Algebraic Cryptanalysis of the HADES Design Strategy: Application to Poseidon and Poseidon2

Tomer Ashur[1,2], Thomas Buschman[3], and Mohammad Mahzoun[4]

[1] 3MI Labs, Leuven, Belgium
[2] Polygon Research
tomer@cryptomeria.tech
[3] Eindhoven University of Technology
t.buschman@student.tue.nl
[4] Eindhoven University of Technology
mail@mahzoun.me

Abstract. Arithmetization-Oriented primitives are the building block of advanced cryptographic protocols such as Zero-Knowledge proof systems. One approach to designing such primitives is the HADES design strategy which aims to provide an efficient way to instantiate generalizing substitution-permutation networks to include partial S-box rounds. A notable instance of HADES, introduced by Grassi *et al.* at USENIX Security '21, is Poseidon. Because of its impressive efficiency and low arithmetic complexity, Poseidon is a popular choice among the designers of integrity-proof systems. An updated version of Poseidon, namely, Poseidon2 was published at AfricaCrypt '23 aiming to improve the efficiency of Poseidon by optimizing its linear operations. In this work, we show some caveats in the security argument of HADES against algebraic attacks and quantify the complexity of Gröbner basis attacks. We show that the complexity of the attack is lower than claimed with the direct implication that there are cases where the recommended number of rounds is insufficient for meeting the claimed security. Concretely, the complexity of a Gröbner basis attack for an instance of Poseidon with 1024 bits of security is 731.77 bits and the original security argument starts failing already at the 384-bit security level. Since the security of Poseidon2 is derived from the security of Poseidon, the same analysis applies to the instances of Poseidon2. The results were shared with the designers and the security arguments were updated accordingly.

Keywords: HADES · Poseidon · Hash functions · Zero-Knowledge proof systems · Gröbner basis attacks

1 Introduction

Arithmetization-Oriented (AO) primitives are a common building block for advanced cryptographic protocols such as Zero-Knowledge (ZK) proofs, Multiparty Computation (MPC), and Fully Homomorphic Encryption (FHE). AO primitives are usually defined over a finite field of large order and designed to have a simple and efficient algebraic representation. Examples of such primitives are *Rescue* [4], *Rescue-Prime* [41], *RPO* [6], and *Chaghri* [7] which are designed based on the Marvellous design strategy [4], *Griffin* [26] and *Anemoi* [14] which are Feistel-like designs, *Poseidon* [29] and *Poseidon2* [30] based on HADES design strategy, and more examples such as *MiMC* [2], *LowMC* [3], *Kreyvium* [15], *FLIP* [38], *Rasta* [21], *Dasta* [33], *Pasta* [22], *Fasta* [17], *Elisabeth* [18], *Rubato* [32], *Tip5* [42], and *XHash* [5] to name just a few.

A promising approach for designing efficient AO primitives is the HADES design strategy. Poseidon and its successor Poseidon2 are the most important and widely used hash functions designed based on the HADES approach which are sponge functions [10] instantiated by the Poseidon$^\pi$ and Poseidon2$^\pi$ permutations. Poseidon2, optimizes the linear layers of Poseidon to improve efficiency and mitigate the attack proposed in [9].

The HADES design strategy allows the designers to optimize the combination of substitution-permutation network (SPN) and Partial SPN [25] (PSPN) to derive an efficient design while not jeopardizing security. In HADES, SPN is referred to as full layers and used to justify arguments for the primitive's resistance against statistical attacks using the wide trail strategy. Then, PSPN is referred to as partial rounds with the goal to not only improve the performance of the design but also in combination with full layers ensure resistance against algebraic attack. Indeed, it is claimed that both full and partial layers provide *same* resistance in case of algebraic attacks [31].

The security of the HADES approach is based on an extensive analysis of various techniques such as:

- Statistical attacks: Differential cryptanalysis, linear cryptanalysis.
- Algebraic attacks: Interpolations attacks [34], Gröbner basis attacks [19], Higher-Order differential attacks [36], and Zero-Sum partitions attacks [13].

The security arguments of HADES and its instances were scrutinized by third-party cryptanalysts which presented security vulnerabilities exploiting the partial layers [12,35]. It has been observed that in some cases, the linear layer results in invariant subspaces in partial layers. Subsequently, the security arguments and suggested secure parameters were updated accordingly by imposing additional constraints on the choice of the linear layer in the partial layer. Later, [9] showed how to bypass two full rounds as an auxiliary approach for mounting algebraic attacks but no parameter sets were yet shown to be vulnerable. Sauer designed an algebraic attack on Poseidon [39] and showed that the resistance of Poseidon against Gröbner basis attacks is overstated. However, he did not provide any instance that is indeed vulnerable.

Our contributions. We investigate the feasibility of Gröbner basis attacks against Poseidon and Poseidon2 as the most important instances of HADES. Our approach is to conduct a thorough security analysis against the CICO problem in the context of Gröber basis attacks, which are considered to be among the most promising algebraic attacks against AO designs [1].[5]

[5] See also [29, Sec. 5.2]

To estimate the complexity of Gröbner basis attacks, we approximate the solving degree by computing the Gröbner basis for toy parameters and extrapolate the solving degree. Extrapolation of the solving degree is a common approach in the cryptography community to compute the Gröbner basis complexity [4,41,6,42,14,9]. The solving degree shows that the polynomial system describing Poseidon is not regular, and using the degree of regularity to bound the solving degree results in over-estimation of the provided security [39]. Moreover, in the design of HADES, it is believed that partial rounds provide the same security resistance against algebraic attacks as full rounds.[6] We demonstrate that in cases where the state size is larger than two, partial rounds offer less resistance against Gröbner basis attacks than full rounds. To show the impact of our observations and as a proof of concept, we demonstrate an instance with 1024 bits of security which is broken by our approach; the complete parameter set can be found in Table 1.

λ	$\log_2(p)$	α	t	r	R_F	R_P	$\mathcal{C}_{\text{GB}}$
1024	128	3	24	8	8	85	**712.98**

Table 1. an instance of Poseidon hash function with security parameter λ, state size t, rate r, and (R_F, R_P) the number of full and partial rounds, respectively. $\mathcal{C}_{\text{GB}}$ is the complexity of the Gröbner basis attack.

The proposed algebraic attacks suggest that the partial layers do not provide the expected level of security against algebraic attacks, requiring that the security argument be re-evaluated for instances following this design strategy.

Additionally, we conducted a more thorough investigation into Poseidon's security argument with respect to the claimed resistance against algebraic attacks. We revealed three distinct flaws in these arguments, each of which has implications for the required number of rounds. First, we show a typo in the security argument against Gröbner basis attacks in the full round setting. Then, we show that the logical reasoning for the security argument against the Gröbner basis attack is not sound. Finally, we present an error in the symbolic computation of bounds that undermines security.

Structure of the Paper. In Section 2, the notations used throughout the paper and the required background materials are described. In Section 3, an overview of Poseidon and Poseidon2 designs, their security arguments, and the flaws in the security arguments are outlined. In Section 4, a Gröbner basis attack is proposed, and vulnerable instances are demonstrated. In Section 4.4, we compare the complexity of computing the Gröbner basis and the running time of the algorithm and discuss that running time is usually faster than the suggested complexity. Finally, in Section 5, the paper is summarized, the steps

[6] See [31, Sec. 2: "*Crucial Points of the Hades Strategy*"] later retracted in Version 121947 of [27, Sec. 2.2: "*Interaction Between Full and Partial Rounds*"].

taken toward disclosure are outlined, and possible directions for future research are discussed.

2 Preliminaries

2.1 Notations and Definitions

In this paper, we define λ as the security parameter. To show an inclusive range of numbers, we use $[a, b] = \{a, \ldots, b\}$. Vectors are denoted by bold capital letters such as $\mathbf{X}, \mathbf{Y}, \mathbf{Z}, \ldots$ and the elements of the vector $\mathbf{X}$ are denoted by $(x_1, \ldots, x_n)$. Matrices are denoted by calligraphic capital letters such as $\mathcal{M}, \mathcal{N}$ where $\mathcal{M}_{i,j}$ is the j^{th} element in the i^{th} row.

Definition 1 (Macaulay Matrix [37]). *Let $\mathcal{P} \in K[x_1, \ldots, x_n]$ be a polynomial system with monomial ordering $\prec$, the Macaulay matrix $\mathcal{M}[d](\mathcal{P})$ of degree d is a matrix with coefficients in K, where $\mathcal{M}[d]_{i,j}$ is the coefficient of the j^{th} biggest monomial with respect to $\prec$ in the i^{th} polynomial in the extended system. For example, let $\mathcal{P} = \{P_1, P_2\} = \{x^2 + xy, y\}$. Then $\mathcal{M}[2](\mathcal{P})$ for degrevlex order is defined as:*

$$\mathcal{M}[2](\mathcal{P}) = \begin{array}{c} \begin{array}{cccccc} x^2 & xy & y^2 & x & y & 1 \end{array} \\ \left(\begin{array}{cccccc} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{array}\right) \end{array} \begin{array}{c} \\ P_1 \\ P_2 \\ xP_2 \\ yP_2 \end{array}.$$

Definition 2 (Linear Algebra Constant (ω) [43]). *In the rest of this paper, $2 < \omega \leq 2.3716$ is defined as the linear algebra constant and is the complexity of matrix multiplication.*

Definition 3 (CICO resistance). *A hash function $H : D_1 \times D_2 \rightarrow R_1 \times R_2$ is CICO resistant if it is computationally infeasible to find $x \in D_1$ and $y' \in R_2$ such that $H(\{x, x'\}) = \{y, y'\}$ for given $x' \in D_2, y \in R_1$.*

2.2 Multivariate polynomial systems

Let K be a field, the polynomial ring $K[x_1, \ldots, x_n]$ is a set of all polynomials in the variables $x_1, \ldots, x_n$ and coefficients in K. A polynomial system is a finite set of polynomials $P_1, \ldots, P_m \in K[x_1, \ldots, x_n]$ such that:

$$\begin{cases} P_1(x_1, \ldots, x_n) = 0 \\ P_2(x_1, \ldots, x_n) = 0 \\ \quad\quad \vdots \\ P_m(x_1, \ldots, x_n) = 0 \end{cases}$$

The polynomial systems typically describing AO hash functions span a zero-dimensional ideal, meaning that the set containing all their solutions is finite. One can use Gröbner basis to solve multivariate polynomial systems. The steps to solve a multivariate polynomial system using one of the Gröbner basis algorithms are:

1. Compute a Gröbner basis with respect to *degrevlex* term order.
2. Convert the Gröbner basis to *lex* term order.
3. Find the roots of the polynomial system by factoring univariate polynomials and extending the partial solutions.

The primary motivation for first computing the Gröbner basis in *degrevlex* order is its lower complexity compared to other term orderings. **Complexity of Step 1.** The complexity of computing a Gröbner basis in *degrevlex* term order is upper bounded by [11]:

$$O\left(\binom{n+d_{sol}}{d_{sol}}^{\omega}\right), \tag{1}$$

where n is the number of variables in the multivariate polynomial system and d_{sol} is the solving degree of the polynomial system [20]. Solving degree is defined as the highest degree of the polynomials involved in the computation of the Gröbner basis using the F5 algorithm. In the case of regular systems, the solving degree matches the Macaulay bound which is defined as:

$$d_{sol} = \sum_{i=1}^{m}(d_i - 1) + 1,$$

where d_i is the polynomial degree of P_i for $1 \leq i \leq m$.

However, most of the AO primitives, when modeled as a polynomial system, are not regular systems [4,39,9], and in most of the cases, the solving degree grows slower than the Macaulay bound.

To determine the solving degree, the current approach used in design and cryptanalysis is to compute the solving degrees of round-reduced versions of the system, and extrapolate a bound for it [4,1,9,39,14].

Complexity of Step 2. The computed Gröbner basis in *degrevlex* order is usually complicated and not useful for solving the system. Therefore, it is converted to a Gröbner basis in *lex* order. The conversion is performed using the FGLM [23] algorithm and the complexity is upper bounded by:

$$O\left(nD^3\right),$$

where D is the degree of the zero-dimensional ideal. In some cases, this step can be performed more efficiently using the sparse FGLM [24] algorithm which has asymptotic complexity of:

$$O(\sqrt{6/n\pi}D^{2+\frac{n-1}{n}}). \tag{2}$$

Complexity of Step 3. When the ideal is zero-dimensional (as is the case for us), the Gröbner basis in the lex order contains a unique univariate polynomial that can be factored and is used to iteratively solve the entire system. When the unique univariate polynomial is factored, it results in a partial solution to the system. In an iterative process, partial solutions are substituted in other polynomials, and these are factored in a similar way until a full solution is obtained. solving a univariate polynomial system of degree D defined over the finite field $\mathbb{F}_p$ the Cantor/Zassenhaus [16] algorithm can be used with complexity [40]:

$$O(D^2(\log D \log\log D)(\log p + \log D)).$$

2.3 The Sponge Construction

The Sponge construction [10] is a generic method for constructing a hash function from a fixed-length public permutation. Let $\mathbb{F}_p$ be a finite field of order p and $f : \mathbb{F}_p^n \to \mathbb{F}_p^n$ be a fixed-length transformation operating over a state of size n with elements in $\mathbb{F}_p$. The sponge function F with rate r and capacity c where $r + c = n$, takes as input $\mathbf{M}$ of arbitrary length, and after applying a padding function, generates the output $\mathbf{H}$. The length of the padded input and the output of F is a multiple of r. The sponge function works as follows:

1. Let $\mathbf{S}$ be the state of the sponge function of length $n = r + c$.
2. The state $\mathbf{S}$ is initialized to $(0, \dots, 0)$.
3. Absorbing phase: The padded message $\mathbf{M}$ is split into χ blocks $\mathbf{M}_1, \mathbf{M}_2, \dots, \mathbf{M}_\chi$ of length r. For each $i \in 1, 2, \dots, \chi$, the $\mathbf{M}_i$ is added to the first r blocks of $\mathbf{S}$ and the function f is applied *i.e.*,
$$\mathbf{S} = f(\mathbf{S} + \mathbf{M}_i || \mathbf{0})$$
4. Squeezing phase: once all blocks of the padded message have been absorbed, the squeezing phase starts to generate the output. In this phase, the function outputs blocks $\mathbf{H}_1, \dots, \mathbf{H}_{\chi'}$ of length r and update the internal state $\mathbf{S}$ by applying the function f.

In Figure 1, a schematic construction of the sponge function is illustrated. To study more about sponge construction in the context of ZK-friendly hash functions, we refer to [8].

Assuming that f is computationally indistinguishable from a random permutation, a sponge function with capacity of c elements offers $2^{\log_2(p)c/2}$ bits of collision resistance and preimage resistance [10].

2.4 The HADES Design Strategy

The HADES design strategy is a paradigm for developing efficient and secure AO primitives. HADES uses two types of SPN networks, known as *full layers*—placed at the beginning and at the end of the permutation—and *partial layers*, placed in the middle. Each full round of HADES works as follows:

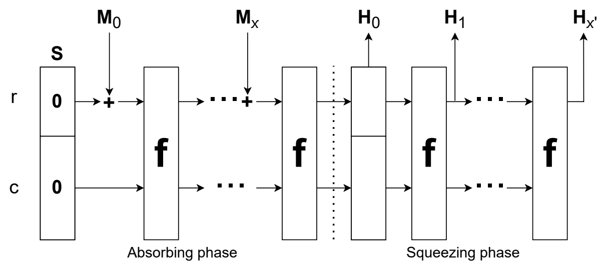

Fig. 1. A sponge function with rate r, capacity c, and internal permutation f.

1. Add round keys.
2. Substitution (non-linear) layer applied to all the elements in the state.
3. Permutation (linear) layer.

Each partial round of HADES works as follows:

1. Add round keys.
2. Substitution (non-linear) layer applied to specific elements, usually the first one, in the state.
3. Permutation (linear) layer.

In Figure 2 an overview of the HADES design strategy is depicted.

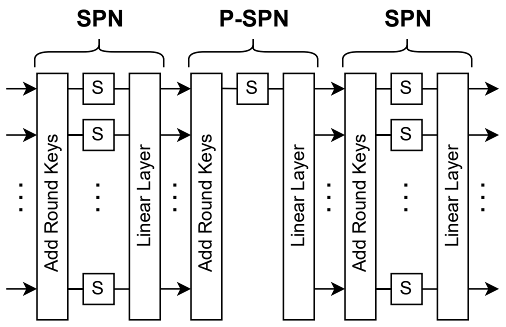

Fig. 2. The SPN layer at the beginning and the end are full layers. The PSPN in the middle is the partial layer where the S-box only applies to the first element of the state.

3 Poseidon and Poseidon2

Let us denote the set of vectors over the finite field $\mathbb{F}_p$ with arbitrary length with $\mathbb{F}_p^*$. Poseidon: $\mathbb{F}_p^* \rightarrow (\mathbb{F}_p^r)^{\chi'}$ is a hash function operating over $\mathbb{F}_p$ with output of χ'

blocks of length r. It is constructed by using the Poseidon$^\pi$ permutation in the sponge construction with rate r and capacity c. Poseidon$^\pi$ is a permutation with a state size of t and consists of $R = R_F + R_P$ rounds, where $R_F = R_f + R_f$ rounds are full rounds with t S-boxes, and R_P rounds are partial rounds with only one S-box applied to the first element of the state. The Poseidon$^\pi$ permutation is illustrated in Figure 3 and works as follows:

1. Add round constants: $ARC_{\mathbf{C}} : \mathbb{F}_p^t \to \mathbb{F}_p^t,\ ARC_{\mathbf{C}}(\mathbf{X}) = \mathbf{X} + \mathbf{C}$.
2. Substitution layer: $S_\alpha : \mathbb{F}_p \to \mathbb{F}_p,\ S_\alpha(x) = x^\alpha$, where the S-box is applied to the first element (in partial rounds), or all elements of the state (in full rounds).
3. Linear layer: $L_{\mathcal{M}} : \mathbb{F}_p^t \to \mathbb{F}_p^t,\ L_{\mathcal{M}}(\mathbf{X}) = \mathcal{M} \cdot \mathbf{X}^\intercal$ where $\mathcal{M}$ is a MDS matrix.

Where $\mathcal{M}$ is a Cauchy matrix [44] and is defined as follows:

$$\mathcal{M}_{i,j} = \frac{1}{x_i + y_j},$$

for pairwise distinct x_i and y_j with the condition that $x_i + y_j \neq 0$ for all $i, j \in [1, t]$.

To improve the efficiency of Poseidon, the authors designed Poseidon2. Poseidon2 uses different partial rounds and different linear layers. Posedion2 works as follows:

1. Initial linear layer: $L_{\mathcal{M}'} : \mathbb{F}_p^t \to \mathbb{F}_p^t,\ L_{\mathcal{M}'}(\mathbf{X}) = \mathcal{M}' \cdot \mathbf{X}^\intercal$ where $\mathcal{M}'$ is an MDS matrix.
2. For R rounds:
   (a) Add round constants: $ARC_{\mathbf{C}} : \mathbb{F}_p^t \to \mathbb{F}_p^t$
   $$ARC_{\mathbf{C}}(\mathbf{X}) = \mathbf{X} + \mathbf{C}.$$
   In the case of partial rounds, $\mathbf{C} = (c_1, 0, \ldots, 0)$.
   (b) Substitution layer: $S_\alpha : \mathbb{F}_p \to \mathbb{F}_p,\ S_\alpha(x) = x^\alpha$, where the S-box is applied to the first element (in partial rounds), or all elements of the state (in full rounds).
   (c) Linear layer: $L_{\mathcal{M}} : \mathbb{F}_p \to \mathbb{F}_p,\ L_{\mathcal{M}}(\mathbf{X}) = \mathcal{M} \cdot \mathbf{X}^\intercal$.

Where $\mathcal{M}$ is an MDS matrix. In case of full rounds, $\mathcal{M} = \mathcal{M}'$ and is same as the MDS matrices defined for Griffin-π [26]. In case of partial rounds, $\mathcal{M} = \mathcal{M}''$ is defined in [30, Section 5.2]. Figure 3 depicts how Poseidon and Poseidon2 work and the updated operations in Poseidon2 are denoted by dashed lines.

Instances of Poseidon and Poseidon2 that provide λ bits of security, guarantee that any algorithm that finds collision or preimage requires a complexity of at least 2^λ. In the case of the CICO problem, as long as finding $x \| x' \in D$ using exhaustive search is not possible with complexity less than 2^λ, any algorithm that finds such $x \| x' \in D$ requires a complexity of at least 2^λ.

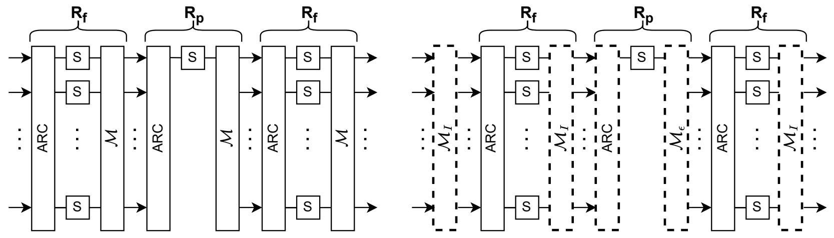

Fig. 3. Construction of Poseidon$^\pi$ (bottom), and Poseidon2$^\pi$ (top) permutations. The modified steps are shown with dashed line

3.1 Security Claims for Poseidon and Poseidon2

The security of Poseidon is analyzed against various attacks in order to compute the reliable number of required rounds. The authors provided a script to facilitate the computation of the number of rounds[7]. In our analysis, we utilized this script to calculate the necessary number of rounds required to ensure the security of our chosen parameters. Currently, the scripts are updated after communicating the results with the authors.

3.2 Flaws in the Security Analysis of Poseidon

In addition to the flaw described in Section 4, we identified three more minor flaws which, when combined, increase the likelihood of an attack to exist. In Section 3.2, we demonstrate that using loose bounds in security arguments leads to incorrect conclusions. In Section 3.2, we investigate the security argument against Gröbner basis attacks in the case of $\chi = 1$, where the system is already a Gröbner basis in the full-permutation setting. We highlight a typo causing an underestimation of the required number of rounds. Finally, in Section 3.2, we identify a flaw in the symbolic computations of round-level Gröbner basis analysis that led to an overclaiming of the security.

Improper Logic The argument for determining the number of rounds that is safe against a Gröbner basis attack can be summarized as follows [29, Section 5.5.2]:

1. Compute the complexity of the attack as a function of the Poseidon parameters $\alpha, R_F, R_P, t, r, \chi, \lambda$.
2. Optionally, derive an upper bound for the computed complexity that is easier to manipulate.
3. Calculate the maximum number of rounds R_F^* and R_P^* that can be attacked given the parameters of Poseidon.

[7] https://extgit.iaik.tugraz.at/krypto/hadeshash

4. Assume that all values for R_F, R_P higher than R_F^*, R_P^* cannot be attacked and are secure.

The problem arises due to Step 2, where a lower bound should be used. As a result, Step 4 concludes resistance against adversaries that Step 3 did not handle.

Consider a simple example: let us assume that a sponge construction with rate r uses an N-round permutation; further, assume an attack with complexity 2^{3Nr}. However, this expression may be challenging to work with (*e.g.*, because 3 is odd and we wish to take a square root) so we attempt to simplify it by noting that $2^{3Nr} \leq 2^{4Nr}$, although this is not a tight upper bound. Using the argumentation shown above, we find N^* from:

$$2^{4N^*r} = 2^{\lambda}.$$

Solving for N^* yields

$$N^* = \frac{\lambda}{4r}.$$

Consequently, for all $0 \leq N \leq N^*$, a sponge function using the N-round permutation can be attacked. This is still a true statement. Using the above argumentation, it is then conjectured that the sponge function is safe from attacks for all $N \geq N^* = \frac{\lambda}{4r}$. This is not a true statement as now using the proper expression to find a safe number of rounds N_s, we obtain

$$2^{3N_s r} = 2^{\lambda},$$

and find:

$$N_s = \frac{\lambda}{3r}.$$

Therefore, for all $0 \leq N \leq N_s$, the sponge function can be attacked, and for all $N > N_s$, the sponge function is safe for the given security level. The problem is that for $N^* \leq N \leq N_s$, we argued that the sponge construction with N rounds is safe, while it is not the case. When using Step 3 and Step 4 outlined earlier, one should use a lower bound rather than an upper bound in Step 2, as it may result in an overestimation of the resistance of the sponge function against attacks.

Similarly, the resistance of Poseidon against a round-level Gröbner basis attack is found to be (up to reasonable approximation) [29]:

$$\mathcal{C}_{\text{GB}} = 2^{Cq - C'},$$

with

$$C = 2\log_2\left(\frac{\alpha^{\alpha}}{(\alpha-1)^{\alpha-1}}\right)$$
$$C' = \log_2\left(\frac{2\pi(\alpha-1)q}{\alpha}\right)$$
$$q = (t-1)R_F + R_P + \chi$$

That concludes Step 1. In Step 2, this approximation was upper-bounded by:

$$\mathcal{C}_{\mathrm{GB}} = 2^{C\cdot q - C'} \leq 2^{C\cdot q}, \tag{3}$$

Ultimately, resistance against the round-level attack is assumed as long as:

$$(t-1)R_F + R_P \geq C^{-1}\min\{\lambda, \log_2(p)\} - 1, \tag{4}$$

Since (3) is not a tight bound (4) necessarily underestimates the required number of round. The effect of this omission is more noticeable when α, state size t, and rate r grow.

Transcription Error Full-Permutation Equation. In the full round equation setting [29, Section C.2.2], a system of equations for the entire R rounds is derived by considering each input as a variable and applying the round functions to them. When the number of input variables χ is the same as the number of output variables, the resulting system will consist of χ equations in χ variables, and the degree of each polynomial is upper-bounded by $D_\alpha(R) = \alpha^R$.

When $\chi = 1$, the system consists of a single polynomial of degree at most α^R in one variable, which is already a Gröbner basis in lex order. Therefore, the only step required to complete the attack is the factorization of the univariate polynomial. Per the security argument provided in [29, Section C.2.2], one should have:

$$\log_2\left(\alpha^{\omega R}\right) \geq \log_2\left(\alpha^{2R}\right) \geq \min\{\lambda, \log_2(p)\},$$

which implies:

$$R \geq \left\lceil \frac{\min\{\lambda, \log_2(p)\}}{2\log_2\alpha} \right\rceil = \log_\alpha(2)\cdot\min\{\frac{\lambda}{\mathbf{2}}, \frac{\log_2(p)}{2}\},$$

where $R = R_F + R_P$. Later, the designers in [29, Equation 11], write the constraint for the full round attack as:

$$R_F + R_P \geq \log_\alpha(2)\cdot\min\{\frac{\lambda}{\mathbf{3}}, \frac{\log_2(p)}{2}\}, \tag{5}$$

where the denominator of the fraction $\frac{\lambda}{3}$ is 3 instead of 2. This mistake results in an overestimation of the security that the Poseidon permutation provides against Gröbner basis attacks in the case where $\chi = 1$.

As an example of how the mistake influences the number of rounds, the constraint in [29, Equation 5] would imply that 6 full rounds and 22 partial rounds are sufficient for $\alpha = 3, t = 2, p \approx 2^{1024}$, and a desired security level of 128 bits, whereas to gain that security level for these parameters, at least 35 partial rounds are required.

Symbolic Computation Error In [29, Section C.2.2] it is shown that for security level of λ, the maximum number of rounds that can be attacked using Gröbner basis is:

$$(t-1)R_F + R_P + \chi \leq C^{-1} \cdot \min\{\lambda, \log_2(p)\chi\}, \tag{6}$$

with

$$C = 2\log_2\left(\frac{\alpha^\alpha}{(\alpha-1)^{\alpha-1}}\right).$$

The designers argue that the maximal number of rounds that can be attacked is when $\chi = 1$ [29, Section C.2.2] but this is not true. Rewriting (6), we get

$$(t-1)R_F + R_P \leq C^{-1} \cdot \min\{\lambda - \chi C, \chi(\log_2(p) - C)\}.$$

Here, the first argument of the minimum function is indeed maximized for $\chi = 1$, but the last argument is maximized for $\chi = t-1$ because $\lambda - C$ is positive for the suggested parameters of Poseidon. Ultimately, security is conjectured if:

$$(t-1)R_F + R_P \geq C^{-1} \cdot \min\{\lambda, \log_2(p)\} + t - 2,$$

but if we address the algebra error, we obtain:

$$(t-1)R_F + R_P \geq C^{-1} \cdot \min\{\lambda + C(t-2), \log_2(p)(t-1)\}.$$

Previously, the constraint for this kind of Gröbner basis attack appeared to be less restrictive than the other attacks, as it was subsumed by the constraints for the other kinds of Gröbner basis attacks [29, Equation 11]. However, once the error is addressed, this is no longer true. More importantly, there are parameter sets for which this constraint would require the highest number of partial rounds to be secure. For example, for $\alpha = 3, \log_2(p) \approx 256, \lambda = 1536, R_F = 8, t = 8$, an interpolation attack would be thwarted if $R_P \geq 158$, a subspace attack would fail if $R_P \geq 80$, and a full-permutation attack requires $R_P \geq 73$, *but* a round-level Gröbner basis attack require $R_P \geq 230$ to achieve required resistance. Therefore, [29, Equation. 5] requires three constraints rather than two and this omission does affect the required number of rounds for some parameter sets.

4 The Gröbner Basis Attack

The CICO resistance of Poseidon and Poseidon2 is analyzed using the Gröbner basis attacks in this section. To solve the CICO problem, we first model Poseidon and Poseidon2 as a system of multivariate polynomials with known output and unknown input and we solve the system to find the desired input.

4.1 Poseidon: Polynomial Modeling

Poseidon is modeled for the case where $\alpha = 3$, the number of input blocks of size r is $\chi = 1$ and the underlying permutation is applied only once. In Poseidon, which is a sponge function, the first r elements of the input state of the permutation are absorbed from the input, and the next c elements are initialized to a constant value. Without loss of generality, we can assume that the last c element of the input state is initialized to 0.

while it is possible to model Poseidon in various ways using algebraic relations describing them, the model that minimizes the complexity of the Gröbner basis attack is the preferred one.

After a thorough analysis of various methods for polynomial modeling, we identified the approach used by Sauer [39] that aims to minimize the solving degree of the system results in the lowest theoretical complexity.

In the described polynomial system, $\mathbf{C}_i = \{c_{i,1}, \ldots, c_{i,t}\}$ denotes the round constants for round $i \in \{1, \ldots, R\}$. $\mathbf{X}_i = \{x_{i,1}, \ldots, x_{i,t}\}$ are the variables that describe the state of the round $i \in \{0, \ldots, R\}$, where $\mathbf{X}_0 = (x_1, \ldots, x_r, 0, \ldots, 0)$ is the input and $\mathbf{X}_R = (H_1, \ldots, H_r, x_{R,r+1}, \ldots, x_{R,t})$ is the output. The first round of Poseidon before multiplication by $\mathcal{M}$ can be described as:

$$x_{1,j} - (x_{0,j} + c_{1,j})^{\alpha} = 0 \qquad\qquad j \in [1, r],$$

that has $2r$ variables and r polynomials. The state after the first and second S-box layers is modeled as follows:

$$x_{2,j} - \left(\left(\sum_{k=1}^{r} \mathcal{M}_{j,k} \cdot x_{1,k} + \sum_{k=r+1}^{t} \mathcal{M}_{j,k} \cdot c_{1,k}^{\alpha} \right) + c_{2,j} \right)^{\alpha} = 0,$$

where $j \in [1, t]$. The described polynomials add t new variables and t new polynomials to the system. The next R_f full rounds (i.e., $3 \leq i \leq R_f$) are modeled as:

$$x_{i,j} - \left(\left(\sum_{k=1}^{t} \mathcal{M}_{j,k} \cdot x_{i-1,k} \right) + c_{i,j} \right)^{\alpha} = 0 \qquad\qquad j \in [1, t],$$

adding $(R_f - 2)t$ new variables and $(R_f - 2)$ new polynomials to the system. We introduce a variable $\mathbf{Y}$ to simplify the equations for partial rounds and it is initialized as:

$$\mathbf{Y}^{\intercal} = \mathcal{M} \cdot (x_{R_f,1}, \ldots, x_{R_f,t})^{\intercal}.$$

The partial rounds $R_f < i \leq R_f + R_P$ are modeled as:

$$\begin{aligned} x_{i,1} - (y_1 + c_{i,1})^{\alpha} &= 0 \\ y_j &= \mathcal{M}_{j,1} \cdot x_{i,1} + \sum_{k=2}^{t} \mathcal{M}_{j,k} \cdot (y_k + c_{i,k}) \qquad\qquad j \in [1, t], \end{aligned}$$

adding R_P new variables and R_P new polynomials to the system. The last R_f rounds $R_f + R_p < i \leq R-1$ are modeled as:

$$\begin{aligned} x_{i,j} - (y_j + c_{i,j})^\alpha &= 0 && j \in [1,t] \\ y_j &= \sum_{k=1}^{t} \mathcal{M}_{j,k} \cdot x_{i,k} && j \in [1,t], \end{aligned}$$

that add $(R_f - 1)t$ variables in $(R_f - 1)t$ polynomials to the system. Finally, the last round is modeled as:

$$\sum_{k=1}^{t} \mathcal{M}^{-1}_{j,k} \cdot x_{R,k} - (y_j + c_{R,j})^\alpha = 0 \qquad j \in [1,t].$$

The last round adds c new variables and t polynomial to the system. The final system has $r + (R_F - 1)t + R_P$ polynomials of degree α in $r + (R_F - 1)t + R_P$ variables.

4.2 Poseidon2: Polynomial Modeling

Poseidon and Poseidon2 differ in the linear layer, constant addition layer for partial rounds, and the initial round. The first round can be modeled as follows:

$$x_{1,j} - \left(\sum_{k=1}^{t} \mathcal{M}''[j,k] \cdot x_{0,k} + c_{1,j} \right)^\alpha = 0 \qquad j \in [1,r],$$

The full rounds are modeled in the same way as in Section 4.1 with different coefficients coming from $\mathcal{M}''$. The partial rounds are modeled as follows:

$$\begin{aligned} x_{i,1} - (y_1 + c_{i,1})^\alpha &= 0 \\ y_j &= \mathcal{M}'_{j,1} \cdot x_{i,1} + \sum_{k=2}^{t} \mathcal{M}'_{j,k} \cdot (y_k) && j \in [1,t], \end{aligned}$$

where $\mathbf{Y}$ is defined in the same way as Section 4.1. The final system, similar to Poseidon's system, has $r + (R_F - 1)t + R_P$ polynomials of degree α in $r + (R_F - 1)t + R_P$ variables.

4.3 Complexity of the Attack

Both Poseidon and Poseidon2 can be modeled as a sequence of polynomials that are not regular. Therefore, the Macaulay bound used in [29] is a loose upper bound for the solving degree of the system. In order to extrapolate the solving degree, we run the attack for smaller parameters and derive a formula for the solving degree. We calculated the solving degree for more than 100 different parameter sets of Poseidon and Poseidon2, with state sizes up to five and rate up to four, using different values for α. Our experiments show that changing the value of α, does **not** affect the asymptotic *growth* of the solving degree when the number of rounds is increased. More precisely, the solving degree grows at the same rate independently of α in all the cases we analyzed (*i.e.*, the derivative of the solving degree does not change). In addition, the field size seems to have no effect on the value or the growth of solving degrees. In the case of Poseidon2, the solving degree is the same as Poseidon's solving degree when $R_F > 2$. Using the collected data, and the conjecture that the solving degree grows linearly as a function of a number of variables, we propose the following heuristic formula for the solving degree.

$$d_{sol} = r\frac{R_F}{2} + 0.8R_P + \alpha. \tag{7}$$

Our experiments show that partial rounds do not provide the same level of resistance against algebraic attacks as full rounds and the solving degree growth is a function of the rate instead of the state size.

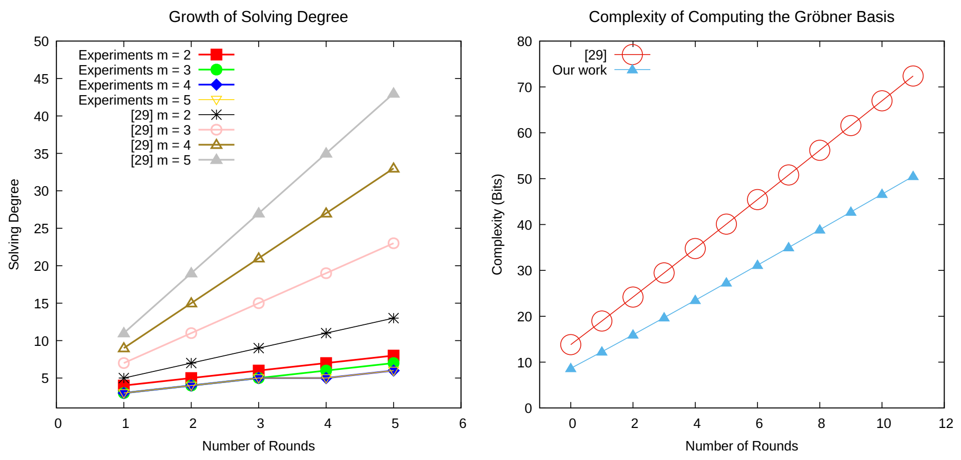

Fig. 4. Comparison of the claimed solving degree [29] and the solving degree of the system describing Poseidon for different state sizes when $R_P = 0$, and $rate = 1$. The solving degree growth is a linear function of the number of rounds and not the state size. The plot on the right shows the overestimation of the complexity of Poseidon in [**?**] and the complexity of this work.

In Table 2, we describe instances of Poseidon and Poseidon2 in which the complexity of a Gröbner basis attack is less than their claimed security level.

4.4 Discussion on Complexity and Running time

In an attempt to optimize the polynomial system describing Poseidon from [39], one can remove the variables corresponding to the output capacity which results

λ	α	t	r	R_F	R_P	$\mathcal{C}_{\text{GB}}$	$\mathcal{C}_{\text{SFGLM}}$	$\mathcal{C}_{\text{Elim}}$
1024	3	24	8	8	85	**712.98**	**705.67**	**466.18**
512	5	12	4	8	57	**430.71**	615.40	413.62
384	7	9	3	8	47	**351.56**	593.25	400.61

Table 2. Examples of Poseidon and Poseidon2 hash functions with security parameter λ over the finite field $\mathbb{F}_p$ with $\log_2(p) \approx 128$. $\mathcal{C}_{\text{GB}}$ is the complexity of computing the Gröbner basis in *degrevlex* order, $\mathcal{C}_{\text{SFGLM}}$ is the asymptotic complexity of sparse FGLM, and $\mathcal{C}_{\text{Elim}}$ is the complexity factoring univariate polynomials and recovering their roots.

in a polynomial system with **larger** solving degree and fewer variables. Comparing the theoretical complexity of the attack of both systems shows that removing these variables makes the system harder to solve. However, computing the Gröbner basis for the system with fewer variables is more efficient in practice and has a noticeably lower running time than suggested by the theoretical complexity. For example, modeling an instance with $(R_F = 10, R_P = 2, t = 2, r = 1)$ using the first approach resulted in a complexity of $> 2^{52}$ and the code did not terminate after 7 days.[8] On the other hand, when modeling the same instance using the second approach, the complexity for computing the Gröbner basis is $2^{63.3}$ and the code terminates after $2^{16.64}$ seconds. These two instances are described in Table 3.

Polynomial Modeling	Theoretical Complexity (field multiplications)	Expected Running Time (seconds)	Actual Running Time (seconds)
Minimize solving degree	2^{52}	$2^{18.42}$	$> 2^{19.2}$
Minimize variables	$2^{63.3}$	$2^{31.3}$	$2^{16.64}$

Table 3. POSEIDON instances with running time that does not match the expected complexity. The expected running time is computed using the number of basic CPU operations that can be done in 1 second on the host machine.

A similar observation is made in [28] and the authors use $\omega = 1$ to ensure a security margin. An explanation for the faster running time is that the F4/F5 algorithms that are commonly used in computing the Gröbner basis do not work with the full Macaulay matrix, but rather smaller submatrices that are selected based on *ad-hoc* criteria. Moreover, the matrices are highly sparse, and therefore the running time of the algorithm is noticeably faster than what the complexity suggests. Therefore, using $\omega = 2$ in analyzing the security of the design may lead to an overestimation of the actual resources the attacker requires to break it.

[8] At this point it was killed by the cluster's scheduler.

5 Conclusion

In this paper, we analyzed the security of Poseidon and Poseidon2 which are primitives based on the HADES design strategy. We studied Gröbner basis attacks, and showed that partial rounds are not providing the claimed resistance. Using Gröbner basis attacks, we break instances of Poseidon and Poseidon2 claiming 1024 bits of security using an attack whose complexity is upper bounded by $2^{731.77}$ and show that the original security argument does not hold for instances with as small as 384 bits of claimed security. Since the vulnerabilities presented in this paper are more pronounced in non-standard security levels we are not aware of any practical instance directly affected. However, we argued that the actual running time of the attack is significantly less than what is suggested by the complexity analysis and hence we encourage the potential users to consider in their risk assessment the ongoing erosion in the security of HADES instances.

References

1. 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: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology – ASIACRYPT 2019. pp. 371–397. Springer International Publishing, Cham (2019)
2. Albrecht, M.R., 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.) Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10031, pp. 191–219 (2016). `https://doi.org/10.1007/978-3-662-53887-6_7`, `https://doi.org/10.1007/978-3-662-53887-6_7`
3. Albrecht, M.R., Rechberger, C., Schneider, T., Tiessen, T., Zohner, M.: Ciphers for mpc and fhe. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology – EUROCRYPT 2015. pp. 430–454. Springer Berlin Heidelberg, Berlin, Heidelberg (2015)
4. Aly, A., Ashur, T., Ben-Sasson, E., Dhooghe, S., Szepieniec, A.: Design of Symmetric-Key Primitives for Advanced Cryptographic Protocols. IACR Trans. Symmetric Cryptol. **2020**(3), 1–45 (2020). `https://doi.org/10.13154/tosc.v2020.i3.1-45`, `https://doi.org/10.13154/tosc.v2020.i3.1-45`
5. Ashur, T., Kindi, A., Mahzoun, M.: Xhash8 and xhash12: Efficient stark-friendly hash functions. IACR Cryptol. ePrint Arch. p. 1045 (2023), `https://eprint.iacr.org/2023/1045`
6. Ashur, T., Kindi, A., Meier, W., Szepieniec, A., Threadbare, B.: Rescue-prime optimized. Cryptology ePrint Archive, Paper 2022/1577 (2022), `https://eprint.iacr.org/2022/1577`, `https://eprint.iacr.org/2022/1577`
7. Ashur, T., Mahzoun, M., Toprakhisar, D.: Chaghri - a fhe-friendly block cipher. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. p. 139–150. CCS '22, Association for Computing Machin-

ery, New York, NY, USA (2022). https://doi.org/10.1145/3548606.3559364, https://doi.org/10.1145/3548606.3559364

8. Aumasson, J., Khovratovich, D., Mennink, B., Quine, P.: Safe: Sponge api for field elements. Cryptology ePrint Archive, Paper 2023/522 (2023), https://eprint.iacr.org/2023/522, https://eprint.iacr.org/2023/522
9. Bariant, A., Bouvier, C., Leurent, G., Perrin, L.: Algebraic attacks against some arithmetization-oriented primitives. IACR Transactions on Symmetric Cryptology **2022**(3), 73–101 (Sep 2022). https://doi.org/10.46586/tosc.v2022.i3.73-101, https://tosc.iacr.org/index.php/ToSC/article/view/9850
10. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: On the indifferentiability of the sponge construction. In: Smart, N. (ed.) Advances in Cryptology – EUROCRYPT 2008. pp. 181–197. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
11. Bettale, L., Faugère, J., Perret, L.: Solving polynomial systems over finite fields: improved analysis of the hybrid approach. In: van der Hoeven, J., van Hoeij, M. (eds.) International Symposium on Symbolic and Algebraic Computation, ISSAC'12, Grenoble, France - July 22 - 25, 2012. pp. 67–74. ACM (2012). https://doi.org/10.1145/2442829.2442843, https://doi.org/10.1145/2442829.2442843
12. Beyne, T., Canteaut, A., Dinur, I., Eichlseder, M., Leander, G., Leurent, G., Naya-Plasencia, M., Perrin, L., Sasaki, Y., Todo, Y., Wiemer, F.: Out of oddity – new cryptanalytic techniques against symmetric primitives optimized for integrity proof systems. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology – CRYPTO 2020. pp. 299–328. Springer International Publishing, Cham (2020)
13. Boura, C., Canteaut, A., De Cannière, C.: Higher-order differential properties of keccak and luffa. In: Joux, A. (ed.) Fast Software Encryption. pp. 252–269. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
14. 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. Cryptology ePrint Archive, Paper 2022/840 (2022), https://eprint.iacr.org/2022/840, https://eprint.iacr.org/2022/840
15. Canteaut, A., Carpov, S., Fontaine, C., Lepoint, T., Naya-Plasencia, M., Paillier, P., Sirdey, R.: Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression. Journal of Cryptology **31**(3), 885–916 (Jul 2018). https://doi.org/10.1007/s00145-017-9273-9, https://doi.org/10.1007/s00145-017-9273-9
16. Cantor, D.G., Zassenhaus, H.: A new algorithm for factoring polynomials over finite fields. Mathematics of Computation **36**(154), 587–592 (1981), http://www.jstor.org/stable/2007663
17. Cid, C., Indrøy, J.P., Raddum, H.: Fasta – a stream cipher for fast fhe evaluation. In: Galbraith, S.D. (ed.) Topics in Cryptology – CT-RSA 2022. pp. 451–483. Springer International Publishing, Cham (2022)
18. Cosseron, O., Hoffmann, C., Méaux, P., Standaert, F.: Towards case-optimized hybrid homomorphic encryption - featuring the elisabeth stream cipher. In: Agrawal, S., Lin, D. (eds.) 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. Lecture Notes in Computer Science, vol. 13793, pp. 32–67. Springer (2022). https://doi.org/10.1007/978-3-031-22969-5_2, https://doi.org/10.1007/978-3-031-22969-5_2
19. Cox, D., Little, J., O'Shea, D.: Ideals, varieties, and algorithms. an introduction to computational algebraic geometry and commutative algebra (2007), https://link.springer.com/book/10.1007/978-0-387-35651-8

20. Ding, J., Schmidt, D.: Solving Degree and Degree of Regularity for Polynomial Systems over a Finite Fields, pp. 34–49. Springer Berlin Heidelberg, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-42001-6_4, https://doi.org/10.1007/978-3-642-42001-6_4
21. Dobraunig, C., Eichlseder, M., Grassi, L., Lallemand, V., Leander, G., List, E., Mendel, F., Rechberger, C.: Rasta: A cipher with low anddepth and few ands per bit. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology – CRYPTO 2018. pp. 662–692. Springer International Publishing, Cham (2018)
22. Dobraunig, C., Grassi, L., Helminger, L., Rechberger, C., Schofnegger, M., Walch, R.: Pasta: A case for hybrid homomorphic encryption. Cryptology ePrint Archive, Paper 2021/731 (2021), https://eprint.iacr.org/2021/731, https://eprint.iacr.org/2021/731
23. Faugère, J., 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). https://doi.org/https://doi.org/10.1006/jsco.1993.1051, https://www.sciencedirect.com/science/article/pii/S0747717183710515
24. Faugère, J.C., Mou, C.: Sparse fglm algorithms. Journal of Symbolic Computation **80**, 538–569 (2017). https://doi.org/https://doi.org/10.1016/j.jsc.2016.07.025, https://www.sciencedirect.com/science/article/pii/S0747717116300700
25. Gérard, B., Grosso, V., Naya-Plasencia, M., Standaert, F.X.: Block ciphers that are easier to mask: How far can we go? In: Bertoni, G., Coron, J.S. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2013. pp. 383–399. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
26. Grassi, L., Hao, Y., Rechberger, C., Schofnegger, M., Walch, R., Wang, Q.: Horst meets fluid-spn: Griffin for zero-knowledge applications. Cryptology ePrint Archive, Paper 2022/403 (2022), https://eprint.iacr.org/2022/403, https://eprint.iacr.org/2022/403
27. Grassi, L., Kales, D., Khovratovich, D., Roy, A., Rechberger, C., Schofnegger, M.: Starkad and poseidon: New hash functions for zero knowledge proof systems. IACR Cryptol. ePrint Arch. p. 458 (2019), https://eprint.iacr.org/2019/458
28. 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. Cryptology ePrint Archive, Paper 2023/1025 (2023), https://eprint.iacr.org/2023/1025, https://eprint.iacr.org/2023/1025
29. Grassi, L., Khovratovich, D., Rechberger, C., Roy, A., Schofnegger, M.: Poseidon: A new hash function for Zero-Knowledge proof systems. In: 30th USENIX Security Symposium (USENIX Security 21). pp. 519–535. USENIX Association (Aug 2021), https://www.usenix.org/conference/usenixsecurity21/presentation/grassi
30. Grassi, L., Khovratovich, D., Schofnegger, M.: Poseidon2: A faster version of the poseidon hash function. In: El Mrabet, N., De Feo, L., Duquesne, S. (eds.) Progress in Cryptology - AFRICACRYPT 2023. pp. 177–203. Springer Nature Switzerland, Cham (2023)
31. Grassi, L., Lüftenegger, R., Rechberger, C., Rotaru, D., Schofnegger, M.: On a generalization of substitution-permutation networks: The HADES design strategy. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part

II. Lecture Notes in Computer Science, vol. 12106, pp. 674–704. Springer (2020). https://doi.org/10.1007/978-3-030-45724-2_23, https://doi.org/10.1007/978-3-030-45724-2_23

32. Ha, J., Kim, S., Lee, B., Lee, J., Son, M.: Rubato: Noisy ciphers for approximate homomorphic encryption. In: Dunkelman, O., Dziembowski, S. (eds.) Advances in Cryptology – EUROCRYPT 2022. pp. 581–610. Springer International Publishing, Cham (2022)
33. Hebborn, P., Leander, G.: Dasta - alternative linear layer for rasta. IACR Trans. Symmetric Cryptol. **2020**(3), 46–86 (2020). https://doi.org/10.13154/tosc.v2020.i3.46-86, https://doi.org/10.13154/tosc.v2020.i3.46-86
34. Jakobsen, T., Knudsen, L.R.: The interpolation attack on block ciphers. In: Biham, E. (ed.) Fast Software Encryption. pp. 28–40. Springer Berlin Heidelberg, Berlin, Heidelberg (1997)
35. Keller, N., Rosemarin, A.: Mind the middle layer: The hades design strategy revisited. In: Canteaut, A., Standaert, F.X. (eds.) Advances in Cryptology – EUROCRYPT 2021. pp. 35–63. Springer International Publishing, Cham (2021)
36. Knudsen, L.R.: Truncated and higher order differentials. In: Preneel, B. (ed.) Fast Software Encryption. pp. 196–211. Springer Berlin Heidelberg, Berlin, Heidelberg (1995)
37. MacAulay, F.S.: Some formulæ in elimination. Proceedings of the London Mathematical Society **s1-35**(1), 3–27 (1902). https://doi.org/https://doi.org/10.1112/plms/s1-35.1.3, https://londmathsoc.onlinelibrary.wiley.com/doi/abs/10.1112/plms/s1-35.1.3
38. Méaux, P., Journault, A., Standaert, F.X., Carlet, C.: Towards stream ciphers for efficient fhe with low-noise ciphertexts. In: Fischlin, M., Coron, J.S. (eds.) Advances in Cryptology – EUROCRYPT 2016. pp. 311–343. Springer Berlin Heidelberg, Berlin, Heidelberg (2016)
39. Sauer, J.F.: Gröbner basis-attacking a tiny sponge. Tech. rep., AS Discrete Mathematics (2021), https://asdm.gmbh/2021/06/28/gb_experiment_summary/
40. Shoup, V.: Factoring polynomials over finite fields: Asymptotic complexity vs. reality (1993)
41. Szepieniec, A., Ashur, T., Dhooghe, S.: Rescue-prime: a standard specification (sok). IACR Cryptol. ePrint Arch. p. 1143 (2020), https://eprint.iacr.org/2020/1143
42. Szepieniec, A., Lemmens, A., Sauer, J.F., Threadbare, B., Al-Kindi: The tip5 hash function for recursive starks. Cryptology ePrint Archive, Paper 2023/107 (2023), https://eprint.iacr.org/2023/107, https://eprint.iacr.org/2023/107
43. Williams, V.V., Xu, Y., Xu, Z., Zhou, R.: New bounds for matrix multiplication: from alpha to omega (2023)
44. Youssef, A., Mister, S., Tavares, S.: On the design of linear transformations for substitution permutation encryption networks pp. 40–48 (09 1997)