An analysis of the feasibility of a Gröbner basis attack on the Poseidon hash function

Buschman, Thomas

Award date:
2022

Eindhoven University of Technology
Department of Mathematics and Computer Science

An analysis of the feasibility of a Gröbner basis attack on the Poseidon hash function

November 22, 2022

Thomas Buschman
1458396

Supervisors:
Tomer Ashur and Mohammad Mahzoun

Abstract

With the introduction of advanced cryptographic protocols, such as multi-party computation, and zero-knowledge proofs, alternatives need to be found to traditional hash functions like SHA, as these have been optimised for different metrics. Where traditional hash functions were optimised for memory usage and processing time, advanced cryptographic protocols require hash functions that have been optimized with respect to a new metric, arithmetic complexity.

Recently, one such hash function was proposed and analysed by Grassi *et al.* In this thesis, I explore some of the error in their analysis of Gröbner basis attacks. First, I show that the security argument was constructed sloppily, using faulty logic. Then, I perform experiments to empirically test one key assumption. Ultimately, I find this key assumption to be incorrect, and conclude that the Poseidon permutation does not provide as much security against Gröbner basis attacks as Grassi *et al.* claim.

Contents

1 Introduction

Hash functions have been used in cryptography for over five decades for encrypting and securing information. In that period, many different hash functions have been proposed and rigorously analysed. The hash functions currently used, such as SHA2 have been optimised for memory requirements and running time, but they are often inefficient for new, advanced cryptographic protocols, such as multi-party computation, or zero-knowledge proofs [1].

To cater to the ever-evolving needs of these advanced cryptographic protocols, several design strategies, with new security arguments have been proposed, such as the *Marvellous* design strategy [1], and the Hades design strategy [2]. In this thesis, I will investigate the security claims of the Hades-based Poseidon permutation.

This thesis is structured as follows: in Section 2, I recall the definitions and notions that will be needed for the rest of the thesis, and their prerequisites. In Section 2, I also explain the type of attack that I will be performing, after which I cover the basics of Poseidon, and its security argument in Section 3. Afterwards, I uncover numerous fundamental flaws in the Poseidon security argument in Section 4. In Section 5 I mention 2 trivial oversights in the Poseidon paper. Finally, I give a short conclusion in Section 6. In Appendix A, I give a short summary of the practical difficulties involving Gröbner basis computations. Thereafter, I systematically explore a subset of the possible ways to model the Poseidon permutation in Appendix B.

2 Preliminaries

Before we can properly communicate about any of the issues in the Poseidon paper, we need to agree on the definitions of concepts, and the notation used for them. First, the algebraic concepts that are relevant for our discussion are defined, and then some of the relevant cryptographic applications of permutations like Poseidon are explained. Finally, the specifics of a Gröbner basis attack are explained.

2.1 Algebra

Let us start by refreshing the notion of a set:

Definition 1 (Set, Folklore). A **set** S is any collection of objects, without considering an order on those objects, or the multiplicity of those objects. An object o is either in a set S, denoted as $o \in S$, or it is not in the set $o \notin S$. A set is uniquely defined by its elements.

Now, we can continue with the notions of rings and fields, which we will need to understand what an 'Ideal' is.

Definition 2 (Commutative ring, [3]; App. A.1.2). A **commutative ring** consists of a set R and two binary operations $" + "$ and $" \cdot "$ defined over R for which the following conditions are satisfied $\forall a, b, c \in R$:

1. $(a + b) + c = a + (b + c)$ and $a \cdot (b \cdot c) = (a \cdot b) \cdot c$ (associativity)
2. $a + b = b + a$ and $a \cdot b = b \cdot a$ (commutativity)
3. $a \cdot (b + c) = a \cdot b + a \cdot c$ (distributivity)
4. There are $0, 1 \in R$ such that $a + 0 = a \cdot 1 = a$ (identities)
5. Given $a \in R$ there exists $a_- \in R$ such that $a + a_- = 0$ (additive inverses)

Definition 3 (Field, [3]; App. A.1.1). A **field** consists of a set $\mathbb{F}$ and two binary operations $" + "$ and $" \cdot "$ defined over $\mathbb{F}$ for which the following conditions are satisfied $\forall a, b, c \in \mathbb{F}$:

1. $(a + b) + c = a + (b + c)$ and $a \cdot (b \cdot c) = (a \cdot b) \cdot c$ (associativity)
2. $a + b = b + a$ and $a \cdot b = b \cdot a$ (commutativity)
3. $a \cdot (b + c) = a \cdot b + a \cdot c$ (distributivity)
4. There are $0, 1 \in \mathbb{F}$ such that $a + 0 = a \cdot 1 = a$ (identities)
5. Given $a \in \mathbb{F}$ there exists $a_- \in \mathbb{F}$ such that $a + a_- = 0$ (additive inverses)
6. given $a \in \mathbb{F}, a \neq 0$ there exists $a^{-1} \in \mathbb{F}$ such that $a \cdot a^{-1} = 1$ (multiplicative inverses)

For the standard notions of $" + "$ and $" \cdot "$, sets like $\mathbb{Q}, \mathbb{R}$, and $\mathbb{C}$ are fields but also $\mathbb{Z}/p\mathbb{Z}$, i.e. the integers modulo p for any prime number p, which is often denoted as $\mathbb{F}_p$. Also note that all fields are commutative rings, but there are commutative rings that are not fields, such as $\mathbb{Z}$, or $\mathbb{Z}/n\mathbb{Z}$ with n a non-prime integer, using the standard notions for $" + "$ and $" \cdot "$.

Ultimately, the goal of a Gröbner basis attack is to find a set of solutions for a system of polynomials. To do that, the system of polynomials is treated as an ideal:

Definition 4 (Ideal, [3]; Chp. 1.4.1). Let R be a commutative ring. A subset $I \subseteq R$ is an **ideal** if it satisfies:

1. $0 \in I$
2. $a, b \in I \implies a + b \in I$
3. $(a \in I \wedge r \in R) \implies a \cdot r \in I$

Before we can continue to the definition of a Gröbner basis, we need to make a small detour to define polynomials and some of their key properties:

Definition 5 (Monomial, [3]; Chp. 1.1.1). A **monomial** in the variables $x_1, \ldots, x_n$ is a product of the form

$$x_1^{\alpha_1} \cdot x_2^{\alpha_2} \ldots x_n{}^{\cdot \alpha_n},$$

where the α_i are non-negative integers. The total degree of this monomial is $\sum_{i=1}^{n} \alpha_i$ We can use a more compact notation by letting $\alpha = (\alpha_1, \ldots \alpha_n) \in \mathbb{Z}_{\geq 0}^n$ be an n-tuple of non-negative integers. Then, the monomial x^α is given by

$$x^\alpha = x_1^{\alpha_1} \cdot x_2^{\alpha_2} \ldots x_n{}^{\cdot \alpha_n},$$

and the total degree is written as

$$|\alpha| = \sum_{i=1}^{n} \alpha_i.$$

Definition 6 (Polynomial, [3]; Chp. 1.1.2). A **polynomial** f in the variables $x_1, \ldots, x_n$ over the field $\mathbb{F}$ is a finite linear combination of monomials with coefficients in $\mathbb{F}$. Such a polynomial can be written as

$$f = \sum_{\alpha} a_\alpha x^\alpha, \qquad a_\alpha \in k$$

Definition 7 (Polynomial ring, [3]; p. 1). The **commutative polynomial ring in variables $\mathbf{x_1}, \ldots, \mathbf{x_n}$** over the field $\mathbb{F}$ is the set of all polynomials in $x_1, \ldots x_n$ with coefficients in $\mathbb{F}$, and it is denoted as $\mathbb{F}[x_1, \ldots, x_n]$.

Definition 8 (Polynomial ideal, [3]; 1.4.2). Let $\mathbb{F}$ be a field, $f_1, \ldots, f_s$ be polynomials in the ring $\mathbb{F}[x_1, \ldots, x_n]$. Define

$$\langle f_1, \ldots, f_s \rangle := \left\{ \sum_{i=1}^{s} h_i f_i \mid h_1, \ldots, h_s \in \mathbb{F}[x_1, \ldots, x_n] \right\}.$$

$\langle f_1, \ldots, f_s \rangle$ is the **ideal** generated by the polynomials $f_1, \ldots, f_s$

To define a Gröbner basis of an ideal, we need to determine which monomials are larger than others. For that, we define the notion of an ordering on monomials:

Definition 9 (Total ordering, [3]; p. 55). A **total ordering** is a binary relation $>$ on some set X which satisfies $\forall a, b, c \in X$:

- $(a > b \wedge b > c) \implies a > c$
- exactly one of $a > b$, $a = b$, and $b > a$ is true

Definition 10 (Well-ordering, [3]; Chp. 2.2.1.iii). A binary relation $>$ on some set X is a **well-ordering** if every nonempty subset of X has a smallest element under $>$

Definition 11 (monomial ordering, [3]; Chp. 2.2.1). A **monomial ordering** $>$ on some polynomial ring $\mathbb{F}[x_1, \ldots, x_n]$ is a relation on the set of monomials $x^\alpha, \alpha \in \mathbb{Z}^n_{\geq 0}$, or equivalently, a relation $>$ on $\mathbb{Z}^n_{\geq 0}$ satisfying:

- $>$ is a total ordering on $\mathbb{Z}^n_{\geq 0}$.
- If $\alpha > \beta$ and $\gamma \in \mathbb{Z}^n_{\geq 0}$ then $\alpha + \gamma > \beta + \gamma$
- $>$ is a well-ordering on $\mathbb{Z}^n_{\geq 0}$

Given some monomial ordering $>$, we say that $\alpha \geq \beta$ if either $\alpha > \beta$ or $\alpha = \beta$.
Implicitly, it is assumed that for single-degree monomials, $x_1, \ldots x_n$ we have

$$x_1 > x_2 > \ldots > x_n$$

The two orderings that are important in this report are *lex* and *grevlex*:

Definition 12 (Lexicographic, or Lex order, [3]; Chp. 2.2.3). Let $\alpha = (\alpha_1, \ldots, \alpha_n), \beta = (\beta_1, \ldots, \beta_n) \in \mathbb{Z}^n_{\geq 0}$. We say $\alpha >_{\text{lex}} \beta$ if the leftmost nonzero entry of $\alpha - \beta \in \mathbb{Z}^n$ is positive, and say $x^\alpha >_{\text{lex}} x^\beta$ if $\alpha >_{\text{lex}} \beta$.

Definition 13 (Graded reverse Lex, Grevlex, or Degrevlex order, [3]; Chp. 2.2.6). Let $\alpha, \beta \in \mathbb{Z}^n_{\geq 0}$. We say $\alpha >_{\text{grevlex}} \beta$ if $|\alpha| = \sum_{i=1}^n \alpha_i > |\beta| = \sum_{i=1}^n \beta_i$, or if $|\alpha| = |\beta|$ and the rightmost nonzero entry of $\alpha - \beta$ is negative

Definition 14 (Multidegree, LC, LM, and LT [3]; Chp. 2.2.7). Let $f = \sum_\alpha a_\alpha x^\alpha$ be a nonzero polynomial in the polynomial ring $\mathbb{F}[x_1, \ldots, x_n]$ and let $>$ be a monomial order.

- The **Multidegree** of f is

  $$\text{multideg}(f) = \max(\alpha \in \mathbb{Z}^n_{\geq 0} \mid a_\alpha \neq 0)$$

  (where the maximum is taken with respect to $>$).

- The **leading coefficient** of f is

  $$\text{LC}(f) = a_{\text{multideg}(f)} \in \mathbb{F}.$$

- The **leading monomial** of f is

  $$\text{LM}(f) = x^{\text{multideg}(f)}$$

  (with coefficient 1).

- The **leading term** of f is

  $$\text{LT}(f) = \text{LC}(f) \cdot \text{LM}(f).$$

Now, we only need to introduce the notion of a **leading term ideal**, which is a **monomial ideal**, before we can define a Gröbner basis:

Definition 15 (Monomial ideal, [3]; Chp. 2.4.1). An ideal $I \subseteq \mathbb{F}[x_1, \ldots, x_n]$ is a **monomial ideal** if there is a subset $A \subseteq \mathbb{Z}^n_{\geq 0}$ such that I consists of all polynomials which are finite sums of the form $\sum_{\alpha \in A} h_\alpha x^\alpha$, where $h_\alpha \in \mathbb{F}[x_1, \ldots, x_n]$. In other words, I is the ideal generated by all monomials corresponding to elements in A. In this case, we write $= \langle x^\alpha \mid \alpha \in A \rangle$

Definition 16 (Leading term ideal, [3]; Chp. 2.5.1). Let $I \subseteq \mathbb{F}[x_1, \ldots, x_n]$ be a nonzero ideal, and fix a monomial ordering on $\mathbb{F}[x_1, \ldots, x_n]$. Then:

- We denote by $LT(I)$ the set of the leading terms of the nonzero elements of I, i.e.

$$\mathrm{LT}(I) = \{cx^\alpha \mid \exists f \in I \backslash \{0\} \text{ such that } \mathrm{LT}(f) = cx^\alpha\}.$$

- We denote by $\langle \mathrm{LT}(I) \rangle$ th **leading term ideal**, the ideal generated by the elements of $\mathrm{LT}(I)$.

Finally, we can define a Gröbner basis:

Definition 17 (Gröbner basis, [3]; Chp. 2.5.5). Fix a monomial order on the polynomial ring $\mathbb{F}[x_1, \ldots, x_n]$. A finite subset $G = \{g_1, \ldots, g_t\}$ of an ideal $I \subseteq \mathbb{F}[x_1, \ldots, x_n]$ different from $\{0\}$ is said to be a **Gröbner basis** for I if

$$\langle \mathrm{LT}(g_1), \ldots \mathrm{LT}(g_t) \rangle = \langle \mathrm{LT}(I) \rangle.$$

For a system of polynomials, computing the Gröbner basis is in general computationally difficult. One class of tools that can be used for computing Gröbner bases are Macaulay matrices:

Definition 18 (Macaulay matrix, [4]; Sec. 2.3). Given a set of polynomials $f_1, \ldots, f_s \in \mathbb{F}[x_1, \ldots, x_n]$, that have degrees $d_1, \ldots, d_n$, the Macaulay matrix of degree d is the matrix containing the coefficients of all monomial multiples of the f_i such that the degree of the products is at most d. For example, the Macaulay matrix of degree 2 of the polynomials $f_1 = x^2 + y + 3$ and $f_2 = 3y - x + 1$ in the variables x and y, is given by

$$\begin{array}{c} \\ f_1 \\ f_2 \\ yf_2 \\ xf_2 \end{array} \begin{array}{c} \begin{array}{cccccc} x^2 & xy & y^2 & x & y & 1 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & -1 & 3 & 1 \\ 0 & -1 & 3 & 0 & 1 & 0 \\ -1 & 3 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array}.$$

2.2 Cryptography

Definition 19 (Permutation, Folklore). Let X be a set. A **permutation** $P : X \mapsto X$ of X is a bijection from X to itself.

Definition 20 (Hash function, [5]; Chp. 9.2.1, 9.2.2). A **hash function** h is a function satisfying:

- *compression:* h maps an input x of arbitrary finite bit length to an output $h(x)$ of fixed bit length n.
- *ease of computation* given h and x, $h(x)$ is easy to compute.

A cryptographic hash functions is a hash function satisfying the following three properties:

- *Preimage resistance:* for any output y, it is computationally infeasible to find an x such that $h(x) = y$.
- *Second preimage resistance:* for any input x_1, it is computationally infeasible to find an x_2 such that $h(x_1) = h(x_2)$.
- *Collision resistance:* It is computationally infeasible to find a pair x_1, x_2 such that $h(x_1) = h(x_2)$.

Definition 21 (Sets of bitstrings, [6]; Chp. 2.1.1).

- The set of all bitstrings is denoted by $\mathbb{Z}_2^*$
- The set of infinite-length bitstrings is denoted by $\mathbb{Z}_2^\infty$

Definition 22 (Sponge construction, [6]; Chp. 2.2). A **sponge construction** is a construction used in cryptography to create many different kinds of cryptographic primitives from some permutation.
The sponge construction builds a sponge function $S : \mathbb{Z}_2^* \mapsto \mathbb{Z}_2^\infty$, using a fixed-length permutation $f : \{0,1\}^b \mapsto \{0,1\}^b$, a padding rule, and a parameter r, which is called the bit rate. The sponge construction has a state of b bits. First, all the bits of the state are initialised to 0, and the input is padded and cut into blocks of r bits. Then, the construction proceeds in two phases: the absorbing phase, followed by the squeezing phase. In these phases, the first r bits of the state s are treated differently than the last $c = b - r$ bits of the state. The first r bits are called the outer part $\bar{s}$, and the last c bits are called the inner part $\hat{s}$. The length c of the inner part is called the capacity. The two phases are:

1. **The absorbing phase**:
   In the absorbing phase, the r-bit input message blocks are XORed into the outer part of the state, interleaved with applications of the function f to the entire state s. When all blocks have been processed, the construction switches to the squeezing phase.

2. **The squeezing phase**:
   The outer part of the state is iteratively returned as output blocks, interleaved with applications of the function f to the entire state s. The number of iterations is determined by the requested number of bits, l.

Finally, after the squeezing phase, the output is truncated to its first l bits.

The definition given here uses states of b bits, but it can be adapted to instead use states of m field elements from a finite field $\mathbb{F}_q$. In that case, the function $f : \mathbb{F}_q^m \mapsto \mathbb{F}_q^m$ is defined over $\mathbb{F}_q^m$, and the XOR addition in the absorbing step is replaced by addition in $\mathbb{F}_q$

An illustration of the workings of a sponge construction is shown in Figure 1. For further detail on sponges, the reader is referred to [6].

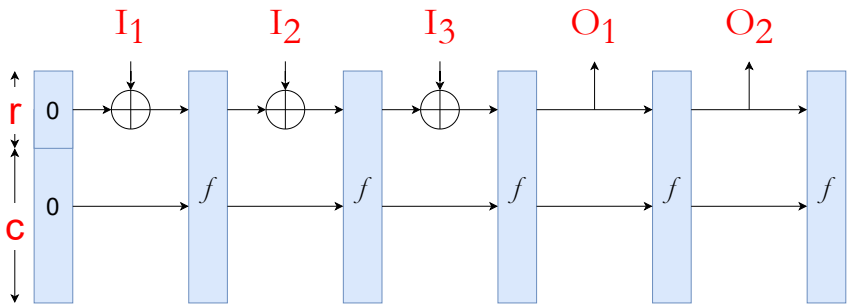

Figure 1: Illustration of a sponge construction. This example has an input that was padded into three blocks of r bits each, that were fed into the construction as I_1, I_2 and I_3. The requested number of bits for this example was $2r$, which was retrieved as O_1 and O_2

2.3 Gröbner basis Attacks

Within cryptography, there are many kinds of attacks to break cryptographic primitives, such as structural or statistical attacks. In this report, I will discuss the viability of Gröbner basis attacks on the sponge construction proposed by Grassi *et al.* In a Gröbner basis attack, the attacker tries to recover the secret (this can be the secret key in a keyed algorithm, or a secret input in a sponge construction) by setting up a system of equations linking the secret to the known quantities, and then using a Gröbner basis to solve that system of equations. Such an attack consists of three steps:

1. Computing the Gröbner basis of the system of equations in a *grevlex* order;
2. transforming the *grevlex* order Gröbner basis into a Gröbner basis in *lex* order;
3. solving the system of equations by computing the variety of this *lex* order Gröbner basis, which generally involves factorizing a single-variable polynomial, and back-substituting its roots.

These first two steps might seem like a convoluted way of finding the Gröbner basis in *lex* order, but it is generally agreed that for most systems of polynomials, finding a *grevlex* Gröbner basis and then transforming it to a *lex* order is much faster than directly finding a Gröbner basis in *lex* order.

In cryptography, security against a Gröbner basis attack is generally argued through the complexity of the first step, as pioneered in [1]. The complexity of finding a Gröbner basis in *grevlex* order is given by

$$C_{\text{GB}} = \mathcal{O}\left(\binom{\mathcal{V} + D_{\text{reg}}}{D_{\text{reg}}}^{\omega}\right), \tag{1}$$

where $\mathcal{V}$ is the number of variables used in the system of equations, and D_{reg} is the degree of regularity[1], which is the lowest degree d for which the diogonalisation of the Macaulay matrix yields a Gröbner basis. Finally, $2 \leq \omega \leq 3$ is the linear algebra constant. The complexity of transforming the Gröbner basis from a *grevlex* order to a *lex* order, using the FGLM algorithm is given by [7]

$$C_{\text{FGLM}} = \mathcal{O}\left(\mathcal{V}D^3\right), \tag{2}$$

with D the degree of the corresponding ideal. Finally, the last step, which factors the univariate polynomial is significantly less costly than the first two steps, and for a polynomial with degree D, it is given by

$$C_{\text{F}} = \mathcal{O}\left(D^{\omega}\right), \tag{3}$$

where w is, as before, the linear algebra constant.

[1] In [1], this degree of regularity is referred to as the 'concrete degree of regularity'.

3 POSEIDON hash function

3.1 Poseidon permutation

In [2], Grassi et al. introduced the POSEIDON hash function, POSEIDON : $\mathbb{F}_p^* \to \mathbb{F}_p^c$, which maps arbitrarily long strings over $\mathbb{F}_p$ to strings of fixed length c. This hash function builds upon the sponge construction, with the POSEIDON$^\pi$ permutation, POSEIDON$^\pi$: $\mathbb{F}_p^t \to \mathbb{F}_p^t$, instantiated with a fixed, known key. Here, p is a prime number. The permutation consists of a total of $2R_f + R_P$ rounds each consisting of three components. An illustration is shown in Figure 2. The three components in each round are:

1. *AddRoundConstants*, denoted by $ARC(\cdot)$;
2. *SubWords*, denoted by S-box$(\cdot)$, or S$(\cdot)$;
3. *MixLayer*, denoted by $M(\cdot)$.

In the ARC step, round constants are added to the state (which is a vector of length t, with entries in $\mathbb{F}_p$). In the S-box step, an element-wise power map $x \mapsto x^\alpha$ (with α equal to 3, 5 or -1) is applied to the state. In the first and last R_f rounds, this power map is applied to all entries of the state, and in the remaining R_P rounds, this power map is only applied to one entry of the state. In the MixLayer step, the state is multiplied with M, a $t \times t$ MDS matrix. For the details about the round constants and the MDS matrix, the reader is referred to [2].

3.2 Different types of Gröbner basis attacks

In their paper, Grassi *et al.* provide security arguments against several types of attacks, but in this thesis, we focus only on Gröbner basis attacks. Within this niche field of Gröbner basis attacks, they distinguish among two kinds of Gröbner basis attacks, and they discuss an optimisation to one of the attacks that is beyond the scope of this work. The two different kinds of Gröbner basis attacks are:

1. **Full-Permutation attack**:
   For this kind of attack, an attacker uses exactly one equation and one variable for each of the χ unknown field elements, where χ is the rate parameter of the sponge construction. This kind of attack has few variables, and few equations with high degree.
2. **Round-level attack**:
   For a **Round-level attack**, an attacker sets up variables and equations for each intermediate state of the $r = R_F + R_P$ rounds used in the permutation. This kind of attack has many variables, and equations with low degree.

In Section 4, I will show that the arguments given by Grassi *et al.* are often faulty, by closely examining the **Round-level attack**, which constitute the clearest example of the errors made by Grassi *et al.* Therefore, I will display the corresponding argument here [2].

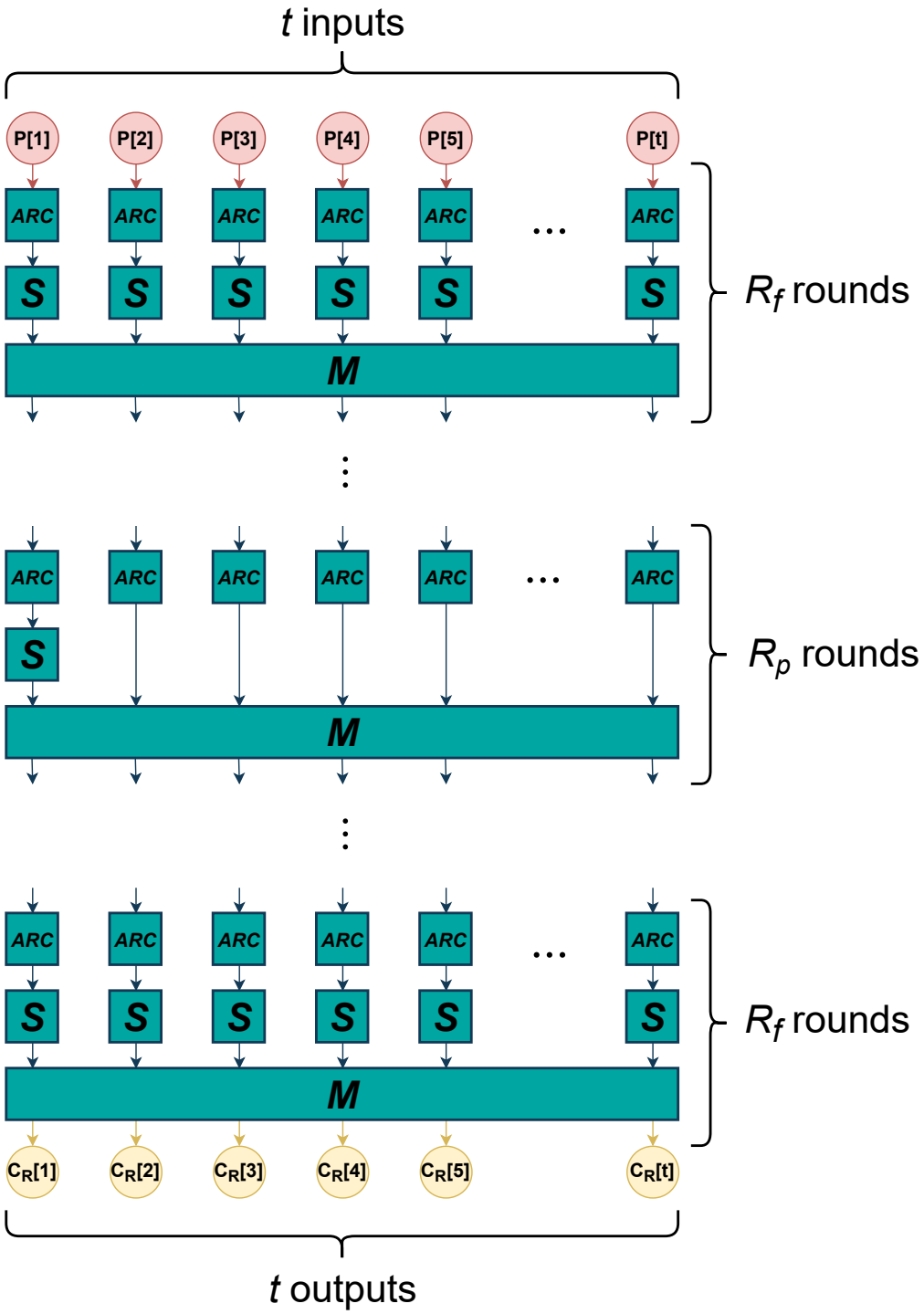

Figure 2: Illustration of the POSEIDON$^{\pi}$ permutation.

3.3 Round-level attack

For this type of Gröbner basis attack, on a sponge construction with rate parameter χ, Grassi *et al.* assume the attacker uses $q = (t-1)R_F + R_P + \chi$ variables, and a total of q equations, all with degree α. Then, they assume that the degree of regularity for the system of equations can be reasonably approximated by the Macaulay bound for this system. In general, the Macaulay bound for a system of k polynomials, each with degree d_i ($i \in \{1, \ldots, k\}$) is given by:

$$D_{\text{Mac}} = 1 + \sum_{i=1}^{k} (d_i - 1). \tag{4}$$

In this specific case, the degree of all q equations is α, such that Grassi *et al.* estimate the degree of regularity to be $D_{\text{reg}} \approx 1 + (\alpha - 1)q$. Then, using Equation (1), and Stirling's approximation, the complexity of the Gröbner basis attack, $\mathcal{C}_{\text{GB}}$, is estimated to be:

$$\begin{aligned}\mathcal{C}_{\text{GB}} &= \binom{\mathcal{V} + D_{\text{reg}}}{D_{\text{reg}}}^2 = \binom{q + 1 + (\alpha - 1)q}{q}^2 = \binom{\alpha q + 1}{q}^2 \approx \binom{\alpha q}{q}^2 = \left(\frac{(\alpha q)!}{q! \cdot (\alpha q - q)}\right)^2 \\ &\approx \left(\frac{\alpha^\alpha}{(\alpha - 1)^{\alpha - 1}}\right)^{2q} \cdot \left(\frac{2\pi(\alpha - 1)q}{\alpha}\right)^{-1} = 2^{2q \log_2\left(\frac{\alpha^\alpha}{(\alpha-1)^{\alpha-1}}\right) - \log_2\left(\frac{2\pi(\alpha-1)q}{\alpha}\right)} = 2^{C \cdot q - C'},\end{aligned}$$

with

$$C := 2 \log_2 \left(\frac{\alpha^\alpha}{(\alpha - 1)^{\alpha - 1}}\right), \qquad C' = \log_2 \left(\frac{2\pi(\alpha - 1)q}{\alpha}\right).$$

Now, the attack complexity, *i.e.* the resistance against this attack, is bounded with

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

Consequently, for a desired security level M Grassi *et al.* concluded that a Poseidon instantiation with R_F full rounds and R_P partial rounds can be attacked if:

$$\mathcal{C}_{\text{GB}} \approx 2^{C \cdot q} = 2^{C((t-1)R_F + R_P + \chi)} \leq 2^{\min\{M, \log_2(p)\chi\}}.$$

Here, the first argument of the minimum function is the desired security level and the second argument is the generic security of the sponge construction itself
which can be written as a constraint on the number of S-boxes:

$$(t - 1)R_F + R_P \leq C^{-1} \min\{M, \log_2(p)\chi\} - \chi. \tag{5}$$

Then, Grassi *et al.* argue that the number of rounds that can be attacked is maximised for $\chi = 1$. Subsequently, they add $t - 1$ extra S-boxes to thwart any attack using sub-spaces of insecure MDS matrices. Finally, they conjecture security if the number of S-boxes satisfies:

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

Flaws in this argument will be demonstrated in Section 4 and Section 5.

4 Improper logic

There are some trivial errors in the Poseidon argument that should be pointed out, which I will do in Section 5. First, I will show the fundamental flaws of the security argument given by Grassi *et al.* will. These will be separated into two different categories: Logic errors and Erroneous assumptions.

4.1 Argument soundness

The security argument demonstrated in Section 3.3 can be reduced to the following steps:

1. Find how many field operations are necessary to complete the attack, *i.e.* the attack resistance, as a function of the parameters $\alpha, R_F, R_P, t, \chi$;
2. Optionally, find an upper bound for this resistance that is easier to manipulate;
3. For a given security level M, find the highest values, R_F^*, R_P^* for the round parameters R_F, R_P that **can** be attacked;
4. Assume that all values for R_F, R_P higher than R_F^*, R_P^* **cannot** be attacked.

Individually, these steps can all be part of a sound security argument, but steps 2 and 4 **together** can lead to incorrect conclusions, especially if the upper bound used in step 2 is not tight. If this upper bound is not tight, the resistance to the attack is overestimated significantly in step 3, after which the values found for R_F^*, R_P^* are too small. Finally, step 4 fails, as there could be values for R_F, R_P in-between the incorrect R_F^*, R_P^*, and their correct values, R_F', R_P'.

Consider a simple example; for the sake of argument, assume there is an r-round permutation which is used in a sponge construction. We further assume that on this sponge function, only one attack is possible. After some sound reasoning, it is found that the resistance against this attack is $2^{3r\chi}$, with χ the rate parameter of the sponge construction. Now, imagine that we find this expression difficult to manipulate. Therefore, we note that $2^{3r\chi} \leq 2^{4r\chi}$. This is not a tight upper bound at all. Nonetheless, using the argumentation shown above, we find r^* from

$$2^{4r^*\chi} = 2^M.$$

Solving for r^* yields

$$r^* = \frac{M}{4\chi}.$$

Now, we have that for all $0 \leq r \leq r^*$, the sponge construction using the $r-$ round permutation can be attacked. This is still a true statement. Using the argumentation as above, it is then conjectured that the sponge construction is safe from attacks for all $r \geq r^* = \frac{M}{4\chi}$. This is **not** a true statement. We now use the proper expression to find the correct safety threshold, r_s. We solve

$$2^{3r_s\chi} = 2^M,$$

and find

$$r_s = \frac{M}{3\chi}.$$

Now, we know that for all $0 \leq r \leq r_s$, the sponge construction can be attacked, and for all $r > r_s$, the sponge construction is safe for the given security level. The problem is that for $r_a \leq r \leq r_s$, we argued that the sponge construction with r rounds is safe, while it very clearly is not. If we want to use the steps 3 and 4 that were lined out before, we better not use an

upper bound in step 2, as it leads us to overestimate the resistance of our sponge construction against attacks. If we must find a quantity that is easier to work with, we should find a Lower bound instead.

We compare this exaggerated example to the security argument outlined in Section 3.3. There, the resistance against a **round-level** Gröbner basis attack was found to be (up to reasonable approximation):

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

with

$$C := 2\log_2\left(\frac{\alpha^\alpha}{(\alpha-1)^{\alpha-1}}\right), \qquad C' = \log_2\left(\frac{2\pi(\alpha-1)q}{\alpha}\right).$$

That concludes step 1. Now, for step 2, the resistance was bounded from above by

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

which is not a tight bound. Ultimately, Grassi *et al.* conjecture security against this **round-level** attack if:

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

after which they add $t-1$ extra S-boxes in order to account for the possibility of a **subspace** attack.
As explained, the use of this upper bound can lead to round parameters R_F, R_P that are considered secure when they are not. This is visualised in Figure 3, where the resistance against the attack is shown in red, and the resistance deduced from Equation (6) is shown in blue.

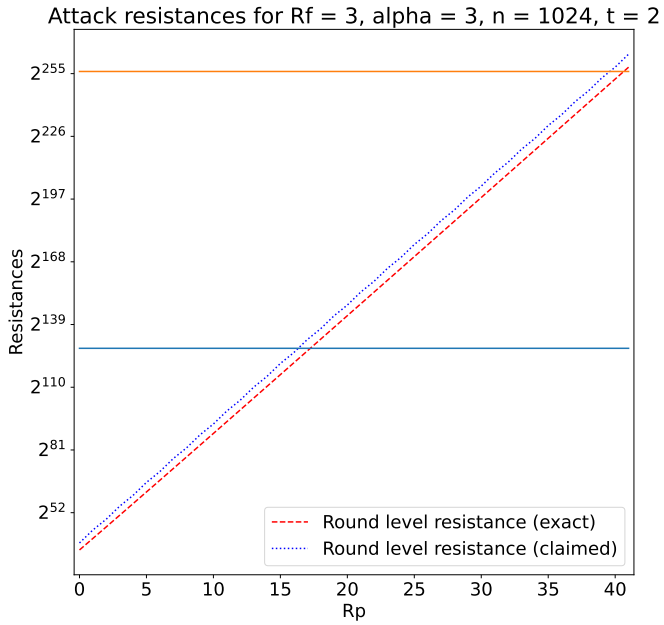

Figure 3: Attack resistances calculated for 6 full rounds, $\alpha = 3, t = 2$ for several numbers of partial rounds. Red dashed line: exact resistance given by the binomial expression. Blue dotted line: resistance deduced from Equation (6).

4.2 Degree of regularity

Finding the degree of regularity that determines the resistance against the Gröbner basis attack is not always possible without performing the full Gröbner basis computation. This degree of regularity is bounded from above by the Macaulay bound, which is a tight bound with overwhelming probability if the system of polynomials were randomly generated. For a system of $\mathcal{N}$ polynomials, each with degree d_i, the Macaulay bound is given by

$$D_{\text{Mac}} = 1 + \sum_{i=1}^{\mathcal{N}} (d_i - 1). \tag{7}$$

In the argument for Poseidon's security against Gröbner basis attacks, the system of polynomials was assumed to be similar to a randomly generated one, such that the degree of regularity of the system is equal to the Macaulay bound. Yet, the polynomials involved in Gröbner basis attacks are highly structured. Therefore, they generally will not reach the Macaulay bound. For small values of R_F, R_P, I have verified this fact experimentally by writing out the system of polynomials, and performing the Gröbner basis computation. The type of attack I performed was a **round-level** attack, for which there are different ways to model the Poseidon permutation. A subset of all possible modelings is given in Appendix B. Out of these, two were implemented in sageMath, and they will be referred to as the sparse modelling and the substituted modelling.

Sparse modelling

The sparse modelling is the modelling in which each polynomial has at most one term with degree larger than 1. In this modelling, the j^{th} element of the state of the sponge construction in round i, after the ARC step is called $A_i[j]$. The j^{th} element of the input (plain text) of the sponge construction is called $P[j]$, the j^{th} element of the i^{th} round constant is called $K_i[j]$, and the j^{th} element of the output of the sponge construction is called $C_R[j]$. Then, the Poseidon instantiation with $R = R_F + R_P = 2R_f + R_P$ rounds is modelled by:

$$(A_i[j])^\alpha - \sum_{k=1}^{t} \left(M^{-1}[j,k] \cdot (A_{i+1}[k] - K_{i+1}[k]) \right) \qquad \begin{array}{l} \text{for } i \in \{1, \dots, R_f\} \cup \{R_f + R_P + 1, \dots, R - 1\}, \\ j \in \{1, \dots, t\} \end{array}$$

$$\left\{ \begin{array}{ll} (A_i[1])^\alpha - \sum_{k=1}^{t} \left(M^{-1}[1,k] \cdot (A_{i+1}[k] - K_{i+1}[k]) \right) & \\ A_i[j] - \sum_{k=1}^{t} \left(M^{-1}[j,k] \cdot (A_{i+1}[k] - K_{i+1}[k]) \right) & j \in \{2, \dots, t\} \end{array} \right\}, \qquad \text{for } i \in \{R_f + 1, \dots, R_f + R_P\}.$$

Then, to link A_1 to the input, and A_R to the output:

$$A_1[j] - P[j] - K_1[j], \qquad A_R[j]^\alpha - \sum_{k=1}^{t} M^{-1}[j,k] C_r[k]$$

An illustration is shown in Figure 4. In this sparse modelling, there are $(R+1)t$ variables, with one equation/polynomial per variable. $R_f \cdot t + R_P$ of these polynomials have degree α, and the others have degree 1, implying that $(t-1)R_P$ variables can be written as a linear combination of other variables. The resistance against a Gröbner basis computation depends both on the number of variables, **and** the degree of regularity in a relatively complex manner. In some situations, it might be beneficial attempt to decrease the degree of regularity by adding a new variable. In other situations, the opposite might be preferable, but which of the two is true is difficult to predict before performing the computation. These $(R+1)t$ variables that are a linear combination of other variables can be eliminated, and their elimination might not

increase the degree of regularity of the resulting system. Even if their elimination slightly increases the degree of regularity, the Gröbner basis for the new system might still be easier to compute, as it has fewer variables. To verify whether eliminating these variables is beneficial, the substituted modelling was introduced.

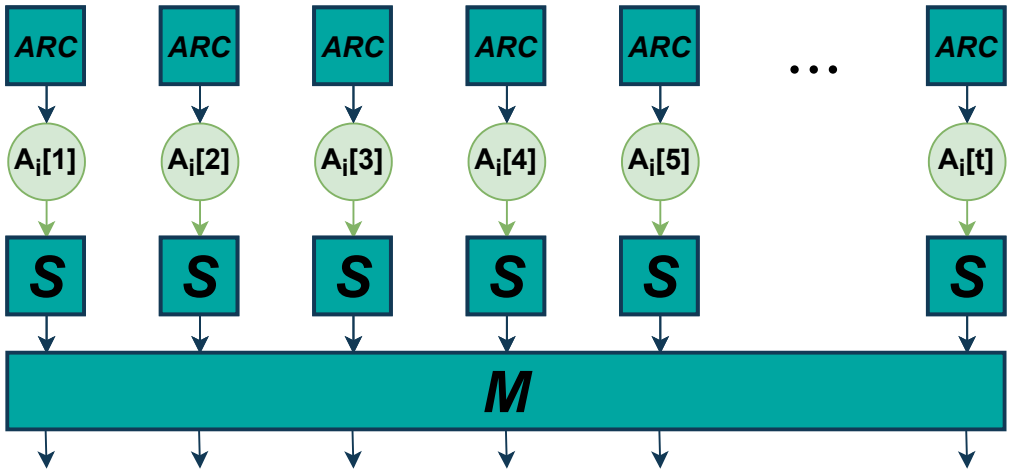

Figure 4: Overview of a round of the Poseidon permutation. The state at the green dots is written as A_i.

Substituted modelling

The substituted modelling was made in order to minimise the number of variables and equations used, while the degree of each equation remains at most α. To that end, an initial modelling was considered, using variables $C_i[j]$, that represent the j^{th} element of the state after $i \in \{0, 1, \ldots, R\}$ rounds. Here $C_0 = P$ is the input of the permutation (plain text). In this initial modelling, the Poseidon permutation is modelled by:

$$\sum_{k=1}^{t} \left(M^{-1}[j,k] \cdot C_{i+1}[k] \right) - (C_i[j] + K_{i+1}[j])^{\alpha} = 0, \qquad \begin{array}{l} \text{for } i \in \{0, \ldots, R_f - 1\} \cup \{R_f + R_P, \ldots, R - 1\}, \\ j \in \{1, \ldots, t\} \end{array}$$

$$\left\{ \begin{array}{ll} \sum_{k=1}^{t} \left(M^{-1}[1,k] \cdot C_{i+1}[k] \right) - (C_i[1] + K_{i+1}[1])^{\alpha} = 0 & \\ \sum_{k=1}^{t} \left(M^{-1}[j,k] \cdot C_{i+1}[k] \right) - (C_i[j] + K_{i+1}[j]) = 0 & j \in \{2, \ldots, t\} \end{array} \right\} \quad \text{for } i \in \{R_f, \ldots, R_f + R_P - 1\}.$$

An illustration is shown in Figure 5. This initial modeling has a total of $R \cdot t$ variables and equations, of which $R_f \cdot t + R_P$ have degree α, whereas the other $R_P(t-1)$ have degree 1. These $R_P(t-1)$ equations of degree 1 are then used to eliminate $R_P(t-1)$ variables, such that the final substituted modelling has a total of $R_f \cdot t + R_P$ variables and equations (all of degree α).

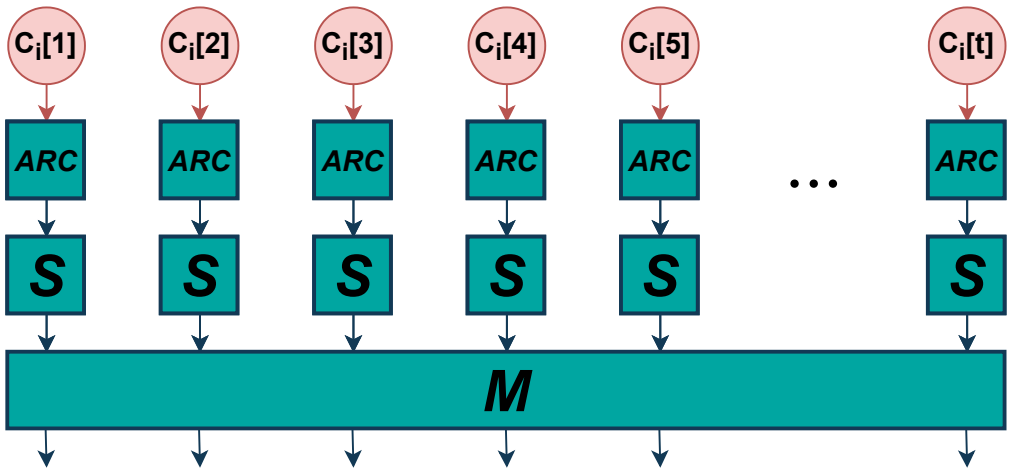

Figure 5: Overview of a round of the Poseidon permutation. The state at the red dots is written as C_i.

Gröbner basis computation

To test the validity of the assumptions made by Grassi *et al.*, I have performed the Gröbner basis computation for both modellings, for $\alpha = 3, t = 2$, R_F ranging from 2 up to 8, and R_P ranging from 0 up to at most 8 (depending on R_F). First, I demonstrate that the degree of regularity for these two modellings is significantly lower than their Macaulay bound. For both systems, the Macaulay bound, D_{Mac}, is given by $D_{\text{Mac}} = 1 + (\alpha - 1)(R_f \cdot t + R_p)$. For both modellings, the experimentally determined degree of regularity is displayed alongside this bound D_{Mac} in Table 1. Evidently, the degree of regularity D_{reg} is considerably smaller than the bound, at approximately $D_{\text{reg}} = 0.5 D_{\text{Mac}}$

Table 1: Experimentally found degree of regularity for both modellings, alongside the Macaulay bound D_{Mac}, and the number of variables $\mathcal{V}$

R_F	R_P	sparse $\mathcal{V}$	substituted $\mathcal{V}$	D_{Mac}	sparse D_{reg}	substituted D_{reg}
2	0	6	4	9	5	4
2	1	8	5	11	5	5
2	2	10	6	13	5	5
2	3	12	7	15	6	6
2	4	14	8	17	7	7
2	5	16	9	19	8	8
2	6	18	10	21	9	9
2	7	20	11	23	10	-
2	8	22	12	25	11	-

Since the degree of regularity depends on the way in which the permutation is modelled, and on the number of variables used, it is arguably less useful to compare the degree of regularity of the sparse modelling, and of the substituted modelling to the estimate used by Grassi *et al.* Instead, the resistances $\mathcal{C}_{\text{sp}}$, $\mathcal{C}_{\text{su}}$ of these (sparse and substituted) attacks should be compared to the expected attack resistance, $\mathcal{C}_{\text{exp}}$ given by Grassi *et al.* As explained in Section 4.1, the final value for the attack resistance given by Grassi *et al.* is incorrect. Therefore, I will display the attack resistance given by the binomial coefficient, for $\chi = 1$:

$$\mathcal{C}_{\text{bi}} = \binom{\alpha q + 1}{q}^2 = \binom{1 + \alpha((t-1)R_F + R_P + \chi)}{(t-1)R_F + R_P + \chi}.$$

In Table 2 and Table 3, these three attack resistances are compared for varying different numbers of rounds. Evidently, the attack has a lower attack resistance than conjectured by Grassi *et al.*, implying that the assumption that the degree of regularity is equal to the Macaulay bound is false. A comprehensive table with all experimental findings is shown in Section 4.2.2. In these experiments, I observed that either the first or the second partial round has no effect on the degree of regularity. Beyond that, the degree of regularity and the number of rounds appear to be linearly related. Assuming the linear behaviour observed for small R_F, R_P also holds for large R_F, R_P, the degree of regularity can be extrapolated to large R_F and R_P. Using the experiments shown in Section 4.2.2, this extrapolation yields

$$D_{\text{reg,sp}} = 0.41R_F + 0.42R_P + 3.13$$

for the sparse modelling, and

$$D_{\text{reg,su}} = 0.45R_F + 0.39R_P + 3.04.$$

The attack resistances given by these extrapolated degrees of regularity are visualised in Figure 6. Again, the attack resistance predicted by Grassi *et al.* is significantly lower than the resistance found in these experiments.

Table 2: Comparison of attack resistances for constant $R_F = 2$ and variable R_P. Here, $\mathcal{C}_{\text{sp}}$ and $\mathcal{C}_{\text{su}}$ are the complexities of finding a Gröbner basis in *grevlex* ordering for both modellings, and $\mathcal{C}_{\text{bi}}$ is the resistance claimed by Grassi *et al.* Finally, $\mathcal{C}_{\text{F, sp}}$ and $\mathcal{C}_{\text{F, su}}$ are the complexities of the FGLM transformation.

R_F	R_P	$\log_2(\mathcal{C}_{\text{sp}})$	$\log_2(\mathcal{C}_{\text{su}})$	$\log_2(\mathcal{C}_{\text{bi}})$	$\log_2(\mathcal{C}_{\text{F,sp}})$	$\log_2(\mathcal{C}_{\text{F,su}})$
2	0	17.70	12.26	13.81	11.51	11.51
2	1	20.66	15.95	18.96	16.85	16.59
2	2	23.10	17.70	24.19	22.02	21.60
2	3	28.36	21.49	29.46	27.1	26.58
2	4	33.65	25.30	34.76	32.11	31.53
2	5	38.98	29.14	40.09	37.09	36.45
2	6	44.32	32.99	45.44	42.04	41.36
2	7	49.68	-	50.81	46.96	46.25
2	8	55.06	-	56.18	51.87	51.13

Table 3: Comparison of attack resistances for constant $R_P = 0$ and variable R_F

R_F	R_P	$\log_2(\mathcal{C}_{\text{sp}})$	$\log_2(\mathcal{C}_{\text{su}})$	$\log_2(\mathcal{C}_{\text{bi}})$	$\log_2(\mathcal{C}_{\text{F,sp}})$	$\log_2(\mathcal{C}_{\text{F,su}})$
2	0	17.70	12.26	13.81	11.51	11.51
4	0	23.1	20.66	24.19	22.02	22.02
6	0	30.48	28.36	34.76	32.11	32.11
8	0	37.75	35.81	45.44	42.04	42.04

4.2.1 Completing the attack

Now, since a Gröbner basis attack is not yet complete once the Gröbner basis has been found, one might think that the two remaining steps could still be expensive enough to thwart the attack. Of the two remaining steps, the FGLM transformation to a *lex* order is the most expensive one, with a complexity of $\mathcal{V}D^3$ if the ideal formed by the system of polynomials has degree D. This ideal can be found easily without computing the Gröbner basis itself. In my experiments, I found that the degree of the ideal was given by

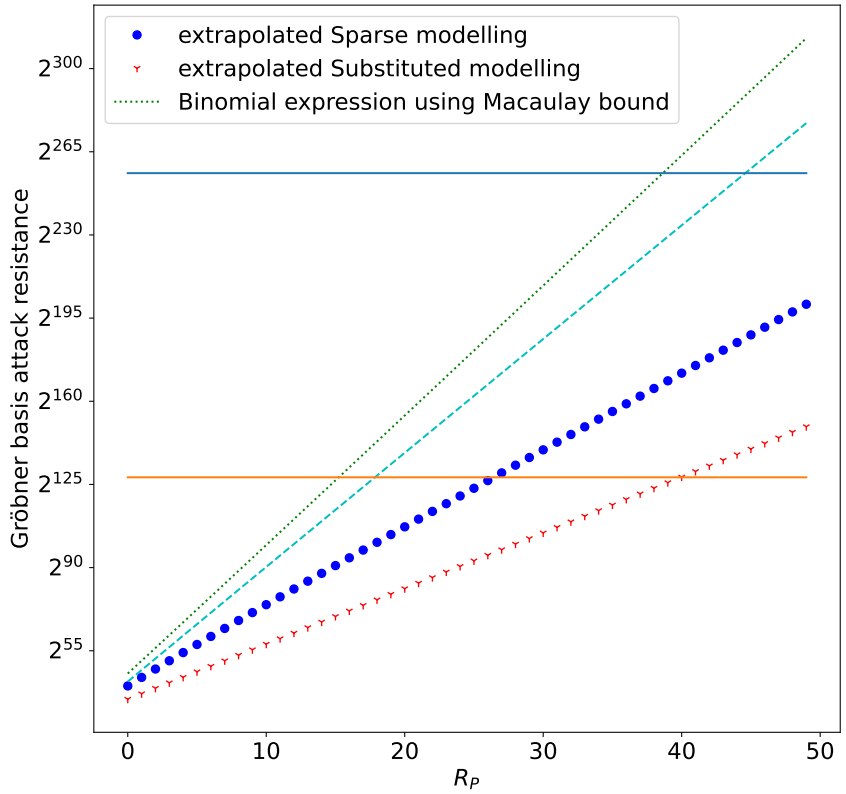

Figure 6: Attack resistances calculated for 8 full rounds, $\alpha = 3, t = 2$ for several numbers of partial rounds. Red symbols: extrapolation for the sparse modelling. Blue, thick, dotted line: extrapolation for the substituted modelling. Green dotted line: resistance given by binomial expression. Cyan dashed line: cost of performing the FGLM transformation

$$D = \alpha^{R_F+R_P},$$

for both the sparse and substituted modellings. The attack resistance for this second step is also shown in Table 2, Table 3 and Figure 6. This FGLM transformation step is notably more expensive than finding the Gröbner basis in *grevlex* order, but it is still less expensive than the claimed resistance. Therefore, the Poseidon permutation is vulnerable to a Gröbner basis attack.

4.2.2 Complexity table

All experimental data, and all resulting resistances are shown in Table 4. There, the complexities for finding a Gröbner basis in *grevlex* order, the complexities for performing the FGLM base transformation, and all their prerequisites are shown.

Table 4: Experiments performed for $\alpha = 3, t = 2, \chi = 1$. Here, $\mathcal{V}_{sp}$ and $\mathcal{V}_{su}$ are the numbers of variables for the sparse modelling, and the substituted modelling, respectively. $D_{reg,sp}$ and $D_{reg,su}$ are the experimentally found degrees of regularity for both modellings. $\mathcal{C}_{sp}$ and $\mathcal{C}_{su}$ are the complexities of finding a Gröbner basis in *grevlex* ordering for both modellings, and $\mathcal{C}_{bi}$ is the resistance that should be reached according to Grassi *et al.* D is the dimension of the ideal generated by the polynomials, and finally, $\mathcal{C}_{F,sp}$ and $\mathcal{C}_{F,su}$ are the complexities of the FGLM transformation.

R_F	R_P	$\mathcal{V}_{sp}$	$\mathcal{V}_{su}$	$D_{reg,sp}$	$D_{reg,su}$	$\log_2 \mathcal{C}_{sp}$	$\log_2 \mathcal{C}_{su}$	$\log_2 \mathcal{C}_{bi}$	D	$\log_2 \mathcal{C}_{F,sp}$	$\log_2 \mathcal{C}_{F,su}$
2	0	6	4	5	4	17.7	12.26	13.81	9	12.09	11.51
2	1	8	5	5	5	20.66	15.95	18.96	27	17.26	16.59
2	2	10	6	5	5	23.1	17.7	24.19	81	22.34	21.6
2	3	12	7	6	6	28.36	21.49	29.46	243	27.36	26.58
2	4	14	8	7	7	33.65	25.3	34.76	729	32.34	31.53
2	5	16	9	8	8	38.98	29.14	40.09	2187	37.28	36.45
2	6	18	10	9	9	44.32	32.99	45.44	6561	42.21	41.36
2	7	20	11	10	-	49.68	-	50.81	19683	47.12	46.25
2	8	22	12	11	-	55.06	-	56.18	59049	52.01	51.13
4	0	10	8	5	5	23.1	20.66	24.19	81	22.34	22.02
4	1	12	9	5	6	25.19	24.58	29.46	243	27.36	26.94
4	2	14	10	6	6	30.48	25.93	34.76	729	32.34	31.85
4	3	16	11	7	7	35.81	29.92	40.09	2187	37.28	36.74
4	4	18	12	8	8	41.15	33.89	45.44	6561	42.21	41.62
4	5	20	13	9	-	46.51	-	50.81	19683	47.12	46.49
4	6	22	14	10	-	51.89	-	56.18	59049	52.01	51.36
6	0	14	12	6	6	30.48	28.36	34.76	729	32.34	32.11
6	1	16	13	6	6	32.37	29.46	40.09	2187	37.28	36.98
6	2	18	14	7	7	37.75	33.65	45.44	6561	42.21	41.85
6	3	20	15	8	-	43.14	-	50.81	19683	47.12	46.7
6	4	22	16	9	-	48.53	-	56.18	59049	52.01	51.55
8	0	18	16	7	7	37.75	35.81	45.44	6561	42.21	42.04
8	1	20	17	7	-	39.52	-	50.81	19683	47.12	46.88

5 Sloppy execution

Now that the fundamental flaws in the Poseidon security argument have been demonstrated, I will show two instances where sloppy execution has led to Poseidon instantiations that are incorrectly assumed to be secure.

5.1 Transcription error

In the setting of **Full-permutation attacks**, where one polynomial is used for each of the χ unknown inputs of the sponge construction, Grassi *et al.* [8] have made a significant transcription error. At the top left of page 25, they conjecture security against a full-round Gröbner basis attack by requiring

$$R_F + R_P \geq \min\left\{\left\lceil \frac{M}{2\log_2(\alpha)} \right\rceil, \left\lceil \frac{n}{2\log_2(\alpha)} \right\rceil\right\} = \log_\alpha(2) \cdot \min\left\{\left\lceil \frac{M}{2} \right\rceil, \left\lceil \frac{n}{2} \right\rceil\right\}. \tag{8}$$

Yet, in their equation (11), the constraint for the full round attack instead requires

$$R_F + R_P \geq \log_\alpha(2) \cdot \min\left\{\left\lceil \frac{M}{3} \right\rceil, \left\lceil \frac{n}{2} \right\rceil\right\}. \tag{9}$$

This mistake leads Grassi *et al.* to significantly overestimate the security that the Poseidon permutation provides against Gröbner basis attacks, as illustrated by Figure 7. For a fixed $\alpha, t, R_P, R_F, n = \log_2(p)$, this figure shows the resistance to the attack (red), and 2^{M_+}, where M_+ is the highest security level M that satisfies the first constraint of Equation (9), for a fixed set of parameters (blue). If the security argument were sound, the blue line should always be (marginally) below the red line. For example, the constraint in equation 5 of [2] would imply that 6 full rounds and 22 partial rounds are sufficient for $\alpha = 3, t = 2$ and $n = 1024$, and a desired security level of 128, whereas to gain that security level for these parameters, 35 partial rounds would be needed.

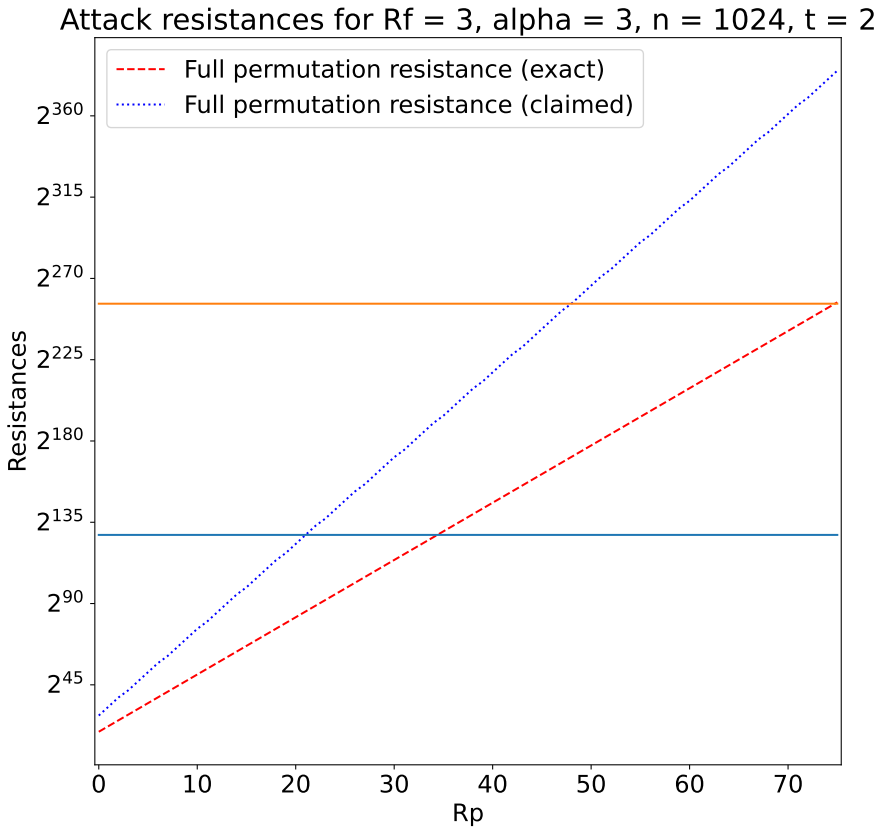

Figure 7: Attack resistances for 6 full rounds, $\alpha = 3, t = 2$ for several numbers of partial rounds. Red dashed line: exact resistance given by the binomial expression. Blue dotted line: claimed resistance (*i.e.* resistance deduced from the first expression in equation 5 from [2])

5.2 Algebra error

In the setting of **Round-level attacks**, where polynomials are set up for each individual S-box, an algebra mistake was made. Recall Equation (5), where Poseidon is shown to be susceptible to an attack if

$$(t-1)R_F + R_P + \chi \leq C^{-1} \cdot \min\{M, n\chi\}, \tag{10}$$

with

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

It is then argued that the maximal number of rounds that can be attacked is achieved for $\chi = 1$, but this is not true. Equation (10) can be rewritten to

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

Here, the first argument of the minimum function is indeed maximised for $\chi = 1$, but the last argument is maximised for $\chi = t-1$, as C is generally not more than 10, such that $n - C$ should be positive. Ultimately, security is conjectured if

$$(t-1)R_F + R_P \geq C^{-1} \cdot \min\{M, n\} + t - 2,$$

but if we address the algebra error, we obtain

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

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. However, once the algebra 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 $\alpha = 3, n = 256, M = 1536, R_F = 8, t = 8$, 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 would require $R_P \geq 73$, **but,** a **round-level** Gröbner basis attack would need $R_P \geq 230$. Therefore, equation 5 of [2] should have 3 constraints, instead of 2.

6 Conclusion

To summarise, the security argument for Poseidon, put forth by Grassi *et al.* was found to have significant errors, ranging from trivial transcription errors to erroneous assumptions, and faulty logic. In particular, Grassi *et al.* used loose upper bounds for the resistance against Gröbner basis attacks, which led to an overestimation of security against these attacks. Furthermore, Grassi *et al.* assumed that the degree of regularity determining the resistance against a Gröbner basis attack on a system is equal to the Macaulay bound for that system, but this assumption was was found to be incorrect. The experimentally determined degree of regularity was significantly lower, such that the resistance against a Gröbner basis attack is much lower than claimed by Grassi *et al.* Consequently, Grassi *et al.* need to revisit their security argument against Gröbner basis attacks, and perform the experiments to determine the correct degree of regularity.

References

[1] A. Aly, T. Ashur, E. Ben-Sasson, S. Dhooghe, and A. Szepieniec, "Design of symmetric-key primitives for advanced cryptographic protocols," *IACR Transactions on Symmetric Cryptology*, vol. 2020, no. 3, pp. 1–45, 2020, ISSN: 2519173X. DOI: 10.13154/tosc.v2020.i3.1-45.

[2] L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofnegger, "POSEIDON: A new hash function for zero-knowledge proof systems," *Proceedings of the 30th USENIX Security Symposium*, pp. 519–535, 2021.

[3] D. A. Cox and D. O. Shea, *Ideals , Varieties , and Algorithms*, ISBN: 9783319167206.

[4] K. Batselier, P. Dreesen, and B. De Moor, "On the null spaces of the Macaulay matrix," *Linear Algebra and Its Applications*, vol. 460, no. 2014, pp. 259–289, 2014, ISSN: 00243795. DOI: 10.1016/j.laa.2014.07.035. [Online]. Available: http://dx.doi.org/10.1016/j.laa.2014.07.035.

[5] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*. CRC press, 2018.

[6] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche, "Cryptographic sponge functions," 2011.

[7] J. C. Faugère, D. Lazard, and T. Mora, *Efficient Computation of Zero-Dimensional Grobner Bases by change of Ordering*, 1996.

[8] M. R. Albrecht, C. Cid, L. Grassi, *et al.*, "Algebraic Cryptanalysis of STARK-Friendly Designs: Application to MARVELlous and MiMC," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11923 LNCS, pp. 371–397, 2019, ISSN: 16113349. DOI: 10.1007/978-3-030-34618-8{_}13.

A Practical difficulties of computing Gröbner Bases

Up until now, I have been discussing the computation of a Gröbner Basis without mentioning the issues that might arise when using SageMath for efficient GB computation on a Windows system. I will discuss these now.

A.1 SageMath F4

The default GB algorithm of SageMath is quite slow compared to Faugère's F4 and F5 algorithms, but the options of the SageMath groebner_basis function do appear to include an F4 algorithm. It is a bit less straightforward than that, because this option uses the Macaulay2 software package, which is not included with SageMath out of the box. This software package is open-source, just like SageMath, but installing it on a windows system brings difficulties of its own. The software package does not have a windows version; it can only be installed on a MacOSX or Linux system. There are several ways to run Linux on a Windows computer; I tried to use Windows Subsystem for Linux (WSL), and a virtual machine. Macaulay2 does work on WSL, but I installed SageMath with a package manager by following the instructions on the SageMath website, but the SageMath application in WSL threw an error that I was unable to remedy. I was more successful with a virtual machine from Oracle VM. With this software, I was able to use a virtual Linux machine, on which I SageMath and Macaulay2 worked properly, so ultimately, I was able to use the F4 algorithm in SageMath.

A.2 SageMath F5

I also attempted using Faugère's F5 algorithm for the GB computations, but I was ultimately unable to do that. This F5 algorithm is not included in the options of groebner_basis. I did track down a toy implementation of F5 ([insert link/citation]), but this implementation was a proof of concept, and not necessarily an optimised implementation. Furthermore, it was written using a very old version of SageMath, and some of the SageMath functions used in this implementation were either deprecated or renamed, and I was unable to get this code to work.

A.3 Macaulay2 RAM issues

After getting the F4 algorithm of Macaulay2 to work in SageMath, I used it to compute the GB's for larger and larger numbers of rounds, but at some point Macaulay2 crashed near the end of the computation because it was using too much RAM, whereas the default implementation was able to complete the computation for the same number of rounds.

A.4 Retrieving the degree of regularity

For the purposes of this report, I also wanted to retrieve the degree of regularity of the Gröebner Bases, but the groebner_basis does not allow for that. There is an optional parameter 'prot' that should print what is happening during the computation, but it is experimental, and does not work for the algorithms that I used. Macaulay2 does show the degree of regularity of the GB's but only using its default GB algorithm, but not with its F4 implementation. Later, I discovered that the FGB package does have an F4 implementation that shows the degree of regularity.

B Polynomial Descriptions

To be able to experimentally verify (or refute) the security claims made by Grassi *et al.*, I have carried out a **round-level** Gröbner basis attack on some Poseidon instantiations with a small number of rounds. There are several different ways to model a Poseidon instantiation with polynomials, and since the resistance against the attack depends both on the degree of regularity, and the number of variables, these different models can all have different attack resistances. For example, an attacker can introduce extra variables to decrease the degree of the polynomials, and therefore decrease the degree of regularity, or he can try to do the opposite. Furthermore, the polynomials are supposed to link certain state variables before or after a certain operation. For some operations, the polynomial describing them can be less efficient (read less sparse, and with higher degree) than the polynomial describing the inverse of the operation. I have systematically investigated a subset of the different ways to model Poseidon instantiations, and I have performed the Gröbner basis attack for 2 of these.

First, I will clarify the notation that will follow; The Poseidon permutation consists of three steps, so a specific name is given to the states in between these steps. (Figure 8 explains this in more detail). A_i, B_i, C_i are the state after the round constant addition, after the S-box, and after matrix multiplication in the i-th round, respectively (with $0 < i \leq 2R_f + R_P$). P is the input (plain text, which consists of χ unknown variables, and $t - \chi$ zeroes), $C_{2R_f+R_P}$ is the output, consisting of χ known outputs, and $t - \chi$ variables. Furthermore, the states are vectors of length t. Then, $A_i[j]$ represents the $j-$th element of the state after the i-th round. Unless stated otherwise, $j \in \{1, \ldots, t\}$

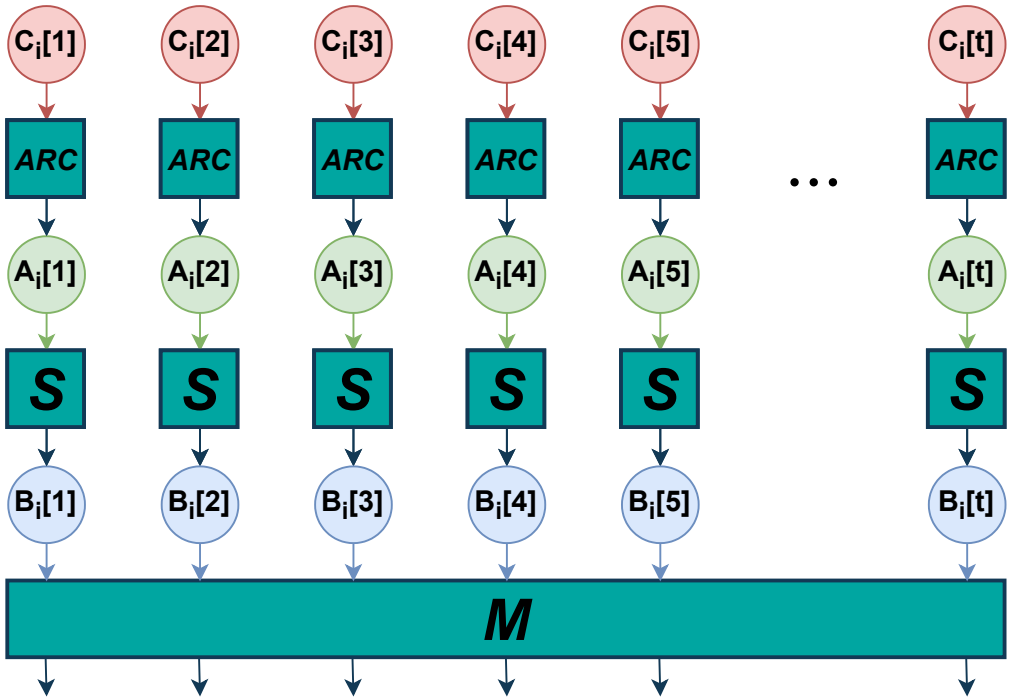

Figure 8: Overview of a round of the Poseidon permutation. The state at the green dots is written as A_i. The state at the blue dots is written as B_i. The state at the red dots is written as C_i.

To write out the Poseidon permutation as a set of equations, we can use the A_i, B_i, or C_i (or some combination of these, which will not be done in this thesis). There is also a choice to either work backwards or forwards to relate state i to state $i + 1$ (or a combination).

The K_i are just the round constants, and I write $R = 2R_f + R_P$.

B.1 Basic relations

These A_i, B_i, C_i are related with each other in the following way:

$$A_i[j] = C_{i-1}[j] + K_i[j], \qquad C_{i-1}[j] = A_i[j] - K_i[j] \qquad \text{for } i = 1 \dots R$$

$$\left\{ \begin{array}{ll} B_i[j] = (B_i[j])^\alpha, A_i[j] = (B_i[j])^{\frac{1}{\alpha}} & \text{if } i = 1 \dots R_f \vee i = R_f + R_P + 1 \dots 2R_f + R_P \\ B_i[1] = (A_i[1])^\alpha, A_i[1] = (B_i[1])^{\frac{1}{\alpha}} & \text{if } i = R_f + 1, \dots R_f + R_P \\ B_i[j] = A_i[j]), A_i[j] = B_i[j] & \text{if } i = R_f + 1, \dots R_f + R_P \text{ and } j = 2 \dots t \end{array} \right\}$$

$$C_i[j] = \sum_{k=1}^{t} \left(M[j,k] \cdot B_i[k] \right), \qquad B_i[j] = \sum_{k=1}^{t} \left(M^{-1}[j,k] \cdot C_i[k] \right) \qquad \text{for } i = 1 \dots R$$

B.2 Using the A_i

In this subsection, I treat the possibilities for using the A_i as variables for each state i. Here, the number of equations, variables and constants is the same for all writings given. There are R distinct A_i, which each consist of t variables, and then there are the P, C_r which together have t variables, for a total of $t \cdot (R+1)$ variables. There are a total of $t \cdot (R+1)$ equations. $2tR_f + R_P$ have degree α, and the rest has degree 1.

B.2.1 Connecting the beginning

To connect the first A_1 to the input, P, we use

$$A_1[j] = P[j] + K_1[j].$$

Note that these equations should hold for all allowed j (so that the previous expression shows a total of t equations at once).

B.2.2 Connecting the end

To connect A_R to C_R, there are three options:
Fully forward:

$$\sum_{k=1}^{t} \left(M[j,k] \cdot (A_R[k])^\alpha \right) - C_R[j] = 0,$$

or meeting halfway:

$$(A_R[j])^\alpha - \sum_{k=1}^{t} \left(M^{-1}[j,k] C_R[k] \right) = 0,$$

or fully backward:

$$\sum_{k=1}^{t} \left(M^{-1}[j,k] \cdot (C_R[k]) \right)^{\frac{1}{\alpha}} - A_R[j] = 0.$$

B.2.3 Inbetween

To connect A_i to A_{i+1}, we can use:

Fully forward:

$$\sum_{k=1}^{t} (M[j,k] \cdot (A_i[k])^{\alpha}) + K_{i+1}[j] - A_{i+1}[j] = 0, \qquad \text{for } i = 1 \dots R_f \vee i = R_f + R_P + 1, \dots, R-1$$

$$\sum_{k=2}^{t} (M[j,k] \cdot A_i[k]) + M[1,k] \cdot A_i[1] + K_{i+1}[j] - A_{i+1}[j] = 0, \qquad \text{for } i = R_f + 1 \dots R_f + R_P,$$

meeting at C_{i+1} (this gives the same as the previous one):

$$\sum_{k=1}^{t} (M[j,k] \cdot (A_i[k])^{\alpha}) + K_{i+1}[j] - A_{i+1}[j] = 0, \qquad \text{for } i = 1 \dots R_f \vee i = R_f + R_P + 1, \dots, R-1$$

$$\sum_{k=2}^{t} (M[j,k] \cdot A_i[k]) + (M[j,1] \cdot A_i[1]) + K_{i+1}[j] - A_{i+1}[j] = 0, \qquad \text{for } i = R_f + 1 \dots R_f + R_P,$$

or meeting at B_i:

$$(A_i[j])^{\alpha} - \sum_{k=1}^{t} \left(M^{-1}[j,k] \cdot (A_{i+1}[k] - K_{i+1}[k]) \right) = 0, \qquad \text{for } i = 1 \dots R_f \vee i = R_f + R_P + 1, \dots, R-1$$

$$\left\{ \begin{array}{ll} (A_i[1])^{\alpha} - \sum_{k=1}^{t} \left(M^{-1}[1,k] \cdot (A_{i+1}[k] - K_{i+1}[k]) \right) = 0 & \\ A_i[j] - \sum_{k=1}^{t} \left(M^{-1}[j,k] \cdot (A_{i+1}[k] - K_{i+1}[k]) \right) = 0 & j = 2, \dots t \end{array} \right\}, \qquad \text{for } i = R_f + 1 \dots R_f + R_P,$$

or fully backwards:

$$A_i[j] - \left(\sum_{k=1}^{t} \left(M^{-1}[j,k] \cdot (A_{i+1}[k] - K_{i+1}[k]) \right) \right)^{\frac{1}{\alpha}} = 0, \qquad \text{for } i = 1 \dots R_f \vee i = R_f + R_P + 1, \dots, R-1$$

$$\left\{ \begin{array}{ll} A_i[1] - \left(\sum_{k=1}^{t} \left(M^{-1}[1,k] \cdot (A_{i+1}[k] - K_{i+1}[k]) \right) \right)^{\frac{1}{\alpha}} = 0 & \\ A_i[j] - \sum_{k=1}^{t} \left(M^{-1}[j,k] \cdot (A_{i+1}[k] - K_{i+1}[k]) \right) = 0 & j = 2, \dots t \end{array} \right\}, \qquad \text{for } i = R_f + 1 \dots R_f + R_P,$$

B.3 Using the B_i

The amount of equations/variables for these is the same as for the A_i, namely $t \cdot (R+1)$

B.3.1 Connecting the beginning

To connect B_1 to P, we use:

Fully forward:

$$(P[j] + K_1[j])^{\alpha} - B_1[j] = 0,$$

meeting halfway:

$$P[j] + K_1[j] - (B_1[j])^{\frac{1}{\alpha}} = 0,$$

or fully backward (basically the same as the previous one):

$$(B_1[j])^{\frac{1}{\alpha}} - K_1[j] - P[j] = 0$$

B.3.2 Connecting the end

To connect B_R to C_R, we can use:

Forward:

$$\sum_{k=1}^{t} (M[j,k] \cdot B_R[k]) - C_R[j] = 0,$$

or backward:

$$\sum_{k=1}^{t} \left(M^{-1}[j,k] \cdot C_R[k] \right) - B_R[j] = 0,$$

B.3.3 Inbetween

To connect B_i to B_{i+1}, we can use:

Fully forward:

$$\left(\sum_{k=1}^{t} (M[j,k] \cdot B_i[k]) + K_{i+1}[j] \right)^{\alpha} - B_{i+1}[j] = 0, \qquad \text{for } i = 1 \dots R_f - 1 \vee i = R_f + R_P, \dots, R-1$$

$$\left\{ \begin{array}{ll} \left(\sum_{k=1}^{t} (M[1,k] \cdot B_i[k]) + K_{i+1}[1] \right)^{\alpha} - B_{i+1}[1] = 0 & \\ \sum_{k=1}^{t} (M[j,k] \cdot B_i[k]) + K_{i+1}[j] - B_{i+1}[j] = 0 & j = 2, \dots t \end{array} \right\} \quad \text{for } i = R_f \dots R_f + R_P - 1,$$

or meeting at A_{i+1}:

$$\sum_{k=1}^{t} (M[j,k] \cdot B_i[k]) + K_{i+1}[j] - (B_{i+1}[j])^{\frac{1}{\alpha}} = 0, \qquad \text{for } i = 1 \dots R_f - 1 \vee i = R_f + R_P, \dots, R-1$$

$$\left\{ \begin{array}{ll} \sum_{k=1}^{t} (M[1,k] \cdot B_i[k]) + K_{i+1}[1] - (B_{i+1}[1])^{\frac{1}{\alpha}} = 0 & \\ \sum_{k=1}^{t} (M[j,k] \cdot B_i[k]) + K_{i+1}[j] - B_{i+1}[j] = 0 & j = 2, \dots t \end{array} \right\} \quad \text{for } i = R_f \dots R_f + R_P - 1,$$

or meeting at C_i (this gives the same as the previous one):

$$\sum_{k=1}^{t} (M[j,k] \cdot B_i[k]) + K_{i+1}[j] - (B_{i+1}[j])^{\frac{1}{\alpha}} = 0, \qquad \text{for } i = 1 \dots R_f - 1 \vee i = R_f + R_P, \dots, R-1$$

$$\left\{ \begin{array}{ll} \sum_{k=1}^{t} (M[1,k] \cdot B_i[k]) + K_{i+1}[1] - (B_{i+1}[1])^{\frac{1}{\alpha}} = 0 & \\ \sum_{k=1}^{t} (M[j,k] \cdot B_i[k]) + K_{i+1}[j] - B_{i+1}[j] = 0 & j = 2, \dots t \end{array} \right\} \quad \text{for } i = R_f \dots R_f + R_P - 1,$$

or fully backwards:

$$B_i[j] - \sum_{k=1}^{t} \left(M^{-1}[j,k] \cdot ((B_{i+1}[k])^{\frac{1}{\alpha}} - K_{i+1}[k]) \right) = 0, \qquad \text{for } i = 1 \dots R_f - 1 \vee i = R_f + R_P, \dots, R-1$$

$$B_i[j] - \sum_{k=2}^{t} \left(M^{-1}[j,k] \cdot (B_{i+1}[k] - K_{i+1}[k]) \right) - M^{-1}[j,1] \cdot ((B_{i+1}[1])^{\frac{1}{\alpha}} - K_{i+1}[1]) = 0, \text{for } i = R_f \dots R_f + R_P - 1$$

B.4 Using the C_i

For the C_i there are only $t \cdot R$ equations, and $t \cdot R$ variables. Here, we denote $C_0 = P$ for the plain text

B.4.1 Connecting the beginning

Not necessary here.

B.4.2 Connecting the endpoint

Not necessary here.

B.4.3 Inbetween

To connect C_i to C_{i+1}, we use:

Fully forward:

$$C_{i+1}[j] - \sum_{k=1}^{t} \left(M[j,k] \cdot (C_i[k] + K_{i+1}[k])^{\alpha}\right) = 0, \qquad \text{for } i = 0 \dots R_f - 1 \vee i = R_f + R_P, \dots, R-1$$

$$C_{i+1}[j] - \left(M[j,1] \cdot (C_i[1] + K_{i+1}[1])^{\alpha} + \sum_{k=2}^{t} \left(M[j,k] \cdot (C_i[k] + K_{i+1}[k])\right) \right) = 0, \qquad \text{for } i = R_f, \dots, R_f + R_P - 1$$

Meeting at the B_{i+1}:

$$\sum_{k=1}^{t} \left(M^{-1}[j,k] \cdot C_{i+1}[k]\right) - (C_i[j] + K_{i+1}[j])^{\alpha} = 0, \qquad \text{for } i = 0 \dots R_f - 1 \vee i = R_f + R_P, \dots, R-1$$

$$\left\{ \begin{array}{ll} \sum_{k=1}^{t} \left(M^{-1}[1,k] \cdot C_{i+1}[k]\right) - (C_i[1] + K_{i+1}[1])^{\alpha} = 0 & \\ \sum_{k=1}^{t} \left(M^{-1}[j,k] \cdot C_{i+1}[k]\right) - (C_i[j] + K_{i+1}[j]) = 0 & j = 2, \dots t \end{array} \right\} \quad \text{for } i = R_f \dots R_f + R_P - 1,$$

Meeting at the A_{i+1}:

$$C_i[j] + K_{i+1}[j] - \left(\sum_{k=1}^{t} M^{-1}[j,k] \cdot C_{i+1}[k] \right)^{\frac{1}{\alpha}} = 0, \qquad \text{for } i = 0 \dots R_f - 1 \vee i = R_f + R_P, \dots, R-1$$

$$\left\{ \begin{array}{ll} C_i[1] + K_{i+1}[1] - \left(\sum_{k=1}^{t} M^{-1}[1,k] \cdot C_{i+1}[k]\right)^{\frac{1}{\alpha}} = 0 & \\ C_i[j] + K_{i+1}[j] - \left(\sum_{k=1}^{t} M^{-1}[j,k] \cdot C_{i+1}[k]\right) = 0 & j = 2, \dots t \end{array} \right\} \quad \text{for } i = R_f \dots R_f + R_P - 1,$$

Fully backward (this gives the same as the previous one):

$$C_i[j] + K_{i+1}[j] - \left(\sum_{k=1}^{t} M^{-1}[j,k] \cdot C_{i+1}[k] \right)^{\frac{1}{\alpha}} = 0, \qquad \text{for } i = 0 \dots R_f - 1 \vee i = R_f + R_P, \dots, R-1$$

$$\left\{ \begin{array}{ll} C_i[1] + K_{i+1}[1] - \left(\sum_{k=1}^{t} M^{-1}[1,k] \cdot C_{i+1}[k]\right)^{\frac{1}{\alpha}} = 0 & \\ C_i[j] + K_{i+1}[j] - \left(\sum_{k=1}^{t} M^{-1}[j,k] \cdot C_{i+1}[k]\right) = 0 & j = 2, \dots t \end{array} \right\} \quad \text{for } i = R_f \dots R_f + R_P - 1,$$