XHash8 and XHash12: Efficient STARK-friendly Hash Functions

Tomer Ashur
3MI Labs, Leuven, Belgium
Polygon Research
tomer@3milabs.tech

Amit Singh Bhati
COSIC, KU Leuven, Belgium
amitsingh.bhati@esat.kuleuven.be

Al Kindi
Polygon
al-kindi-0@protonmail.com

Mohammad Mahzoun
Eindhoven University of Technology
mail@mahzoun.me

Abstract

Zero-knowledge proofs are widely used in real-world applications for authentication, access control, blockchains, and cryptocurrencies, to name a few. A core element in some Zero-Knowledge proof systems is the underlying pseudorandom function, which is usually modeled as a hash function. This underlying hash function must be efficient over finite fields of large prime order, which means that straightforward choices such as SHA2 are not practical. The need for efficient hash functions has led to the development of a new paradigm known as Arithmetization-Oriented designs.

In this work, we propose two new AO hash functions, XHash8 and XHash12 which are inspired by the Marvellous design strategy and outperform the current offering of this family. Based on our experiments, XHash8 performs ≈ 2.5 times faster than RPO, and XHash12 performs ≈ 1.7 times faster than RPO, while at the same time inheriting the security and robustness of the Marvellous design strategy.

1 Introduction

Zero-knowledge (ZK) proof systems are advanced cryptographic protocols used to prove the validity of a statement to a party (a verifier) without revealing any further information. ZK systems are used in a variety of applications such as blockchains and banking systems. A pioneer ZK proof system is ZK-STARK [BSBHR18] which is a scalable proof system widely used in industrial applications such as Polygon Miden and Starknet. A core building block in a ZK-STARK is the underlying hash function. The pursuit for efficient ZK-STARKs has motivated the need for Arithmetization Oriented (AO) primitives, i.e., cryptographic primitives with low arithmetic complexity. Examples of such AO designs are [AMT22a, GHR+22, BBC+22, ARS+15, CCF+18, MJSC16, DEG+18, HL20, DGH+21, CIR22, CHMS22, HKL+22, SLST23]. However, the two main AO hash functions in practical use for ZK proof systems are Rescue [AAB+20] and Poseidon [GKR+21].

The design of Rescue was a major improvement in the domain of AO primitives by introducing non-procedural computations. While Rescue is competitive inside the STARK, its security-first approach leaves room for improvement when executed on more standard platforms such as a CPU or an FPGA. Indeed, subsequent improvements were obtained in follow-up works such as Rescue-Prime [SAD20] and more recently, RPO [AKM+22] where Ashur, Kindi, Meier, Szepieniec, & Threadbare published an optimized variant for the specific case of 2-to-1 compression of elements from a finite field F_p with $p = 2^{64} - 2^{32} + 1$ for 128- and 160 bits of security. RPO offers the same performance as Rescue-Prime when evaluated inside the STARK but is about 40% faster on a standard CPU. The lion's share of this improvement was achieved by finding a particularly efficient MDS matrix but without changing the overall design or introducing new operations.

In this work, we take the next step in designing AO primitives. We start by analyzing the remaining bottlenecks in RPO and improve the efficiency by introducing a new operation: exponentiation over an extension field. Arithmetic over an extension field is quite common in the design of traditional symmetric-key algorithms such as AES [DR02a]. In the context of AO primitives, this approach was carried over to Vision [AAB+20] and Chaghri [AMT22b] as well as the now-defunct Starkad [GKK+19]; but it was not yet considered for primitives operating over large prime fields.

In this work we use, for the first time to the best of our knowledge, S-boxes employing extension field arithmetic over large prime fields. Concretely, we propose two new permutations:

1. XHash12: interleaving between Rescue rounds and a new type of round;
2. XHash8: a more aggressively optimized variant of XHash12 which does not apply the expensive $x^{1/\alpha}$ S-box to all elements.

Related work. There are multiple hash functions designed to be efficient when used in ZK-proof systems. One of these is Poseidon, an efficient hash function that is widely used. The partial layers in the design of Poseidon were subject to multiple attacks [KR21a, BCD+20] subsequently resulting in updated parameters. Recently, in [ABM23a] the authors analyzed the security of Poseidon and showed that in some synthetic cases, the number of rounds proposed for providing a specific security level is insufficient. In the same work, the authors showed some flaws in Poseidon's security proof.

2 Background

2.1 Notation

In the rest of this work, $p = 2^{64} - 2^{32} + 1$ and $\mathbb{F}_p$ is a finite field of order p. Vectors and matrices are denoted by capital Latin letters. For example, the vector S of size n is denoted by $S = (S[0], \dots, S[n-1])$ and the elements of a matrix M with dimensions $n \times m$ are denoted by $M[i][j]$ where $0 \le i < n, 0 \le j < m$. $\boxplus$ is used to denote addition over the finite field $\mathbb{F}_p$.

2.2 Rescue-Prime Optimized

Rescue-Prime Optimized (RPO) is a sponge function instantiated with a permutation over $\mathbb{F}_p^{12}$. The permutation consists of seven rounds and each round can be described in terms of four components:

- an S-box $\pi_0 : x \mapsto x^7$;
- an S-box $\pi_1 : x \mapsto x^{\frac{1}{7}}$;
- an MDS matrix M; and
- constant addition $\boxplus c$.

With respect to the state vector $S \in \mathbb{F}_p^{12}$ these components allow to define two types of steps:

- an (F)-step works as follows: first, the S-box π_0 is applied to each of the state elements to provide non-linearity. Then, the state vector is multiplied with an MDS matrix to spread local properties over the entire state. Finally, a different round constant is added to each element to avoid self-symmetry between different rounds;
- the (B)-step is almost the same, only that π_1 is applied instead of π_0. That is, first the S-box π_1 is applied to each of the state elements to provide non-linearity. Then, the state vector is multiplied with an MDS matrix to spread local properties over the entire state. Finally, a different round constant is added to each element to avoid self-symmetry between different rounds.

With this notation, a typical round for a Rescue-like function is $(F)(B)$. Concretely, RPO can be written as

$$(F)(B)(F)(B)(F)(B)(F)(B)(F)(B)(F)(B)(F)(B)\,,$$

except that the last M and constants injection are moved to the beginning for reasons explained in [AAB+20, Sec. 4.3].

3 XHash8 and XHash12 — Design Rationale

So far, three main components were used in Marvellous designs: S-boxes, linear layers, and injection of round constants. Together, S-boxes, MDS multiplication, and constant injection result in a dense polynomial representation of high degree. Building on Rescue's S-boxes, the designers of RPO fixed the MDS matrix to one that can be efficiently computed over $p = 2^{64} - 2^{32} + 1$. A natural question then arises—whether the efficiency can be improved any further without sacrificing security. Noting that the costs of π_0 and constant injection are almost negligible and M has been aggressively optimized, the remaining bottleneck has been observed to be the π_1 S-box. On a CPU, each call to π_1 requires about 70 multiplications, there are 12 calls in each (B)-step, and seven (B) steps; totaling in $70 \cdot 12 \cdot 7 = 5880$ multiplications. This is by far the largest cost driver in the entire permutation.

Before attempting to reduce this cost, one must determine what purpose the (B)-step serves. Citing [AAB+20], the two S-boxes (π_0, π_1) are motivated as follows:

The difference between π_0 and π_1 is in their degree. They should be chosen such that π_0 has a high degree when evaluated forward (i.e., in the direction of the encryption) and a low degree when evaluated backward (i.e., in the direction of the decryption). The other S-box, namely π_1, is chosen with the opposite goal (i.e., to have a low degree in the forward direction and a high degree in the backward direction). This choice serves to achieve three objectives: (i) no matter which direction an adversary is trying to attack, the degree is guaranteed to be high; (ii) it results in the same cost for the encryption and decryption functions, and (iii) owing to non-procedural computation, the low-degree representation of each S-box can be evaluated efficiently.

We remind the reader that the stated goal of [AAB+20] was to design a general-purpose primitive, usable not only for hashing, not only in STARKs, and to work with any sufficiently large prime field. In the context of Polygon Miden, Goal (ii) is irrelevant since we expect honest users only to compute the primitive "forward", never requiring efficient "backward" evaluation. Hence, we can abandon Goal (ii) altogether.

For similar reasons, goal (i) can also be relaxed. Avoiding attacks still requires that the degree in either direction be high—but it no longer needs to be the same in both directions. First, observe that the high degree in the "backward" direction is ensured by the (F)-rounds, which we are not looking to optimize. Therefore, the degree remains sufficiently high and the security argument does not need to be re-evaluated.

We now conjecture the possibility that the (B)-step may offer "too much security" for our purposes. Concretely, [AAB+20] observes that the (B)-step achieves maximal degree already after two rounds. Given that the minimal number of rounds is set to ten in Rescue, eight in Rescue-Prime, and seven in RPO, positing a (B')-step that can achieve, hypothetically speaking, a maximal degree in four rounds; the overall performance can be improved without affecting security. Getting ahead of ourselves, this is exactly what we will be doing. To reduce the cost of achieving a compact polynomial description, we propose to use power maps over an extension field. Exponentiation in an extension field simultaneously provides diffusion by mixing the base field elements as well as confusion by doing so in a non-linear way.

Revisiting the design rationale for general Marvellous designs we see that in the context of a hash function operating over F_p^{12} with p prime, the interpolation attack would be the main concern in case the polynomial degree is not sufficiently high. Considering in detail the argument against the interpolation attack we see that resistance is achieved when the univariate polynomial describing the cipher is:

1. dense; and
2. of maximal degree.

Intuitively, the composition of an MDS matrix (which ensures optimal diffusion) with a power map ensures density due to the Multinomial Theorem.

We are left to ensure a maximal degree. Considering that

$$x^{\frac{1}{7}} = x^{(2p-1)/7} = x^{10540996611094048183}$$

achieves an almost maximal degree already in a single call to π_1 we can informally argue that this is an "overkill" and that an S-box resulting in a lower degree per step may still be sufficient to resist the interpolation attack as long as it reaches a maximal degree within a reasonable number of rounds. Thus, we are looking for a power map β such that $7 < \beta < \frac{p-1}{7}$. However, by definition, such an S-box will be inefficient to compute inside the STARK both directly and folded.

Seeing that a better power map is not readily available, we can instead apply the same power map, but to fewer elements. However, with this approach, only the elements to which the power map was applied are "protected". If this approach is taken, the partial S-box layer must be complemented with another operation that would mix high-degree elements with the rest in a non-linear way. This, if done right, would ensure that all polynomials are both dense and of high degree.

4 Specification

We complement the S-boxes (π_0, π_1) described in Section 2.2 with a third type of S-box:

- π_2 is similar to π_0 in that it takes a field element and raises it to the 7th power. However this time, the element is in F_{p^3} rather than F_p.

Using the new S-box π_2, we define XHash8 and XHash12. XHash12 uses a full layer of π_1 S-boxes applied to all elements of the state. XHash8 further improves efficiency using a partial layer of π_1 S-boxes that are applied to 8 out of 12 elements in the state.

4.1 XHash12: With Full $x^{\frac{1}{7}}$ Layer

XHash12 employs a full layer of π_1 S-boxes, and works as follows:

- A version of the (F)-step starting with round constants injection, followed by an MDS multiplication, and concluded by applying π_0 to each word of the state;
- A version of the (B)-step starting with MDS multiplication, followed by round constants injection, and concluded by applying π_1 to each word of the state;
- A (P3)-step starting with constants injection; then, the 12-element state is restructured as a 4-element vector in a cubic extension field, *i.e.*,

$$(s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9, s_{10}, s_{11}) \mapsto (S_{0,1,2}, S_{3,4,5}, S_{6,7,8}, S_{9,10,11})$$

  where each $s_{i,j,k} \in F_{p^3}$. This is followed by an application of π_2 to each of these extension field elements, such that

$$(S_{0,1,2}, S_{3,4,5}, S_{6,7,8}, S_{9,10,11}) \mapsto (S_{0,1,2}^7, S_{3,4,5}^7, S_{6,7,8}^7, S_{9,10,11}^7).$$

  At this point, the state is decomposed back into a 12-element vector in F_p;
- The last step is a special one, denoted by (MC). It consists of an MDS multiplication followed by round constants injection.

The state consists of 12 field elements in F_p where $p = 2^{64} - 2^{32} + 1$. The permutation consists of 10 steps in total, described as follows:

$$(F)(B)(P3)(F)(B)(P3)(F)(B)(P3)(MC),$$

and depicted in Figure 1.

4.2 XHash8: With Partial $x^{\frac{1}{7}}$ Layer

XHash8 is a more aggressively optimized version XHash12. The difference is that it employs partial layers of π_1 S-boxes and works as follows:

- A version of the (F)-step starting with a constant addition, followed by an MDS multiplication, and concluded by applying π_0 to the each word of the state;

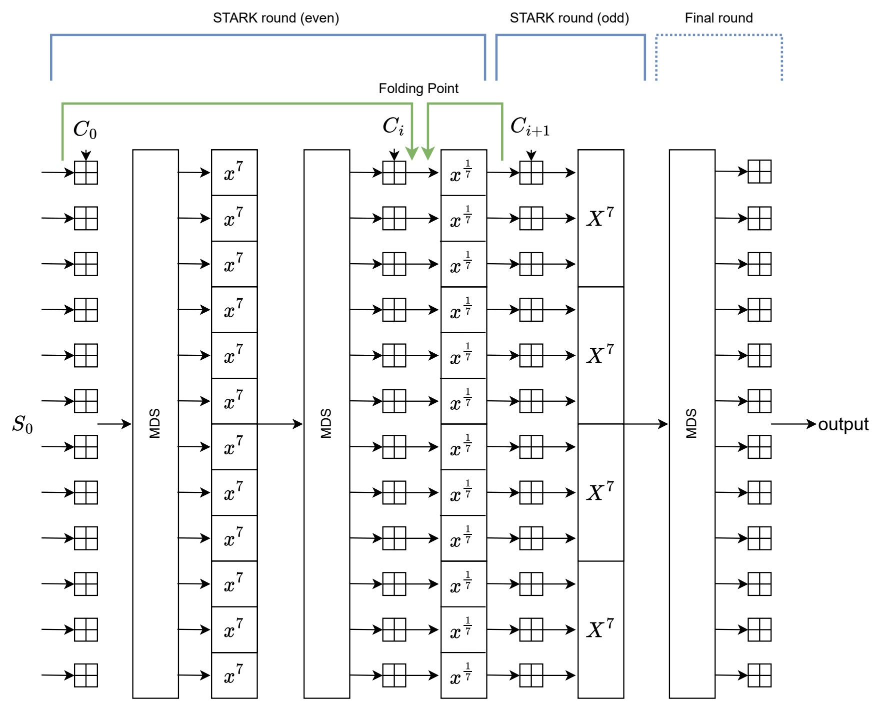

Figure 1: The last four steps of the XHash12 permutation. The folding point depicts how to compress the (F) and (B) rounds into a single low-degree polynomial. This allows the verifier to evaluate a sequence of (F)(B)(P3) steps as two STARK rounds (blue lines). The dotted blue line represents the finalization step.

- A modified version of the (B)-step, namely the (B')-step, starting MDS multiplication, followed by constant addition, and concluded by applying π_1 to 8 out of the 12 state elements:

$$(s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9, s_{10}, s_{11}) \mapsto (s_0^{\frac{1}{7}}, s_1, s_2^{\frac{1}{7}}, s_3^{\frac{1}{7}}, s_4, s_5^{\frac{1}{7}}, s_6^{\frac{1}{7}}, s_7, s_8^{\frac{1}{7}}, s_9^{\frac{1}{7}}, s_{10}, s_{11}^{\frac{1}{7}})\,;$$

- A (P3)-step starting with constants injection; then, the 12-element state is restructured as a 4-element vector in a cubic extension field, *i.e.*,

$$(s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9, s_{10}, s_{11}) \mapsto (S_{0,1,2}, S_{3,4,5}, S_{6,7,8}, S_{9,10,11})$$

  where each $s_{i,j,k} \in F_{p^3}$. This is followed by an application of π_2 to each of these extension field elements, such that

$$(S_{0,1,2}, S_{3,4,5}, S_{6,7,8}, S_{9,10,11}) \mapsto (S_{0,1,2}^7, S_{3,4,5}^7, S_{6,7,8}^7, S_{9,10,11}^7).$$

  At this point, the state is decomposed back into a 12-element vector in F_p.

- The last step is a special one, denoted by (MC). It consists of an MDS multiplication followed by round constants injection.

Again, the state consists of 12 field elements in F_p where $p = 2^{64} - 2^{32} + 1$ and the permutation consists of 10 steps:

$$(F)(B')(P3)(F)(B')(P3)(F)(B')(P3)(MC)\,;$$

as depicted in Figure 2.

4.3 The Hash Function

A hash function offering 128-bit security is obtained by using either of these permutations in a sponge construction with the elements $(s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7)$ as the outer part, and $(s_8, s_9, s_{10}, s_{11})$ as the inner part. The round constants are randomly selected and the MDS is the same as the one used in RPO. Domain separation is handled in the same way as in RPO by designating certain values of an inner part element to encode the domain.

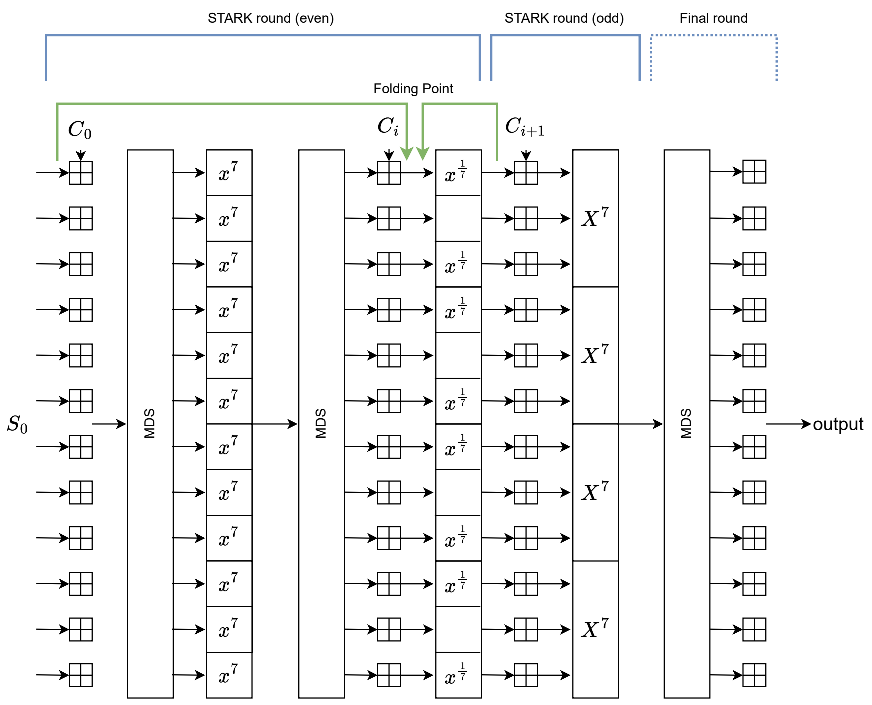

Figure 2: The last four steps of the XHash8 permutation. The folding point depicts how to compress the (F) and (B') rounds into a single low-degree polynomial. This allows the verifier to evaluate a sequence of (F)(B')(P3) steps as two STARK rounds (blue lines). The dotted blue line represents the finalization step.

5 Security Rationale

We analyze the security of both XHash8 and XHash12 against standard attacks.

5.1 Differential cryptanalysis

We analyze the resistance of XHash8 against differential cryptanalysis by counting active S-boxes. The resistance of XHash8 also ensures the resistance of XHash12 since, for any trail pattern, XHash12 activates the same number or more S-boxes than XHash8. Since the same set of S-boxes is used in both, the probability of the best differential characteristic for XHash8 upper bounds the probability of the best different characteristic for XHash12.

The analysis follows the standard argument: we find a lower bound on the number of active S-boxes and an upper bound on the probability of the best differential transition in any S-box; the quantity obtained from raising the latter to the power of the former upper bounds the probability of the best differential characteristic.

The aspects of our new function that require special care are:

1. The (B') step uses a partial S-box layer;
2. The (P3) step mixes base field elements in a non-linear way.

As a result of these particularities, the Two-Round Propagation Theorem originally stated by Daemen and Rijmen in [DR02b, Thm. 9.3.1] for the case of alternating block ciphers cannot be applied directly to XHash8.

To address these observations we provide a step-by-step detailed analysis. First, we observe that in the first step the adversary controls only the outer part of the sponge and therefore they can only introduce a difference in $1 \leq d_{(I)} \leq r = 8$ field elements. Consequently, after the application of the MDS matrix $5 \leq d_{(F)} \leq 12$ S-boxes are active in the (F)-step. For the (B')-step, the adversary can activate $0 \leq d_{(B')} \leq 8$ S-boxes, followed by $1 \leq d_{(P3)} \leq 4$ S-boxes in the (P3)-step.[1] In total, over a

[1]We note that the S-boxes in the (P3)-step are of different type than those in the (F)- and (B)- steps.

triplet of consecutive (F)(B')(P3) steps, at least nine S-boxes are activated from (π_0, π_1), and at least one S-box of type π_2.

In theorem 5.1 we show that S-boxes of type π_2 are $(\gamma - 1)$-uniform which in our setting means that their differential transition probability is upper bounded by 2^{-186}. Completing the argument, we see that the probability of the best differential transition over a triplet of consecutive (F)(B')(P3) is upper bounded by $2^{9\cdot(-60)} \cdot 2^{1\cdot(-186)} = 2^{-540-186} = 2^{-726}$; which is already enough to resist differential attacks at the 128-bit security level.

Theorem 5.1. *Let $\mathbb{F}_q$ be a finite field of order $q = p^n$ and characteristic p. Let $F(x) = x^\gamma$ be a power map defined over $\mathbb{F}_q$, then F is differentially $(\gamma - 1)$-uniform.*

Proof. Given $\alpha, \beta \in \mathbb{F}_q, \alpha \neq 0$, the cardinality of the set $\mathcal{D} = \{x \in \mathbb{F}_q | F(x + \alpha) - F(x) = \beta\}$ can be computed as the number of roots for the following polynomial:

$$(x + \alpha)^\gamma - x^\gamma = \sum_{i=0}^{\gamma-1} \binom{\gamma}{i} \alpha^{\gamma-i} x^i, \tag{1}$$

which is a polynomial of degree $\gamma - 1$. It is well known that a polynomial of degree α has at most α roots. We therefore obtain the upper bound $|\mathcal{D}| \leq \gamma - 1$. □

5.2 Algebraic Attacks

Polynomial Degree We again analyze only the polynomial degree of XHash8 and use it also as a lower bound for the polynomial degree of XHash12. Similar to other algorithms from the Marvellous family, the high polynomial degree is obtained by applying a large power map to the elements of the state. However, when we use XHash8, π_1 is applied only to part of the state. Supposedly, even if we ignore the (P3) round, the next application of M will distribute the high-degree terms to the entire state. However, this is hard to argue formally without reverting to complicated case analysis and furthermore, it is not clear that a linear transformation is enough to spread algebraic properties in a sufficient way.[2] Thus, we do not abstract the (P3)-step and instead analyze its diffusion properties.

Let $(x_0, x_1, x_2) \in F_p^3$ and $x_{0,1,2} \in F_{p^3}$. For brevity, we consider only a single squaring operation: Consider the three polynomials describing the base field elements after a single squaring operation:

$$\begin{aligned} x_{0,1,2}^2 = (&c_0 x_0^2 + c_1 x_1 \cdot x_2, \\ &x_1 \cdot (c_3 x_0 + c_4 x_2) + c_5 x_2^2, \\ &c_6 x_0 \cdot x_2 + c_7 x_1^2 + c_8 x_2^2), \end{aligned}$$

where $c_i \in \mathbb{F}_p$ for all $0 \leq i \leq 8$. Taking into account that $x_0 = \pi_1(y_0) = y_0^{1/7}, x_2 = \pi_1(y_2) = y_2^{1/7}$ and setting $x_1 = y_1$ we get

$$\begin{aligned} x_{0,1,2}^2 = (&y_0^{2/7} + y_1 \cdot y_2^{1/7}, \\ &y_1 \cdot (y_0^{1/7} + y_2^{1/7}) + y_2^{2/7}, \\ &y_0^{1/7} \cdot y_2^{1/7} + y_1^2 + y_2^{2/7}), \end{aligned}$$

and conclude that every possible polynomial description of the initial state is of a high degree even before applying the MDS matrix. While this observation is already enough to argue resistance against interpolation attacks, in Appendix A we work out the complete case for x^7 in F_{p^3}.

Density Several works have recently noticed that the solving degree of a polynomial system describing an AO primitive and consisting of π_0 S-boxes alone is smaller than the degree of regularity. In particular, Sauer observed in [Sau21a] that for Rescue-like functions (i.e., functions mixing π_0 and π_1 S-boxes), the solving degree grows at the same rate as the Macaulay bound; whereas this is not the case for Poseidon-like ciphers. This observation was later independently confirmed by Ashur, Kindi, Meier, Szepieniec, & Threadbare in the design of RPO and more recently quantified and leveraged into an attack by Ashur, Buschman, and Mahzoun in [ABM23b].

[2]See [KR21b] for the binary case and [ABM23b] for the prime case.

To explain this observation Sauer introduces a new metric, "involvement", as a proxy for the difficulty of finding a Gröbner basis [Sau21b]. Two notions of involvement are suggested based on vectors of origin; one based on the normalized average of polynomials from the original system required to describe each element in the Gröbner basis and the other on the number of non-zero coefficients in the matrix describing the vectors of origin. Sauer presents heat maps suggesting that the latter is a reasonable, even if noisy, proxy for the difficulty of finding a Gröbner basis.

Here, we suggest that the notion of density is sufficient to capture the quality we are interested in when designing a symmetric-key primitive. As an instructive example, consider the polynomial modeling of a single Rescue round consisting of consecutive (F)(B) steps:

$$\left(\sum_{k=0}^{m-1} M[j,k]S_{2i-1}[k]^{\alpha}\right) - \left(\sum_{k=0}^{m-1} M^{-1}[j,k](S_{2i+1}[k] - K_{2i+1}[k])\right)^{\alpha} + K_{2i}[j] = 0\,. \tag{2}$$

Note that in the second term, the power map α is applied to the entire sum, creating a complicated expression due to the multinomial theorem. For $\alpha = 7$ and $m = 12$ as in the case of RPO, each such polynomial consists of $12 + \binom{18}{11}$ monomials in 24 variables. Comparably, the modeling of an (F)-round

$$\left(\sum_{k=0}^{m-1} M[j,k]S_{i-1}[k]^{\alpha}\right) + K_{2i}[j] - S_i[j] = 0 \tag{3}$$

consists of $12+1 = 13$ monomials in $12+1 = 13$ variables. It is straightforward to see that polynomials of type (2) are denser than polynomials of type (3) and since clearly density implies involvement (but not necessarily the other way around) we conjecture density to be the explanation to Sauer's observations.

XHash8 achieves density differently. Considering an even STARK round consisting of one pair of (F)(B') steps. This round gives rise to two types of polynomials:

$$\left(\sum_{k=0}^{m-1} M[j,k]S_{i-1}[k]^{\alpha}\right) + K_{2i}[j] - S_i^7[j] = 0 \qquad j \not\equiv 1 \pmod 3$$
$$\left(\sum_{k=0}^{m-1} M[j,k]S_{i-1}[k]^{\alpha}\right) + K_{2i}[j] - S_i[j] = 0 \qquad j \equiv 1 \pmod 3$$

neither of which is dense. Density arises from the polynomial modeling of (P3). The polynomial description of (P3) can be found in Appendix A showing that each base element can be modeled as a 3-variate polynomial of degree seven with 31–34 monomials. Following the MDS, each state element is modeled by a polynomial consisting of $36 \cdot 4 = 144$ monomials in 13 variables.

Resistance to Gröbner basis attacks is a delicate matter and no good method to evaluate it has been established as of yet. The state-of-the-art approach is the one described in [AAB+20] where the behavior of toy examples is compared to the expected behavior of a regular system and then extrapolated to larger cases. We performed similar experiments and the results are reported in Appendix B.

6 Performance

A performance comparison of XHash8 and XHash12 with RPO is described in Table 1. The benchmarks are showing the result of 2-to-1 hashing for random values over F_p with $p = 2^{64} - 2^{32} + 1$. The code was written using Rust 1.68 and executed on an Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz.

Table 1: Cost of hashing random values using RPO, XHash8, and XHash12 over a CPU

Hash Function	Time(μs)
RPO	8.1474
XHash8	2.9547
XHash12	4.0906

A Polynomial representation of π_2

Let $(x_0, x_1, x_2) \in F_p^3$ and $(y_0, y_1, y_2) \in F_p^3$ such that

$$\pi_2(x_0, x_1, x_2) = (y_0, y_1, y_2);$$

then

$$\begin{aligned}
&y_0 = \\
&\quad x_0^7 + 35x_0^4x_1^3 + 21x_0^2x_1^5 + 7x_0x_1^6 + x_1^7 + 42x_0^5x_1x_2 + 140x_0^3x_1^3x_2 + 105x_0^2x_1^4x_2 + \\
&\quad 42x_0x_1^5x_2 + 14x_1^6x_2 + 105x_0^4x_1x_2^2 + 210x_0^3x_1^2x_2^2 + 210x_0^2x_1^3x_2^2 + 210x_0x_1^4x_2^2 + \\
&\quad 42x_1^5x_2^2 + 35x_0^4x_2^3 + 140x_0^3x_1x_2^3 + 420x_0^2x_1^2x_2^3 + 280x_0x_1^3x_2^3 + 105x_1^4x_2^3 + \\
&\quad 70x_0^3x_2^4 + 210x_0^2x_1x_2^4 + 315x_0x_1^2x_2^4 + 140x_1^3x_2^4 + 63x_0^2x_2^5 + 168x_0x_1x_2^5 + 105x_1^2x_2^5 + \\
&\quad 35x_0x_2^6 + 49x_1x_2^6 + 9x_2^7, \\
&y_1 = \\
&\quad 7x_0^6x_1 + 35x_0^4x_1^3 + 35x_0^3x_1^4 + 21x_0^2x_1^5 + 14x_0x_1^6 + 2x_1^7 + 42x_0^5x_1x_2 + 105x_0^4x_1^2x_2 + \\
&\quad 140x_0^3x_1^3x_2 + 210x_0^2x_1^4x_2 + 84x_0x_1^5x_2 + 21x_1^6x_2 + 21x_0^5x_2^2 + 105x_0^4x_1x_2^2 + \\
&\quad 420x_0^3x_1^2x_2^2 + 420x_0^2x_1^3x_2^2 + 315x_0x_1^4x_2^2 + 84x_1^5x_2^2 + 70x_0^4x_2^3 + 280x_0^3x_1x_2^3 + \\
&\quad 630x_0^2x_1^2x_2^3 + 560x_0x_1^3x_2^3 + 175x_1^4x_2^3 + 105x_0^3x_2^4 + 420x_0^2x_1x_2^4 + 525x_0x_1^2x_2^4 + \\
&\quad 245x_1^3x_2^4 + 105x_0^2x_2^5 + 294x_0x_1x_2^5 + 189x_1^2x_2^5 + 63x_0x_2^6 + 84x_1x_2^6 + 16x_2^7, \\
&y_2 = \\
&\quad 21x_0^5x_1^2 + 35x_0^3x_1^4 + 21x_0^2x_1^5 + 7x_0x_1^6 + 2x_1^7 + 7x_0^6x_2 + 105x_0^4x_1^2x_2 + 140x_0^3x_1^3x_2 + \\
&\quad 105x_0^2x_1^4x_2 + 84x_0x_1^5x_2 + 14x_1^6x_2 + 21x_0^5x_2^2 + 105x_0^4x_1x_2^2 + 210x_0^3x_1^2x_2^2 + \\
&\quad 420x_0^2x_1^3x_2^2 + 210x_0x_1^4x_2^2 + 63x_1^5x_2^2 + 35x_0^4x_2^3 + 280x_0^3x_1x_2^3 + 420x_0^2x_1^2x_2^3 + \\
&\quad 420x_0x_1^3x_2^3 + 140x_1^4x_2^3 + 70x_0^3x_2^4 + 315x_0^2x_1x_2^4 + 420x_0x_1^2x_2^4 + 175x_1^3x_2^4 + \\
&\quad 84x_0^2x_2^5 + 210x_0x_1x_2^5 + 147x_1^2x_2^5 + 49x_0x_2^6 + 63x_1x_2^6 + 12x_2^7.
\end{aligned}$$

Similarly

$$\pi_2(x_0^{(2p-1)/7}, x_1, x_2^{(2p-1)/7}) = (y_0, y_1, y_2)$$

gives

$$\begin{aligned}
y_0 = {} & x_0^{(2p-1)/7} + 35x_0^{(8p-4)/7}x_1^3 + 21x_0^{(4p-2)/7}x_1^5 + 7x_0^{(2p-1)/7}x_1^6 + x_1^7 + 42x_0^{(10p-5)/7}x_1x_2^{(2p-1)/7} \\
& + 140x_0^{(6p-3)/7}x_1^3x_2^{(2p-1)/7} + 105x_0^{(4p-2)/7}x_1^4x_2^{(2p-1)/7} + 42x_0^{(2p-1)/7}x_1^5x_2^{(2p-1)/7} + 14x_1^6x_2^{(2p-1)/7} \\
& + 105x_0^{(8p-4)/7}x_1x_2^{(4p-2)/7} + 210x_0^{(6p-3)/7}x_1^2x_2^{(4p-2)/7} + 210x_0^{(4p-2)/7}x_1^3x_2^{(4p-2)/7} \\
& + 210x_0^{(2p-1)/7}x_1^4x_2^{(4p-2)/7} + 42x_1^5x_2^{(4p-2)/7} + 35x_0^{(8p-4)/7}x_2^{(6p-3)/7} + 140x_0^{(6p-3)/7}x_1x_2^{(6p-3)/7} \\
& + 420x_0^{(4p-2)/7}x_1^2x_2^{(6p-3)/7} + 280x_0^{(2p-1)/7}x_1^3x_2^{(6p-3)/7} + 105x_1^4x_2^{(6p-3)/7} + 70x_0^{(6p-3)/7}x_2^{(8p-4)/7} \\
& + 210x_0^{(4p-2)/7}x_1x_2^{(8p-4)/7} + 315x_0^{(2p-1)/7}x_1^2x_2^{(8p-4)/7} + 140x_1^3x_2^{(8p-4)/7} + 63x_0^{(4p-2)/7}x_2^{(10p-5)/7} \\
& + 168x_0^{(2p-1)/7}x_1x_2^{(10p-5)/7} + 105x_1^2x_2^{(10p-5)/7} + 35x_0^{(2p-1)/7}x_2^{(12p-6)/7} + 49x_1x_2^{(12p-6)/7} + 9x_2^{(2p-1)},
\end{aligned}$$

$$\begin{aligned}
y_1 = {} & 7x_0^{(12p-6)/7}x_1 + 35x_0^{(8p-4)/7}x_1^3 + 35x_0^{(6p-3)/7}x_1^4 + 21x_0^{(4p-2)/7}x_1^5 + 14x_0^{(2p-1)/7}x_1^6 \\
& + 2x_1^7 + 42x_0^{(10p-5)/7}x_1x_2^{(2p-1)/7} + 105x_0^{(8p-4)/7}x_1^2x_2^{(2p-1)/7} + 140x_0^{(6p-3)/7}x_1^3x_2^{(2p-1)/7} \\
& + 210x_0^{(4p-2)/7}x_1^4x_2^{(2p-1)/7} + 84x_0^{(2p-1)/7}x_1^5x_2^{(2p-1)/7} + 21x_1^6x_2^{(2p-1)/7} + 21x_0^{(10p-5)/7}x_2^{(4p-2)/7} \\
& + 105x_0^{(8p-4)/7}x_1x_2^{(4p-2)/7} + 420x_0^{(6p-3)/7}x_1^2x_2^{(4p-2)/7} + 420x_0^{(4p-2)/7}x_1^3x_2^{(4p-2)/7} \\
& + 315x_0^{(2p-1)/7}x_1^4x_2^{(4p-2)/7} + 84x_1^5x_2^{(4p-2)/7} + 70x_0^{(8p-4)/7}x_2^{(6p-3)/7} + 280x_0^{(6p-3)/7}x_1x_2^{(6p-3)/7} \\
& + 630x_0^{(4p-2)/7}x_1^2x_2^{(6p-3)/7} + 560x_0^{(2p-1)/7}x_1^3x_2^{(6p-3)/7} + 175x_1^4x_2^{(6p-3)/7} + 105x_0^{(6p-3)/7}x_2^{(8p-4)/7} \\
& + 420x_0^{(4p-2)/7}x_1x_2^{(8p-4)/7} + 525x_0^{(2p-1)/7}x_1^2x_2^{(8p-4)/7} + 245x_1^3x_2^{(8p-4)/7} + 105x_0^{(4p-2)/7}x_2^{(10p-5)/7} \\
& + 294x_0^{(2p-1)/7}x_1x_2^{(10p-5)/7} + 189x_1^2x_2^{(10p-5)/7} + 63x_0^{(2p-1)/7}x_2^{(12p-6)/7} + 84x_1x_2^{(12p-6)/7} + 16x_2^{(2p-1)},
\end{aligned}$$

$$\begin{aligned}
y_2 = {} & 21x_0^{(10p-5)/7}x_1^2 + 35x_0^{(6p-3)/7}x_1^4 + 21x_0^{(4p-2)/7}x_1^5 + 7x_0^{(2p-1)/7}x_1^6 + 2x_1^7 + 7x_0^{(12p-6)/7}x_2^{(2p-1)/7} \\
& + 105x_0^{(8p-4)/7}x_1^2x_2^{(2p-1)/7} + 140x_0^{(6p-3)/7}x_1^3x_2^{(2p-1)/7} + 105x_0^{(4p-2)/7}x_1^4x_2^{(2p-1)/7} \\
& + 84x_0^{(2p-1)/7}x_1^5x_2^{(2p-1)/7} + 14x_1^6x_2^{(2p-1)/7} + 21x_0^{(10p-5)/7}x_2^{(4p-2)/7} + 105x_0^{(8p-4)/7}x_1x_2^{(4p-2)/7} \\
& + 210x_0^{(6p-3)/7}x_1^2x_2^{(4p-2)/7} + 420x_0^{(4p-2)/7}x_1^3x_2^{(4p-2)/7} + 210x_0^{(2p-1)/7}x_1^4x_2^{(4p-2)/7} \\
& + 63x_1^5x_2^{(4p-2)/7} + 35x_0^{(8p-4)/7}x_2^{(6p-3)/7} + 280x_0^{(6p-3)/7}x_1x_2^{(6p-3)/7} + 420x_0^{(4p-2)/7}x_1^2x_2^{(6p-3)/7} \\
& + 420x_0^{(2p-1)/7}x_1^3x_2^{(6p-3)/7} + 140x_1^4x_2^{(6p-3)/7} + 70x_0^{(6p-3)/7}x_2^{(8p-4)/7} + 315x_0^{(4p-2)/7}x_1x_2^{(8p-4)/7} \\
& + 420x_0^{(2p-1)/7}x_1^2x_2^{(8p-4)/7} + 175x_1^3x_2^{(8p-4)/7} + 84x_0^{(4p-2)/7}x_2^{(10p-5)/7} + 210x_0^{(2p-1)/7}x_1x_2^{(10p-5)/7} \\
& + 147x_1^2x_2^{(10p-5)/7} + 49x_0^{(2p-1)/7}x_2^{(12p-6)/7} + 63x_1x_2^{(12p-6)/7} + 12x_2^{(2p-1)}.
\end{aligned}$$

B Gröbner Basis Resistance

To compute the Gröbner basis, each round is divided into two steps: The first step is called the basic step and contains π_0, linear layer, constant addition, and π_1. The second step is called the extension step which contains constant addition, π_2, linear layer, and constant addition. In our experiments, we used a toy instance with state size $m = 3$, rate $r = 1$, and capacity $c = 2$. We denote the number of steps in the polynomial system by N and the number of rounds in the polynomial system by $R = 2N$.

Polynomial description. The input is denoted by $X_0 = (X_0[1], \ldots, X_0[r], 0, \ldots, 0)$, the state after the step i by

$$X_i = (X_i[1], \ldots, X_i[m]),$$

and the output by

$$Y = (H[1], \ldots, H[r], Y[r+1], \ldots, Y[m])$$

where $H[i]$ is the i^{th} output of the hash function. The MDS matrix is denoted by M and constants used at step i are denoted by

$$K_i = (K_i[1], \ldots, K_i[m]), K'_i = (K'_i[1], \ldots, K'_i[m]).$$

In the case of the XHash12, the i^{th} basic step is modeled as:

$$\sum_{k=0}^{m-1} M[j,k] \cdot X_i[k]^\alpha + K_i[j] - (X'_i)^\alpha - K'_i[j] = 0$$

In the case of the XHash8, the i^{th} basic step is modeled as:

$$\begin{aligned}
&\sum_{k=0}^{m-1} M[j,k] \cdot X_i[k]^\alpha + K_i[j] - (X'_i)^\alpha - K'_{i+1}[j] = 0 && k \not\equiv 1 \pmod 3 \\
&\sum_{k=0}^{m-1} M[j,k] \cdot X_i[k]^\alpha + K_i[j] - X'_i - K'_i[j] = 0 && k \equiv 1 \pmod 3.
\end{aligned}$$

The i^{th} extended step is modeled as:

$$\sum_{k=0}^{m-1} M^{-1}[j,k] \cdot (X_{i+1}[k] + K_{i+1}[k]) - \pi_{2,\beta} = 0 \qquad \beta = k \pmod 3$$

The results of the Gröbner basis algorithm for XHash12 with $m = 3, r = 1, p = 65519$ are described in Table 2.

Introducing partial layers in XHash8 results, as expected, in a smaller solving degree; interestingly, the actual running time actually increases. These experiments on toy parameters are described in Table 3.

The total complexity of computing the Gröbner basis in degrevlex order for the hash function with N steps, state size m, and rate r is:

$$\mathcal{C}_{gb} \geq \binom{\mathcal{V} + d_{sol}}{d_{so}}^2, \tag{4}$$

Table 2: Experimental results for finding a Gröbner basis in degrevlex order for XHash12

R	N	$\mathcal{V}$	solving degree	Macaulay bound	complexity	time	memory
1	1	3	7	19	13.81	0.0	0.15
1	2	6	12	37	28.36	72.38	1.13
2	3	9	21	55	47.54	26338	12.3

Table 3: Experimental results of finding Gröbner basis in the degrevlex order for XHash8

R	N	$\mathcal{V}$	solving degree	Macaulay bound	complexity	time	memory
1	1	3	8	19	14.73	0.0	0.15
1	2	6	12	37	28.36	61.33	1.12
2	3	9	17	55	43.15	74593	31.58

$$\mathcal{C}_{gb} \approx 2^{423.59}.$$

By taking a very conservative approximation and assuming the solving degree remains 17 after the third step, the complexity of **both** instances is lower bounded by

$$\mathcal{C}_{gb} \geq 2^{136.87}.$$

C RPX Standard Specification

The XHash12 function defined in Section 4.1 has been implemented in optimized form by Polygon Miden. We provide here a canonical specification of the function implemented in [Mid23] which we refer to as RPX.

MDS Matrix. RPX uses the same MDS matrix found by the RPO project. That is, a circulant matrix whose first row is

$$[7, 23, 8, 26, 13, 10, 9, 7, 6, 22, 21, 8].$$

Round Constants. To ensure "nothing-up-my-sleeve" round constants we reuse the ones drived for RPO. Recall that the RPO constants were derived by employing the following procedure:

- Start from the string `RPO(%i,%i,%i,%i)`;
- Populate the wildcards "`%i`" with the ASCII decimal expansion of the integer parameters p, m, c, λ, in that order;
- Use SHAKE256 to expand this ASCII string into $9 \cdot 2 \cdot N \cdot m$ pseudorandom bytes;
- For every chunk of 9 bytes, compute the matching integer by interpreting the byte array as the integer's base-256 expansion with the least significant digit first;
- Reduce the obtained integer modulo p;
- Collect all such integers. The list of obtained field elements constitutes the list of round constants.

Primitive. The permutation used in RPX is depicted in Figure 3. It is different from the one depicted in Figure 1 in that the injection of the first set of round constants in even rounds is swapped with the MDS matrix. Since multiplication is distributive with respect to addition, injecting C before applying the MDS, or $M(C)$ after applying it is equivalent in terms of cryptanalytic resistance and therefore, all security arguments for XHash12 carry over to RPX.

Hashing Mode. To hash a sequence of field elements in $p = 2^{64} - 2^{32} + 1$ we instantiate the sponge construction with the RPX permutation. The state consists of 12 field elements, of which eight are designated as rate, and the remaining four are designated as capacity. Absorbing is done in *overwrite mode* (i.e., the topmost eight elements of the state are overwritten by new values every time the permutation is invoked) and the squeezing phase outputs eight field elements, of which the first four are returned as output and the rest are discarded.

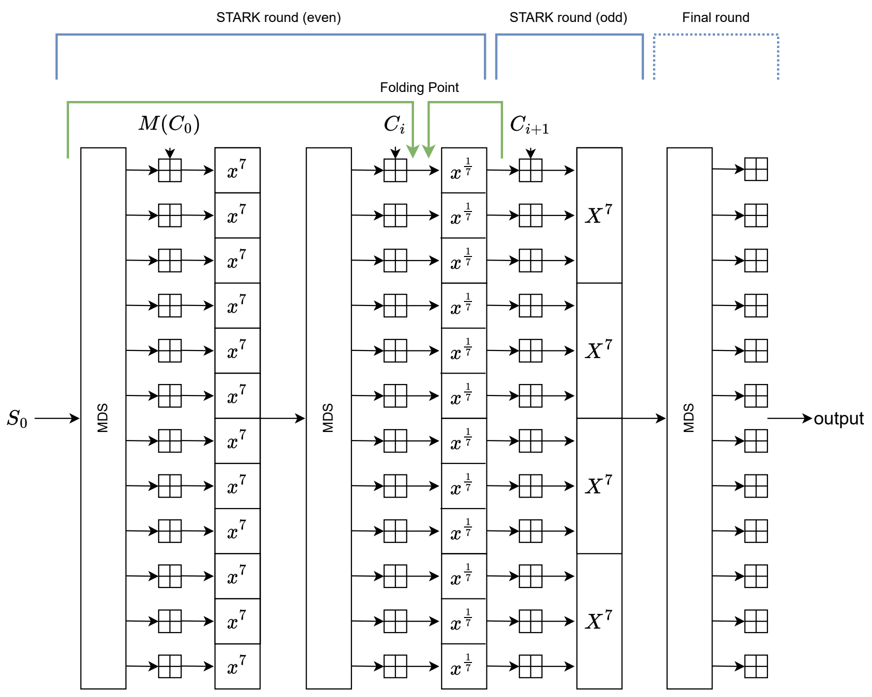

Figure 3: One round of the XHash8 permutation

Padding. For hashing field elements we use the zero-padding scheme; that is, if the length of the last block is smaller than $r = 8$ field elements, a sufficient amount of [0] elements are appended to complete it. We note that this padding rule alone is not sponge compliant. To avoid bijectivity problems we partition the input space into eight input domains: all messages whose last block is of length 8 are designated to the 0-domain; all messages whose last block is of length 7 are designated to the 1-domain, etc.

Domain Separation. We enforce domain separation by fixing the topmost inner part element (i.e., the ninth state element) to the domain identifier. Designating one capacity element to encode the domain identifier effectively ensures that two messages whose last blocks differ in length will be processed differently. We provide our proof intuition for this result in the following paragraphs.

Proof Intuition. We first note that RPX's security (such as indifferentiability, collision resistence, preimage resistance, etc.) remains same even when the padded zeros are pushed to the start i.e., when padding type is changed from being a post-padding to a pre-padding. This holds because both paddings are bijective to each other which means that the prepadded RPX can be seen as RPX with differently ordered a.k.a. permuted input set yet with the exact same output multiset and hence the same output distribution. Therefore, it is sufficient to argue the security of prepadded RPX.

We highlight that prepadded RPX processes messages in the exact same way as prepadded overwrite sponge except that the valid input space for the first permutation call is now larger due to the domain separation. We also note that this increase can at most double the input space for any number of domain separations. More specifically, the i^{th} domain can only increase the input space by p^i many elements and there can be at most r many domains which makes the final input space size of the first permutation $= p + p^2 + \ldots + p^r \leq 2p^r$.

One can notice that this change affects the pre-image resistance of prepadded RPX by one bit. More concretely, for q many permutation calls, the pre-image bound of prepadded overwirte Sponge is $p^r(q/p^b)$ with p^r representing the possible number of valid inputs for the first permutation call. For prepadded RPX, this reduces to $2p^r(q/p^b)$. On the other hand, this increased input space has no effect on the probability of finding collisions as they happen in the permutation outputs and remains $p^{-c/2}$.

Indeed RPX's preimage resistance is reduced now from $c \log_2 p$-bit to $c \log_2 p - 1$; however, noting that $c \log_2 p = 2 \cdot \kappa = 256$ this does not form a bottleneck and the security is still compatible with overwrite sponge bounds found in the literature (e.g., [BDPVA08]); i.e., at least 128-bit security.

D Test Vectors

The following test vectors are generated using the reference implementation [Mid23].

```
[0 0 0 0 0 0 0 0]  ->
[15293807115397414812 15290017247514670316 10548590320248089637  9459855167724924903]

[0 1 0 0 0 0 0 0]  ->
[12186327779210739392 12437198001472812457 17431583359007807548  5889070798901825636]

[0 1 2 0 0 0 0 0]  ->
[10984154334898375 5 17705465395673162594  5228101643025463311  7748133072458912307]

[0 1 2 3 0 0 0 0]  ->
[12729520246190904536  6715713369175329478 13802021724186903884 16589532398625893763]

[0 1 2 3 4 0 0 0]  ->
[3191491209909564984  4336372174992679659  3812090377223784023 16173224027531585338]

[0 1 2 3 4 5 0 0]  ->
[6461289079179018348 10449674711255412289  5054891760098348434 10721040246835958771]

[0 1 2 3 4 5 6 0]  ->
[16191592956183275197   7465323344470807 22 15358793909583453268  9513601171909830185]

[0 1 2 3 4 5 6 7]  ->
[12373829276206882697 10138650388065685463 15520480835694974951  2510219987660336228]

[0 1 2 3 4 5 6 7 8 0 0 0 0 0 0 0]  ->
[14898769958092295192 14076282783168040015  8476014900264177995 17336863755113979084]

[0 1 2 3 4 5 6 7 8 9 0 0 0 0 0 0]  ->
[17237194195242105781  6087397938124003113  1345882193144969073 14783461183116020251]

[0  1  2  3  4  5  6  7  8  9 10  0  0  0  0  0]  ->
[ 4575950952442466526 10298089839422454303 14861479923204285799 11880231458488351907]

[0  1  2  3  4  5  6  7  8  9 10 11  0  0  0  0]  ->
[ 9169920211008402116 12659190867532264163 13563500138844524911 12617975739035351823]

[0  1  2  3  4  5  6  7  8  9 10 11 12  0  0  0]  ->
[17454638445264588716  8802637143045803178 13982112504343449988 17442048147529824646]

[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13  0  0]  ->
[11373557723159380221 17180935309137919099  3242047510064238430 12672923945735822946]

[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14  0]  ->
[6214573915685641755 17951587517596484461 11692428935571224516  6628032869761165814]

[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15]  ->
[586102497461023489 11384107678327501002 10422108750253329853  7699259539482247907]

[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16  0  0  0  0  0  0  0]  ->
[16846145822493683059  6007639340046859794 13049520400071115122  5060263239960030371]
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17  0  0  0  0  0  0]  ->
[6509160877964314093 12642155348170163940  7507001761825557252  4565405860198708542]

[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18  0  0  0  0  0]  ->
[17905682982576162590  5720278714894771907  9596600499219832172     5974292660959196]
```

References

[AAB+20] Abdelrahaman Aly, Tomer Ashur, Eli Ben-Sasson, Siemen Dhooghe, and Alan Szepieniec. Design of symmetric-key primitives for advanced cryptographic protocols. *IACR Trans. Symmetric Cryptol.*, 2020(3):1–45, 2020.

[ABM23a] Tomer Ashur, Thomas Buschman, and Mohammad Mahzoun. Algebraic cryptanalysis of poseidon. Cryptology ePrint Archive, Paper 2023/537, 2023. https://eprint.iacr.org/2023/537.

[ABM23b] Tomer Ashur, Thomas Buschman, and Mohammad Mahzoun. Algebraic cryptanalysis of poseidon. Technical report, 2023. Under submission.

[AKM+22] Tomer Ashur, Al Kindi, Willi Meier, Alan Szepieniec, and Bobbin Threadbare. Rescue-prime optimized. *IACR Cryptol. ePrint Arch.*, page 1577, 2022.

[AMT22a] Tomer Ashur, Mohammad Mahzoun, and Dilara Toprakhisar. Chaghri - a fhe-friendly block cipher. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, page 139–150, New York, NY, USA, 2022. Association for Computing Machinery.

[AMT22b] Tomer Ashur, Mohammad Mahzoun, and Dilara Toprakhisar. Chaghri - A fhe-friendly block cipher. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, pages 139–150. ACM, 2022.

[ARS+15] Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for mpc and fhe. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, pages 430–454, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[BBC+22] Clémence Bouvier, Pierre Briaud, Pyrros Chaidos, Léo Perrin, Robin Salen, Vesselin Velichkov, and Danny Willems. New design techniques for efficient arithmetization-oriented hash functions:anemoi permutations and jive compression mode. Cryptology ePrint Archive, Paper 2022/840, 2022. https://eprint.iacr.org/2022/840.

[BCD+20] Tim Beyne, Anne Canteaut, Itai Dinur, Maria Eichlseder, Gregor Leander, Gaëtan Leurent, María Naya-Plasencia, Léo Perrin, Yu Sasaki, Yosuke Todo, and Friedrich Wiemer. Out of oddity – new cryptanalytic techniques against symmetric primitives optimized for integrity proof systems. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020*, pages 299–328, Cham, 2020. Springer International Publishing.

[BDPVA08] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the indifferentiability of the sponge construction. In Nigel Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, pages 181–197, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[BSBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Paper 2018/046, 2018. https://eprint.iacr.org/2018/046.

[CCF+18] Anne Canteaut, Sergiu Carpov, Caroline Fontaine, Tancrède Lepoint, María Naya-Plasencia, Pascal Paillier, and Renaud Sirdey. Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression. *Journal of Cryptology*, 31(3):885–916, Jul 2018.

[CHMS22] Orel Cosseron, Clément Hoffmann, Pierrick Méaux, and François-Xavier Standaert. Towards case-optimized hybrid homomorphic encryption - featuring the elisabeth stream cipher. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part III*, volume 13793 of *Lecture Notes in Computer Science*, pages 32–67. Springer, 2022.

[CIR22] Carlos Cid, John Petter Indrøy, and Håvard Raddum. Fasta – a stream cipher for fast fhe evaluation. In Steven D. Galbraith, editor, *Topics in Cryptology – CT-RSA 2022*, pages 451–483, Cham, 2022. Springer International Publishing.

[DEG+18] Christoph Dobraunig, Maria Eichlseder, Lorenzo Grassi, Virginie Lallemand, Gregor Leander, Eik List, Florian Mendel, and Christian Rechberger. Rasta: A cipher with low anddepth and few ands per bit. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018*, pages 662–692, Cham, 2018. Springer International Publishing.

[DGH+21] Christoph Dobraunig, Lorenzo Grassi, Lukas Helminger, Christian Rechberger, Markus Schofnegger, and Roman Walch. Pasta: A case for hybrid homomorphic encryption. Cryptology ePrint Archive, Paper 2021/731, 2021. https://eprint.iacr.org/2021/731.

[DR02a] Joan Daemen and Vincent Rijmen. *The design of Rijndael: AES — the Advanced Encryption Standard*. Springer-Verlag, 2002.

[DR02b] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.

[GHR+22] Lorenzo Grassi, Yonglin Hao, Christian Rechberger, Markus Schofnegger, Roman Walch, and Qingju Wang. Horst meets fluid-spn: Griffin for zero-knowledge applications. Cryptology ePrint Archive, Paper 2022/403, 2022. https://eprint.iacr.org/2022/403.

[GKK+19] Lorenzo Grassi, Daniel Kales, Dmitry Khovratovich, Arnab Roy, Christian Rechberger, and Markus Schofnegger. Starkad and poseidon: New hash functions for zero knowledge proof systems. *IACR Cryptol. ePrint Arch.*, page 458, 2019.

[GKR+21] Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger. Poseidon: A new hash function for zero-knowledge proof systems. In Michael Bailey and Rachel Greenstadt, editors, *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, pages 519–535. USENIX Association, 2021.

[HKL+22] Jincheol Ha, Seongkwang Kim, Byeonghak Lee, Jooyoung Lee, and Mincheol Son. Rubato: Noisy ciphers for approximate homomorphic encryption. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022*, pages 581–610, Cham, 2022. Springer International Publishing.

[HL20] Phil Hebborn and Gregor Leander. Dasta - alternative linear layer for rasta. *IACR Trans. Symmetric Cryptol.*, 2020(3):46–86, 2020.

[KR21a] Nathan Keller and Asaf Rosemarin. Mind the middle layer: The hades design strategy revisited. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021*, pages 35–63, Cham, 2021. Springer International Publishing.

[KR21b] Nathan Keller and Asaf Rosemarin. Mind the middle layer: The HADES design strategy revisited. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and*

Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part II, volume 12697 of *Lecture Notes in Computer Science*, pages 35–63. Springer, 2021.

[Mid23] Polygon Miden. https://github.com/0xPolygonMiden/crypto/blob/next/benches/README.md, 2023.

[MJSC16] Pierrick Méaux, Anthony Journault, François-Xavier Standaert, and Claude Carlet. Towards stream ciphers for efficient fhe with low-noise ciphertexts. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016*, pages 311–343, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[SAD20] Alan Szepieniec, Tomer Ashur, and Siemen Dhooghe. Rescue-prime: a standard specification (sok). Cryptology ePrint Archive, Paper 2020/1143, 2020. https://eprint.iacr.org/2020/1143.

[Sau21a] Jan Ferdinand Sauer. Gröbner basis-attacking a tiny sponge. Technical report, AS Discrete Mathematics, 2021. https://asdm.gmbh/2021/06/28/gb_experiment_summary/.

[Sau21b] Jan Ferdinand Sauer. Towards lower bounds: "involvement" of a polynomial system and explained gröbner bases. Technical report, AS Discrete Mathematics, 2021. https://asdm.gmbh/2021/05/28/involvement/.

[SLST23] Alan Szepieniec, Alexander Lemmens, Jan Ferdinand Sauer, and Bobbin Threadbare. The tip5 hash function for recursive starks. Cryptology ePrint Archive, Paper 2023/107, 2023. https://eprint.iacr.org/2023/107.