# Algebraic cryptanalysis of POSEIDON

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

[1] Polygon Research
[2] Cryptomeria, Leuven, Belgium tomer@cryptomeria.tech
[3] Eindhoven University of Technology, Eindhoven, Netherlands, t.buschman@student.tue.nl
[4] Eindhoven University of Technology, Eindhoven, Netherlands, m.mahzoun@tue.nl

**Abstract.** POSEIDON is a hash function proposed by Grassi *et al.* in the USENIX Security '21 conference. Due to its impressive efficiency and low arithmetic complexity it has garnered the attention of designers of integrity-proof systems such as SNARKS, STARKS, and Bulletproofs. In this work, we show some caveats in Poseidon's security argument. Most notably, we extend on previous work by Sauer and quantify the rate at which the degree of regularity increases as a function of full and partial rounds. We observe that this degree grows slower than originally assumed, suggesting that there are cases where the recommended number of rounds is insufficient to meet claimed security.
The findings presented in this paper are asymptotic in nature and do not affect all parameter sets equally. As a proof of concept, we present a full attack for an instance at the 1024-bit security level. We present two more parameter sets at the 512- and 384-bit security levels where the original security argument does not hold, but for which we were not able to demonstrate a full attack due to other aspects of the design. We were not able to find parameter sets in the 128- and 256-bit levels that are vulnerable.

**Keywords:** POSEIDON · Hash functions · Zero-Knowledge proof systems · Gröbner basis attacks

## 1 Introduction

Arithmetization-oriented primitives are highly favored for use in advanced cryptographic protocols, such as Zero-Knowledge (ZK) proofs, Multiparty Computation (MPC) protocols, and Fully Homomorphic Encryption (FHE) protocols. Among these primitives is POSEIDON[GKK+19], an efficient hash function constructed by employing the POSEIDON$^\pi$ permutation in a sponge construction. POSEIDON$^\pi$ is an SP-network based on the HADES design strategy[GLR+20] and operates over the finite field $\mathbb{F}_p$ for a prime $p$.

POSEIDON is claimed to resist statistical and algebraic attacks in the literature. In most cases, statistical attacks do not pose a significant threat to arithmetic-oriented hash functions due to the large finite fields they operate on. Therefore, designers and attackers focus on algebraic attacks when analyzing the security of these designs. The security of POSEIDON is analyzed against four different types of algebraic attacks: Interpolations attacks [JK97], Gröbner basis attacks [CLO07], Higher-Order differential attacks [Knu95], and Zero-Sum partitions attacks [BCC10], and Gröbner basis attacks are considered to be among the most promising algebraic attacks against arithmetic-oriented designs [ACG+19]. The primary objective of our research is to conduct a thorough security analysis of POSEIDON in the context of Gröber basis attacks, specifically with regard to preimages resistance.

We present a Gröbner basis attack and show that the required number of rounds to ensure preimage resistance is underestimated. Two reasons contribute to this underestimation.

Firstly, the degree of regularity was assumed to be the same as Macaulay bound[Mac02], which we show to be inaccurate. Secondly, it was believed that partial rounds provide the same security resistance against algebraic attacks as full rounds. Partial rounds were used in previous attacks against POSEIDON [BCD+20, KR20] and the security argument of POSEIDON was updated accordingly. We demonstrate that in cases where the state size is larger than two, partial rounds offer significantly less security against Gröbner basis attacks than full rounds.

In addition to the proposed Gröbner basis attack against POSEIDON, we conducted a comprehensive study to analyze the security argument presented for POSEIDON's resistance to algebraic attacks. Our rigorous analysis revealed three distinct flaws in these arguments, each of which has implications for the required number of rounds needed to ensure POSEIDON's security against algebraic attacks. First, we show that is a transcription error in the security argument against Gröbner basis attack in the full round setting. Second, the logical reasoning of the security argument against the Gröbner basis attack is not sound. Third, there exists an error in the symbolic computation of bounds in the security argument that undermines the resistance of POSEIODN.

**Structure of the paper.** In Section 2, the notations used throughout the paper are introduced and an overview of POSEIDON's design is provided. In Section 3, the security claims against statistical and algebraic attacks are presented, and the number of required rounds for arbitrary parameters to ensure security is discussed. In Section 4, a Gröbner basis attack is proposed and vulnerable instances are demonstrated which shows the efficacy of the proposed attack by breaking instances of the POSEIDON hash function. In Section 5, flaws in the security argument of [GKK+19] are investigated, and their impact on the security of POSEIDON is discussed. Finally, in Section 6, the concluding remarks on our findings and their implications for the security of POSEIDON in addition t the limitations of our work and suggested future research directions are discussed.

## 2 Preliminaries

### 2.1 Notations

POSEIDON hash function operates on the prime field $\mathbb{F}_p = GF(p)$. We define $\lambda$ as the security parameter, and $[a, b] = \{a, \ldots, b\}$. 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 $(x_1, \ldots, x_n)$. The MDS matrix used in POSEIDON is denoted by $\mathcal{M}$ where $\mathcal{M}_{i,j}$ is the $j^{th}$ element in the $i^{th}$ row.

### 2.2 Sponge construction

Sponge construction [BDPVA08] is a mode of operation used to create sponge functions with variable-length inputs and outputs using a fixed-length permutation and padding rule. Sponge functions are widely used in cryptography and can be utilized for many cryptographic primitives, including hash functions and stream ciphers. Let $f : \mathbb{K} \to \mathbb{K}$ be a fixed-length transformation and let $P$ be a padding function that transforms an arbitrary-length input message $\mathbf{M} \in \mathbb{F}_p^*$ into a sequence of $\chi$ blocks of length $r$, where $r$ is the rate of the sponge function. The sponge function $F$ with the input $\mathbf{M}$, generates the output $\mathbf{H} = \mathbf{H}_1, \ldots, \mathbf{H}_{\chi'}$, where for all $1 \leq i \leq \chi'$ , $\mathbf{H}_i \in \mathbb{F}_p^r$. The sponge function works as follows:

1. Let $S$ be the state of the sponge function of length $t = r + c$, where $r$ is the rate of the sponge and $c$ is the capacity of the sponge.
2. The state $\mathbf{S}$ is initialized to $(0, \ldots, 0)$.

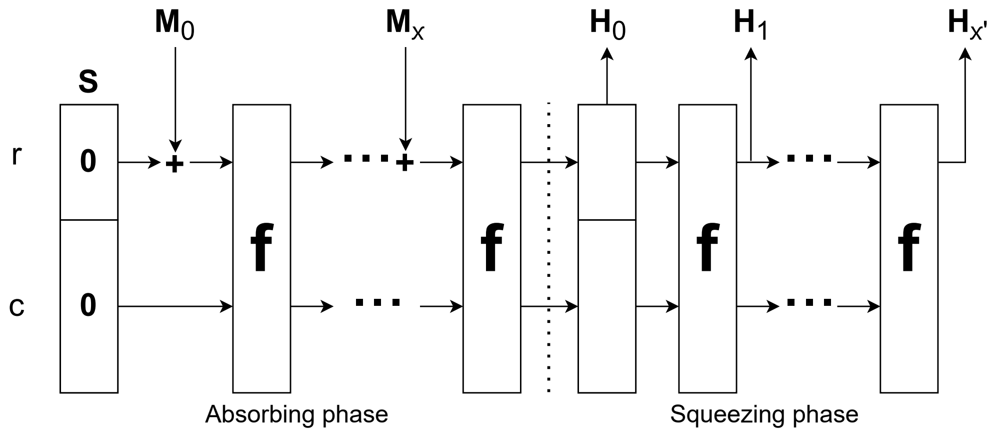

**Figure 1:** A sponge function with rate $r$, capacity $c$, and internal permutation $f$.

3. Absorbing phase: The padded message $\mathbf{M}$ is split into $\chi$ 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 on $\mathbf{S}$.
$$\mathbf{S} = f(\mathbf{S} + \mathbf{M}_i)$$

4. Squeezing phase: once all blocks of the padded message have been absorbed, the sponge function enters the squeezing phase. In this phase, the function outputs blocks $\mathbf{H}_1, \dots, \mathbf{H}_{\chi'}$ of length $r$ and update internal state $\mathbf{S}$ by applying the function $f$ until enough output are produced.

In Figure 1, the construction of the sponge function is illustrated.

Assuming that $f$ is computationally indistinguishable from a random permutation, a sponge function with capacity $c$ offers $2^{c/2}$ bits of collision and preimage resistance[BDPVA08].

## 2.3 POSEIDON hash function description

POSEIDON: $\mathbb{F}_p^* \to (\mathbb{F}_p^r)^{\chi'}$ is a hash function operating over $\mathbb{F}_p$ with output of $\chi'$ blocks of length $r$. It is constructed using the POSEIDON$^\pi$ permutation in the sponge construction with rate $r$ and capacity $c$. POSEIDON$^\pi$ is an SP-Network with 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. The POSEIDON$^\pi$ permutation is illustrated in Figure 2 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 sbox is applied to the first element (in partial rounds), or all elements of the state (in full rounds).
3. Linear layer: $L_{\mathcal{M}} : \mathbb{F}_p \to \mathbb{F}_p$, $L_{\mathcal{M}}(\mathbf{X}) = \mathcal{M} \cdot \mathbf{X}^\intercal$ where $\mathcal{M}$ is a MDS matrix.

POSEIDON offers two types of sboxes:

1. $S(x) = x^\alpha$, where $\alpha$ is an odd integer,
2. $S(x) = x^{-1}$.

We study the case where $S(x) = x^\alpha$ in this paper.

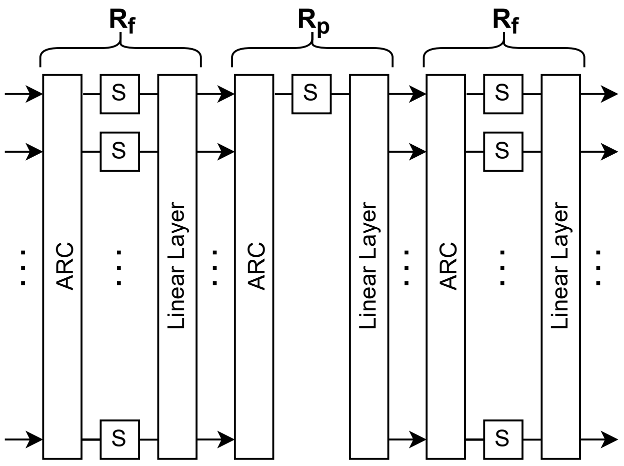

**Figure 2:** Construction of POSEIDON$^{\pi}$ permutation.

## 3 Security claim

The analysis of POSEIDON's security involves evaluating the system's vulnerability to two categories of attacks, namely statistical attacks and algebraic attacks. The authors established a constraint on the secure number of rounds $R = R_F + R_P$ and the desired security level $\lambda$ by ensuring that none of the attacks can be executed with a complexity of less than $\lambda$ steps, thereby ensuring the system's resilience against potential attacks. In addition to this, the authors incorporated a security margin to minimize the risk of any unpredicted weaknesses in the system. The security margins are:

1. Two additional full rounds ($+2R_F$), and
2. 7.5% of more partial rounds ($+7.5\%R_P$).

### 3.1 Generic security

The sponge construction provides a generic security level of $2^{c/2}$ bits. In addition, an ideal hash function with security parameter $\lambda$ is expected to provide $2^{\lambda}$ bits of security against preimage attacks. To ensure resistance against generic attacks to the hash functions, POSEIDON design ensures that $\lambda \leq \frac{c}{2}$ and $\lambda \leq r$.

### 3.2 Statistical attacks

In [GKK$^+$19, Equation. 2], the minimum number of rounds to ensure security against statistical attacks for sboxes of the form $S(x) = x^{\alpha}$ is described as:

$$\begin{cases} 6 & \text{if } \lambda \leq (\lfloor \log_2 p \rfloor - \log_2(\alpha - 1)) \cdot (t+1) \\ 10 & \text{otherwise.} \end{cases} \tag{1}$$

### 3.3 Algebraic attacks

Evaluating the security of POSEIDON against algebraic attacks suggests that Interpolation attacks and Gröbner basis attacks have the lowest complexity. Therefore, the constraints on the number of rounds derived from these attacks are sufficient to provide resistance to other types of algebraic attacks.

#### 3.3.1 Interpolation attacks

In [GKK+19, Equation. 3], it is asserted that for security level of $\lambda$ bits, the maximum number of rounds vulnerable to the interpolation attack is:

$$R \leq \lceil \log_\alpha(2). \min\{\lambda, \log_2(p)\}\rceil + \lceil \log_\alpha(t)\rceil \tag{2}$$

#### 3.3.2 Gröbner basis attacks

In [GKK+19, Equation. 5, 6], it is asserted that for the security level of $\lambda$ bits, the number of rounds vulnerable to Gröbner basis is:

$$\begin{cases} R \leq \log_\alpha(2). \min\left\{\frac{\lambda}{3}, \frac{\log_2(p)}{2}\right\} \\ R \leq t - 1 + \min\left\{\frac{\log_\alpha(2)\cdot\lambda}{t+1}, \frac{\log_\alpha(2)\log_2(p)}{2}\right\} \end{cases} \tag{3}$$

### 3.4 Sufficient number of rounds

By applying the constraints specified in Equations (1)–(3) and the security margin, the total number of rounds to ensure the resistance of POSEIODN to studied attacks can be computed. A Python script is provided to facilitate the computation of the number of rounds[1]. In our analysis, we utilized this script to calculate the necessary number of rounds required to ensure the security of our chosen parameters.

## 4 Gröbner basis attacks

In this section, we provide an overview of Gröbner basis attacks and analyze the security of POSEIDON against such attacks. To find a secret used in a cryptographic primitive, the attacker first needs to write the primitive as a system of multivariate polynomials, with the secrets being the variables and solve the system to find the secrets. In case of preimage attack to a hash function, the input can be considered as the secret.

### 4.1 The methodology of the Gröbner basis attacks

To efficiently solve a polynomial system using the Gröbner basis algorithms, the following steps are taken:

1. Compute the 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 to compute the Gröbner basis in the *degrevlex* order is its low complexity cost compared to other term orderings. The complexity of computing a Gröbner basis in *degrevlex* term order is [BFP12]:

$$O\left(\binom{n + d_{sol}}{d_{sol}}^{\omega}\right), \tag{4}$$

where $n$ is the number of unknows in the multivariate polynomial system, $2 < \omega \leq 2.3727$ [Wil14] is the linear algebra constant, and $d_{sol}$ is the degree of the regularity of the polynomial system [DS13].

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

There are multiple notions of degrees in the literature that aim to capture the complexity of the Gröbner basis computation[CG21]. In this work, we use the solving degree, which is defined as the largest degree of the polynomials in the extended Macaulay matrix involved in the computation of the Gröbner basis. The complexity analysis of the Gröbner basis algorithm suggests that the linear algebra operations on the Macaulay matrix are the dominant step in the complexity of the algorithm. Therefore, the dimensions of the Macaulay matrix are widely used as a measurement for the complexity of the Gröbner basis algorithms[BFS13].

To determine the solving degree of the polynomial system used to model POSEIDON, we analyzed the solving degrees of round-reduced versions of the system, and performed linear regression on the resulting data to establish bounds for real-world parameters. We opted for linear regression over polynomial regression because empirical observations[GKK$^+$19, AAB$^+$20, ACG$^+$19], along with known theoretical bounds in the literature like the Macaulay bound, show a linear growth of solving degree with respect to the number of polynomials in the system.

The computed Gröbner basis in *degrevlex* order needs to be converted to a Gröbner basis in *lex* order. Using FGLM [FGLM93] algorithm, the complexity of converting the *degrevlex* order to *lex* order is:

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

where $D$ is the degree of the zero-dimensional ideal. The second step can be done more efficiently using the sparse FGLM [FM17], which has the asymptotic complexity of:

$$O\Big(\sqrt{\frac{6}{n\pi}}D^{2+\frac{n-1}{n}}\Big). \tag{5}$$

After computing the Gröbner basis in *lex* order, the univariate polynomials need to be factored in order to compute partial solutions and extend them to the solutions of the system. The complexity of factoring a univariate polynomial over a finite field is [HSK12]:

$$O\left(\textstyle\sum_i d_i^{\omega}\right),$$

where each $d_i$ is the degree of the univariate polynomial appearing at the $i^{\text{th}}$ elimination step. To factor a polynomial with degree $D$ over the finite field $GF(P)$, we use the Cantor/Zassenhaus [CZ81] algorithm with the complexity [Sho93]:

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

## 4.2 Polynomial modeling

In the analysis of POSEIDON against Gröbner basis attacks, the authors considered the input size $\chi$ to be equal to the rate $r$ of the sponge function. In the polynomial representation of POSEIDON, the first $\chi$ inputs are considered unknown variables, while the last $c$ inputs and the output are considered known constants. The resulting polynomial system is determined and is assumed it is a regular system. Therefore, he Macaulay bound is to estimate the degree of regularity used in the complexity estimation of the Gröbner basis computation[GKK$^+$19, Section C.2.2].

Cryptographic primitives can be modeled in various ways using algebraic relations describing them but the polynomial system that minimizes the complexity of the Gröbner basis attack is the preferred one. For the instances of POSEIDON with $S(x) = x^\alpha$, two approaches are used to model the sponge function. The first approach is called the full-permutation equation and involves using one equation for each output element based on the input variables which results in a small number of equations with high degrees.

The second approach, called round-level equations, uses new variables in each round to form a system of polynomials with small degrees.

After a thorough analysis of various methods for polynomial modeling, we identified two approaches that we consider to be the most efficient ways to model POSEIDON in a round-level configuration.

**The first approach**[2], aims to minimize the solving degree of the system which is described in Section 4.2.1. Our results demonstrate that a reduction in the solving degree of the polynomial system can result in a significant decrease in its computational complexity when a limited number of additional variables are introduced.

**The second approach** objective is to use the minimum number of variables in the polynomial system and is described in Section 4.2.2. Despite the higher theoretical complexity associated with reducing the number of variables, our empirical results demonstrate that this approach is more efficient in practice. Specifically, we found that the running time of this approach is significantly lower than the alternative method, and is also much lower than the complexity suggested by prior research [BFP12]. Evidence supporting this assertion is presented in Appendix A.

To define the polynomial system that models the POSEIDON hash function, we use $\mathbf{C}_i = \{c_{i,1}, \dots, c_{i,t}\}$ to denote the round constants for the round $i \in \{1, \dots, R\}$. $\mathbf{X}_i = \{x_{i,1}, \dots, x_{i,t}\}$ are the variables that are describing the state in the round $i \in \{0, \dots, R\}$, where $\mathbf{X}_0 = (x_1, \dots, x_r, 0, \dots, 0)$ is the input and $\mathbf{X}_R = (H_1, \dots, H_r, x_{R,r+1}, \dots, x_{R,t})$ is the output.

### 4.2.1 Minimizing the solving degree

To minimize the solving degree, we experimented with different approaches to observe the impact of the polynomial system on the solving degree. While it is challenging to suggest a general method for minimizing the solving degree, our analysis suggests that reducing the number of variables and excluding linear equations from the system are effective strategies in our case. The first round before multiplication by $\mathcal{M}$ is:

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

which adds $2r$ new variables and $r$ polynomials to the system. The relation of the state after the first and second Sbox layer is modeled as:

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

which adds $t$ 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:

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

which add $(R_f - 2)t$ 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:

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

[2]This approach is inspired by https://asdm.gmbh/2021/06/28/gb_experiment_summary/

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

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

which add $R_P$ 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:

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

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 $\alpha$ in $r + (R_F - 1)t + R_P$ variables.

Considering the $t$ and $r$ as constants, the linear regression of $d_{sol}$ on number of rounds $R_F, R_P$ based on the experimental data is:

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

In Table 2, the details of our experiments to solve the polynomial system for the system are described.

### 4.2.2 Minimizing the number of variables

To reduce the number of variables in the round-level setting, the variables corresponding to the output capacity in Section 4.2.1 are removed while the polynomials for the first $R - 1$ rounds remain unchanged. The last round is modeled as:

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

The system has $(R_F - 2)t + R_P + 2r$ polynoamis of degree $\alpha$ in $(R_F - 2)t + R_P + 2r$ variables. In the Table 3, the details of our experiments to solve the polynomial system for the system are described. Considering the $t$ and $r$ as constants, the linear regression of $d_{sol}$ on number of rounds $R_F, R_P$ based on the experimental data is:

$$d_{sol} = r\frac{R_F}{2} + (\alpha - 1)R_P + \alpha$$

## 4.3 Complexity of the attack and broken parameters

We conducted experiments to compute the Gröbner basis for different instances of POSEIDON and determined the solving degrees, degree of the ideal, and degree of polynomials in the final Gröbner basis. The results of these computations are presented in detail in Appendix A.

Our first observation is that none of the systems we analyzed were regular, and the solving degree did not reach the Macaulay bound used in the analysis of POSEIDON in [GKK+19]. Therefore, the complexity of a Gröbner basis attack suggested by designers is overestimated.

Our second observation is that partial rounds do not provide the same level of resistance against algebraic attacks as full rounds. The addition of partial rounds increases the solving degree by at most one in each round, and there are cases where partial rounds do not increase the solving degree at all.

Minimizing the solving degree theoretically minimizes the theoretical complexity of the attack. 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. In some cases, the system of Section 4.2.2 was approximately 200 times faster than the system of Section 4.2.1.

The complexity of the Gröbner basis attack can be summarized as:

1. **Computing the Gröbner basis in *degrevlex* order**. The solving degree computed by linear regression of experimental data is:

$$d_{sol} = r\frac{R_F}{2} + R_P + \alpha,$$

and by substituting the values for $d_{sol}$ and number of variables in Equation (4), we obtain the complexity as:

$$\binom{(R_F-1)\,t + R_P + r + r\frac{R_F}{2} + R_P + \alpha}{r\frac{R_F}{2} + R_P + \alpha}^{2.3727}$$

2. **Changing the term order from *degrevlex* to *lex* order**. The degree of the corresponding zero-dimensional ideal is:

$$d_I = \alpha^{r\cdot R_F + R_P},$$

and the asymptotic complexity of the sparse FGLM is:

$$O\left(\sqrt{\frac{6}{((R_F-1)t + R_P + r)\pi}}(\alpha)^{\left(2+\frac{(R_F-1)t+R_P+r-1}{(R_F-1)t+R_P+r}\right)\cdot(r\cdot R_F+R_P)}\right).$$

3. **Complexity of finding the variety**. The largest degree of polynomials in the final Gröbner basis with respect to *lex* order is:

$$d_{\mathrm{GB}} = \alpha^{r\cdot(R_F-1)+R_P},$$

and the complexity of finding the variety is:

$$O\left(\left(3^{r\cdot(R_F-1)+R_P}\right)^2\left(\log\left(3^{r\cdot(R_F-1)+R_P}\right)\log\log\left(3^{r\cdot(R_F-1)+R_P}\right)\right)\left(\log p + \log\left(3^{r\cdot(R_F-1)+R_P}\right)\right)\right)$$

In Table 1, an example of parameters for POSEIDON that is susceptible to our attack is described.

**Table 1:** Examples of POSEIDON hash function with security parameter $\lambda$. $\mathcal{C}_{\mathrm{GB}}$ is the complexity of computing the Gröbner basis in *degrevles* order, $\mathcal{C}_{\mathrm{SFGLM}}$ is the asymptotic complexity of sparse FGLM, $\mathcal{C}_{\mathrm{FGLM}}$ is the complexity of FGLM and $\mathcal{C}_{\mathrm{Elim}}$ is the complexity factoring univariate polynomials and recovering their roots.

| $\lambda$ | $\log_2(p)$ | $\alpha$ | $t$ | $r$ | $R_F$ | $R_P$ | $\mathcal{C}_{\mathrm{GB}}$ | $\mathcal{C}_{\mathrm{SFGLM}}$ | $\mathcal{C}_{\mathrm{FGLM}}$ | $\mathcal{C}_{\mathrm{Elim}}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 1024 | 128 | 3 | 24 | 8 | 8 | 85 | **731.77** | **705.67** | **716.5** | **466.18** |
| 512 | 128 | 5 | 12 | 4 | 8 | 57 | **435.10** | 615.40 | 627.13 | 413.62 |
| 384 | 128 | 7 | 9 | 3 | 8 | 47 | **361.81** | 593.25 | 604.78 | 400.61 |

## 5 Other flaws in the security analysis of POSEIDON

During our analysis of the security proof provided by the designers of POSEIDON, we identified three flaws that led to an overestimation of the security against Gröbner basis attacks. In Section 5.1, we demonstrate that using loose bounds in security arguments leads to incorrect conclusions. In Section 5.2, we analyze the security argument of POSEIDON against Gröbner basis attacks when $\chi = 1$, where the system is already a Gröbner basis in the full-permutation setting. We show a transcription error that resulted in underestimating the required number of rounds. Finally, in Section 5.3, we identify a flaw in the symbolic computations of round-level Gröbner basis analysis that led to an overestimation of the number of rounds.

### 5.1 Improper logic

The logical reasoning of resistance against Gröbner basis attack can be summarized as follow:

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

While each step in the security argument can be sound on its own, combining steps 2 and 4 can lead to erroneous conclusions, particularly if the upper bound used in step 2 is not tight. In such cases, the resistance to attack will be overestimated in step 3, resulting in values for the number of rounds $R_F^*$ and $R_P^*$ that are lower than what is necessary for security. Thus, step 4 will be unsuccessful in guaranteeing security, as there may be intermediate values for $R_F, R_P$ between the underestimated number of rounds $R_F^*, R_P^*$, and the correct number of rounds to ensure the security, $R'_F, R'_P$.

Consider a simple example: let us assume that a sponge construction with rate $r$ uses an $N$-round permutation, and there is an attack with complexity $2^{3Nr}$. However, this expression may be challenging to work with, 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

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

This means for all $0 \leq N \leq N^*$, the 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. We now use the proper expression to find the correct safety threshold, $N_s$. We solve:

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

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 steps 3 and 4 outlined earlier, it is advisable to avoid using an upper bound in step 2, as it may result in an overestimation of the resistance of the sponge function against attacks. Instead, we should find a lower bound that is easier to work with and use it in step 2 to avoid possible underestimations for the number of rounds in step 3.

Similarly, the resistance of POSEIDON against a round-level Gröbner basis attack is found to be (up to reasonable approximation) [GKK+19]:

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

with

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

That concludes step 1. For step 2, the resistance is upper-bounded by:

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

which is not a tight bound. Ultimately, resistance against the round-level attack is assumed as long as:

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

after which $t-1$ extra S-boxes are added to account for the possibility of a subspace attack.

## 5.2 Transcription error

**Full-Permutation Equation**. In the full round equation setting, a system of equations for the entire $R$ rounds is derived by considering each input as a variable and applying round functions to the input variable. When the number of input variables $\chi$ and the number of output variables are equal, the resulting system will consist of $\chi$ equations in $\chi$ variables, and the degree of each polynomial is upper-bounded by $D_\alpha(R) = \alpha^R$.

In the case that number of input variables is $\chi = 1$, the system is one polynomial of degree at most $\alpha^R$ 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. As the security argument provided in [GKK+19, 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 [GKK+19, Equation. 11], write the constraint for the full round attack as:

$$R_F + R_P \geq \log_\alpha(2) \cdot \min\{\frac{\lambda}{\mathbf{3}}, \frac{\log_2(p)}{2}\}, \tag{7}$$

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 [GKK+19, Equation. 5] would imply that 6 full rounds and 22 partial rounds are sufficient for $\alpha = 3, t = 2, p \approx 2^{1024}$, and the desired security level of 128 bits, whereas to gain that security level for these parameters, at least 35 partial rounds are required.

## 5.3 Symbolic computation error

In [GKK+19, Section. C.2.2]is shown that for the security level of $\lambda$, the maximum number of rounds that can be attacked is using Gröbner basis is:

$$(t-1)R_F + R_P + \chi \leq C^{-1} \cdot \min\{\lambda, \log_2(p)\chi\}, \tag{8}$$

with

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

The designers argued that the maximal number of rounds that can be attacked is when $\chi = 1$[GKK+19, Section C.2.2]. However, Equation 8 can be rewritten to

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

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 [GKK+19, 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, [GKK+19, Equation. 5] requires 3 constraints, instead of 2.

## 6 Conclusion

In this paper we demonstrated that POSEIDON is vulnerable to Gröbner basis attacks, which stems from the inaccurate security analysis and the use of incorrect bounds to estimate the attack complexity. While some parameters remain unaffected by the attack, some parameters are indeed compromised. Notably, we find that POSEIDON cannot offer high-security levels such as 1024 bits.

We presented two polynomial modelings with distinct objectives: one aimed at minimizing the solving degree, and another aimed at minimizing the number of variables. While the former method boasts a smaller theoretical complexity, the latter is found to terminate more expeditiously in practice. We posit that the efficiency of the latter approach is attributable to the smaller number of variables in the polynomial system. Hence, we recommend increasing the security margins or augmenting the number of rounds in future designs, as the theoretical complexity may not be a definitive metric for security evaluation.

In the pursuit of designing more secure ciphers, further investigation into the behavior of symmetric primitives with regard to Gröbner basis attacks, as well as the analysis of the interplay between theoretical complexities and actual running time of such attacks, would be highly valuable. Within the scope of our work, we employed a conservative linear regression technique for determining the solving degree, but we chose to neglect instances in which the solving degree remains static during some rounds. Hence, we posit that by adopting a less restrictive approach, we may improve the efficacy of the attack and be able to successfully break a larger set of parameters.

## 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.

[ACG+19] Martin R. Albrecht, Carlos Cid, Lorenzo Grassi, Dmitry Khovratovich, Reinhard Lüftenegger, Christian Rechberger, and Markus Schofnegger. Algebraic cryptanalysis of stark-friendly designs: Application to marvellous and mimc. Cryptology ePrint Archive, Paper 2019/419, 2019. https://eprint.iacr.org/2019/419.

[BCC10] Christina Boura, Anne Canteaut, and Christophe De Cannière. Higher-order differential properties of keccak and luffa. Cryptology ePrint Archive, Paper 2010/589, 2010. https://eprint.iacr.org/2010/589.

[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.

[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.

[BFP12] Luk Bettale, Jean-Charles Faugère, and Ludovic Perret. Solving polynomial systems over finite fields: improved analysis of the hybrid approach. In Joris van der Hoeven and Mark van Hoeij, editors, *International Symposium on Symbolic and Algebraic Computation, ISSAC'12, Grenoble, France - July 22 - 25, 2012*, pages 67–74. ACM, 2012.

[BFS13] Magali Bardet, Jean-Charles Faugère, and Bruno Salvy. On the complexity of the f5 gröbner basis algorithm, 2013.

[CG21] Alessio Caminata and Elisa Gorla. Solving degree, last fall degree, and related invariants. *CoRR*, abs/2112.05579, 2021.

[CLO07] David Cox, John Little, and Donal O'Shea. Ideals, varieties, and algorithms. an introduction to computational algebraic geometry and commutative algebra. 2007.

[CZ81] David G. Cantor and Hans Zassenhaus. A new algorithm for factoring polynomials over finite fields. *Mathematics of Computation*, 36(154):587–592, 1981.

[DS13] Jintai Ding and Dieter Schmidt. *Solving Degree and Degree of Regularity for Polynomial Systems over a Finite Fields*, pages 34–49. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[FGLM93] J.C. Faugère, P. Gianni, D. Lazard, and T. Mora. Efficient computation of zero-dimensional gröbner bases by change of ordering. *Journal of Symbolic Computation*, 16(4):329–344, 1993.

[FM17] Jean-Charles Faugère and Chenqi Mou. Sparse fglm algorithms. *Journal of Symbolic Computation*, 80:538–569, 2017.

[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.

[GLR+20] Lorenzo Grassi, Reinhard Lüftenegger, Christian Rechberger, Dragos Rotaru, and Markus Schofnegger. On a generalization of substitution-permutation networks: The HADES design strategy. In Anne Canteaut and Yuval Ishai, editors, *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*, volume 12106 of *Lecture Notes in Computer Science*, pages 674–704. Springer, 2020.

[HSK12] Ryuichi Harasawa, Yutaka Sueyoshi, and Aichi Kudo. Improving the berlekamp algorithm for binomials xn - a. In Ferruh Özbudak and Francisco Rodríguez-Henríquez, editors, *Arithmetic of Finite Fields*, pages 225–235, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[JK97] Thomas Jakobsen and Lars R. Knudsen. The interpolation attack on block ciphers. In Eli Biham, editor, *Fast Software Encryption*, pages 28–40, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[Knu95] Lars R. Knudsen. Truncated and higher order differentials. In Bart Preneel, editor, *Fast Software Encryption*, pages 196–211, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.

[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.

[Mac02] F. S. MacAulay. Some formulæ in elimination. *Proceedings of the London Mathematical Society*, s1-35(1):3–27, 1902.

[Sho93] Victor Shoup. Factoring polynomials over finite fields: Asymptotic complexity vs. reality. 1993.

[Wil14] Virginia Williams. Breaking the coppersmith-winograd barrier. 09 2014.

## A Experiments data

In this section, the raw data obtained from the experiments are presented. In the Table 2, the data for the modeling described in Section 4.2.1 is described and in the Table 3, the data for the modeling described in Section 4.2.2 is described.

Table 2: The result of computing the Gröbner basis for the polynomial system described in Section 4.2.1

| $t$ | $r$ | $(R_F, R_P)$ | $n$ | $d_{sol}$ | $d_I$ | $GB_{LEX}$ degree | time(s) | mem(GB) |
|---|---|---|---|---|---|---|---|---|
| 2 | 1 | (2, 0) | 3 | 4 | $3^2$ | 3 | 1 | 0.14 |
| | | (4, 0) | 7 | 5 | $3^4$ | $3^3$ | 1 | 0.14 |
| | | (6, 0) | 11 | 6 | $3^6$ | $3^5$ | 1 | 0.14 |
| | | (8, 0) | 15 | 7 | $3^8$ | $3^7$ | 80 | 1.4 |
| | | (10, 0) | 19 | 8 | $3^{10}$ | $3^9$ | 152371 | 66.2 |
| | | (2, 1) | 4 | 5 | $3^3$ | $3^2$ | 1 | 0.14 |
| | | (2, 2) | 5 | 5 | $3^4$ | $3^3$ | 1 | 0.14 |
| | | (2, 3) | 6 | 6 | $3^5$ | $3^4$ | 1 | 0.14 |
| | | (2, 4) | 7 | 7 | $3^6$ | $3^5$ | 1 | 0.14 |
| | | (2, 5) | 8 | 8 | $3^7$ | $3^6$ | 5.3 | 0.63 |
| | | (2, 6) | 9 | 9 | $3^8$ | $3^7$ | 65 | 0.91 |
| | | (2, 7) | 10 | 10 | $3^9$ | $3^8$ | 937 | 3.3 |
| | | (2, 8) | 11 | 11 | $3^{10}$ | $3^9$ | 31550 | 25.2 |
| | | (2, 9) | 12 | 12 | $3^{11}$ | $3^{10}$ | 76714 | 96.2 |
| | | (4, 1) | 8 | 6 | $3^5$ | $3^4$ | 1 | 0.14 |
| | | (4, 2) | 9 | 7 | $3^6$ | $3^5$ | 7.6 | 0.62 |
| | | (4, 3) | 10 | 7 | $3^7$ | $3^6$ | 7.3 | 0.68 |
| | | (4, 4) | 11 | 8 | $3^8$ | $3^7$ | 90 | 1.2 |
| | | (4, 5) | 12 | 9 | $3^9$ | $3^8$ | 2042 | 6 |
| | | (4, 6) | 13 | 10 | $3^{10}$ | $3^9$ | 103376 | 48 |
| | | (6, 1) | 12 | 6 | $3^7$ | $3^6$ | 4 | 0.65 |
| | | (6, 2) | 13 | 7 | $3^8$ | $3^7$ | 55 | 0.9 |
| | | (6, 3) | 14 | 8 | $3^9$ | $3^8$ | 3649 | 7 |
| | | (6, 4) | 15 | 9 | $3^{10}$ | $3^9$ | 161080 | 66 |
| | | (6, 5) | 16 | 9 | $3^{11}$ | $3^{10}$ | 115239 | 72 |
| | | (6, 6) | 17 | 10 | $3^{12}$ | $3^{11}$ | 54981 | 109 |
| | | (8, 1) | 16 | 7 | $3^9$ | $3^8$ | 3239 | 7 |
| | | (8, 2) | 17 | 8 | $3^{10}$ | $3^9$ | 174948 | 66 |
| | | (8, 3) | 18 | 9 | $3^{11}$ | $3^{10}$ | 310977 | 159 |
| | | (10, 2) | 21 | 8 | $3^{12}$ | $3^{11}$ | 162441 | 206 |
| 3 | 1 | (2, 0) | 4 | 3 | $3^2$ | 3 | 1 | 0.14 |
| | | (4, 0) | 10 | 4 | $3^4$ | $3^3$ | 3.3 | 0.61 |
| | | (6, 0) | 16 | 5 | $3^6$ | $3^5$ | 86 | 1.34 |
| | | (8, 0) | 22 | 5 | $3^8$ | $3^7$ | 55759 | 67.9 |
| | | (2, 1) | 5 | 4 | $3^3$ | $3^2$ | 1 | 0.14 |
| | | (2, 2) | 6 | 5 | $3^4$ | $3^3$ | 1 | 0.14 |
| | | (2, 3) | 7 | 6 | $3^5$ | $3^4$ | 1 | 0.17 |
| | | (2, 4) | 8 | 6 | $3^6$ | $3^5$ | 1 | 0.17 |
| | | (2, 5) | 9 | 7 | $3^7$ | $3^6$ | 3.94 | 0.93 |
| | | (2, 6) | 10 | 8 | $3^8$ | $3^7$ | 112 | 7.16 |
| | | (2, 7) | 11 | 9 | $3^9$ | $3^8$ | 3655 | 67.21 |
| | | (4, 1) | 11 | 8 | $3^5$ | $3^4$ | 1 | 0.17 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | (6, 1) | 17 | 8 | $3^7$ | $3^6$ | 11 | 0.82 |
| | | (2, 0) | 5 | 5 | $3^4$ | $3^2$ | 1 | 0.14 |
| | | (4, 0) | 11 | 7 | $3^8$ | $3^6$ | 6.73 | 0.64 |
| | | (6, 0) | 17 | x | $3^8$ | $3^6$ | x | x |
| | | (2, 1) | 6 | 5 | $3^5$ | $3^3$ | 1 | 0.14 |
| 3 | 2 | (2, 2) | 7 | 6 | $3^6$ | $3^4$ | 1 | 0.16 |
| | | (2, 3) | 8 | 7 | $3^7$ | $3^5$ | 1 | 0.21 |
| | | (2, 4) | 9 | 8 | $3^8$ | $3^6$ | 1.98 | 0.65 |
| | | (2, 5) | 10 | 9 | $3^9$ | $3^7$ | 43.67 | 4.5 |
| | | (4, 1) | 17 | 8 | $3^9$ | $3^7$ | 53 | 1.17 |
| | | (2, 0) | 5 | 3 | $3^2$ | 3 | 3.2 | 0.61 |
| | | (4, 0) | 13 | 4 | $3^4$ | $3^3$ | 2.2 | 0.61 |
| | | (6, 0) | 21 | 5 | $3^6$ | $3^5$ | 18.4 | 0.63 |
| | | (8, 0) | 29 | 5 | $3^8$ | $3^7$ | 904 | 3.9 |
| | | (2, 1) | 6 | 4 | $3^3$ | $3^2$ | 1 | 0.14 |
| | | (2, 2) | 7 | 4 | $3^4$ | $3^3$ | 1 | 0.14 |
| | | (2, 3) | 8 | 5 | $3^5$ | $3^4$ | 1 | 0.17 |
| 4 | 1 | (2, 4) | 9 | 6 | $3^6$ | $3^5$ | 1 | 0.29 |
| | | (2, 5) | 10 | 7 | $3^7$ | $3^6$ | 5.21 | 1.65 |
| | | (2, 6) | 11 | 7 | $3^8$ | $3^7$ | 122.28 | 13.32 |
| | | (4, 1) | 14 | 5 | $3^5$ | $3^4$ | 1 | 0.17 |
| | | (4, 2) | 15 | 5 | $3^6$ | $3^5$ | 3.26 | 0.63 |
| | | (6, 1) | 22 | 5 | $3^7$ | $3^6$ | 25.4 | 86 |
| | | (6, 2) | 23 | 6 | $3^8$ | $3^7$ | 419 | 3.07 |
| | | (8, 2) | 31 | 6 | $3^{10}$ | $3^9$ | 155505 | 288.24 |
| | | (2, 0) | 6 | 4 | $3^4$ | $3^2$ | 1.6 | 0.14 |
| | | (4, 0) | 14 | 6 | $3^8$ | $3^6$ | 30 | 2.5 |
| | | (6, 0) | 22 | 8 | $3^{12}$ | $3^{10}$ | 84953 | 119 |
| | | (2, 1) | 7 | 5 | $3^5$ | $3^3$ | 1 | 0.14 |
| 4 | 2 | (2, 2) | 8 | 6 | $3^6$ | $3^4$ | 1 | 0.17 |
| | | (2, 3) | 9 | 6 | $3^7$ | $3^5$ | 1 | 0.27 |
| | | (2, 4) | 10 | 7 | $3^8$ | $3^6$ | 3.59 | 0.99 |
| | | (2, 5) | 11 | 8 | $3^9$ | $3^7$ | 93.41 | 7.77 |
| | | (4, 1) | 15 | 7 | $3^9$ | $3^8$ | 83 | 1.6 |
| | | (2, 0) | 6 | 3 | $3^2$ | 3 | 1.6 | 0.14 |
| | | (4, 0) | 16 | 4 | $3^4$ | $3^3$ | 1.04 | 0.14 |
| | | (6, 0) | 36 | 5 | $3^6$ | $3^5$ | 3 | 0.6 |
| | | (8, 0) | 36 | 5 | $3^8$ | $3^7$ | 933 | 3.7 |
| | | (2, 1) | 7 | 4 | $3^3$ | $3^2$ | 1 | 0.14 |
| 5 | 1 | (2, 2) | 8 | 4 | $3^4$ | $3^3$ | 1 | 0.14 |
| | | (2, 3) | 9 | 5 | $3^5$ | $3^4$ | 1 | 0.17 |
| | | (2, 4) | 10 | 6 | $3^6$ | $3^5$ | 1 | 0.43 |
| | | (2, 5) | 11 | 6 | $3^7$ | $3^6$ | 11 | 1.96 |
| | | (2, 6) | 12 | 7 | $3^8$ | $3^7$ | 219 | 13.8 |
| | | (4, 1) | 17 | 4 | $3^5$ | $3^4$ | 6.7 | 0.62 |
| | | (6, 1) | 27 | 5 | $3^7$ | $3^6$ | 170 | 0.89 |
| | | (2, 0) | 7 | 4 | $3^4$ | $3^2$ | 1.11 | 0.14 |
| 5 | 2 | (4, 0) | 17 | 6 | $3^8$ | $3^6$ | 67 | 6.6 |
| | | (2, 1) | 8 | 4 | $3^5$ | $3^3$ | 1.11 | 0.14 |
| | | (2, 0) | 8 | 5 | $3^6$ | $3^3$ | 1.11 | 0.14 |
| 5 | 3 | (4, 0) | 18 | x | $3^{12}$ | $3^9$ | 1.04 | 0.14 |
| | | (2, 1) | 9 | 6 | $3^7$ | $3^4$ | 1.11 | 0.14 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 4 | (2, 0) | 9 | 6 | $3^8$ | $3^4$ | 82 | 0.14 |
| | | (4, 0) | 19 | 10 | $3^{12}$ | $3^8$ | 50877 | 79.85 |
| | | (2, 1) | 10 | 7 | $3^9$ | $3^5$ | 5.68 | 0.42 |
| | | (4, 1) | 20 | 10 | $3^{13}$ | $3^9$ | x | x |

Table 3: The result of computing the Gröbner basis for the polynomial system described in Section 4.2.1

| $t$ | $r$ | $(R_F, R_P)$ | $n$ | $d_{sol}$ | $d_I$ | $GB_{LEX}$ degree | time(ms) | mem(GB) |
|---|---|---|---|---|---|---|---|---|
| 2 | 1 | (2, 0) | 2 | Already GB | $3^2$ | 3 | 1 | 0.14 |
| | | (4, 0) | 6 | 6 | $3^4$ | $3^3$ | 1 | 0.14 |
| | | (6, 0) | 10 | 7 | $3^6$ | $3^5$ | 1 | 0.61 |
| | | (8, 0) | 14 | 8 | $3^8$ | $3^7$ | 84 | 1.032 |
| | | (10, 0) | 18 | 8 | $3^{10}$ | $3^9$ | 31813 | 37.78 |
| | | (2, 1) | 3 | 6 | $3^3$ | $3^2$ | 1 | 0.14 |
| | | (2, 2) | 4 | 8 | $3^4$ | $3^3$ | 1 | 0.14 |
| | | (2, 3) | 5 | 10 | $3^5$ | $3^4$ | 1 | 0.14 |
| | | (2, 4) | 6 | 12 | $3^6$ | $3^5$ | 1 | 0.14 |
| | | (2, 5) | 7 | 14 | $3^7$ | $3^6$ | 1 | 0.28 |
| | | (2, 6) | 8 | 16 | $3^8$ | $3^7$ | 5.66 | 0.67 |
| | | (2, 7) | 9 | 18 | $3^9$ | $3^8$ | 198 | 1.2 |
| | | (2, 8) | 10 | 20 | $3^{10}$ | $3^9$ | 2725 | 5.61 |
| | | (2, 9) | 11 | 22 | $3^{11}$ | $3^{10}$ | 62139 | 44.87 |
| | | (4, 1) | 7 | 7 | $3^5$ | $3^4$ | 1 | 0.14 |
| | | (4, 2) | 8 | 8 | $3^6$ | $3^5$ | 1 | 0.14 |
| | | (4, 3) | 9 | 9 | $3^7$ | $3^6$ | 1 | 0.21 |
| | | (4, 4) | 10 | 10 | $3^8$ | $3^7$ | 16 | 0.82 |
| | | (4, 5) | 11 | 10 | $3^9$ | $3^8$ | 1494 | 2.65 |
| | | (4, 6) | 12 | 11 | $3^{10}$ | $3^9$ | 33642 | 17.94 |
| | | (6, 1) | 11 | 7 | $3^7$ | $3^6$ | 20 | 0.62 |
| | | (6, 2) | 12 | 8 | $3^8$ | $3^7$ | 96 | 0.96 |
| | | (6, 3) | 13 | 9 | $3^9$ | $3^8$ | 605 | 3.93 |
| | | (6, 4) | 14 | 10 | $3^{10}$ | $3^9$ | 24342 | 32 |
| | | (8, 1) | 15 | 8 | $3^9$ | $3^8$ | 1159 | 4 |
| | | (8, 2) | 16 | 9 | $3^{10}$ | $3^9$ | 38418 | 36 |
| 3 | 1 | (2, 0) | 2 | Already GB | $3^2$ | 3 | 1 | 0.14 |
| | | (4, 0) | 8 | 5 | $3^4$ | $3^3$ | 1 | 0.15 |
| | | (6, 0) | 14 | 6 | $3^6$ | $3^5$ | 1 | 0.15 |
| | | (8, 0) | 20 | 7 | $3^8$ | $3^7$ | 282 | 1.34 |
| | | (10, 0) | 26 | 8 | $3^8$ | $3^7$ | 193678 | 69.1 |
| | | (2, 1) | 3 | 6 | $3^3$ | $3^2$ | 1 | 0.14 |
| | | (2, 2) | 4 | 8 | $3^5$ | $3^4$ | 1 | 0.14 |
| | | (2, 3) | 5 | 10 | $3^7$ | $3^6$ | 1 | 0.14 |
| | | (2, 4) | 6 | 12 | $3^8$ | $3^7$ | 1 | 0.14 |
| | | (2, 5) | 7 | 14 | $3^9$ | $3^8$ | 1 | 0.14 |
| | | (2, 6) | 8 | 16 | $3^{10}$ | $3^9$ | 10 | 0.71 |
| | | (2, 7) | 9 | 18 | $3^{11}$ | $3^{10}$ | 253 | 1.49 |
| | | (2, 8) | 10 | 20 | $3^{12}$ | $3^{11}$ | 6912 | 8.3 |
| | | (2, 9) | 11 | 22 | $3^{13}$ | $3^{13}$ | 102725 | 69.43 |
| 3 | 2 | (2, 0) | 4 | 6 | $3^4$ | $3^2$ | 1 | 0.14 |
| | | (4, 0) | 10 | 9 | $3^8$ | $3^6$ | 2.4 | 0.62 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | (2, 1) | 5 | 8 | $3^5$ | $3^3$ | 1 | 0.14 |
| | | (2, 2) | 6 | 10 | $3^6$ | $3^5$ | 1 | 0.14 |
| | | (2, 3) | 7 | 12 | $3^7$ | $3^6$ | 1 | 0.14 |
| | | (2, 4) | 8 | 14 | $3^8$ | $3^7$ | 1 | 0.14 |
| | | (2, 5) | 9 | 16 | $3^9$ | $3^8$ | 12 | 0.69 |
| | | (2, 6) | 10 | 18 | $3^{10}$ | $3^9$ | 280 | 1.36 |
| | | (2, 7) | 11 | 20 | $3^{11}$ | $3^{10}$ | 280 | 1.36 |
| | | (2, 8) | 12 | 22 | $3^{12}$ | $3^{11}$ | 280 | 1.36 |
| | | (4, 1) | 11 | 9 | $3^9$ | $3^7$ | 28 | 0.8 |
| | | (6, 1) | 17 | 10 | $3^9$ | $3^7$ | 147881 | 116 |
| 4 | 1 | (2, 0) | 2 | Already GB | $3^2$ | 3 | 1 | 0.14 |
| | | (4, 0) | 10 | 5 | $3^4$ | $3^3$ | 1 | 0.24 |
| | | (6, 0) | 18 | 6 | $3^6$ | $3^5$ | 1 | 0.16 |
| | | (8, 0) | 26 | 6 | $3^8$ | $3^7$ | 106 | 1.7 |
| 4 | 2 | (2, 0) | 4 | 6 | $3^4$ | $3^2$ | 1.6 | 0.14 |
| | | (4, 0) | 12 | 8 | $3^8$ | $3^6$ | 2.27 | 0.62 |
| | | (2, 1) | 5 | 8 | $3^5$ | $3^3$ | 1 | 0.14 |
| | | (4, 1) | 13 | 9 | $3^9$ | $3^7$ | 1 | 0.14 |
| 4 | 3 | (2, 0) | 6 | 8 | $3^6$ | $3^3$ | 1.6 | 0.14 |
| | | (4, 0) | 14 | 11 | $3^{12}$ | $3^9$ | 18223 | 13.5 |
| | | (2, 1) | 7 | 10 | $3^7$ | $3^4$ | 1 | 0.14 |
| 5 | 1 | (2, 0) | 2 | Already GB | $3^2$ | 3 | 1.6 | 0.14 |
| | | (4, 0) | 12 | 5 | $3^4$ | $3^3$ | 1.04 | 0.2 |
| | | (6, 0) | 22 | 6 | $3^6$ | $3^5$ | 1.05 | 0.2 |
| | | (8, 0) | 32 | 6 | $3^8$ | $3^7$ | 94 | 1.83 |
| | | (10, 0) | 42 | 7 | $3^{10}$ | $3^9$ | 60714 | 130 |
| 5 | 2 | (2, 0) | 4 | 6 | $3^4$ | $3^2$ | 1 | 0.14 |
| | | (4, 0) | 14 | 8 | $3^8$ | $3^6$ | 4 | 0.61 |
| 5 | 4 | (2, 0) | 8 | 10 | $3^8$ | $3^4$ | 1.11 | 0.24 |