitely many solutions in its algebraic closure. Furthermore, the Gröbner basis remains a Gröbner basis of the reduced system obtained by substituting a specific solution for one variable, thereby producing a univariate polynomial for the subsequent variable. We refer to [54] for more details on Gröbner basis computation.

The complexity of computing a Gröbner basis can be estimated using the *degree of regularity* of the system of equations. This metric provides an upper bound on the degree of the polynomials that arise during the computation of a Gröbner basis using algorithms such as F_4 [26] and F_5 [27]. Given the degree of regularity d_{reg}, the complexity of computing a Gröbner basis of the system is known to be

$$O\left(\binom{n + d_{reg}}{d_{reg}}^{\omega}\right)$$

where ω is the matrix multiplication exponent, with $2 \leq \omega \leq 3$. Consider a system of m equations in n variables $\{f_i\}_{i=1}^{m}$ over a field $\mathbb{F}_p$ for some large prime p. Let d_i denote the degree of f_i for $i = 1, 2, \ldots, m$. Then $1+\sum_{i=1}^{m}(d_i-1)$ provides an upper bound of the degree of regularity. However, there is no theoretical method to determine degree of regularity for a system derived from a symmetric primitive. In most cases, the degree of regularity observed experimentally is lower than the theoretical upper bound. In some instances, this upper bound is taken as the degree of regularity [36,3], or it is estimated based on the experiments conducted with small parameters [33,15]. The limitation of both approaches is that their complexity cannot be theoretically proven.

The complexity of the FGLM algorithm for term order change is $O(nD_I^3)$, where n is the number of variables and D_I is the ideal degree, representing the number of solutions of the system counted with multiplicity. A probabilistic variant [28] improves this to $O(nD_I^\omega)$. Additionally, the sparse FGLM algorithm [30] has $O(D_I(N_1 + n\log(D_I)))$ complexity, where N_1 is the number of non-zero entries in the $D_I \times D_I$ multiplication matrix.

3 New S-box Design Using Power Residues

3.1 Design Rationale

The base expansion method in Reinforced Concrete is a common approach to mapping the elements of $\mathbb{F}_p$ to size-limited tables. However, a significant drawback of this approach is the high computational cost required to evaluate the Bars layer in the ZK setting. Although the Bars layer is used only once out of the 15 layers in Reinforced Concrete, it accounts for 75% of the constraints in the ZK setting. In general, it is standard to design a permutation or a block cipher by iterating a simpler layer over multiple rounds, while it seems infeasible to reduce the cost of the Bars layer.

In order to address this limitation, we propose the power residue method. This method allows one to apply a lookup table to the elements of $\mathbb{F}_p$ (for a large prime $p \approx 2^{256}$) with fewer gates. The m-th power residue takes $m+1$ distinct values, enabling each possible value to serve as an input to a lookup table T of size $m+1$, with $O(\log m)$ Plonk gates.

3.2 Specification

Suppose that p is prime, $m = 2^l$ is a divisor of $p-1$ (which is a power of two), g is a generator of the multiplication group $\mathbb{F}_p^*$, and σ is a permutation on $\{0, 1, \ldots, m-1\}$. Then S-box $S : \mathbb{F}_p \to \mathbb{F}_p$ is defined as follows:

$$S(x) = \begin{cases} 0 & \text{if } x = 0, \\ g^{-qm+rm+\sigma(r)} & \text{if } x = g^{qm+r}, \text{ where } 0 \le q < \frac{p-1}{m} \text{ and } 0 \le r < m. \end{cases} \tag{1}$$

The purpose of having rm in the exponent is to ensure that the outputs of table T defined as

$$T\left[\left(\frac{x}{p}\right)_m\right] = g^{rm+\sigma(r)+r}$$

differ for distinct values of r. With this condition, the attacker's complexity in guessing the power residue is estimated as $O(m)$. Additionally, we emphasize that the S-box remains well-defined as long as m is a divisor of $p-1$, and importantly, m does not need be a power of two. However, we restrict our consideration to power of two because, in ZK protocols, the prime p is typically chosen such that $p-1$ contains many factors of 2, enabling the use of FFT. This choice also aligns with achieving optimal efficiency in Plonk cost.

We now prove the bijectivity of the S-box S as follows.

Lemma 1 (Bijectivity). *Let $S : \mathbb{F}_p \to \mathbb{F}_p$ be the function defined in (1). Then S is bijective.*

Proof. It is sufficient to prove that S is injective since the domain and the codomain of S are an identical finite set. Suppose that $x \neq x' \in \mathbb{F}_p$. If either $x = 0$ or $x' = 0$, then it is obvious that $S(x) \neq S(x')$.

Now we consider nonzero distinct x and x'. Let $x = g^{qm+r}$ and $x' = g^{q'm+r'}$ where $0 \leq q, q' < \frac{p-1}{m}$ and $0 \leq r, r' < m$. If $S(x) = S'(x)$, then we have $g^{(q-q'-r+r')m} = g^{\sigma(r)-\sigma(r')}$, and hence

$$(q - q' - r + r')m \equiv \sigma(r) - \sigma(r') \pmod{p-1}.$$

Since $m \mid (p-1)$, $-m < \sigma(r) - \sigma(r') < m$ and σ is a permutation, we have $\sigma(r) = \sigma(r')$, and hence $r = r'$. Furthermore, since

$$(q - q')m \equiv 0 \pmod{p-1}$$

and $-p < (q - q')m < p$, we have $q = q'$. Since $r = r'$ and $q = q'$, we have $x = x'$, which contradicts the assumption that $x \neq x'$. So we conclude that $S(x) \neq S(x')$, which completes the proof. $\square$

3.3 Plonk Constraints

Our S-box is designed with a focus on the efficiency in the Plonk evaluation. First, we define a lookup table T, as given in Table 3. Next, we establish the constraints to prove that $y = S(x)$, as defined in Constraints (3.3). Overall, each S-box requires $\log m + 3$ multiplications and 1 table lookup. The completeness and the soundness of the constraints are proved in Lemma 2 and Lemma 3, respectively.

w_{l+2}	g^0	g^1	$\dots$	g^{m-1}	0
w_{l+4}	$g^{\sigma(0)}$	$g^{m+\sigma(1)}$	$\dots$	$g^{(m-1)m+\sigma(m-1)}$	0

Table 3: Lookup table T for Plonk constraints.

$$\begin{aligned}w_i \cdot w_i &= w_{i+1} \text{ for } 1 \le i \le l,\\ w_{l+1} \cdot w_{l+2} &= x,\\ w_{l+1} \cdot w_{l+3} &= 1,\\ (w_{l+2}, w_{l+4}) &\in T,\\ w_{l+3} \cdot w_{l+4} &= y,\end{aligned} \tag{2}$$

where $l = \log m$.

Lemma 2 (Completeness). *Let $S : \mathbb{F}_p \to \mathbb{F}_p$ be the function defined in (1). For any $x \in \mathbb{F}_p$ and $y = S(x)$, it is possible to assign $w_1, \ldots, w_{l+4} \in \mathbb{F}_p$ such that the Constraints (3.3) are satisfied.*

Proof. If $x = 0$, let $w_1 = \cdots = w_{l+1} = w_{l+3} = 1$ and let $w_{l+2} = w_{l+4} = 0$. With this assignment, all the constraints are satisfied.

If $x \neq 0$, then $x = g^{qm+r}$ where $0 \le q < \frac{p-1}{m}$ and $0 \le r < m$. In this case, let $w_i = (g^q)^{2^{i-1}}$ for $i = 1, \ldots, l+1$, $w_{l+2} = g^r$, $w_{l+3} = g^{-qm}$, and $w_{l+4} = g^{rm+\sigma(r)}$. With this assignment, all the constraints are satisfied. □

Lemma 3 (Soundness). *Let $S : \mathbb{F}_p \to \mathbb{F}_p$ be the function defined in (1). For any set of witnesses $(x, w_1, \ldots, w_{l+4}, y)$ satisfying the Constraints (3.3), $y = S(x)$.*

Proof. Let $(x, w_1, \ldots, w_{l+4}, y)$ be the set of witnesses that satisfies the Constraints (3.3). We distinguish two cases.

First, suppose that $x = 0$, and hence $S(0) = 0$. In this case, we have $w_{l+2} = 0$ since $w_{l+1} \cdot w_{l+2} = 0$ and $w_{l+1} \cdot w_{l+3} = 1$, which implies $w_{l+1} \neq 0$. Since $(w_{l+2}, w_{l+4}) \in T$, it follows that $w_{l+4} = 0$. Therefore, $y = w_{l+3} \cdot w_{l+4} = 0$.

Next, suppose that $x = g^{qm+r} \neq 0$ for some q and r, where $0 \le q < \frac{p-1}{m}$ and $0 \le r < m$. In this case, suppose that two sets of witnesses $(x, w_1, \ldots, w_{l+4}, y)$ and $(x, w'_1, \ldots, w'_{l+4}, y')$ satisfy the Constraints (3.3). Then, since $w_{l+1} \cdot w_{l+2} = 1$ and $w'_{l+1} \cdot w'_{l+2} = 1$, we have w_{l+1}, w_{l+2}, w'_{l+1}, and w'_{l+2} are all nonzero. So, let $w_{l+1} = g^a$, $w_{l+2} = g^b$, $w'_{l+1} = g^c$, and $w'_{l+2} = g^d$ for some a, b, c, d (where $0 \le a, b, c, d < p$). Since $(w_{l+2}, w_{l+4}) \in T$ and $(w'_{l+2}, w'_{l+4}) \in T$, we have $0 \le b, d < m - 1$. Moreover, since $w_{l+1} = w_1^m$ and $w'_{l+1} = (w')_1^m$, we have $m \mid a$ and $m \mid c$.

Since $x = g^a \cdot g^b = g^c \cdot g^d$, we have $g^{a-c} = g^{d-b}$. Since $-m < d-b < m$ and $m \mid (a-c)$, we have $b = d$, and hence $a = c$. So we have $(w_{l+1}, w_{l+2}) = (w'_{l+1}, w'_{l+2})$. By the constraints, we also have $(w_{l+3}, w_{l+4}) = (w'_{l+3}, w'_{l+4})$. In other words, for the set of witnesses $(x, w_1, \ldots, w_{l+4}, y)$ satisfying Constraints (3.3),

$$(w_{l+1}, w_{l+2}, w_{l+3}, w_{l+4})$$

is uniquely determined.

From the proof of Lemma 2, we see that $w_{l+3} = g^{-qm}$ and $w_{l+4} = g^{rm+\sigma(r)}$. Thus, we conclude that $y = w_{l+3} \cdot w_{l+4} = g^{-qm+rm+\sigma(r)} = S(x)$. □

4 Polocolo Hash Function

We design the $\mathsf{Polocolo}^\pi$ permutation by incorporating the S-box introduced in Section 3 within a substitution-permutation networks (SPN). This permutation is then converted into the Polocolo hash function using the sponge construction [11].

4.1 Linear Layers

A linear layer in a ZK-friendly hash function is typically defined as

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

for a matrix $M \in \mathbb{F}_p^{t\times t}$ and a constant vector $\vec{c}$. In most block ciphers and permutations, an MDS matrix is chosen for M since such a matrix provides nice diffusion properties. On the other hand, certain ZK-friendly hash functions such as Arion [53], Griffin [33] and Poseidon2 [38] employ non-MDS matrices for their linear layers. Instead of taking advantage of MDS matrices in terms of resilience against statistical attacks, they opted for non-MDS matrices that provide better performance in the Plonk setting, where addition is not free. Specifically, multiplication of a $t \times t$ matrix by a vector of size t requires $t(t-1)$ additions, which becomes costly for a large t.

In Polocolo, we chooses an MDS matrix M since the round number is relatively small with the heavy S-box. On the other hand, we take a heuristic approach to minimize Plonk constraints in linear layers. Depending on the matrix structure, matrix multiplication might can be performed with fewer than $t(t-1)$ addition gates. For example:

$$M_3 = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix} \text{ and } M_4 = \begin{pmatrix} 5 & 7 & 1 & 3 \\ 4 & 6 & 1 & 1 \\ 1 & 3 & 5 & 7 \\ 1 & 1 & 4 & 6 \end{pmatrix}$$

require only 5 and 8 addition gates, respectively. These matrices are introduced in [22], and we use them for $t = 3, 4$. For $t \in \{5, 6, 7, 8\}$, M is generated using our heuristic algorithm. Details of the algorithm, matrices, and the corresponding constraints are provided in Appendix A.1.

The numbers of addition gates for the linear layers of various ZK-friendly hash functions are shown in Table 4. Our matrix provides stronger security (against statistical attacks) without significantly degrading efficiency compared to the matrices used in existing hash functions.

4.2 Specification

A permutation $\mathsf{Polocolo}^\pi$ with R rounds is defined as:

$$\mathsf{Polocolo}^\pi := \mathsf{LinLayer}^{(R)} \circ \mathsf{SBoxLayer} \circ \cdots \circ \mathsf{LinLayer}^{(1)} \circ \mathsf{SBoxLayer} \circ \mathsf{LinLayer}^{(0)}$$

Table 4: The number of addition gates for linear layers in each hash function. "-" is used to indicate that the hash function is not defined for the given value of t. Poseidon2-f (resp. Poseidon2-p) refers to the full (resp. partial) rounds in Poseidon2. ✗ is used to indicate that the corresponding matrix is not MDS.

Hash functions	Addition gates					
	$t=3$	$t=4$	$t=5$	$t=6$	$t=7$	$t=8$
Polocolo	5	8	13	17	24	31
Reinforced Concrete	5	-	-	-	-	-
Rescue, Rescue-Prime	6	12	20	30	42	56
Poseidon	6	12	20	30	42	56
Poseidon2-f	5	8	-	-	-	24✗
Poseidon2-p	5	7✗	-	-	-	15✗
Griffin	5	8	-	-	-	24✗
Arion	6	12	16✗	20✗	24✗	28✗

where $\mathsf{LinLayer}^{(i)}$, $i = 0, 1, \ldots, R$, are linear layers and $\mathsf{SBoxLayer}$ is a non-linear layer.

$\mathsf{LinLayer}^{(i)}$ is defined as follows:

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

where matrix M depends on the parameter t, and the constant vector $\vec{c}^{(i)}$ is chosen uniformly at random from $(\mathbb{F}_p)^t$. For $i = R$, we have that $\vec{c}^{(i)} = \{0, \ldots, 0\}$ because it does not affect the security. The concrete specification of M is given in Appendix A.1.

$\mathsf{SBoxLayer}$ is defined as follows.

$$\mathsf{SBoxLayer}(\vec{x}) = (S(x_1), S(x_2), \ldots, S(x_t))$$

where S-box S is defined in Section 3. The underlying permutation σ used in the S-box is chosen uniformly at random from the set of the permutations on $\{0, 1 \ldots, m-1\}$ subject to the following constraints:

1. The function f derived by interpolating $(g^r, g^{rm+\sigma(r)})$ for $r = 0, \ldots, m-1$ has the maximum degree $m-1$, where the coefficients of x^i, $i = 0, \ldots, m-1$, are all nonzero.
2. The function h derived by interpolating $(g^{r(p-1)/m}, g^{r(m+1)+\sigma(r)})$ for $r = 0, \ldots, m-1$ has the maximum degree $m-1$, where the coefficients of x^i, $i = 0, \ldots, m-1$, are all nonzero.

We consider these constraints when choosing σ in order to enhance its security against algebraic attacks. The details are discussed in Section 5.2. For each parameter t, the round number R and the table size m are recommended as in Table 1. The SageMath code that generates constant vectors $\vec{c}^{(i)}$ and permutation σ is also available online.[4]

[4] https://github.com/KAIST-CryptLab/Polocolo/blob/main/param_gen.sage

5 Security Analysis

In this section, we provide security analysis of Polocolo. The attacks are classified into two categories: statistical attacks and algebraic attacks. In statistical attacks, we determine the minimum round numbers required for Polocolo such that any statistical characteristic requires at least 2^{128} permutation evaluations. In algebraic attacks, we focus on the CICO problem, which is related to preimage attacks. A secure hash function must resist both preimage and collision attacks. While collision attacks are intuitively easier than preimage attacks, it remains unclear how to model a polynomial system for collision attacks without implying preimage attacks. For this reason, our focus is put on preimage attacks [8,44,2,5,24,62].

From the statistical attack analysis, we conclude that 4 rounds are sufficient to provide security. For the recommended parameters in Table 1, we include an additional 1 round as a security margin, so $R \geq 5$ is recommended. Regarding algebraic attacks, the most effective method involves guessing the power residues for each S-box and solving the univariate equations, as described in Guessing Power Residues for Each S-box section. Our parameter selection ensures that the complexity of this attack is at least 2^{160}.

5.1 Statistical Attacks

In this section, we analyze the security of Polocolo against various statistical attacks, including classical differential attack, truncated differential attack, boomerang attack, rebound attack, and linear cryptanalysis. Our analysis shows that Polocolo remains secure against statistical attacks with a minimum of 4 rounds.

Classical Differential Attack. To analyze differential attack on Polocolo, we need to determine the probability distribution of input and output differences. The differential probability (DP) of an output difference β given an input difference α and the permutation P is given as

$$\mathrm{DP}_P(\alpha, \beta) = \frac{|\{x \in \mathbb{F}_p : P(x+\alpha) - P(x) = \beta\}|}{p}.$$

Before calculating the differential probability, we assume that all rounds are independent of each other. Since the rounds are independent, the differential probability of differential trail is the product of the differential probabilities of each round. We now prove a Lemma 4, which bounds the differential probability of SBoxLayer.

Lemma 4. *Let p be a prime and $S : \mathbb{F}_p \to \mathbb{F}_p$ be defined as Equation 1. Given any $\alpha, \beta \in \mathbb{F}_p$, we have*

$$DP_S(\alpha, \beta) \leq \frac{2m^2+2}{p}.$$

Proof. For any $x \in \mathbb{F}_p \setminus \{0, -\alpha\}$, both x and $x+\alpha$ can be uniquely expressed as $x = g^{qm+r}$, and $x + \alpha = g^{q'm+r'}$, where $0 \le q, q' < \frac{p-1}{m}$ and $0 \le r, r' < m$. Then,

$$S(x) = g^{-qm+\sigma(r)+mr} = x^{-1} g^{(m+1)r+\sigma(r)}$$

and

$$S(x+\alpha) = g^{-q'm+\sigma(r')+mr'} = (x+\alpha)^{-1} g^{(m+1)r'+\sigma(r')}.$$

Therefore, $S(x+\alpha) - S(x) = \beta$ if and only if

$$x^{-1} g^{(m+1)r+\sigma(r)} - (x+\alpha)^{-1} g^{(m+1)r'+\sigma(r')} = \beta.$$

Multiplying both sides by $x(x+\alpha)$, we get:

$$(x+\alpha) g^{(m+1)r+\sigma(r)} - x g^{(m+1)r'+\sigma(r')} = \beta x(x+\alpha),$$

which has at most two solutions for each pair (r, r'). Thus, we have:

$$\mathrm{DP}_S(\alpha, \beta) \le \frac{2m^2+2}{p},$$

where the additional two accounts for the cases where $x \in \{0, -\alpha\}$. $\square$

Hence, the minimum number of rounds required to provide security against differential attack with security parameter κ is given by:

$$\left(\frac{2m^2+2}{p}\right)^{(1+t)\left\lfloor \frac{R}{2} \right\rfloor} \le 2^{-2.5\kappa},$$

since Polocolo uses MDS matrix. The factor of 2.5 is included to account the clustering effect, as considered in other ZK-friendly hash functions [33,3]. Therefore, 2 rounds are sufficient to prevent differential cryptanalysis.

Truncated Differential Attack. Truncated differential attack [43] is a variant of the differential attack. Even if a truncated differential exists in a single round, our MDS matrix effectively mixes the affected part with other parts of state. As a result, more than two rounds are sufficient to render the truncated differential attack impossible.

Boomerang Attack. Boomerang attack [59] divides the cipher into two parts and uses differentials to attack each part independently. Let E_1 and E_2 represent the first and second parts of the cipher, respectively. According to the Boomerang attack, the probability of successfully attacking the cipher is at most $p_1 p_2^2 p_1'$, where p_1, p_2, and p_1' represent the probabilities of differentials occuring in E_1, E_2, and E_1^{-1}, respectively.

In our case, each single round has a differential probability of at most $(2m^2 + 2)/p$. Therefore, the Boomerang attack has a success probability of at most $(2m^2+2)^3/p^3$. This probability is significantly smaller than that of a classic differential attack.

Rebound Attack. Rebound attack [48] consists of three phases: one inbound phase and two outbound phases. In the outbound phases, the attacker attempts to find a (truncated) differential characteristic with a high probability. While such a (truncated) differential might exist within a single round, it cannot be extended beyond one round because the maximum distance separable (MDS) matrix ensures maximum diffusion. This means that each outbound phase cannot span more than one round. In the inbound phase, the attacker aims to find many solutions that can connect to the outbound phases. However, similarly to a classical differential attack, the attacker cannot find a differential characteristic that spans more than one round. Therefore, rebound attack cannot be effective with more than three rounds, meaning that four rounds are sufficient to defend against this attack.

Linear Cryptanalysis. In the binary fields, linear cryptanalysis [46] searches for linear combinations of input, output, and key bits that are biased toward 0 or 1. This approach has been generalized to prime fields [7], where it examines linear relations among input, output, and key elements. We define the correlation and linear probabilities of the S-box as follows:

Definition 1. *Let p be a prime and $a, b \in \mathbb{F}_p$ be non-zero. For a function $S : \mathbb{F}_p \to \mathbb{F}_p$, the correlation of (a, b) with respect to a function S is defined as*

$$|\mathsf{Corr}_S(a,b)| = \frac{1}{p} \cdot \left| \sum_{x \in \mathbb{F}_p} \exp\left(\frac{2\pi i}{p}\left(ax + bS(x)\right)\right) \right|. \tag{1}$$

Also, the linear probabilities of masks (a, b) with respect to a function S is defined as:

$$\mathsf{LP}_S(a,b) = |\mathsf{Corr}_S(a,b)|^2. \tag{2}$$

We do not provide a theoretical bound for the linear probabilities of Polocolo. Instead, we offer informal arguments supported by experimental results. Let p be an odd prime, and $S : \mathbb{F}_p \to \mathbb{F}_p$ be defined as Equation 1. Define N_r as a set of elements in $\mathbb{F}_p$, where each element is expressed in the form $x = g^{qm+r}$. In other words:

$$N_r = \{g^r, g^{m+r}, \dots, g^{p-1-m+r}\}.$$

Then, by definition:

$$\begin{aligned} |\mathsf{Corr}_S(a,b)| &= \frac{1}{p} \cdot \left| \sum_{x \in \mathbb{F}_p} \exp\left(\frac{2\pi i}{p}\left(ax + bS(x)\right)\right) \right| \\ &\leq \frac{1}{p} \cdot \sum_{0 \leq r < m} \left| \sum_{x \in N_r} \exp\left(\frac{2\pi i}{p}\left(ax + bS(x)\right)\right) \right|. \end{aligned}$$

Since $S(x) = x^{-1} g^{(m+1)r+\sigma(r)}$ and the term $g^{(m+1)r+\sigma(r)}$ is constant when r is fixed, we have

$$\left| \sum_{x \in N_r} \exp\left(\frac{2\pi i}{p}\left(ax + bS(x)\right)\right) \right| = \left| \sum_{x \in N_0} \exp\left(\frac{2\pi i}{p}\left(a'x + b'x^{-1}\right)\right) \right|,$$

for some $a', b' \in \mathbb{F}_p \setminus \{0\}$.

The similar form

$$\left|\sum_{x\in\mathbb{F}_p} \exp\left(\frac{2\pi i}{p}\left(a'x+b'x^{-1}\right)\right)\right|$$

is bounded by

$$\left|\sum_{x\in\mathbb{F}_p} \exp\left(\frac{2\pi i}{p}\left(a'x+b'x^{-1}\right)\right)\right| \leq 2\sqrt{p},$$

as introduced in [60]. However, we have no theoretical bounds when the condition of x is $x \in N_0$, instead of $x \in \mathbb{F}_p$. While $\sum_{x\in N_0} \exp\left(\frac{2\pi i}{p}\left(a'x+b'x^{-1}\right)\right)$ is a partial sum of $\sum_{x\in\mathbb{F}_p} \exp\left(\frac{2\pi i}{p}\left(a'x+b'x^{-1}\right)\right)$, and no theoretical bounds exist for it, we believe the probability of a significant bias occuring when $x \in N_0$ is very low, given that p is large value. Therefore, we make the following conjecture.

Conjecture 1. Let p be a prime, m be a divisor of $p-1$, $N_0 = \{g^0, g^m, \ldots, g^{p-1-m}\}$ be a set of elements in $\mathbb{F}_p$ and $S : \mathbb{F}_p \to \mathbb{F}_p$ be defined as Equation 1. Given any $a, b \in \mathbb{F}_p$, we have

$$\left|\sum_{x\in N_0} \exp\left(\frac{2\pi i}{p}\left(ax+bx^{-1}\right)\right)\right| \leq 4\sqrt{p}.$$

We experimentally verified that for every prime $p < 1000$ and $m \in \{2, 4, 8, 16\}$, the inequality

$$\left|\sum_{x\in N_0} \exp\left(\frac{2\pi i}{p}\left(ax+bx^{-1}\right)\right)\right| \leq 2\sqrt{p}$$

holds, which supports Conjecture 1. The experimental result is provided in Appendix B. Based on Conjecture 1, we derive the following upper bound for $|\mathsf{Corr}_S(a, b)|$:

$$\begin{aligned}
|\mathsf{Corr}_S(a,b)| &= \frac{1}{p}\cdot\left|\sum_{x\in\mathbb{F}_p} \exp\left(\frac{2\pi i}{p}\left(ax+bS(x)\right)\right)\right| \\
&\leq \frac{1}{p}\cdot\sum_{0\leq r<m}\left|\sum_{x\in N_r} \exp\left(\frac{2\pi i}{p}\left(ax+bS(x)\right)\right)\right| \\
&\leq \frac{4m\sqrt{p}}{p}.
\end{aligned}$$

Consequently, the linear probability of the S-box is:

$$\mathsf{LP}_S(a,b) = |\mathsf{Corr}_S(a,b)|^2 \leq \frac{16m^2}{p},$$

For the R-round $\mathsf{Polocolo}$, the linear probability is given by:

$$\mathsf{LP}_{\mathsf{Polocolo}}(a,b) \le \left(\frac{16m^2}{p}\right)^{(1+t)\left\lfloor \frac{R}{2} \right\rfloor}.$$

An attack is considered feasible when the distinguisher's data complexity is approximately $d \approx \frac{1}{\mathsf{LP}_{\mathsf{Polocolo}}(a,b)}$ [7]. Therefore, to ensure that $\mathsf{LP}_{\mathsf{Polocolo}}(a,b) \le 2^{-128}$, it is sufficient to use 2 rounds to prevent linear cryptanalysis.

5.2 Algebraic Attacks

In algebraic attacks, the attacker's goal is to solve a CICO problem stated in Problem 2. Specifically, the attacker aims to find $x_2, \ldots, x_t, y_2, \ldots, y_t \in \mathbb{F}_p$ such that:

$$\mathsf{Polocolo}^{\pi}(x_1, \ldots, x_{t-1}, 0) = y_1, \ldots, y_{t-1}, 0.$$

The attacker represents this condition algebraically then attempts to solve it. In this section, we analyze how to construct the system and the time complexity of various strategies.

Bypassing the First Round. Before introducing the systems, we begin with a strategy to bypass the first round. This strategy is not a weakness unique to $\mathsf{Polocolo}$, but rather a generic attack on hash functions constructed with sponge construction and SPN networks. For example, it is possible to bypass up to 3 rounds in Griffin and 1 round in Arion [8].

Let the input vector be $\vec{x} = (x_1, x_2, ..., x_t = 0)$, let $\vec{a} = (a_1, \ldots, a_t) = \mathsf{LinLayer}^{(0)}(\vec{x})$, and let $\vec{b} = (b_1, \ldots, b_t) = \mathsf{SBoxLayer}(\vec{a})$. The relationship $a_1v_1 + \ldots a_tv_t + v_0 = x_t$ holds for some constants $v_0, v_1, \ldots, v_t$ with $v_1, \ldots, v_t \neq 0$, derived from $\mathsf{LinLayer}^{(0)}$. By setting:

$$a_2 = -v_1v_2^{-1}a_1, a_3 = -v_3^{-1}v_0, a_4 = \cdots = a_t = 0,$$

the condition $x_t = 0$ is satisfied. Moreover, $b_3, \ldots, b_t$ are constants, and since

$$\left(\frac{a_2}{p}\right)_m = \left(\frac{a_1}{p}\right)_m \cdot \left(\frac{-v_1v_2^{-1}}{p}\right)_m,$$

b_2 becomes vb_1 for some constant v once $\left(\frac{a_1}{p}\right)_m$ is given. Therefore, if $\left(\frac{a_1}{p}\right)_m$ is given to an attacker, they can bypass the first linear layer and S-box layer. Guessing $\left(\frac{a_1}{p}\right)_m$ is feasible because the attacker only needs to test $m+1$ possible values of $\left(\frac{a_1}{p}\right)_m$. Moreover, depends on the value of $\left(\frac{-v_1v_2^{-1}}{p}\right)_m$ and σ, possible v values can be fewer than $m+1$. For instance, when $\left(\frac{-v_1v_2^{-1}}{p}\right)_m = 1$, v is fixed to $-v_1^{-1}v_2$, independent of $\left(\frac{a_1}{p}\right)_m$.

In conclusion, once $\left(\frac{a_1}{p}\right)_m$ is given to an attacker, they can bypass the first round and construct the equation with a single variable, b_1. The graphical overview is given in Figure 4.

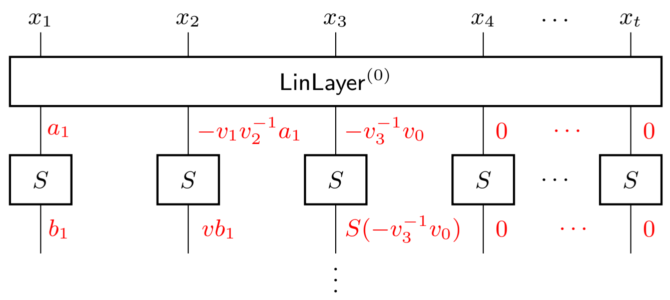


Fig. 4: Graphical overview of the strategy for bypassing the first round of Polocolo.

Guessing Power Residues for Each S-box. An attacker can employ a strategy where they first guess the power residue of each S-box. We refer to this strategy as *guessing power residue attack*. The attack consists of two phases:

1. Guessing phase: The attacker guesses the outputs of the lookup tables in all S-boxes.
2. Determining phase: Using the results from the guessing phase, the attacker constructs a polynomial system and computes its solution. If the solution does not match the expected output, the attacker returns to the guessing phase and tries a different set of guesses.

Since the remaining system after the guessing phase no longer involves any lookup operations, it simplifies to a straightforward SPN network with an inverse nonlinear layer. The similar approaches have been discussed in the arithmetic-oriented hash function named Grendel, which uses a S-box $x \mapsto x^\alpha \times \left(\frac{x}{p}\right)_2$ [37].

The m-th power residue takes on $m+1$ distinct values. However, the probability of m-th power residue is zero is only $1/p$. Therefore an attacker can effectively ignore the case where the power residue is zero and guess among the remaining m possible values for each S-box. In Polocolo$^\pi$, which consists of tR S-boxes, this approach requires m^{tR} guesses. With the bypassing the first round strategy, the number of required guesses is reduced to $m^{(t-1)R}$.

In the determining phase, an attacker must solve an SPN network with an S-box, with the equation:

$$xy = g^{(m+1)r+\sigma(r)},$$

where r is the guessed power residue of the input value x, and y is the output value of the S-box. Since $g^{(m+1)r+\sigma(r)}$ is a constant when r is fixed, the S-box can be considered an inverse S-box. This allows the attacker to derive a univariate equation for b_1 with degree t^{R-1}. The computational steps to obtain this univariate equation are provided in Appendix C.

For a large prime field $\mathbb{F}_p$, solving a univariate equation of degree d has a complexity of $O(d\log(d)(\log(d)+\log(p)\log(\log(d))))$ field operations [62]. Therefore, the overall attack complexity of the guessing power residue attack can be estimated as:

$$m^{t(R-1)} \times O(d\log(d)(\log(d) + \log(p)\log(\log(d)))), \tag{3}$$

where $d = t^{R-1}$. The attack complexity for the original Polocolo parameters and the tight parameters introduced in Section 6.1 is summarized in Table 5.

Table 5: The time complexity of the guessing power residue attack. The time complexities above (resp. below) the dashed line are estimated using the original parameters (resp. modified parameters). The modified parameters are chosen so that the complexity of guessing power residue attack is at least 2^{128}.

Parameters			Time Complexity (log)		
t	R	m	Guess	Determine	Total
3	6	1024	150.00	20.49	170.49
4	5	512	144.00	20.59	164.59
5	5	128	140.00	22.19	162.19
6	5	64	144.00	23.47	167.47
7	5	32	140.00	24.53	164.53
8	5	32	160.00	25.43	185.43
3	5	1024	120.00	18.42	138.42
4	4	1024	120.00	17.95	137.95
6	4	64	108.00	20.27	128.27
8	4	32	120.00	21.84	141.84

Basic Multivariate System. The basic methodology for constructing a multivariate system of Polocolo is to represent $y = S(x)$ as a polynomial in terms of x and introduce every output of S-box as a new variable. Let's examine the degree of y, given that $y = g^{-qm+rm+\sigma(r)} = x^{-1} \cdot g^{r(m+1)+\sigma(r)}$.

Define a function h such that $h(g^{\frac{r(p-1)}{m}}) = g^{(m+1)r+\sigma(r)}$ for $0 \le r < m$. Then, $y = x^{-1} \cdot h\left(\left(\frac{x}{p}\right)_m\right)$. By condition of σ, h is at least degree of $m-1$ and $\left(\frac{x}{p}\right)_m$ has a degree of $\frac{p-1}{m}$.

Therefore, attacker's best strategy is to use the equation:

$$xy - h\left(\left(\frac{x}{p}\right)_m\right) = 0 \tag{4}$$

and its degree it at least $\frac{(m-1)(p-1)}{m}$. This degree is sufficiently high to make solving the multivariate system using Gröbner Basis Attack challenging.

Multivariate System from Plonk constraints. Another approach is to build a system from Plonk constraints in Equation 3.3. From the attacker's flavor, they assume that none of the S-boxes have 0 as an input. Each S-box $y = S(x)$ can then be represented as follows:

$$w_{l+1} - (w_1)^m = 0, \quad w_{l+1}w_{l+2} - x = 0,\ w_{l+1}w_{l+3} - 1 = 0,$$
$$\prod_{i=0}^{m-1}(w_{l+2} - g^i) = 0,\ w_{l+4} - f(w_{l+2}) = 0,\ w_{l+3}w_{l+4} - y = 0,$$

where the function f corresponds to a lookup table T without the $(0,0)$ entry from Table 3. For each S-box, 5 intermediate variables $(w_1, w_{l+1}, w_{l+2}, w_{l+3}, w_{l+4})$ and 5 additional equations are introduced.

By combining this approach with the bypassing the first round technique, we obtain a system of $6t(R-1)+1$ variables and $6t(R-1)+1$ equations, assuming that $\left(\frac{a_1}{p}\right)_m$ is known. We attempted to solve this multivariate system using a Gröbner basis attack. Unfortunately, we could not estimate the complexity of Gröbner basis attack for given system theoretically.

Instead, we experimentally verified that the Gröbner basis attack from Plonk constraints is infeasible. Our experiments is conducted using Sage[5] on an Intel i5-13600K @ 3.90 GHz with 128GB of RAM, used the fgb_sage[6] module for Gröbner basis computation.

We compare the Gröbner basis attack from Plonk constraints with the guessing power residue attack in Table 6. Our results indicate that the Gröbner basis attack using Plonk constraints is less efficient than the guessing power residue attack. Notably, experimental result show that the most time-consuming step in the Gröbner basis attack is the term order change performed using the FGLM algorithm [29]. For each S-box, w_{l+1}, w_{l+2}, w_{l+3}, and w_{l+4} are uniquely determined, while w_1 has m possible values. Therefore, the number of solutions of the system of equations is at least $m^{t(R-1)}$, which implies

$$D_I \geq m^{t(R-1)}.$$

Consequently, the complexity of the term order change exceeds that of the guessing power residue attack, even under optimistic assumptions such as $\omega = 2$ or an unrealistically sparse multiplication matrix with $N_1 \approx D_I$. By guessing the power residue, we could reduce the volume and the time required for FGLM. However, as we guess more values, the attack will be closer to the guessing power residue attack.

6 Performance

6.1 Plonk Cost

We compare Polocolo to existing ZK-friendly hash functions in terms of the number of gates required in fan-in-two Plonk with lookup. Our evaluation is based on the original specifications of the hash functions, while recent algebraic attacks [8,62] have compromised the security of Griffin and Arion. For these hash functions, we have adjusted the round numbers to the minimum required to achieve 128-bit security against the recent attacks, assuming $\omega = 2$. The attacks

5 https://www.sagemath.org/
6 https://github.com/mwageringel/fgb_sage

Table 6: Attack time for toy parameter $p = 113$. "GB" represents the Gröbner basis computation time in degrevlex order, d_{reg} represents the degree of regularity, "Term" represents the computation time for term order change using FGLM algorithm, D_I represents the ideal degree, and "Uni" represents the computation time to solve a univariate equation obtained from a lex order system. "-"(resp. "$>$ 12h") indicates that the computation is aborted due to memory limitations(resp. exceeded 12 hours).

Params			**GBA from Plonk constraints**					**Guessing Power**
R	t	m	GB (ms)	d_{reg}	Term (ms)	D_I	Uni (ms)	**Residue** (ms)
2	3	2	55.05	2	67.09	128	0.04	0.58
		4	89.64	4	4.81×10^4	8192	0.06	5.72
		8	1.15×10^4	8	-	524288	-	35.16
	4	2	66.04	4	1.28×10^3	1024	0.15	1.06
		4	4.33×10^4	5	-	262144	-	18.43
		8	$>$ 12h	-	-	-	-	252.93
3	3	2	71.19	3	6.76×10^4	8192	0.03	3.08
		4	5.38×10^4	5	-	33554432	-	192.38
		8	$>$ 12h	-	-	-	-	1.09×10^4
	4	2	2.83×10^3	6	-	258048	-	17.40
		4	$>$ 12h	-	-	-	-	4.27×10^3
		8	$>$ 12h	-	-	-	-	1.08×10^6

also apply to the full rounds of Anemoi with $t = 2$ (that has been claimed to have security margin of 5 rounds), while there is no concrete description of the attacks for $t > 2$. So for $t > 2$, the original parameters of Anemoi are used in our comparison, while we believe that further analysis on Anemoi might requires one to modify the parameters too, in favor of Polocolo.

We fairly compare the cost of Polocolo to Griffin and Arion without considering security margin. For Polocolo, the round number is fixed such that the complexity of the best attack is at least 2^{128}, instead of 2^{160}. In this setting, the round numbers for Polocolo are provided in Table 7. The adjusted round numbers of Griffin and Arion, along with the equations used to calculate the round numbers, are presented in Appendix D.1. The number of Plonk gates for each hash function is shown in Table 2. Details on how Plonk gates are measured for the hash functions are provided in Appendix D.2.

When $t = 8$, Polocolo requires 21% less Plonk gates compared to Anemoi, which is currently known as the most efficient ZK-friendly hash function. For $t = 3$, Polocolo requires 24% less Plonk gates than Reinforced Concrete, which is one of the recent lookup-based ZK-friendly hash functions.

Table 7: Tight round number for Polocolo. t denotes the size of the permutation in blocks of $\mathbb{F}_p$, R denotes the round number, and m denotes the exponent in the power residue.

t	3	4	6	8
R	5	4	4	4
m	1024	1024	64	32

6.2 Plain Performance

Table 8 compares the performance of the ZK-friendly hash functions in the plain environment using the codes from the ZKFriendlyHashZoo repository for Reinforced Concrete, Poseidon2, Rescue-Prime, and Griffin[7], while the codes for Anemoi and Arion are obtained from their official repositories[8] [9]. The performance is evaluated using the scalar fields of the BLS12 elliptic curves, and the experiments are executed in Intel i5-13600K @ 3.90 GHz with 128GB of RAM. The performance of Polocolo is comparable to the existing hash functions, while our focus is mainly put on reducing the Plonk cost since computation time in plain is much smaller than the proving time in general.

Table 8: Performance of hash functions in plain. "-" is used to indicate that the hash function is not defined for the given value of t. The speed for the hash functions above (resp. below) the dashed line are estimated using the original parameters (resp. modified parameters). The modified parameters are chosen to achieve 128-bit security against all the recent attacks with no security margin.

Hash functions	Speed (μs)			
	$t = 3$	$t = 4$	$t = 6$	$t = 8$
Polocolo	136.77	154.06	254.12	350.20
Reinforced Concrete	1.52	-	-	-
Poseidon2	4.86	8.53	-	13.66
Rescue-Prime	238.41	251.21	278.60	376.79
Anemoi	-	353.59	457.45	610.08
Polocolo (tight)	106.47	117.52	196.03	274.43
Griffin	92.066	93.093	-	102.48
Arion	72.880	72.849	70.707	66.294

[7] https://extgit.iaik.tugraz.at/krypto/zkfriendlyhashzoo
[8] https://github.com/anemoi-hash/anemoi-rust
[9] https://github.com/sca-research/Arion

7 Conclusion

In this work, we present a new ZK-friendly hash function, Polocolo, based on the power residue method. It achieves the smallest Plonk costs compared to existing schemes in this category. We conclude with several open topics for future research.

- While it is evident that performance in the Plonk setting is proportional to the number of Plonk gates, it would be valuable to evaluate Polocolo by implementing it in an actual protocol such as Plonkup.
- In the security analysis of Polocolo, the linear probabilities of S-box relies on conjecture. Providing formal proofs for this conjecture would strengthen the argument for the security of Polocolo.
- Polocolo is based on the scalar fields of the BLS12 and the BN254 elliptic curves. However, certain proof systems that support lookup operations such as Plonky2 [25] are defined over smaller primes such as $p = 2^{64} - 2^{32} + 1$ and $p = 2^{32} - 1$. Tip5 [57] and Monolith [35] are specifically designed for these fields. It would be interesting to design hash functions for the smaller primes with the power residue method.
- We have only evaluated Polocolo in the context of fan-in-two Plonk. However, fan-in-three Plonk and Plonk with custom gates are also studied in the literature. It would be interesting to estimate the performance of Polocolo in such alternative configurations.
- We derive a new MDS matrix through a heuristic approach. Further research could focus on improving this algorithm, determining lower bounds on the number of additional gates required to construct an MDS matrix, and exploring other potential optimizations.

References

1. Albrecht, M., Grassi, L., Rechberger, C., Roy, A., Tiessen, T.: MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. vol. 10031, pp. 191–219. Springer (2016)
2. Albrecht, M.R., Cid, C., Grassi, L., Khovratovich, D., Lüftenegger, R., Rechberger, C., Schofnegger, M.: Algebraic Cryptanalysis of STARK-Friendly Designs: Application to MARVELlous and MiMC. In: ASIACRYPT 2019. pp. 371–397. Springer (2019)
3. Aly, A., Ashur, T., Ben-Sasson, E., Dhooghe, S., Szepieniec, A.: Design of Symmetric-Key Primitives for Advanced Cryptographic Protocols. ToSC 2020 **2020**(3) (Sep 2020). https://doi.org/10.13154/tosc.v2020.i3.1-45
4. Arun, A., Setty, S., Thaler, J.: Jolt: Snarks for virtual machines via lookups. In: EUROCRYPT 2024. pp. 3–33. Springer (2024)
5. Ashur, T., Buschman, T., Mahzoun, M.: Algebraic cryptanalysis of hades design strategy: Application to poseidon and poseidon2. Cryptology ePrint Archive (2023)
6. Ashur, T., Dhooghe, S.: MARVELlous: a STARK-Friendly Family of Cryptographic Primitives. IACR Cryptology ePrint Archive, Report 2018/1098 (2018), https://eprint.iacr.org/2018/1098

7. Baigneres, T., Stern, J., Vaudenay, S.: Linear Cryptanalysis of Non Binary Ciphers: (With an Application to SAFER). In: SAC 2007. pp. 184–211. Springer (2007)
8. Bariant, A., Boeuf, A., Lemoine, A., Ayala, I.M., Øygarden, M., Perrin, L., Raddum, H.: The Algebraic Freelunch Efficient Gröbner Basis Attacks Against Arithmetization-Oriented Primitives. Cryptology ePrint Archive (2024)
9. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive (2018)
10. Ben-Sasson, E., Chiesa, A., Riabzev, M., Spooner, N., Virza, M., Ward, N.P.: Aurora: Transparent succinct arguments for R1CS. In: EUROCRYPT 2019. pp. 103–128. Springer (2019)
11. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Sponge functions. In: ECRYPT hash workshop. vol. 2007 (2007)
12. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: On the indifferentiability of the sponge construction. In: EUROCRYPT 2008. pp. 181–197. Springer (2008)
13. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Keccak. In: EUROCRYPT 2013. pp. 313–314. Springer (2013)
14. Bootle, J., Cerulli, A., Groth, J., Jakobsen, S., Maller, M.: Arya: Nearly linear-time zero-knowledge proofs for correct program execution. In: ASIACRYPT 2018. pp. 595–626. Springer (2018)
15. Bouvier, C., Briaud, P., Chaidos, P., Perrin, L., Salen, R., Velichkov, V., Willems, D.: New Design Techniques for Efficient Arithmetization-Oriented Hash Functions: Anemoi Permutations and Jive Compression Mode. In: CRYPTO 2023. pp. 507–539. Springer (2023)
16. Bowe, S., Grigg, J., Hopwood, D.: Recursive proof composition without a trusted setup. Cryptology ePrint Archive (2019)
17. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: S&P 2018. pp. 315–334. IEEE (2018)
18. Chen, B., Bünz, B., Boneh, D., Zhang, Z.: Hyperplonk: Plonk with linear-time prover and high-degree custom gates. In: EUROCRYPT 2023. pp. 499–530. Springer (2023)
19. Chiesa, A., Hu, Y., Maller, M., Mishra, P., Vesely, N., Ward, N.: Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In: EUROCRYPT 2020. pp. 738–768. Springer (2020)
20. Damgård, I.B.: On the randomness of Legendre and Jacobi sequences. In: CRYPTO 1988. pp. 163–172. Springer (1988)
21. Ding, C., Hesseseth, T., Shan, W.: On the linear complexity of Legendre sequences. IEEE Transactions on Information Theory **44**(3), 1276–1278 (1998)
22. Duval, S., Leurent, G.: MDS matrices with lightweight circuits. ToSC 2018 **2018**(2), 48–78 (2018)
23. Eagen, L., Fiore, D., Gabizon, A.: cq: Cached quotients for fast lookups. Cryptology ePrint Archive (2022)
24. Eichlseder, M., Grassi, L., Lüftenegger, R., Øygarden, M., Rechberger, C., Schofnegger, M., Wang, Q.: An algebraic attack on ciphers with low-degree round functions: application to full MiMC. In: ASIACRYPT 2020. pp. 477–506. Springer (2020)
25. Farmer, B.: Introducing Plonky2 (2022)
26. Faugere, J.C.: A new efficient algorithm for computing Gröbner bases (F_4). Journal of Pure and Applied Algebra **139**(1-3), 61–88 (1999). https://doi.org/10.1016/S0022-4049(99)00005-5

27. Faugère, J.C.: A new efficient algorithm for computing Gröbner bases without reduction to zero (F_5). In: ISSAC 2002. pp. 75–83. ACM (2002). https://doi.org/10.1145/780506.780516
28. Faugère, J.C., Gaudry, P., Huot, L., Renault, G.: Sub-cubic change of ordering for Gröbner basis: a probabilistic approach. In: Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation. pp. 170–177 (2014)
29. Faugere, J.C., Gianni, P., Lazard, D., Mora, T.: Efficient computation of zero-dimensional Gröbner bases by change of ordering. Journal of Symbolic Computation **16**(4), 329–344 (1993)
30. Faugère, J.C., Mou, C.: Sparse FGLM algorithms. Journal of Symbolic Computation **80**, 538–569 (2017)
31. Gabizon, A., Williamson, Z.J.: plookup: A simplified polynomial protocol for lookup tables. Cryptology ePrint Archive (2020)
32. Gabizon, A., Williamson, Z.J., Ciobotaru, O.: Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive (2019)
33. Grassi, L., Hao, Y., Rechberger, C., Schofnegger, M., Walch, R., Wang, Q.: Horst meets Fluid-SPN: Griffin for zero-knowledge applications. In: CRYPTO 2023. pp. 573–606. Springer (2023)
34. Grassi, L., Khovratovich, D., Lüftenegger, R., Rechberger, C., Schofnegger, M., Walch, R.: Reinforced concrete: a fast hash function for verifiable computation. In: CCS 2022. pp. 1323–1335 (2022)
35. Grassi, L., Khovratovich, D., Lüftenegger, R., Rechberger, C., Schofnegger, M., Walch, R.: Hash Functions Monolith for ZK Applications: May the Speed of SHA-3 be With You. IACR Cryptol. ePrint Arch. **2023**, 1025 (2023)
36. Grassi, L., Khovratovich, D., Rechberger, C., Roy, A., Schofnegger, M.: Poseidon: A New Hash Function for Zero-Knowledge Proof Systems. In: USENIX 2021. pp. 519–535. USENIX Association (Aug 2021)
37. Grassi, L., Khovratovich, D., Rønjom, S., Schofnegger, M.: The Legendre Symbol and the Modulo-2 Operator in Symmetric Schemes over $\mathbb{F}_p^n$. Cryptology ePrint Archive (2021)
38. Grassi, L., Khovratovich, D., Schofnegger, M.: Poseidon2: A faster version of the poseidon hash function. In: AFRICACRYPT 2023. pp. 177–203. Springer (2023)
39. Grassi, L., Lüftenegger, R., Rechberger, C., Rotaru, D., Schofnegger, M.: On a generalization of substitution-permutation networks: The HADES design strategy. In: EUROCRYPT 2020. pp. 674–704. Springer (2020)
40. Grassi, L., Rechberger, C., Rotaru, D., Scholl, P., Smart, N.P.: MPC-Friendly Symmetric Key Primitives. In: CCS 2016. p. 430–443. ACM (2016)
41. Groth, J.: On the size of pairing-based non-interactive arguments. In: EUROCRYPT 2016. pp. 305–326. Springer (2016)
42. Gyarmati, K., Mauduit, C., Sárközy, A.: The cross-correlation measure for families of binary sequences. (2014)
43. Knudsen, L.R.: Truncated and higher order differentials. In: Preneel, B. (ed.) FSE '95. vol. 1008, pp. 196–211. Springer (1995)
44. Koschatko, K., Lüftenegger, R., Rechberger, C.: Exploring the Six Worlds of Gröbner Basis Cryptanalysis: Application to Anemoi. Cryptology ePrint Archive (2024)
45. Maram, D., Malvai, H., Zhang, F., Jean-Louis, N., Frolov, A., Kell, T., Lobban, T., Moy, C., Juels, A., Miller, A.: Candid: Can-do decentralized identity with legacy compatibility, sybil-resistance, and accountability. In: S&P 2021. pp. 1348–1366. IEEE (2021)

46. Matsui, M.: Linear Cryptanalysis Method for DES Cipher. In: Helleseth, T. (ed.) EUROCRYPT '93. vol. 765, pp. 386–397. Springer (1994). https://doi.org/10.1007/3-540-48285-7_33
47. Mauduit, C., Sárközy, A.: On finite pseudorandom binary sequences I: Measure of pseudorandomness, the Legendre symbol. Acta Arithmetica **82**(4), 365–377 (1997)
48. Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: The rebound attack: Cryptanalysis of reduced Whirlpool and Grøstl. In: FSE 2009. pp. 260–276. Springer (2009)
49. Pearson, L., Fitzgerald, J., Masip, H., Bellés-Muñoz, M., Muñoz-Tapia, J.L.: Plonkup: Reconciling plonk with plookup. Cryptology ePrint Archive (2022)
50. Perrin, L., Udovenko, A., Biryukov, A.: Cryptanalysis of a theorem: Decomposing the only known solution to the big APN problem. In: CRYPTO 2016. pp. 93–122. Springer (2016)
51. Pertsev, A., Semenov, R., Storm, R.: Tornado cash privacy solution version 1.4. Tornado cash privacy solution version **1**, 7 (2019)
52. Pub, F.: Secure hash standard (shs). Fips pub **180**(4) (2012)
53. Roy, A., Steiner, M.J., Trevisani, S.: Arion: Arithmetization-oriented permutation and hashing from generalized triangular dynamical systems. arXiv preprint arXiv:2303.04639 (2023)
54. Sauer, J.F., Szepieniec, A.: SoK: Gröbner Basis Algorithms for Arithmetization Oriented Ciphers. Cryptology ePrint Archive, Paper 2021/870 (2021), https://eprint.iacr.org/2021/870
55. Szepieniec, A.: On the use of the legendre symbol in symmetric cipher design. Cryptology ePrint Archive (2021)
56. Szepieniec, A., Ashur, T., Dhooghe, S.: Rescue-prime: a standard specification (SoK). Cryptology ePrint Archive (2020)
57. Szepieniec, A., Lemmens, A., Sauer, J.F., Threadbare, B., et al.: The tip5 hash function for recursive starks. Cryptology ePrint Archive (2023)
58. Tóth, V.: Collision and avalanche effect in families of pseudorandom binary sequences. Periodica Mathematica Hungarica **55**, 185–196 (2007)
59. Wagner, D.: The Boomerang Attack. In: Knudsen, L. (ed.) FSE '99. vol. 1636, pp. 156–170. Springer (1999)
60. Weil, A.: On some exponential sums. Proceedings of the National Academy of Sciences **34**(5), 204–207 (1948)
61. Xie, T., Zhang, J., Cheng, Z., Zhang, F., Zhang, Y., Jia, Y., Boneh, D., Song, D.: zkbridge: Trustless cross-chain bridges made practical. In: CCS 2022. pp. 3003–3017 (2022)
62. Yang, H.S., Zheng, Q.X., Yang, J., Liu, Q.f., Tang, D.: A New Security Evaluation Method Based on Resultant for Arithmetic-Oriented Algorithms. Cryptology ePrint Archive (2024)
63. Zapico, A., Gabizon, A., Khovratovich, D., Maller, M., Rafols, C.: Baloo: nearly optimal lookup arguments. Cryptology ePrint Archive (2022)
64. Zhang, Y., Sun, S.F., Gu, D.: Efficient KZG-Based Univariate Sum-Check and Lookup Argument. In: PKC 2024. pp. 400–425. Springer (2024)

Appendix

A Linear Layers

A.1 Linear Layers and Corresponding Plonk Constraints

For each $t \in \{3, \ldots, 8\}$, we provide an MDS matrix M and corresponding Constraints for $\vec{y} = M \cdot \vec{x}$,

- $\mathbf{t = 3}$.

$$M = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix},$$

$$\text{Constraints}_3 = \begin{cases} w_1 = x_1 + x_2, w_2 = w_1 + x_3, w_3 = w_2 + x_1, \\ w_4 = w_2 + x_2, w_5 = w_2 + x_3 \end{cases},$$

$\vec{y} = \{w_3, w_4, w_5\}$.

- $\mathbf{t = 4}$.

$$M = \begin{pmatrix} 5 & 7 & 1 & 3 \\ 4 & 6 & 1 & 1 \\ 1 & 3 & 5 & 7 \\ 1 & 1 & 4 & 6 \end{pmatrix},$$

$$\text{Constraints}_4 = \begin{cases} w_1 = x_1 + x_2, w_2 = x_3 + x_4, w_3 = 2x_2 + w_2, \\ w_4 = 2x_4 + w_1, w_5 = 4w_2 + w_4, w_6 = 4w_1 + w_3, \\ w_7 = w_4 + w_6, w_8 = w_3 + w_5 \end{cases},$$

$\vec{y} = \{w_7, w_6, w_8, w_5\}$.

- $\mathbf{t = 5}$.

$$M = \begin{pmatrix} 39 & 6 & 10 & 28 & 8 \\ 174 & 28 & 32 & 80 & 16 \\ 348 & 58 & 42 & 84 & 2 \\ 39 & 4 & 54 & 100 & 44 \\ 204 & 20 & 300 & 560 & 244 \end{pmatrix},$$

$$\text{Constraints}_5 = \begin{cases} w_1 = 6x_1 + x_2, w_2 = 2x_3 + 4x_4, w_3 = 3x_1 + 8x_5, \\ w_4 = 8w_1 + 3w_2, w_5 = 5w_2 + w_3, w_6 = 4x_3 + 5w_5, \\ w_7 = 8x_4 + 6w_1, w_8 = 2w_1 + 2x_5, w_9 = w_7 + w_5, \\ w_{10} = 2w_4 + 2w_9, w_{11} = w_8 + 7w_4, w_{12} = w_6 + 2w_8, \\ w_{13} = 3w_5 + 5w_{12} \end{cases},$$

$\vec{y} = \{w_9, w_{10}, w_{11}, w_{12}, w_{13}\}$.

– **t = 6**.

$$M = \begin{pmatrix} 1011 & 1470 & 42 & 140 & 508 & 1700 \\ 232 & 70 & 48 & 48 & 264 & 1280 \\ 4227 & 7371 & 3 & 490 & 1420 & 2900 \\ 6744 & 11760 & 60 & 844 & 2272 & 4670 \\ 13281 & 23163 & 9 & 1540 & 4460 & 9100 \\ 48 & 84 & 12 & 35 & 40 & 200 \end{pmatrix},$$

$$\text{Constraints}_6 = \begin{cases} w_1 = 4x_1 + 7x_2, w_2 = 2x_3 + 2x_4, w_3 = x_5 + 5x_6, \\ w_4 = 3x_1 + 4w_3, w_5 = 5w_1 + 4x_5, w_6 = 5w_2 + 5x_6, \\ w_7 = 3w_1 + 3x_3, w_8 = 8w_4 + 3w_2, w_9 = 8w_3 + 7x_4, \\ w_{10} = 2w_9 + 6w_5, w_{11} = w_4 + 7w_{10}, w_{12} = 7w_8 + w_{11}, \\ w_{13} = 2w_5 + 8w_8, w_{14} = 5w_{11} + w_7, w_{15} = 8w_{11} + 6w_6, \\ w_{16} = 3w_{14} + 5w_{10}, w_{17} = 5w_9 + 4w_7 \end{cases},$$

$\vec{y} = \{w_{12}, w_{13}, w_{14}, w_{15}, w_{16}, w_{17}\}$.

– **t = 7**.

$$M = \begin{pmatrix} 3538 & 3090 & 768 & 480 & 720 & 96 & 336 \\ 470862 & 470750 & 1120 & 16380 & 94284 & 136 & 924 \\ 10112885 & 10113269 & 24960 & 352496 & 2023200 & 768 & 18048 \\ 3799380 & 3799524 & 9024 & 132256 & 760128 & 288 & 6783 \\ 94120 & 94080 & 232 & 3276 & 18816 & 5 & 198 \\ 1357780 & 1357788 & 3240 & 47268 & 271632 & 101 & 2454 \\ 270260 & 270260 & 640 & 9402 & 54108 & 64 & 480 \end{pmatrix},$$

$$\text{Constraints}_7 = \begin{cases} w_1 = 5x_1 + 5x_2, w_2 = 8x_3 + 5x_4, w_3 = 3x_5 + 2x_6, \\ w_4 = 8x_1 + 6x_7, w_5 = 6x_5 + 6w_1, w_6 = 2w_2 + 2w_5, \\ w_7 = 8w_6 + 7w_1, w_8 = 7w_5 + 7x_4, w_9 = 7w_4 + 6w_3, \\ w_{10} = x_6 + w_4, w_{11} = 8w_8 + 3x_7, w_{12} = 8w_{11} + 4w_2, \\ w_{13} = 8x_3 + 7w_{12}, w_{14} = 8w_{12} + 6x_2, w_{15} = w_{12} + 2w_1, \\ w_{16} = 8w_3 + 5w_{15}, w_{17} = 6w_{16} + 8w_{14}, w_{18} = 8w_9 + 6w_7, \\ w_{19} = 7w_{16} + 2w_9, w_{20} = 8w_{17} + 7w_7, w_{21} = 5w_{11} + 3w_{17}, \\ w_{22} = w_{13} + 5w_{10}, w_{23} = w_{22} + w_{17}, w_{24} = 4w_{16} + 6w_8 \end{cases},$$

$\vec{y} = \{w_{18}, w_{19}, w_{20}, w_{21}, w_{22}, w_{23}, w_{24}\}$.

– **t = 8**.

$$M = \begin{pmatrix} 3840 & 24 & 4728 & 2952 & 258912 & 99840 & 94222 & 74400 \\ 1386 & 78 & 280 & 1218 & 32256 & 13044 & 8120 & 6496 \\ 6180 & 743 & 10416 & 4428 & 508032 & 194858 & 193984 & 153056 \\ 432 & 400 & 1920 & 144 & 73728 & 27776 & 30400 & 23936 \\ 10122 & 1246 & 5320 & 8526 & 346752 & 136724 & 108570 & 86184 \\ 950 & 1052 & 5424 & 240 & 202944 & 76333 & 84683 & 66656 \\ 2564 & 16 & 3072 & 1920 & 172128 & 66380 & 62528 & 49408 \\ 661 & 35 & 908 & 585 & 43008 & 16448 & 14512 & 11456 \end{pmatrix},$$

$$\text{Constraints}_8 = \begin{cases} w_1 = 3x_1 + 5x_2, w_2 = 4x_3 + 3x_4, w_3 = 8x_5 + 3x_6, \\ w_4 = 5x_7 + 4x_8, w_5 = 4x_6 + 7w_4, w_6 = 5w_2 + w_1, \\ w_7 = 3w_3 + 2w_4, w_8 = 2x_1 + 6w_3, w_9 = 4x_3 + 6w_7, \\ w_{10} = 8w_5 + x_2, w_{11} = 3x_4 + 2w_8, w_{12} = 6w_9 + 6x_7, \\ w_{13} = 5w_1 + 5w_{12}, w_{14} = 4w_{13} + 2w_5, w_{15} = x_7 + 6w_{13}, \\ w_{16} = 8w_{11} + 4w_5, w_{17} = 6w_{16} + 2w_6, w_{18} = 2w_{10} + 5x_6, \\ w_{19} = 5w_{16} + 8w_{12}, w_{20} = 2w_{19} + w_{18}, w_{21} = 3w_2 + 8x_7, \\ w_{22} = w_{10} + 4w_{13}, w_{23} = 6w_{20} + 4w_{21}, w_{24} = 2w_{23} + w_{12}, \\ w_{25} = 4w_{18} + 7w_{17}, w_{26} = 7w_{22} + 3w_{23}, w_{27} = 6w_{16} + 4w_{14}, \\ w_{28} = 7w_{25} + 7w_{14}, w_{29} = w_{20} + 7w_{15}, w_{30} = 2w_8 + 8w_{20}, \\ w_{31} = 4w_{19} + 7w_6 \end{cases},$$

$\vec{y} = \{w_{24}, w_{25}, w_{26}, w_{27}, w_{28}, w_{29}, w_{30}, w_{31}\}$.

A.2 An Algorithm to Find an Efficient MDS Matrix

We propose an algorithm for constructing an efficient t by t MDS matrix M_t using a addition gates. In this algorithm, we iteratively generate a new vectors as linear combinations of two vectors, which is either unit vectors or previously computed vectors, with selected coefficients. Each vector is correspond to a one witness, generated by a one addition gate. Afterward, we check whether the matrix formed from the last t vectors satisfies the MDS property. This process is repeated with different random selections until an MDS matrix is obtained. The algorithm is presented in Algorithm 1.

B Experiment on Linear Correlation of the S-box

In Section 5.1, we introduce Conjecture 1. To support this conjecture, we provide the results of our experimental analysis.

Let $C(a, b)$ be defined as follows:

$$C(a,b) = \left| \sum_{x \in N_0} \exp\left(\frac{2\pi i}{p} \left(ax + bx^{-1} \right) \right) \right|.$$

We compute

$$\max_{a,b \in \mathbb{F}_p \setminus \{0\}} C(a,b)$$

for every prime $p < 1000$ and $m \in \{2, 4, 8, 16\}$. The experimental results confirm that

$$\max_{a,b \in \mathbb{F}_p \setminus \{0\}} C(a,b) \leq 2\sqrt{p}.$$

The results are summarized in Figure 5.

Algorithm 1: An algorithm to find an efficient MDS matrix

```
Data: Size of matrix $t \geq 3$, A number of addition gates $a \geq t$
Result: An MDS matrix $M \in \mathbb{F}_p^{t \times t}$, A constraints set $C$
1  $M \leftarrow t$ by $t$ empty matrix;
2  $C \leftarrow \emptyset$;
3  for $i \in [t]$ do
4      $e_i \leftarrow i$-th unit vector;
5      $x_i \leftarrow$ A witness corresponds to an $i$-th element of the input vector $\vec{x}$;
6  for $i \in [a]$ do
7      $w_i \leftarrow i$-th witness used in constraints set $C$;
8  for $i \in \lfloor t/2 \rfloor$ do
9      $c_1, c_2 \leftarrow_\$ \{1, 2, \ldots, 8\}$;
10     $v_i \leftarrow c_1 e_{2i} + c_2 e_{2i+1}$;
11     $C \leftarrow C \cup \{w_i = c_1 x_{2i} + c_2 x_{2i+1}\}$;
12 if $N$ is odd then
13     $c_1, c_2 \leftarrow_\$ \{1, 2, \ldots, 8\}$;
14     $v_{(N+1)/2} \leftarrow c_1 e_1 + c_2 e_t$;
15     $C \leftarrow C \cup \{w_{(N+1)/2} \leftarrow c_1 x_1 + c_2 x_t\}$;
16 for $i \in \{\lceil (t+1)/2 \rceil, \ldots, a - t\}$ do
17     $c_1, c_2 \leftarrow_\$ \{1, 2, \ldots, 8\}$;
18     $z_1 \leftarrow_\$ \{x_1, \ldots, x_t, w_1, \ldots, w_{|C|}\}$;
19     $z_2 \leftarrow_\$ \{x_1, \ldots, x_t, w_1, \ldots, w_{|C|}\} \setminus \{z_1\}$;
20     if $z_1 \in \{x_1, \ldots, x_t\}$ then
21         $j \leftarrow$ an index that $z_1 = x_j$;
22         $v_i \leftarrow c_1 e_j$;
23     else
24         $j \leftarrow$ an index that $z_1 = w_j$;
25         $v_i \leftarrow c_1 v_j$;
26     if $z_2 \in \{x_1, \ldots, x_t\}$ then
27         $j \leftarrow$ an index that $z_2 = x_j$;
28         $v_i \leftarrow v_i + c_2 e_j$;
29     else
30         $j \leftarrow$ an index that $z_2 = w_j$;
31         $v_i \leftarrow v_i + c_2 v_j$;
32     $C \leftarrow C \cup \{w_i = c_1 z_1 + c_2 z_2\}$;
33 for $i \in \{a - t + 1, \ldots, a\}$ do
34     $c_1, c_2 \leftarrow_\$ \{1, 2, \ldots, 8\}$;
35     $j \leftarrow [i - 1]$;
36     $k \leftarrow [i - 1] \setminus \{j\}$;
37     $v_i \leftarrow c_1 w_j + c_2 w_k$;
38     $C \leftarrow C \cup \{w_i = c_1 w_j + c_2 w_k\}$;
39     $M[i - (a - t)] \leftarrow v_i$;
40 if $M$ is a non-MDS matrix then
41     return $\perp$;
42 return $M, C$;
```

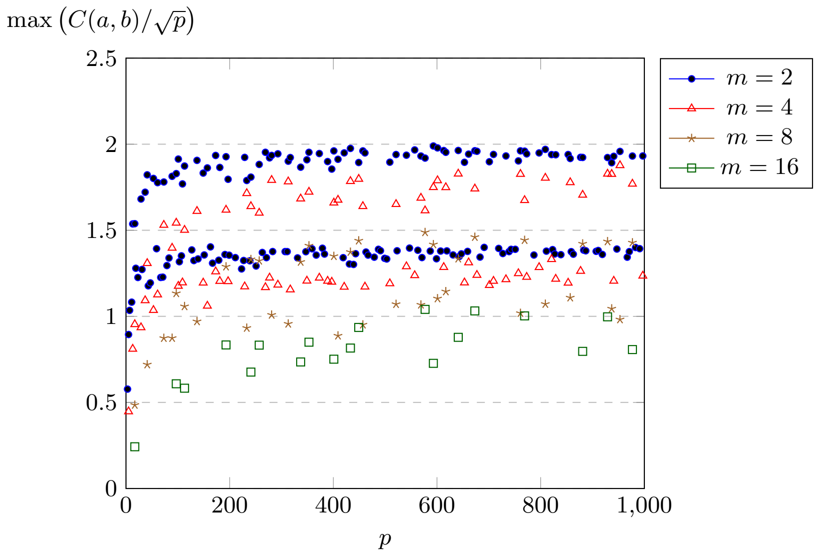


Fig. 5: Maximum values of $C(a, b)/\sqrt{p}$ for each prime $p < 1000$ and $m \in \{2, 4, 8, 16\}$, where $C(a, b) = \left|\sum_{x \in N_0} \exp\left(\frac{2\pi i}{p}\left(ax + bx^{-1}\right)\right)\right|$. The results show that for each m, the maximum value is bounded by $2\sqrt{p}$, supporting Conjecture 1.

C Univariate Equations in the Guessing Power Residue Attack

In this section, we outline the computational steps required to derive the univariate equation in the guessing power residue attack, as described in Section **??**. We also demonstrate that the degree of the resulting univariate equation is t^{R-1}.

Consider a system with t branches and R rounds of $\mathsf{Polocolo}^{\pi}$. Let $\vec{x}^{(i)} = (x_1^{(i)}, \cdots, x_t^{(i)})$ represent the state after the i-th $\mathsf{SBoxLayer}$, and $\vec{w}^{(j-1)} = (w_1^{(j-1)}, \cdots, w_t^{(j-1)})$ represent the state after j-th $\mathsf{LinLayer}$. As the input $\vec{x}$ progresses through the layers, its transformation follows these steps:

$$\begin{aligned}
\vec{w}^{(0)} &= \mathsf{LinLayer}^{(0)}(\vec{x}),\\
\vec{x}^{(1)} &= \mathsf{SBoxLayer}(\vec{w}^{(0)}),\\
\vec{w}^{(1)} &= \mathsf{LinLayer}^{(1)}(\vec{x}^{(1)}),\\
&\vdots\\
\vec{y} = \vec{w}^{(R)} &= \mathsf{LinLayer}^{(R)}(\vec{x}^{(R)}).
\end{aligned}$$

After bypassing the first round, as described in Section **??**, we begin with the state $\vec{x}^{(1)} = (b_1, vb_1, S(-c_3^{-1}c_0), 0, \ldots, 0)$, where b_1 is the variable and v is a constant. After guessing the power residues for each S-box, $y = S(x)$ satisfies $xy = v'$ for some constant v'. To simplify the expressions, we assume $v' = 1$, which does not affect the degree of the variables.

Now we analyze the degree of each variable with respect to b_1. After applying $\mathsf{LinLayer}^{(1)}$, the components $w_1^{(1)}, \cdots, w_t^{(1)}$ are linear with respect to b_1. Following the S-box layer, where $\vec{x}^{(2)} = \mathsf{SBoxLayer}(\vec{w}^{(1)})$, we have $x_i^{(2)} = 1/w_i^{(1)}$ for $i \in [t]$.

After applying $\mathsf{LinLayer}^{(2)}$, each $w_i^{(2)}$ is a linear combination of $x_1^{(2)}, \ldots, x_t^{(2)}$. As a result, we obtain univariate equations $f_i^{(2)}$ and $h_i^{(2)}$ of degree t, where:

$$w_i^{(2)} = \frac{f_i^{(2)}(b_1)}{h_i^{(2)}(b_1)},$$

for $i \in [t]$, since the t linear equations were combined under a common denominator.

Similarly, in the next round, we have

$$x_i^{(3)} = 1/w_i^{(2)} = \frac{h_i^{(2)}(b_1)}{f_i^{(2)}(b_1)}$$

and

$$w_i^{(3)} = \frac{f_i^{(3)}(b_1)}{h_i^{(3)}(b_1)},$$

where $f_i^{(3)}$ and $h_i^{(3)}$ are of degree t^2 for $i \in [t]$, as the t equations of degree t were combined under a common denominator.

By repeating this process,

$$y_1 = w_1^{(R)} = \frac{f_1^{(R)}(b_1)}{h_1^{(R)}(b_1)},$$

where $f_i^{(R)}$ and $h_i^{(R)}$ are of degree t^{R-1}. Finally, we obtain a univariate equation $f_1^{(R)}(b_1) = 0$, which has a degree of $t^{(R-1)}$.

D The Round Numbers and Plonk Gates of Other Hash Functions

Since our comparison is conducted in the BLS12 and BN254 prime fields, the parameter α used in power map is set to $\alpha = 5$, which is the smallest positive number greater than 1 such that $\gcd(\alpha, p-1) = 1$. Additionally, we assume a security parameter of $\kappa = 128$ and a rate of $r = 1$.

D.1 Round Numbers

For Reinforced Concrete and Anemoi, the round numbers are provided in their respective specifications. See [34, Section 2] and [15, Table 1.a] for details. The sections below summarize how to determine the round number for Rescue-Rrime, Poseidon2, Griffin, and Arion. The overall round numbers are summarized in the Table 9.

Table 9: The round numbers for ZK-friendly hash functions. "-" is used to indicate that the hash function is not defined for the given value of t. For Poseidon2, the round numbers are presented in the order of full round number, partial round number. The round number for the hash functions above (resp. below) the dashed line are estimated using the original parameters (resp. modified parameters). The modified parameters are chosen to achieve 128-bit security against all the recent attacks with no security margin.

Hash functions	Round number			
	$t = 3$	$t = 4$	$t = 6$	$t = 8$
Reinforced Concrete	7	-	-	-
Poseidon2	(8, 55)	(8, 55)	-	(8, 56)
Rescue-Prime	14	11	8	8
Anemoi	-	14	12	12
Griffin	17	17	-	18
Arion	8	7	7	6

Poseidon2 For Poseidon2, we calculate the round numbers using [38, Equation 1]. Specifically, the round numbers are given by:

$$R_F = 8, R_P \geq \left\lceil 1.075 \cdot \left\lceil \max\{\frac{\min\{\kappa, \log_2(p)\}}{\log_2(\alpha)} + \log_\alpha(t) - 5, R_{GB}\} \right\rceil \right\rceil,$$

where

$$\begin{aligned} R_{GB} \geq \max\{&R'_{GB}, \\ &\log_\alpha(2) \cdot \min\{\kappa, \log_2(p)\} - 6, \\ &t - 7 + \log_\alpha(2) \cdot \min\{\frac{\kappa}{t+1}, \frac{\log_2(p)}{2}\}, \\ &\frac{\kappa}{2 \cdot \log_2(\alpha)} - 5 \cdot t + 4\}, \end{aligned}$$

and $R'_{GB} \geq 1$ is the smallest integer that satisfies

$$\binom{4t + 5 + 2R'_{GB} + \alpha}{3 + R'_{GB} + \alpha} \geq 2^{\kappa/2}.$$

Rescue-Prime For Rescue-Prime, we calculate round numbers using [56, Section 2.5]. Specifically, the round numbers are given by:

$$\lceil 1.5 \cdot \max(5, \ell) \rceil,$$

where ℓ be the smallest integer that satisfies:

$$\binom{(0.5\alpha + 0.5)t(\ell - 1) + 3}{t(\ell - 1) + 1} > 2^{\kappa/2}.$$

Griffin Griffin has been affected by FreeLunch attack, so we use the estimated time complexity provided in [8, Section 4.1] to calculate the round numbers with $\omega = 2$. The round number R is the smallest integer that satisfies:

$$\begin{cases} (\alpha^{\omega}(2\alpha + 1))^{R-1} \geq 2^{\kappa}, & \text{for } t = 3, 4, \\ 3(\alpha^{\omega}(2\alpha + 1))^{R-2} \geq 2^{\kappa}, & \text{for } t = 8. \end{cases}$$

Arion Similar to Griffin, Arion has also been affected by the FreeLunch attack, so we use the estimated time complexity provided in [8, Section 4.2] to calculate round numbers for $d_1 = 5, d_2 = 257$, and $\omega = 2$. The round number R is the smallest integer that satisfies:

$$3d_1(d_2^{\omega}(2^{t-1}(d_1 + 1) - d_1))^{R-1} \geq 2^{\kappa}.$$

D.2 Plonk Gates

For Reinforced Concrete, we use the values provided in the original paper. See [34, Table 1] for reference.

Poseidon2 The number of Plonk gates in Poseidon2 with R_F full rounds and R_P partial rounds is given by:

$$3(tR_F + R_P) + (R_F + 1) \cdot \#\mathrm{mat}_F + R_P \cdot \#\mathrm{mat}_P,$$

where $\#\mathrm{mat}_F$ and $\#\mathrm{mat}_P$ represent the number of Plonk gates per linear layer for full round and partial round, respectively, as follows:

$$(\#\mathrm{mat}_F, \#\mathrm{mat}_P) = \begin{cases} (5, 5) \text{ for } t = 3, \\ (8, 7) \text{ for } t = 4, \\ (24, 15) \text{ for } t = 8. \end{cases}$$

Rescue-Prime The number of Plonk gates in Rescue-Prime is

$$2R(3t + t(t - 1)).$$

Anemoi As outlined in [15, Section 7.2], the number of Plonk gates in Anemoi is given by:

$$5Rt + (R+1) \cdot \#\text{mat},$$

where #mat is the number of Plonk gates per linear layer, as follows:

$$\#\text{mat} = \begin{cases} 4 \text{ for } t = 4, \\ 10 \text{ for } t = 6, \\ 16 \text{ for } t = 8. \end{cases}$$

Griffin As outlined in [33, Section 7.3], the number of Plonk gates in Griffin is given by:

$$\begin{cases} (R+1) \cdot \#\text{mat} + R(4t-3) \text{ for } t = 3, \\ (R+1) \cdot \#\text{mat} + R(4t-4) \text{ for } t = 4, 8, \end{cases}$$

where #mat is the number of Plonk gates per linear layer, as follows:

$$\#\text{mat} = \begin{cases} 5 \text{ for } t = 3, \\ 8 \text{ for } t = 4, \\ 24 \text{ for } t = 8. \end{cases}$$

Arion As outlined in [53, Lemma 10], the number of Plonk gates in ArionHash with $d_1 = 5$ and $d_2 = 257$ is given by:

$$R(9t-1) + (R+1) \cdot \#\text{mat},$$

where #mat is the number of Plonk gates per linear layer, as follows:

$$\#\text{mat} = \begin{cases} 6 \text{ for } t = 3, \\ 4(t-1) \text{ for } t \geq 4. \end{cases}$$