XHash8 and XHash12: Efficient STARK-friendly Hash Functions

Tomer Ashur
Polygon Research
Cryptomeria, Leuven, Belgium
tomer@cryptomeria.tech

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

Mohammad Mahzoun
Eindhoven University of Technology
m.mahzoun@tue.nl

Abstract

Zero-Knowledge proof systems are widely used as building blocks of different protocols, e.g., such as those supporting blockchains. A core element in Zero-Knowledge proof systems is the underlying PRF, usually modeled as a hash function that needs to be efficient over finite fields of prime order. Such hash functions are part of a newly developed paradigm known as Arithmetization-Oriented designs.

In this paper, we propose two new AO hash functions, XHash8 and XHash12 which are designed based on improving the bottlenecks in RPO [ePrint 2022/1577]. Based on our experiments, XHash8 performs ≈ 2.75 times faster than RPO, and XHash12 performs ≈ 2 times faster than RPO, while at the same time inheriting the security and robustness of the battle-tested Marvellous design strategy.

1 Introduction

Zero-Knowledge (ZK) proof systems are advanced cryptographic protocols used to prove to a party (a verifier) that some statement is true without revealing any private information about that statement. Such a setting is used in many applications such as blockchains, or banking systems. A pioneer ZK proof system is ZK-STARK [BSBHR18] which is a scalable proof system used in Layer 2 projects such as Polygon and Starknet.

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 hash functions balancing between the CPU and STARK running times. We start by analyzing the remaining bottlenecks in RPO and improve the efficiency by introducing a new operation: multiplication over an extension field. Multiplication over an extension field is quite common in the design of traditional symmetric-key algorithms such as AES [DR02a]. In the context of AO ciphers this approach was carried over to Vision [AAB+20] and Chaghri [AMT22] as well as the now defunct Starkad [GKK+19].

We adapt the same technique to prime order fields to achieve diffusion and algebraic complexity at the same time at a reduced cost. Using multiplication over the extension field is an efficient way that results in better diffusion of desired properties, such as high polynomial degree, and therefore improves the efficiency of the design. We propose two candidates:

1. XHash12: interleaving between Rescue rounds and a new type of round;
2. XHash8: a more aggressively optimized variant of XHash 12 which does not apply the expensive $x^{1/\alpha}$ S-box to all elements.

Related work. There are multiple hash functions designed for ZK-proof systems. Posedion is an efficient hash function that is widely used. The partial layers in the design of Posedion were subject to multiple attacks [KR20, BCD+20] subsequently leading to updating the parameters. Recently, in [ABM23a] the authors analyzed the security of Posedion 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 the security proof of Posedion. Recently, a new version of Posedion, called Poseidon2 [GKS23] was published to improve the efficiency of Posedion but without correcting the flaws found in the security arguments.

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 script. For example, the vector S of size n is denoted by $S = (S[0], \ldots, S[n-1])$ and the elements of a matrix M with dimensions $n \times m$ are denoted by $M[i][j]$ where $0 \leq i \leq n, 0 \leq j \leq m$. $\boxplus$ is used to denote addition over the finite field $\mathbb{F}_p$.

2.2 Rescue-Prime Optimized

Rescue-Prime Optimized (RPO) hash 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.
- a (B)-step is almost the same, only that π_1 is applied. 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 constant injection is moved to the beginning for reasons explained in [AAB+20, Sec. 4.3].

3 XHash8 and XHash12 — Design Rationale

The two main operations used in the Marvellous design strategy are S-boxes and linear layers. S-boxes are non-linear maps providing confusion. The linear layer, usually an MDS matrix multiplication, provides diffusion. Together, S-boxes, MDS multiplication, and the key injection result in a complicated polynomial representation with a high degree and high density. To reduce the cost of achieving this kind of description, we propose to use power maps over an extension field. Multiplication 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.

A natural question that arises is whether the efficiency can be improved any further. 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 which 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 and not only in STARKs. Goal (ii) is irrelevant to our use case since we expect honest users only to compute the primitive "forward", never requiring efficient "backward" evaluation. Thus 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, supposing a (B')-step that can achieve, hypothetically speaking, a maximal degree in four rounds; the overall performance can be improved without affecting the security. Getting ahead of ourselves, this is exactly what we will be doing.

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 with the goal of ensuring 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. Intuitively, 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". Generally speaking, the operation we are looking for should combine the high-degree element with the other elements 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 above 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 that are applied to all elements of the state. On the other hand, XHash8 aims to improve the efficiency and uses 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 uses a full layer of π_1 S-boxes, and works as follows:

- An initial (I)-step that consists of constant addition and MDS operation.
- An standard (B)-step where π_1 applies to all elements of the state followed by constant injection without MDS operation.
- A (P3)-step starts by restructuring the 12-element state 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 $s_{i,j,k} \in F_{p^3}$ followed by an application of π_2 to each of these extension field elements, such that

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

  At this point, the state is decomposed back into a 12-element vector in F_p, an MDS matrix is applied, and the step is concluded with constant injection.

The state consists of 12 field elements in F_p where $p = 2^{64} - 2^{32} + 1$. The permutation consists of six steps (three rounds) as follows:

$$(I)(F)(B)(P3)(F)(B)(P3)(F)(B)(P3),$$

as 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:

- An initial (I)-step that consists of constant addition and MDS operation.
- A (B')-step is an adaption of the standard (B)-step where π_1 is applied to 8 out of the 12 state elements, followed by constant injection, but without applying an MDS matrix. Concretely,

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

  and this is followed by the constant injection operation;

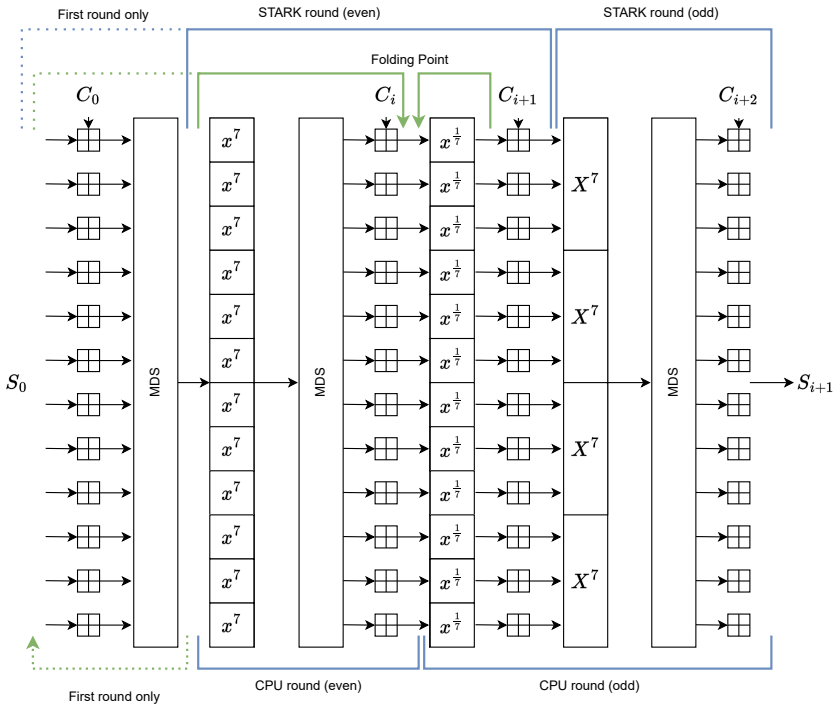

Figure 1: One round of the XHash12 permutation

- A (P3)-step starts by restructuring the 12-element state 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 $s_{i,j,k} \in F_{p^3}$ followed by an application of π_2 to each of these extension field elements, such that

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

  At this point, the state is decomposed back into a 12-element vector in F_p, an MDS matrix is applied, and the step is concluded with constant injection.

We are now ready to introduce the full specification of our new permutation. As before, the state consists of 12 field elements in F_p where $p = 2^{64} - 2^{32} + 1$. The permutation consists of six steps (three rounds) as follows:

$$(I)(F)(B')(P3)(F)(B')(P3)(F)(B')(P3),$$

as depicted in Figure 2.

A hash function offering 128-bit security is obtained by using this permutation 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 RPO by designating certain values of a capacity element to encode the domain.

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. The resistance of XHash8 against differential cryptanalysis attacks also ensures the resistance of XHash12 since any trail of

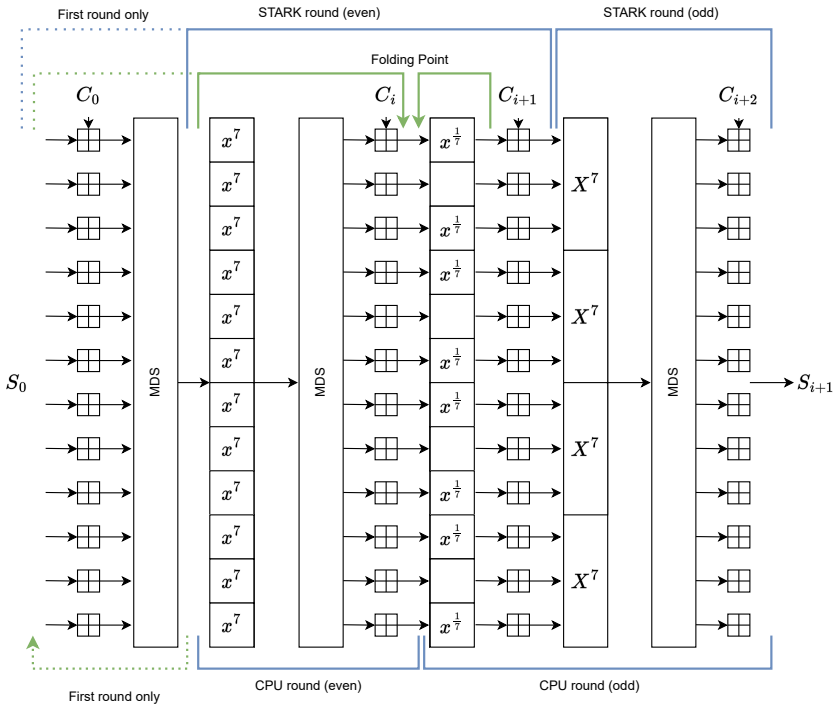

Figure 2: One round of the XHash8 permutation

XHash12 is also a trail of XHash8 with equal or lower probability. Analyzing the resistance of XHash8 against differential cryptanalysis follows the standard argument: we find a lower bound on the number of active S-box and an upper bound on the probability of the best differential transition; 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 (B') and (P3) steps are not separated by an MDS matrix;
3. 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 Damean 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 (I) step the adversary controls only the outer part of the sponge and therefore they can only create a difference in $1 \leq d_{(I)} \leq r = 8$ field elements. Consequently, after the application of the MDS matrix in the (I)-step, $5 \leq d_{(F)} \leq 12$ 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 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 mapp defined over $\mathbb{F}_q$, then F is differentially $(\gamma - 1)$-uniform.*

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

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 $\gamma - 1$. Using the fundamental theorem of algebra, the number of roots of Equation (1) is upper-bounded by $\gamma - 1$. Therefore, $|\mathcal{D}| \leq \gamma - 1$. □

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].

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 in fact sufficient to capture the quality we are interested in when designing a symmetric-key primitive. As an instructive example, consider the

[2] See [KR21] for the binary case and [ABM23b] for the prime case.

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:

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

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

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

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 with N and the number of rounds in the polynomial system with $R = 2N$.

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

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

and the output with

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

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:

$$\sum_{k=0}^{m-1} M[j,k] \cdot X_i[k]^\alpha - X_{i+1}[k]^\alpha - K_i[k] = 0.$$

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

$$\sum_{k=0}^{m-1} M[j,k] \cdot X_i[k]^\alpha - X_{i+1}[k]^\alpha - K_i[k] = 0 \qquad k \not\equiv 1 \pmod 3$$

$$\sum_{k=0}^{m-1} M[j,k] \cdot X_i[k]^\alpha - X_{i+1}[k] - K_i[j] = 0 \qquad k \equiv 1 \pmod 3.$$

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

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

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 also increases. These experiment 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 result of computing the Gröbner basis in the 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 result of computing the 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

where $\mathcal{V} = mN$ is the number of variables in the system, and $d_{sol} \geq mN$ is the solving degree. The complexity for the proposed instance with full layer is:

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

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

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

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.

[AMT22] 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.

[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. Cryptology ePrint Archive, Paper 2020/188, 2020. https://eprint.iacr.org/2020/188.

[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.

[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.

[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.

[GKS23] Lorenzo Grassi, Dmitry Khovratovich, and Markus Schofnegger. Poseidon2: A faster version of the poseidon hash function. Cryptology ePrint Archive, Paper 2023/323, 2023. https://eprint.iacr.org/2023/323.

[KR20] Nathan Keller and Asaf Rosemarin. Mind the middle layer: The hades design strategy revisited. Cryptology ePrint Archive, Paper 2020/179, 2020. https://eprint.iacr.org/2020/179.

[KR21] 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.

[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/.