symmetry

MDPI

Article

Algebraic Attacks against *Grendel*: An Arithmetization-Oriented Primitive with the Legendre Symbol

Jianqiang Ni [1], Jianhui Zhang [2], Gaoli Wang [1,*], Rui Li [3] and Yanzhao Shen [4]

[1] Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai 200062, China; 52255902001@stu.ecnu.edu.cn
[2] R&D Center, Shandong Luruan Digital Technology Co., Ltd., Jinan 250101, China; iah@163.com
[3] Inspur Academy of Science and Technology, Jinan 250014, China; lirui01@inspur.com
[4] Shandong Institute of Blockchain, Jinan 250101, China; shenyanzhao@sdibc.cn
* Correspondence: glwang@sei.ecnu.edu.cn

Abstract: The rise of modern cryptographic protocols such as Zero-Knowledge proofs and secure Multi-party Computation has led to an increased demand for a new class of symmetric primitives. Unlike traditional platforms such as servers, microcontrollers, and desktop computers, these primitives are designed to be implemented in arithmetical circuits. In terms of security evaluation, arithmetization-oriented primitives are more complex compared to traditional symmetric cryptographic primitives. The arithmetization-oriented permutation *Grendel* employs the Legendre Symbol to increase the growth of algebraic degrees in its nonlinear layer. To analyze the security of *Grendel* thoroughly, it is crucial to investigate its resilience against algebraic attacks. This paper presents a preimage attack on the sponge hash function instantiated with the complete rounds of the *Grendel* permutation, employing algebraic methods. A technique is introduced that enables the elimination of two complete rounds of substitution permutation networks (SPN) in the sponge hash function without significant additional cost. This method can be combined with univariate root-finding techniques and Gröbner basis attacks to break the number of rounds claimed by the designers. By employing this strategy, our attack achieves a gain of two additional rounds compared to the previous state-of-the-art attack. With no compromise to its security margin, this approach deepens our understanding of the design and analysis of such cryptographic primitives.

Keywords: arithmetization-oriented hash functions; Legendre symbol; preimage attack; algebraic cryptanalysis; Gröbner basis; *Grendel*

Citation: Ni, J.; Zhang, J.; Wang, G.; Li, R.; Shen, Y. Algebraic Attacks against *Grendel*: An Arithmetization-Oriented Primitive with the Legendre Symbol. *Symmetry* **2023**, *15*, 1563. https://doi.org/10.3390/sym15081563

Academic Editor: Jeng-Shyang Pan

Received: 18 June 2023
Revised: 27 July 2023
Accepted: 7 August 2023
Published: 10 August 2023

1. Introduction

Arithmetization-oriented primitives have recently been widely employed in advanced cryptographic protocols, including Fully Homomorphic Encryption (FHE) protocols, Multi-party Computation (MPC) protocols, and Zero-Knowledge (ZK) proofs. These advanced cryptographic protocols employ arithmetic to convert normal calculations into a sequence of finite field operations, such as addition and multiplication over a large finite field $\mathbb{F}_p$, where p is a big prime integer greater than or equal to 2^{63}. To characterize these finite field operations, arithmetization-oriented primitives are created. The design criterion for arithmetical primitives is to lessen the complexity of multiplication in cryptographic algorithms, as the primary resource consumption in advanced cryptographic protocols comes from the multiplication operation. Using the low-degree round function is an easy route to accomplishing this objective.

Various arithmetization-oriented primitives have been developed, including MiMC [1], GMiMC [2], HadesMiMC/Poseidon [3,4], Masta [5], Pasta [6], Ciminion [7], `Chaghri` [8], and Neptune [9]. These primitives directly use a low-degree round function as power maps $x \mapsto x^d$. More complex ones such as Rescue use the low-degree power map $x \mapsto x^3$ and its inverse $x \mapsto x^{1/3}$ as round functions. A new arithmetization-oriented primitive,

Grendel [10], is designed for zero-knowledge proof systems. *Grendel* uses the Legendre symbol to enhance the round functions in combination with the SHARK-like construction. Here, $\chi_p(\cdot) : \mathbb{F}_p \mapsto \{-1, 0, 1\}$ is defined as $\chi_p(x) := x^{\frac{p-1}{2}} \pmod p$ for the Legendre symbol. The application of the Legendre symbol in cryptography dates back to 1988. Ivan Damgård [11] proposed a new problem of predicting consecutive Legendre (Jacobi) symbols modulo a prime. This problem can be utilized to construct a cryptographically strong pseudorandom bit generator. This concept is closely tied to the distribution of quadratic residues and nonresidues modulo a prime number [12]. In 1997, Mauduit and Sárközy [13] introduced a range of metrics for quantifying the pseudo-randomness of binary sequences. Pseudo-random number generators and pseudo-random bit sequence generators, also known as pseudo-random sequence generators, have widespread applications in numerous scientific, technological, and industrial fields. They are utilized for process modeling, industrial problem solving, and cybersecurity purposes, serving as essential tools in these domains [14]. Tóth [15] and Gyarmati et al. [16] introduced new measures of pseudorandomness (avalanche effect and cross-correlation), and have asserted that those values in the Legendre symbol sequence (known as the Legendre symbol PRF, $x \mapsto \chi_p^k(x) := \chi_p(x + k)$, where k is the private key) are high. In [17], Khovratovich developed a birthday-bound attack for the security analysis of the Legendre symbol PRF. This attack was subsequently enhanced by Beullens et al. [18] and Kaluderovic et al. [19]. According to more recent research by Seres et al., key-recovery attacks against the Legendre symbol PRF may be converted into the solution of a certain set of multivariate quadratic equation systems over a prime field [20].

In symmetric cryptographic schemes, the incorporation of the Legendre symbol into a round function requires the resulting nonlinear layer to be invertible. In [21], the authors proposed the construction of an invertible function using the Legendre symbol as follows: $x \mapsto x \cdot (\chi_p(x) + a)$. The resulting function is invertible when $\chi_p(a^2 - 1) = 1$. By combining the Legendre symbol with the power map, the map $x \mapsto x^d \cdot \chi_p(x)$ is obtained, which is invertible when $\gcd(d + (p-1)/2, p-1) = 1$. In [22], Grassi et al. conducted a further analysis on the generalization of $x \mapsto x \cdot (\chi_p(x) + a)$ to $x \mapsto x^d \cdot (\chi_p(x) + a)$, building upon the foundations of [10,21]. They proposed new invertible functions that combine the Legendre symbol, and analyzed their statistical and algebraic properties.

When operating on large finite fields, arithmetization-oriented ciphers are less susceptible to statistical attacks such as differential [23] and linear [24] attacks. However, they are more vulnerable to algebraic attacks. For example, the cipher `Jarvis` [25] was found to be vulnerable to Gröbner basis attacks. The high-order differential attack is an effective method, as demonstrated in [26] for the high-order differential attack on GMiMC and in [27], where Eichlseder et al. first applied the high-order differential attack on MiMC. Subsequently, Bouvier et al. [28] and Cui et al. [29] analyzed the upper bounds on the algebraic degrees of MiMC, reevaluating its security margin against high-order differentials using different approaches. In [30], Liu et al. proposed an innovative technique called the coefficient grouping technique, which reduces the evaluation of algebraic degrees to a well-structured optimization problem. They applied this technique to launch a high-order differential attack on `Chaghri`, a fully homomorphic encryption scheme. Exploring the application of algebraic methods further to analyze arithmetization-oriented ciphers remains an interesting avenue for future investigation.

Related Works. In [21], the authors proposed the construction of an invertible function using the Legendre symbol, although it was not utilized for cryptographic design. Recently, Grassi et al. [31] re-proposed the use of Legendre symbols for secure multi-party computation (MPC) applications. In the original security analysis of *Grendel* proposed by the designers in [10], the utilization of S-boxes based on the Legendre symbol was highlighted as a notable advantage. This choice allows a higher algebraic degree to be achieved within a relatively small number of rounds, providing resilience against high-order differential [32,33] and interpolation attacks [34]. Consequently, the focus of the

analysis shifted towards the Gröbner basis attack, which presents two distinct approaches for constructing equation systems:

1. The first approach involves the attacker guessing all the Legendre symbols used in the scheme. Subsequently, they can solve the resulting system of equations and verify the correctness of the guessed symbols based on the obtained solution. The complexity of this attack increases by approximately a factor of 2 for each correctly guessed symbol, considering a probability of accurately guessing of around $1/2$.
2. On the other hand, the second approach avoids guessing the Legendre symbols and instead relies on the introduction of auxiliary variables to facilitate the establishment of the equation system. For more detailed information, please refer to [10].

Additionally, after guessing all Legendre symbols, the S-boxes in *Grendel* exhibit a low degree. As a result, it is not necessary to introduce intermediate variables in each round to mitigate the degree of growth. Instead, the attacker can directly solve a higher-degree system of equations. This alternative approach has already been used to attack the full hash function of *Grendel* [22]. In Section 3.1, we provide a detailed description of this attack. Despite proposing a full-round preimage attack on the hash function of Grendel, Ref. [22] only employed basic algebraic attacks. They treated the hash preimage as an unknown variable x, constructed a single-variable equation, and then solved it.

Our Contribution. In this paper, we further analyze the hash function of *Grendel* on the basis of [10,22].

- We introduce the *Constrained Input/Constrained Output* (CICO) problem [4] and exploit its solution to obtain preimages of the hash function of *Grendel*. In this way, we extend the previously proposed technique in [35] and improve the preimage attack by bypassing two additional rounds of the SPN structure. By introducing the CICO problem, our attack is capable of attacking two additional rounds compared to the attack presented in [22], as shown in Table 1.
- Additionally, we employ the CICO problem to formulate a system of multivariate equations for the hash function of *Grendel*. Through an analysis of intermediate variable introduction and the core intricacies of Gröbner basis attacks, we enhance the understanding of constructing equation systems and executing Gröbner basis attacks.

Table 1. For a security level of $s = 128$ and a modulus $p = 2^{256}$, the number of rounds that can be attacked using the univariate root-finding method in different instances of the hash function *Grendel*.

Instance (d, n)	Attacked Rounds in [10]	Attacked Rounds in [22]	Our Result
(2, 3)	28	25	**27**
(2, 4)	21	20	**22**
(2, 8)	11	12	**14**
(2, 12)	7	8	**10**
(3, 3)	22	22	**24**
(3, 4)	16	18	**20**
(3, 8)	8	11	**13**
(3, 12)	6	8	**10**
(5, 3)	16	19	**21**
(5, 4)	12	16	**18**
(5, 8)	6	10	**12**
(5, 12)	4	7	**9**

Organization. Here, we provide a brief overview of the organization of this paper. We present preliminaries in the following section. Section 3 covers the Algebraic Cryptanalysis of the Grendel hash function. In Section 3.3, we discuss our preimage attack on the Grendel hash function, utilizing a construction of a univariate equation system. Section 3.4 delves

into the investigation of the Grendel hash function's security through Gröbner basis attacks. Finally, we conclude the paper in Section 4.

2. Preliminaries

2.1. Notation

In the following, let p be a prime number and let $\mathbb{F}_p$ be a finite field with p elements. Let $\mathbb{F}_p^n$ denote a vector space with n elements and with each element in $\mathbb{F}_p$. The notation $\mathbf{0}^u$ represents a vector of length u in $\mathbb{F}_p^u$, where all components are zero. Considering $\boldsymbol{X} \in \mathbb{F}_p^n$, we denote by X_i its i-th component for each $i \in \{0, 1, \ldots, n-1\}$, i.e., $\boldsymbol{X} = (X_0, X_1, \ldots, X_{n-1})$.

Let $\mathbb{F}_p^n$ be a vector space with standard basis $\{e_0, e_1, \ldots, e_{n-1}\}$. A vector subspace V_u of $\mathbb{F}_p^n$ can be represented as the span of a subset of a standard basis $\{e_0, e_1, \ldots e_{u-1}\}$, where $0 < u < n$. For ease of reference, please review the abbreviated Table 2.

Table 2. List of abbreviations.

Abbreviation	Explanation
$\mathbb{F}_p$	Finite field with p elements
$\mathbb{F}_p^n$	Vector space with n elements over $\mathbb{F}_p$
$\mathbf{0}^u$	Zero vector in $\mathbb{F}_p^u$ of length u
$\boldsymbol{X} = (X_0, X_1, \ldots, X_{n-1})$	Vector in $\mathbb{F}_p^n$
X_i	i-th component of vector $\boldsymbol{X}$
V_u	Vector subspace of $\mathbb{F}_p^n$ spanned by $\{e_0, e_1, \ldots, e_{u-1}\}$

Definition 1 (The Legendre Symbol). *The Legendre symbol $\chi_p(\cdot)$ is a function $\chi_p : \mathbb{F}_p \mapsto \{-1, 0, 1\}$, defined as*

$$\chi_p(x) = \begin{cases} 1 & \text{if } x \text{ is a nonzero quadratic residue modulo } p, \\ -1 & \text{if } x \text{ is a quadratic non-residue modulo } p, \\ 0 & x = 0. \end{cases}$$

Proposition 1 ([36]). *For two prime integers $p, q \geq 3$, the Legendre symbol has the following properties:*

- *If $x \equiv y \pmod{p}$, then $\chi_p(x) = \chi_p(y)$.*
- $\chi_p(x \cdot y) = \chi_p(x) \cdot \chi_p(y)$.
- $\chi_p(q) \cdot \chi_q(p) = (-1)^{\frac{p-1}{2} \cdot \frac{q-1}{2}}$.

2.2. CICO Problem

In the cryptanalysis of traditional symmetric schemes, the goal is to recover the key (or some subkeys) with complexity lower than 2^k. However, the security of arithmetization-oriented hash functions such as the *Grendel* hash function relies on the computational infeasibility of solving the CICO problem.

Definition 2 (CICO Problem [4]). *Let $F : \mathbb{F}_p^n \to \mathbb{F}_p^n$ be a permutation, and let $0 < u < n$ be an integer. For given $(a_0, \ldots a_{n-u-1}), (b_0, \ldots, b_{n-u-1}) \in \mathbb{F}_p^{n-u}$, the CICO problem aims to find $(X_0, \ldots, X_{u-1}), (Y_0, \ldots, Y_{u-1}) \in \mathbb{F}_p^u$ such that*

$$F(X_0, \ldots, X_{u-1}, a_0, \ldots, a_{n-u-1}) = (Y_0, \ldots, Y_{u-1}, b_0, \ldots, b_{n-u-1}).$$

A simpler version of the CICO problem can be defined as follows: when $n = 2$ and $u = 1$, the goal is to find $(X, Y) \in \mathbb{F}_p^2$ such that $F(X, 0) = (Y, 0)$. It can be observed that both

the input and output of the permutation belong to the same vector subspace V_u. The CICO problem is highly relevant to the security of hash functions. Therefore, if the adversary has the ability to solve the problem with a complexity of less than p^{n-u} permutation calls, it is possible to find a preimage or collision of the hash function under the sponge structure. The CICO problem can usually be modelled as a system of equations and solved algebraically.

2.3. Solve the Systems of Algebraic Equations

Our attack is based on modelling cryptographic primitives as a system of polynomial equations. In this section, we present the methods and complexities of solving some univariate and multivariate equations, which are then used to attack the hash function *Grendel*.

We assume that the cryptographic primitive is represented as a well-defined system, specifically, a system of m polynomial equations consisting of n variables $\boldsymbol{X} = (X_0, \ldots, X_{n-1}) \in \mathbb{F}_p^n$,

$$\begin{cases} F_0(X_0, \ldots X_{n-1}) = 0 \\ F_1(X_0, \ldots X_{n-1}) = 0 \\ \vdots \\ F_{m-1}(X_0, \ldots X_{n-1}) = 0. \end{cases}$$

Then, our purpose is to find the ordinary solution of the equation in the hope of obtaining the round key of the encryption schemes or the preimage of the hash function.

2.3.1. Solve a System of Univariate Equations

A univariate equation has only one variable and an equation of $F(x) = 0$. Solving the given system is equivalent to finding the roots of the univariate polynomial $F \in \mathbb{F}_p[x]$ with degree $\mathcal{D}$ of F. Because all operations are performed on the finite field $\mathbb{F}_p$, the computational complexity is measured in terms of field operations.

1. Compute $G = x^p - x \pmod F$.
   The computation of $x^p \pmod F$ requires $\mathcal{O}(\mathcal{D} \cdot \log(p) \cdot \log(\mathcal{D}) \cdot \log(\log(\mathcal{D})))$ field operations with a double-and-add algorithm.
2. Compute $H = \gcd(F, G)$.
   H has the same roots as F in $\mathbb{F}_p$, as $H = \gcd(F, x^p - x)$; however, its degree is likely much lower. This step [37] requires $\mathcal{O}(\mathcal{D} \cdot \log(\mathcal{D})^2)$ field operations.
3. Factor H.
   In general, the polynomial H has only a few roots in $\mathbb{F}_p$. Thus, this step is negligible in complexity.

This root-finding approach using GCD computations is provided in [37], and the final complexity of the algorithm is estimated by

$$\mathcal{O}(\mathtt{M}(\mathcal{D}) \log(\mathcal{D}) \log(\mathcal{D} \cdot p)), \tag{1}$$

where $\mathtt{M}(\mathcal{D}) := 63.43 \cdot \mathcal{D} \log(\mathcal{D}) \log(\log(\mathcal{D})) + \mathcal{O}(\mathcal{D} \log(\mathcal{D}))$ is the complexity of multiplying two polynomials with a degree of at most $\mathcal{D}$ over $\mathbb{F}_p$.

2.3.2. Solve a System of Multivariate Equations

The Gröbner basis attack is a method of recovering a secret from a system of polynomial equations. The first step is to convert the primitive into a system of multivariate equations. Then, a Gröbner basis is computed for the ideal generated by these polynomials. Finally, the Gröbner basis is utilized to compute the target variables in the given system. This attack method involves the following three phases:

1. When launching a Gröbner basis attack, the first step is to construct a set of polynomial equations describing the primitive. After that, a Gröbner basis for the ideal generated by these equations is computed, usually concerning the *degrevlex* ordering for better efficiency. The algorithm used for the computation of the Gröbner basis could be Buchberger's algorithm [38], F4 [39], or F5 [40].
2. After computing the Gröbner basis for the given system of polynomial equations, the next step is to perform a change of term order to facilitate the computation of the elimination ideals and the elimination of variables. This is typically achieved by going from the *degrevlex* term order to the *lex* one using an algorithm such as FGLM [41]. It is worth noting that in many applications, including those in cryptography, the systems of algebraic equations results in zero-dimensional ideals, meaning that they have only finite solutions.
3. The final step of a Gröbner basis attack is to solve the univariate equation for the last variable using a polynomial factoring algorithm. This allows the specific value of the last variable to be obtained; this can then be substituted into the remaining equations to obtain the full solution of the system. This step can use the algorithm mentioned above to find the univariate equation system. When the polynomial has been factored, its roots can be easily found, and correspond to the possible values of the last variable. By substituting each root into the remaining equations, it is possible to obtain all possible solutions to the system of equations.

Cost of Gröbner Basis Computation. For a system of m polynomial equations and n variables, we have

$$F_0(X_0, \ldots, X_{n-1}) = F_1(X_0, ..., X_{n-1}) = \ldots = F_{m-1}(X_0, ..., X_{n-1}) = 0,$$

where $F_i \in \mathbb{F}_p[X_0, ..., X_{n-1}]$, $0 \leq i \leq m$. The complexity of computing a Gröbner basis in *degrevlex* term order [42] is

$$\mathcal{O}\left(\binom{n + D_{reg}}{D_{reg}}^{\omega}\right). \tag{2}$$

In [43], another bound for the complexity of computing the Gröbner basis was provided:

$$\mathcal{O}\left(nD_{reg} \cdot \binom{n + D_{reg} - 1}{D_{reg}}^{\omega}\right), \tag{3}$$

where $2 \leq \omega < 2.3727$ is the linear algebra constant representing the complexity of matrix multiplication and D_{reg} is the degree of regularity. By further comparing these two complexities and computing their ratio, we can observe that

$$\frac{\binom{n+D_{reg}}{D_{reg}}^{\omega}}{nD_{reg} \cdot \binom{n+D_{reg}-1}{D_{reg}}^{\omega}} = \frac{(n + D_{reg}^{\omega})}{n^{\omega+1} \cdot D_{reg}}.$$

When n is small and D_{reg} is large, the complexity calculation of Formula (3) provides a tighter bound. However, the authors of [22] found that when n is small, the complexity of computing the Gröbner basis is asymptotically smaller than the complexity of the FGLM algorithm. Therefore, for small values of n the complexity of the Gröbner basis attack depends on the complexity of the FGLM algorithm. On the other hand, Formula (2) becomes more restrictive when n has a comparatively larger value.

Cost of FGLM algorithm. When employing the FGLM [41] algorithm, the complexity of converting the *degrevlex* order to the *lex* order is

$$\mathcal{O}(n \cdot \mathcal{D}_{\mathcal{I}}^3),$$

where n represents the number of variables and $\mathcal{D}_{\mathcal{I}}$ denotes the degree of the zero-dimensional ideal.

Assuming that the polynomial system is a regular system, we consider n polynomials with the same degree $d_i = \delta, i \in [1, n]$ and n variables. In this case, D_{reg} can be estimated as $1 + \sum_{i=1}^{n} d_i - 1$. If the polynomial system is not regular, then its D_{reg} is less than $1 + \sum_{i=1}^{n} d_i - 1$, a bound known as Macaulay's bound.

2.4. Description of the Grendel Hash Function

The hash function *Grendel* is composed of the *Grendel* permutation combined with a sponge structure. Let $p \geq 3$ be a prime number and let $n \geq 2$ be an integer. The *Grendel* permutation $\mathcal{P} : \mathbb{F}_p^n \to \mathbb{F}_p^n$ is obtained by iteratively applying the *Grendel* round function $\mathcal{F} : \mathbb{F}_p^n \to \mathbb{F}_p^n$ for R rounds. The state size is n. Each round employs a distinct round constant. Each round function consists of three parts: a nonlinear layer, a linear layer, and adding round constants, respectively, denoted as $\mathcal{NL}$, $\mathcal{L}$, and $\mathcal{AC}$.

- The Nonlinear Layer: let $\boldsymbol{X} = (X_0, \ldots, X_{n-1}) \in \mathbb{F}_p^n$; then, $\mathcal{NL} : \mathbb{F}_p^n \to \mathbb{F}_p^n$ consists of independent n identical S-boxes $\mathcal{NL}(X) = (S(X_0), S(X_1), \ldots, S(X_{n-1}))$, where

$$S(X) = X^d \cdot \chi_p(X),$$

  with χ_p being the Legendre symbol. Here, $d \geq 2$ is an integer that satisfies $\gcd(\frac{2d+p-1}{2}, p-1) = 1$.
- The Linear Layer: $\mathcal{L} : \mathbb{F}_p^n \to \mathbb{F}_p^n$ is an $n \times n$ MDS matrix $\mathcal{M} \in \mathbb{F}_p^{n \times n}$.
- The Adding Round Constants Step: this involves the utilization of round constants $c_j^i \in \mathbb{F}_p$, where $0 \leq i \leq R-1$ and $0 \leq j \leq n-1$.

The *Grendel* round function $\mathcal{F}$ consists of three parts, and can be described as $\mathcal{F}(\cdot) = \mathcal{AC} \circ \mathcal{L} \circ \mathcal{NL}(\cdot)$. The *Grendel* permutation $\mathcal{P}$ is iterated over R rounds by $\mathcal{F}$, which can be expressed as $\mathcal{P}(\cdot) = \underbrace{\mathcal{F} \circ \ldots \circ \mathcal{F}(\cdot)}_{R}$. The pseudocode describing the *Grendel* permutation is shown in Algorithm 1.

Algorithm 1 The Grendel Permutation $\mathcal{P}$

Input: $\boldsymbol{X} = (X_0, X_1, \ldots, X_{n-1}) \in \mathbb{F}_p^n$;
Output: $\boldsymbol{Y} = (Y_0, Y_1, \ldots, Y_{n-1}) \in \mathbb{F}_p^n$.

for $r = 0$ to $R-1$ **do**
  for $i = 0$ to $n-1$ **do**
    $X_i \leftarrow X^d \cdot \chi_p(X_i)$;
  end for
  $\boldsymbol{X} \leftarrow \mathcal{M} \cdot \boldsymbol{X}$;
  for $i = 0$ to $n-1$ **do**
    $X_i \leftarrow X_i + c_i^r$;
  end for
end for
$\boldsymbol{Y} \leftarrow \boldsymbol{X}$;
return $\boldsymbol{Y}$;

The sponge construction (Figure 1) [44,45] is a cryptographic framework that utilizes an internal cryptographic permutation or function. It provides versatility in achieving different objectives, including encryption, authentication, and hashing. The construction is based on the concept of a sponge, which consists of an internal permutation that operates on a fixed-size state. By appropriately configuring the sponge, it can be adapted for various cryptographic applications, providing security and flexibility. In this paper, we make slight modifications to the original approach in order to operate on elements of $\mathbb{F}_p$ instead of $\mathbb{F}_2$. Both the input and the output may be of arbitrary size. The state size is $n = r + c$, where r

denotes the rate and c denotes capacity. To process a message m which consists of elements from the field $\mathbb{F}_p$, we utilize the following operations:

1. **Padding:** if the length of the message is already a multiple of r, no padding is necessary; however, if the length is not a multiple of r, we first append $1 \in \mathbb{F}_p$ to the message, then pad the message with 0 until its length becomes a multiple of r.
2. **Absorption:** the message is divided into blocks of size r. Each block is added to the first r blocks of the state using the addition operation. Afterwards, the entire state is processed by applying the permutation function $\mathcal{P}$. Repeat the above operation until all the messages are absorbed.
3. **Squeezing:** in each iteration of the squeezing phase, a block of length r is squeezed out, then the permutation function $\mathcal{P}$ is applied to the entire state and the squeezed block is extracted. This process is repeated until the squeezing phase is completed.

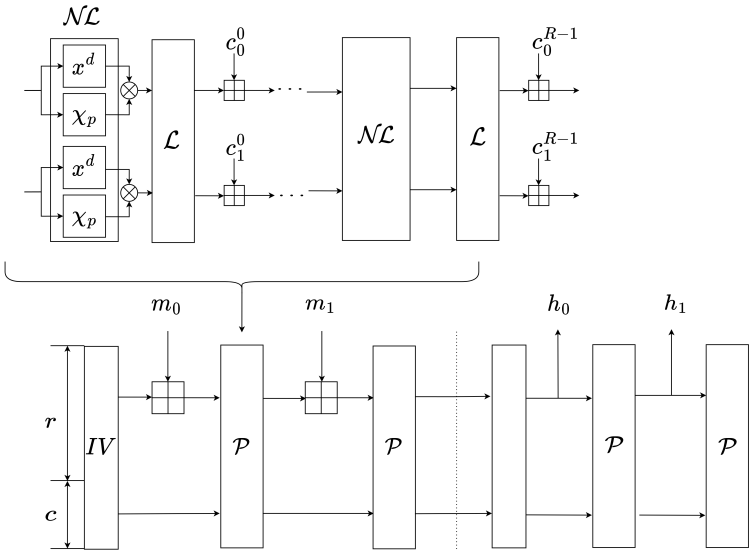

Figure 1. The above is an example of a *Grendel* permutation with a state size of 2. The following is an instance of the hash function *Grendel* with a sponge structure, which is built upon the *Grendel* permutation.

Security. According to the proof presented in [45], when the inner permutation bears resemblance to a random permutation, the sponge construction is indistinguishable from a random oracle up to approximately $p^{c/2}$ queries. Equivalently, in order to provide s bits of security, we need $p^{c/2} \geq 2^s$ and $p^r \geq 2^s$, i.e., $c \geq \lceil 2s \cdot \log_p(2) \rceil$. For such a hash function $\mathcal{H} : \mathbb{F}_p^* \to \mathbb{F}_p^\infty$, it is hard to find

- **collision resistance**: $x, x' \neq x$ such that $\mathcal{H}(x) = \mathcal{H}(x')$
- **preimage resistance**: x, given y such that $\mathcal{H}(x) = y$
- **second-preimage resistance**: x', given $x \neq x'$ such that $\mathcal{H}(x') = \mathcal{H}(x)$.

We assume an output of at least $\lceil 2s/\log_2(p) \rceil$ elements to prevent birthday bound attacks. Furthermore, we require $c \geq \lceil 2s/\log_2(p) \rceil$ for an s-bit security level.

3. Algebraic Cryptanalysis of *Grendel* Hash Function

In this section, we simply review the preimage attack proposed by [22] on a sponge hash function instantiated with the *Grendel* permutation presented in Section 3.1. We then introduce the CICO problem and provide a further analysis of the security of the *Grendel* hash function.

3.1. Preimage Attack on Hash Function Grendel in [22]

Let s denote the security level and let p represent the prime that defines the field. The authors of [22] focused on the case where $p \geq 2^s$, allowing for $r \geq 1$. Here, r defines the

rate of the sponge hash function. In this scenario, the hash function can output a single element from $\mathbb{F}_p$, which aligns with common practice, as p is typically chosen to be large.

For a hash digest $h \in \mathbb{F}_p$, the objective is to find a preimage. For cases where $r \geq 2$, the authors of [22] started by fixing $r-1$ input elements. In contrast to the analysis in [10], they avoided introducing intermediate variables. Instead, they [22] employed polynomials of degree d^R with fixed Legendre symbols, where R denotes the number of attacked rounds. Essentially, the attack on R-round construction involves the following steps:

1. Iterating over all possible sets of Legendre symbols. The probability of a Legendre symbol being ± 1 is approximately $\frac{1}{2}$, while the probability of it being 0 is $\frac{1}{p}$. Consequently, the probability that l Legendre symbols are different from zero can be calculated as $(1-\frac{1}{p})^l$. For a large number of rounds, if p is approximately 2^{32}, this probability exceeds 99.99%. In their attack, $l = nR-(n-1) = n(R-1)+1$. In the first round, it is possible to compute $n-1$ Legendre symbols deterministically because there is no linear layer before the initial application of the S-boxes.
2. Solving the resulting univariate equation to identify a preimage. The authors focused on the case in which the number of hash output elements is 1. By fixing all Legendre symbols, there is only a single unknown (the input variable) and a single equation of degree at most d^R in the end. The equation system consists of only one univariate equation, and can be solved by applying a root-finding algorithm to this equation.
3. Verifying whether the obtained solution is a valid preimage. With the roots discovered, the authors proceeded to verify the validity of the obtained solution. They did this by comparing the computed Legendre symbols to the fixed ones for the given instance. If a inconsistency was found between the computed symbol using their solution and the fixed symbol, they promptly terminated the verification process, indicating an invalid trial. Considering that we only need to compute the first Legendre symbol in each instance with a probability of 50%, the first two symbols with a probability of 25%, etc., we can expect to compute an average of 3 Legendre symbols for each trial before encountering an inconsistency.

3.2. Techniques to Skip SPN Rounds

In this section, we introduce a trick proposed by [35] which can help us skip two rounds without additional consumption when analyzing the permutation based on the SPN structure using the CICO problem.

Let permutation $\mathcal{P} : \mathbb{F}_p^n \to \mathbb{F}_p^n$ be s-secure against the CICO problem. We can split it into two permutations, $\mathcal{F}_0$ and $\mathcal{F}_1$, i.e., $\mathcal{P} = \mathcal{F}_1 \circ \mathcal{F}_0(\cdot)$. Here, V_u is a vector subspace spanned by $\{e_0, \ldots, e_{u-1}\}$. We denote the input state and output state of $\mathcal{P}$ as

$$\boldsymbol{X} = (X_0, X_1, \ldots, X_{u-1}, A_0, \ldots, A_{n-u-1}) \in V_u,$$
$$\boldsymbol{Z} = (Z_0, Z_1, \ldots, Z_{u-1}, C_0, \ldots, C_{n-u-1}) \in V_u,$$

respectively. Here, $(A_0, \ldots, A_{n-u-1}) \in \mathbb{F}_p^{n-u}$ and $(C_0, \ldots, C_{n-u-1}) \in \mathbb{F}_p^{n-u}$ are fixed constants. According to the definition of the CICO problem, if we can find a pair of inputs $\boldsymbol{X}$ and $\boldsymbol{Z}$ such that $\mathcal{P}(\boldsymbol{X}) = \boldsymbol{Z}$ with a complexity less than 2^s, we can conclude that the security margin of the permutation is insufficient.

We denote $\boldsymbol{Y} = (Y_0, Y_1, \ldots, Y_{n-1}) \in \mathbb{F}_p^n$ as the intermediate variable after $\mathcal{F}_0$. If $\boldsymbol{Y}$ can be found to belong to the vector subspace V_u, a polynomial system with $n-u$ outputs can be constructed through $\mathcal{F}_1$, meaning that we can find its root. Ultimately, the value of $\boldsymbol{X}$ can be obtained based on the value of $\boldsymbol{Y}$, which is sufficient to solve the CICO problem. Then, to solve the CICO problem of permutation $\mathcal{P}$, only the $\mathcal{F}_1$ part needs to be dealt with, not the whole permutation $\mathcal{P}$.

To provide a detailed description of this technique, we assume that the permutation $\mathcal{P}$ corresponds to the *Grendel* permutation. $\mathcal{F}_0$ consists of two nonlinear layers, one linear layer, and one round key addition in the *Grendel* round function, while $\mathcal{F}_0$ can be expressed as $\mathcal{F}_0(\cdot) = \mathcal{NL} \circ \mathcal{AC} \circ \mathcal{L} \circ \mathcal{NL}(\cdot)$ and $\mathcal{F}_1$ can be regarded as an $R-2$ round *Grendel* round

function with a linear layer and a round key addition; moreover, S represents the S-box while S^{-1} denotes its inverse.

The S-box needs to satisfy the following property:

$$S(A \cdot X) = S(A) \cdot S(X), \tag{4}$$

where $A, X \in \mathbb{F}_p$.

Let the linear layer MDS matrix $\mathcal{M}$ satisfy

$$\mathcal{M}^{-1} = \begin{bmatrix} m_{0,0} & m_{0,1} & m_{0,2} & \cdots & m_{0,n-1} \\ m_{1,0} & m_{1,1} & m_{1,2} & \cdots & m_{1,n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ m_{n-1,0} & m_{n-1,1} & m_{n-1,2} & \cdots & m_{n-1,n-1} \end{bmatrix}.$$

The round constant is denoted as c_j^i $(0 \le i \le R-1, 0 \le j \le n-1)$. Next, we show how to construct univariate equations with the CICO problem. For the following discussion, we set $u = n-1$ at all times.

When $n = 3$, we set $u = n-1 = 2$; then, V_u is a vector subspace spanned by $\{e_0, e_1\}$. Let the input states of $\mathcal{F}_0$ be $\boldsymbol{X} = (X_1, X_2, A_0) \in V_u$, where A_0 is a fixed constant, and let the states after $\mathcal{F}_0$ be $\boldsymbol{Y} = (Y_0, Y_1, Y_2) \in \mathbb{F}_p^3$. When passing through the first nonlinear layer of $\mathcal{F}_0$, we have

$$\begin{aligned} S(A_0) &= m_{2,0}(S^{-1}(Y_0) - c_0^0) + m_{2,1}(S^{-1}(Y_1) - c_1^0) + m_{2,2}(S^{-1}(Y_2) - c_2^0) \\ &= m_{2,0}S^{-1}(Y_0) + m_{2,1}S^{-1}(Y_1) + m_{2,2}S^{-1}(Y_2) - (m_{2,0}c_0^0 + m_{2,1}c_1^0 + m_{2,2}c_2^0). \end{aligned} \tag{5}$$

We fix Y_2 to a constant value $B_0 = S(m_{2,2}^{-1}(m_{2,0}c_0^0 + m_{2,1}c_1^0 + m_{2,2}c_2^0 + S(A_0)))$. Then, we can simplify Equation (5) as follows:

$$\begin{aligned} & m_{2,0}S^{-1}(Y_0) + m_{2,1}S^{-1}(Y_1) = 0 \\ \Longleftrightarrow & m_{2,0}S^{-1}(Y_0) = -m_{2,1}S^{-1}(Y_1) \\ \Longleftrightarrow & S(m_{2,0}S^{-1}(Y_0)) = S(-m_{2,1}S^{-1}(Y_1)) \\ \Longleftrightarrow & S(m_{2,0})Y_0 = S(m_{2,1})Y_1. \end{aligned} \tag{6}$$

The S-box must satisfy Formula (4) for the above Equation (6) to be established successfully. We find that A_0 and B_0 are fixed and that Y_1 can be represented by Y_0 as $Y_1 = \frac{S(m_{2,0})}{S(m_{2,1})}Y_0$. Then, we have

$$\begin{aligned} \boldsymbol{X} &= (X_0, X_1, A_0) \in V_2, \\ \boldsymbol{Y} &= Y_0(1, \frac{S(m_{2,0})}{S(m_{2,1})}, 0) + (0, 0, B_0) \in V_2. \end{aligned}$$

When $n = 4$, we set $u = n-1 = 3$; then, V_u is a vector subspace spanned by $\{e_0, e_1, e_2\}$. As seen in Figure 2, we denote the input and output states of $\mathcal{F}_0$ as $\boldsymbol{X} = (X_0, X_1, X_2, A_0) \in V_u$ and $\boldsymbol{Y} = (Y_0, Y_1, Y_2, Y_3) \in \mathbb{F}_p^4$, respectively, where A_0 are fixed constants. When passing through the first nonlinear layer of $\mathcal{F}_0$, we have

$$\begin{aligned} S(A_0) &= m_{3,0}(S^{-1}(Y_0) - c_0^0) + m_{3,1}(S^{-1}(Y_1) - c_1^0) + m_{3,2}(S^{-1}(Y_2) - c_2^0) + m_{3,3}(S^{-1}(Y_3) - c_3^0) \\ &= m_{3,0}S^{-1}(Y_0) + m_{3,1}S^{-1}(Y_1) + m_{3,2}S^{-1}(Y_2) + m_{3,3}S^{-1}(Y_3) \\ &\quad - (m_{3,0}c_0^0 + m_{3,1}c_1^0 + m_{3,2}c_2^0 + m_{3,3}c_3^0) \\ &= \sum_{i=0}^{3} m_{3,i}S^{-1}(Y_i) - \sum_{i=0}^{3} m_{3,i}c_i^0. \end{aligned} \tag{7}$$

We fix Y_3 to a constant denoted as B_0, while Y_3 satisfies

$$m_{3,3}S^{-1}(Y_3) = \sum_{i=0}^{3} m_{3,i}c_i^0 + S(A_0).$$

Then, we can obtain

$$m_{3,0}S^{-1}(Y_0) + m_{3,1}S^{-1}(Y_1) + m_{3,2}S^{-1}(Y_2) = 0. \tag{8}$$

In order to simplify the equation, we set $(Y_1, Y_2) = (S(Q_1)Y_0, S(Q_2)Y_0)$ and bring (Y_1, Y_2) into Equation (8); then, we have

$$S^{-1}(Y_0)(m_{3,0} + m_{3,1}Q_1 + m_{3,2}Q_2) = 0.$$

Therefore, if (Q_1, Q_2) and Y_3 satisfy

$$\begin{cases} m_{3,0} + m_{3,1}Q_1 + m_{3,2}Q_2 = 0 \\ Y_3 = S(m_{3,3}^{-1}(\sum_{i=0}^{3} m_{3,i}c_i^0 + S(A_0))), \end{cases}$$

we have

$$\begin{aligned} \boldsymbol{X} &= (X_0, X_1, X_2, A_0) \in V_3, \\ \boldsymbol{Y} &= Y_0(1, Q_1, Q_2, 0) + (0, 0, 0, B_0) \in V_3. \end{aligned}$$

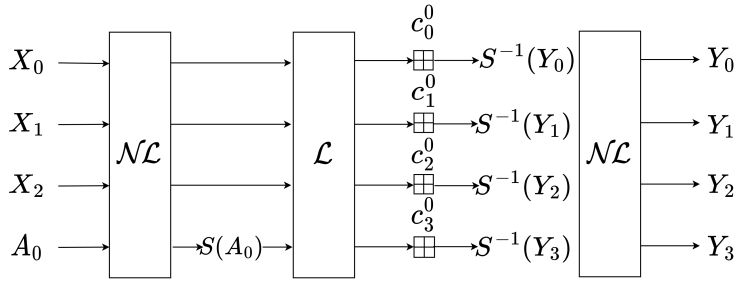

Figure 2. A detailed description of a specific trick with a state size of 4.

When $n \geq 4$, we set $u = n - 1$ in general, while V_{n-1} is a vector subspace spanned by $\{e_0, e_1, \ldots, e_{n-2}\}$. Similarly, the input and output states of $\mathcal{F}_0$ are in the form of $\boldsymbol{X} = (X_0, X_1, \ldots, X_{n-2}, A_0)$ and $\boldsymbol{Y} = (Y_0, Y_1, \ldots, Y_{n-1}) \in \mathbb{F}_p^n$, respectively. Let $A_0 \in \mathbb{F}_p$ be a fixed constant. When passing through the first nonlinear layer of $\mathcal{F}_0$, we have

$$S(A_0) = \sum_{i=0}^{n-1} m_{n-1,i}(S^{-1}(Y_i) - c_i^0) = \sum_{i=0}^{n-1} m_{n-1,i}S^{-1}(Y_i) - \sum_{i=0}^{n-1} m_{n-1,i}c_i^0. \tag{9}$$

We can fix Y_{n-1} to a constant denoted as B_0; then, Y_{n-1} fulfills

$$m_{n-1,n-1}S^{-1}(Y_{n-1}) = \sum_{i=0}^{n-1} m_{n-1,i}c_i^0 + S(A_{n-1}). \tag{10}$$

Just as for $n = 3$ and $n = 4$, we set $(Y_1, Y_2, \ldots, Y_{n-2}) = (S(Q_1)Y_0, S(Q_2)Y_0, \ldots, S(Q_{n-2})Y_0)$. By bringing $(Y_1, Y_2, \ldots, Y_{n-1})$ and the constant Y_{n-1} back into the Equation (9), we can obtain

$$S^{-1}(Y_0)(m_{n-1,0} + \sum_{i=1}^{n-1} m_{n-1,i}Q_i) = 0.$$

Therefore, if $(Q_1, Q_2, \ldots, Q_{n-1})$ and Y_{n-1} satisfy

$$\begin{cases} m_{n-1,0} + \sum_{i=1}^{n-1} m_{n-1,i} Q_i = 0 \\ Y_{n-1} = S(m_{n-1,n-1}^{-1}(\sum_{i=0}^{n-1} m_{n-1,i} c_i^0 + S(A_0))), \end{cases}$$

we have

$$\begin{aligned} \boldsymbol{X} &= (X_0, X_1, \ldots, X_{n-2}, A_0) \in V_{n-1}, \\ \boldsymbol{Y} &= Y_0(1, Q_1, Q_2, \ldots, Q_{n-2}, 0) + (\mathbf{0}^{n-1}, B_0) \in V_{n-1}. \end{aligned}$$

Let $\boldsymbol{Y}$ be the input to $\mathcal{F}_1$, where Y_0 is the only unknown variable. We define the output of $\mathcal{F}_1$ as $\boldsymbol{Z} = (Z_0, Z_1, \ldots, Z_{n-2}, C_0) \in V_{n-1}$, with $C_0 \in \mathbb{F}_p$ being a fixed constant. Considering the final position of the output from $\mathcal{F}_1$, a univariate equation is constructed with Y_0 as its variable, taking the form of

$$F(Y_0) = C_0. \tag{11}$$

With a valid Y_0, we can invariably infer an input $\boldsymbol{X}$ specifically tailored for the R-round permutation $\mathcal{P}$ that projects onto the vector subspace V_{n-1}.

3.3. Application to Grendel Hash Function

In this section, we build upon the full-round preimage attack on the *Grendel* hash function presented in [22] by employing the trick described in the previous section to decrease the degree and complexity of the polynomial system. Consider a security level denoted by s and a prime p that defines the field. We limit ourselves here to the case in which $p \geq 2^s$. The following are the details of our attack:

1. We first divide the *Grendel* permutation into two parts, $\mathcal{F}_0$ and $\mathcal{F}_1$, as before. The *Grendel* permutation consists of R rounds. Consider the *Grendel* hash function with the parameters $n = r + c$. The *Grendel* S-box, denoted as $S(x) : x \mapsto x^d \cdot \chi_p(x)$, satisfies Formula (4), which can be easily proven using Proposition 1. Similarly, we set $u = n - 1$ and let V_u be a vector subspace. The *Grendel* permutation takes an input $\boldsymbol{X} = (X_0, \ldots, X_{r-1}, \mathbf{0}^c)$, where $X_0, \ldots, X_{r-1}$ represent the input messages, and produces an output $\boldsymbol{Z} = (Z_0, \ldots, Z_{r-1}, \mathbf{0}^c)$. The initial value IV of *Grendel* is set to all zeros, and the last c elements of the output $\boldsymbol{Z}$ are also zeros. Consequently, when $\boldsymbol{X} \in V_u$ passes through $\mathcal{F}_0$ it results in $\boldsymbol{Y} = (Y_0, Y_1, \ldots, Y_{n-1}) \in V_u$. As stated in the previous section, we have Y_{n-1}, and $(Q_1, \ldots, Q_{n-1})$ satisfy

$$\begin{cases} m_{n-1,0} + \sum_{i=1}^{n-1} m_{n-1,i} Q_i = 0 \\ Y_{n-1} = \left(m_{n-1,n-1}^{-1}(\sum_{i=0}^{n-1} m_{n-1,i} c_i^0)\right)^3 \cdot \chi_p\left(m_{n-1,n-1}^{-1}(\sum_{i=0}^{n-1} m_{n-1,i} c_i^0)\right). \end{cases}$$

   Thus, for $\mathcal{F}_1$ there is only one unknown input variable Y_0. The subsequent processing can be carried out in a similar manner as described in [22].
2. According to [22], it can be observed that when $p \geq 2^{32}$, the probability of the Legendre symbol being ± 1 is greater than 99.99%. Therefore, we only consider guessing ± 1. Based on the previous step, $\mathcal{F}_1$ has an input $\boldsymbol{Y}$ with only one unknown variable Y_0. The $\mathcal{F}_1$ has $R - 2$ rounds; we must guess the number of Legendre symbols, which is provided by $l = n(R-3) + 1 = nR - 3n + 1$. Because the Legendre symbol of Y_0 only needs to be guessed in the first round of $\mathcal{F}_1$, while the other values are constant, the Legendre symbol is known. Consequently, there are at most $2^l = 2^{nR-3n+1}$ distinct sets of Legendre symbols to guess until the correct set of Legendre symbols is found.
3. After fixing the Legendre symbols, we can construct a polynomial with Y_0 as an unknown variable. The polynomial equation, as defined in Formula (11), has a degree

of $\mathcal{D} = d^{R-2}$. To determine the specific value of Y_0 we can employ the root-finding algorithm in Section 2.3.1. The complexity T_1 of the root-finding algorithm is

$$T_1 = \mathcal{O}(\mathtt{M}(d^{R-2}) \log(d^{R-2}) \log(d^{R-2} \cdot p)),$$
$$\mathtt{M}(d^{R-2}) = 63.43 \cdot d^{R-2} \log(d^{R-2}) \log(\log(d^{R-2})) + \mathcal{O}(d^{R-2} \log(d^{R-2})).$$

4. Upon obtaining the value of Y_0, we need to verify its validity. This requires checking the correctness of each guessed Legendre symbol. According to [22], for each set of guessed Legendre symbols we only need to verify three of them to exclude an invalid set. The complexity T_2 of computing a Legendre symbol [46] is evaluated as $\mathcal{O}(\sigma(\log \sigma)^2 \log(\log(\sigma)))$ for $\sigma = \log(p)$; therefore, the complexity of this step is $3 \cdot T_2$.

Upon obtaining a valid Y_0 according to the CICO definition, we can always deduce $\boldsymbol{X}$ such that they are mapped to the vector subspace V_u through the *Grendel* permutation. The overall computational complexity of this attack, denoted as T, is represented by

$$T = (T_1 + 3 \cdot T_2) \cdot 2^{nR-3n+1}.$$

Therefore, this particular instance is vulnerable to attack if $T \leq 2^s$. As summarized in Table 1, for a security level of $s = 128$ it can be observed that under different parameter settings we are able to perform two additional attacking rounds compared to the previous work [22]. However, our advances do not exceed the newly established security margin in [22].

In our investigation of the *Grendel* hash function, we have ascertained that capitalizing on the CICO problem to devise a univariate equation is feasible solely under the conditions $u = r = n - 1$. This premise holds because in this specific scenario it enables the generation of an intermediate variable intimately associated with the vector subspace V_u, ensuring that the hash output remains confined to this subspace.

3.4. The Gröbner Basis Attacks for the Grendel Hash Function

In this section, we employ the CICO problem to construct a multivariate equation system for the *Grendel* hash function instantiation. Similarly, we consider the message absorption size to be r, resulting in r hash digests being squeezed out after a *Grendel* permutation. To further analyze the complexity, we utilize the Gröbner basis attack method described in Section 2.3.2 and incorporate insights gained from our experimental observations.

Building upon our previous assumption of guessing the Legendre symbols, we delve deeper into the analysis by considering the introduction of intermediate variables to reduce the degree of the polynomials. Based on the presence of intermediate variables, we categorize our attacks into two scenarios: one without the introduction of intermediate variables, and another in which intermediate variables are introduced in each round.

Considering the *Grendel* hash function with input messages $(X_1, X_2, \ldots, X_{r-1})$ of size r and an IV set to all zeros, the *Grendel* permutation takes an input $\boldsymbol{X} = (X_1, X_2, \ldots, X_{r-1}, \mathbf{0}^c)$, where $\mathbf{0}^c$ denotes a vector of zeros with length c. The resulting output $\boldsymbol{Z} = (Z_0, \ldots, Z_{n-1})$ is subject to the CICO problem, where the input $\boldsymbol{X}$ belongs to the vector subspace V_r spanned by $\{e_0, \ldots, e_{r-1}\}$. To satisfy this condition, the last c positions of $\boldsymbol{Z}$, denoted as $C_0, \ldots, C_{c-1}$, are fixed constants. In the following attacks, we always set $r = c = \frac{2}{n}$. Consequently, we can construct a system of multivariate equations.

Without Intermediate Variables. Let $\boldsymbol{X}$ and $\boldsymbol{Z}$ be the input and output of the permutation. No additional intermediate variables, such as $\boldsymbol{Y}$, are introduced. We can build an equation system with c variables and c equations:

$$\begin{cases} F_0(X_0, X_1, \ldots, X_{c-1}) = C_0 \\ F_1(X_0, X_1, \ldots, X_{c-1}) = C_1 \\ \vdots \\ F_{c-1}(X_0, X_1, \ldots, X_{c-1}) = C_{c-1}. \end{cases}$$

We obtain a system of equations with c equations and c variables, where each equation has a degree of $\mathcal{D}_i = d^R, 0 \leq i \leq c-1$. It is evident that the degrees of the equations is much larger than the number of variables. Therefore, Formula (3) is used to calculate the complexity of the Gröbner basis algorithm. Specifically, when setting $d = 2$, we can compare the computational complexities of the Gröbner basis and FGLM algorithms. For a system of c equations in which each equation has a degree of 2^R, we can compute the upper bound on the regularity degree $\mathcal{D}_{reg}$ of the equation system and the zero-dimensional ideal $\mathcal{D}_{\mathcal{I}}$ as follows:

$$\mathcal{D}_{reg} \leq 1 + \sum_{i=0}^{c-1}(\mathcal{D}_i - 1) = (2^R - 1) \cdot c + 1, \qquad \mathcal{D}_{\mathcal{I}} \leq \prod_{i=0}^{c-1} \mathcal{D}_i = 2^{Rc}.$$

Computing the Gröbner basis with respect to the *grevlex* term order using Formula (3) exhibits asymptotic complexity:

$$T_G = nD_{reg} \cdot \binom{n + D_{reg} - 1}{D_{reg}}^{\omega} \leq c \cdot ((2^R - 1) \cdot c + 1) \cdot \binom{2^R \cdot c}{2^R \cdot c - c + 1}.$$

Then, applying a fast variant of the FGLM algorithm to perform the change of term order exhibits asymptotic complexity:

$$T_F = (\mathcal{D}_{\mathcal{I}})^{\omega} = 2^{Rc\omega}.$$

By evaluating T_G and T_F for $c = 2$, we find that

$$T_G = (2^{R+2} - 2) \times \binom{2^{R+1}}{2^{R+1} - 1} = (2^{R+1})^{\omega} \times (2^{R+2} - 2) \leq 2^{Rw+w+r+2},$$
$$T_F = 2^{2Rw}.$$

Based on this, it is clear that T_F is larger than T_G, implying that the FGLM algorithm becomes the bottleneck in the computation of Gröbner bases. As shown in Table 3 for $p = 2^{256}$ and $s = 128$, by setting $c = r = 2/n$, we can evaluate the number of rounds that can be susceptible to Gröbner basis attacks with varying computational complexities. Our practical tests regarding the actual degrees reached in the computation are given in Figure 3.

Intermediate Variables. Using the input and output of the *Grendel* permutation directly may be infeasible due to the high degree and dense nature of the polynomials involved. To overcome this challenge, one possible strategy is to introduce intermediate variables. This approach reduces the degrees in the equation system, thereby reducing the number of monomials, although it introduces additional variables. In each round of *Grendel* permutation, we introduce new variables to prevent an increase in degrees (Figure 4). Let $\boldsymbol{X}$ and $\boldsymbol{Y}^0 = (Y_0^0, \ldots, Y_{n-1}^0) \in \mathbb{F}_p^n$ represent the input and output of the nonlinear layer in the first round, respectively. The relationship between $\boldsymbol{X}$ and $\boldsymbol{Y}^0$ can be expressed through r equations of degree d and c equations of degree 1. Specifically, $Y_0^0 = X_0^d$, in accordance with the definition of the S-box (excluding the consideration of the Legendre symbol, as it is

determined based on the conjecture); consequently, we add n variables in each round. Then, we simply use the output values $C_0, \ldots, C_{c-1}$ to construct the system of equations except for the last one, meaning that we have $r + Rn$ variables and the same number of equations. Among these equations, there are Rn equations with a degree of d and r equations with a degree of 1.

Table 3. For a security margin of $s = 128$ and a modulus $p = 2^{256}$, by setting c = r = 2/n we can evaluate the number of rounds that can be susceptible to Gröbner basis attacks with varying computational complexities.

Instance (d, n)	Attacked Rounds with T_F	Attacked Rounds with T_G
(2, 4)	16	21
(2, 8)	8	10
(2, 12)	5	7
(3, 4)	12	17
(3, 8)	6	8
(3, 12)	4	5
(5, 4)	9	14
(5, 8)	4	7
(5, 12)	3	4

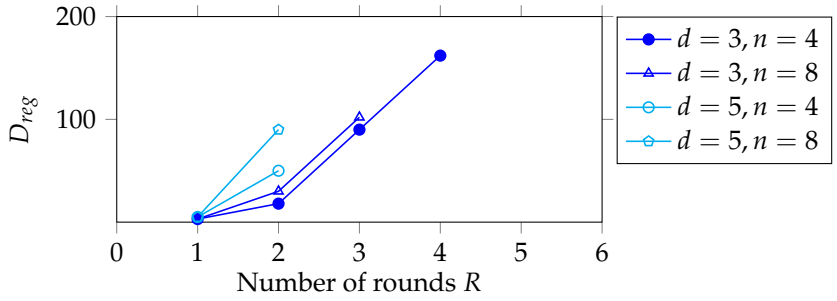

Figure 3. The values for D_{reg} in practice, where $d \in \{3, 5\}$, $n \in \{4, 8\}$, and the number of variables $n_v = n/2$.

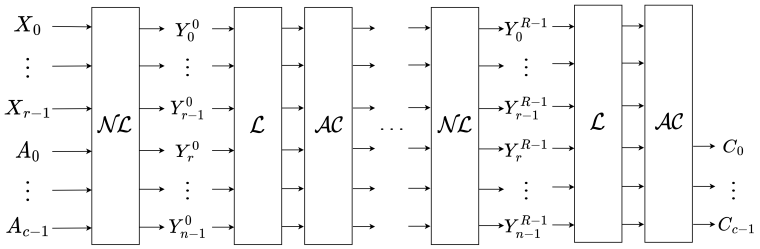

Figure 4. Overview of the introduction of intermediate variables in the Grendel permutation.

When n is large, indicating the presence of a greater number of intermediate variables, then D_{reg} becomes relatively small. Therefore, we utilize Formula (2) to evaluate the complexity of the Gröbner basis attack.

In summary, the complexity of the Gröbner basis attack on the *Grendel* hash function can be divided into three parts. The first part is the complexity of guessing the Legendre symbols, denoted as T_{guess}. The second part is the complexity of the Gröbner basis attack, denoted as T_{GB} (with specific calculations selected from the various scenarios mentioned

earlier). The third part is the complexity of verifying the Legendre symbols, denoted as T_{verify}. Lastly,

$$T_{guess} = 2^{(R-1)n+c},$$
$$T_{verify} = 3 \cdot \mathcal{O}(\sigma(\log \sigma)^2 \log(\log(\sigma))) \text{ for } \sigma = \log(p).$$

The overall complexity of the Gröbner basis attack for the *Grendel* hash function can be evaluated by

$$(T_{GB} + T_{verify}) \cdot T_{guess}.$$

Our practical tests regarding the actual degrees reached in the computation are given in Figure 5.

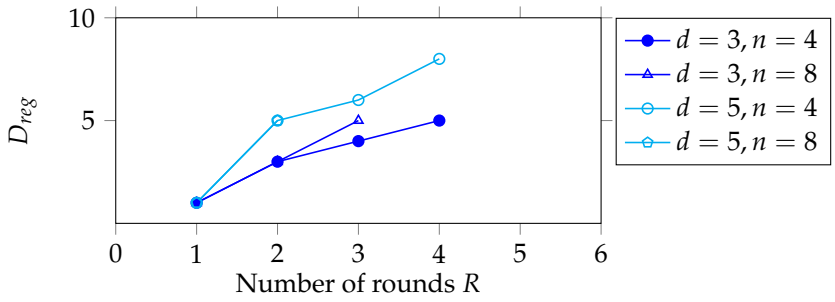

Figure 5. The values for D_{reg} in practice, where $d \in \{3, 5\}$, $n \in \{4, 8\}$, and the number of variables $n_v = n/2 + n * R$.

4. Conclusions

In this paper, we propose a preimage attack on the sponge hash function implemented with full rounds of the *Grendel* permutation, utilizing algebraic approaches. By introducing the CICO problem, we investigate the construction of univariate and multivariate equation systems for the *Grendel* hash function and employ different algorithms to solve these equations, resulting in new analytical findings. This provides additional insights into the factors that designers should consider when developing arithmetization-oriented cryptographic primitives in response to the CICO problem. Moreover, our research highlights the influence that the selection of distinct algebraic methods for equation construction can have on the security analysis of cryptographic primitives.

Further Discussion. It is worthwhile to investigate the potential of merging various algebraic methods for the analysis of arithmetization-oriented cryptographic primitives in the future. Similar to the algebraic techniques used in this study, we introduce the CICO problem and employ specific strategies to bypass one round of the cryptographic algorithm, enabling further equation construction or utilization of alternative techniques to build equation systems. This study has focused solely on the analysis of permutation in SPN structures; however, in the future it may be possible to extend this method to Feistel structures.

Author Contributions: Conceptualization, J.N. and G.W.; methodology, J.Z.; validation, J.N.; formal analysis, R.L. and Y.S.; writing—original draft preparation, J.N.; writing—review and editing, R.L. and Y.S.; supervision, G.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Key Research and Development Program of China (2022YFB2701900), the National Natural Science Foundation of China (No. 62072181), and the Shanghai Trusted Industry Internet Software Collaborative Innovation Center.

Data Availability Statement: All data are contained within the article.

Acknowledgments: The authors would like to thank the anonymous reviewers for their helpful comments.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Albrecht, M.R.; Grassi, L.; Rechberger, C.; Roy, A.; Tiessen, T. MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity. In *Advances in Cryptology—ASIACRYPT 2016, Proceedings of the 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, 4–8 December 2016, Proceedings, Part I*; Cheon, J.H., Takagi, T., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2016; Volume 10031, pp. 191–219. [CrossRef]
2. Albrecht, M.R.; Grassi, L.; Perrin, L.; Ramacher, S.; Rechberger, C.; Rotaru, D.; Roy, A.; Schofnegger, M. Feistel Structures for MPC, and More. In *Computer Security—ESORICS 2019, Proceedings of the 24th European Symposium on Research in Computer Security, Luxembourg, 23–27 September 2019, Proceedings, Part II*; Sako, K., Schneider, S.A., Ryan, P.Y.A., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2019; Volume 11736, pp. 151–171. [CrossRef]
3. Grassi, L.; Lüftenegger, R.; Rechberger, C.; Rotaru, D.; Schofnegger, M. On a Generalization of Substitution-Permutation Networks: The HADES Design Strategy. In *Advances in Cryptology—EUROCRYPT 2020, Proceedings of the 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, 10–14 May 2020, Proceedings, Part II*; Canteaut, A., Ishai, Y., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2020; Volume 12106, pp. 674–704. [CrossRef]
4. Grassi, L.; Khovratovich, D.; Rechberger, C.; Roy, A.; Schofnegger, M. Poseidon: A New Hash Function for Zero-Knowledge Proof Systems. In *USENIX Security 2021, Proceedings of the 30th USENIX Security Symposium, 11–13 August 2021*; Bailey, M., Greenstadt, R., Eds.; USENIX Association; Springer: Berlin/Heidelberg, Germany, 2021; pp. 519–535.
5. Ha, J.; Kim, S.; Choi, W.; Lee, J.; Moon, D.; Yoon, H.; Cho, J. Masta: An HE-Friendly Cipher Using Modular Arithmetic. *IEEE Access* **2020**, *8*, 194741–194751. [CrossRef]
6. Dobraunig, C.; Grassi, L.; Helminger, L.; Rechberger, C.; Schofnegger, M.; Walch, R. Pasta: A Case for Hybrid Homomorphic Encryption. *Iacr Trans. Cryptogr. Hardw. Embed. Syst.* **2023**, *2023*, 30–73. [CrossRef]
7. Dobraunig, C.; Grassi, L.; Guinet, A.; Kuijsters, D. Ciminion: Symmetric Encryption Based on Toffoli-Gates over Large Finite Fields. In *Advances in Cryptology—EUROCRYPT 2021, Proceedings of the 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, 17–21 October 2021, Proceedings, Part II*; Canteaut, A., Standaert, F., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2021; Volume 12697, pp. 3–34. [CrossRef]
8. Ashur, T.; Mahzoun, M.; Toprakhisar, D. Chaghri—A FHE-friendly Block Cipher. In *CCS 2022, Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, Los Angeles, CA, USA, 7–11 November 2022*; Yin, H., Stavrou, A., Cremers, C., Shi, E., Eds.; ACM: New York, NY, USA, 2022; pp. 139–150. [CrossRef]
9. Grassi, L.; Onofri, S.; Pedicini, M.; Sozzi, L. Invertible Quadratic Non-Linear Layers for MPC-/FHE-/ZK-Friendly Schemes over Fnp Application to Poseidon. *IACR Trans. Symmetric Cryptol.* **2022**, *2022*, 20–72. [CrossRef]
10. Szepieniec, A. On the Use of the Legendre Symbol in Symmetric Cipher Design. Paper 2021/984. *Cryptol. ePrint Arch.* 2021. Available online: https://eprint.iacr.org/2021/984 (accessed on 17 June 2023).
11. Damgård, I. On the Randomness of Legendre and Jacobi Sequences. In *Advances in Cryptology—CRYPTO 1988, Proceedings of the 8th Annual International Cryptology Conference, Santa Barbara, CA, USA, 21–25 August 1988, Proceedings*; Goldwasser, S., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1988; Volume 403, pp. 163–172. [CrossRef]
12. Peralta, R. On the Distribution of Quadratic Residues and Nonresidues Modulo a Prime Number. *Math. Comput.* **1992**, *58*, 433–440. [CrossRef]
13. Mauduit, C.; Sárközy, A. On finite pseudorandom binary sequences I: Measure of pseudorandomness, the Legendre symbol. *Acta Arith.* **1997**, *82*, 365–377. [CrossRef]
14. Maksymovych, V.; Shabatura, M.; Harasymchuk, O.; Shevchuk, R.; Sawicki, P.; Zajac, T. Combined Pseudo-Random Sequence Generator for Cybersecurity. *Sensors* **2022**, *22*, 9700. [CrossRef] [PubMed]
15. Tóth, V. Collision and avalanche effect in families of pseudorandom binary sequences. *Period. Math. Hung.* **2007**, *55*, 185–196. [CrossRef]
16. Gyarmati, K.; Mauduit, C.; Sárközy, A. The cross-correlation measure for families of binary sequences. In *Applied Algebra and Number Theory*; Larcher, G., Pillichshammer, F., Winterhof, A., Xing, C., Eds.; Number Theory; Cambridge University Press: Cambridge, UK, 2014; pp. 126–143. [CrossRef]
17. Khovratovich, D. Key recovery attacks on the Legendre PRFs within the birthday bound. Paper 2019/862. *Cryptol. Eprint Arch.* 2019. Available online: https://eprint.iacr.org/2019/862 (accessed on 17 June 2023).
18. Beullens, W.; Beyne, T.; Udovenko, A.; Vitto, G. Cryptanalysis of the Legendre PRF and Generalizations. *IACR Trans. Symmetric Cryptol.* **2020**, *2020*, 313–330. [CrossRef]
19. Kaluđerović, N.; Kleinjung, T.; Kostić, D. Cryptanalysis of the generalised Legendre pseudorandom function. *Open Book Series* **2020**, *4*, 267–282. [CrossRef]
20. Seres, I.A.; Horváth, M.; Burcsi, P. The Legendre Pseudorandom Function as a Multivariate Quadratic Cryptosystem: Security and Applications. Paper 2021/182. *Cryptol. Eprint Arch.* 2021. Available online: https://eprint.iacr.org/2021/182 (accessed on 17 June 2023).
21. Shallue, C.J. Permutation polynomials of finite fields. *arXiv* **2012**, arXiv:1211.6044.

22. Grassi, L.; Khovratovich, D.; Rønjom, S.; Schofnegger, M. The Legendre Symbol and the Modulo-2 Operator in Symmetric Schemes over Fnp Preimage Attack on Full Grendel. *IACR Trans. Symmetric Cryptol.* **2022**, *2022*, 5–37. [CrossRef]
23. Biham, E.; Shamir, A. Differential Cryptanalysis of DES-like Cryptosystems. In *Advances in Cryptology—CRYPTO 1990, Proceedings of the 10th Annual International Cryptology Conference, Santa Barbara, CA, USA, 11–15 August 1990, Proceedings*; Menezes, A., Vanstone, S.A., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1990; Volume 537; pp. 2–21. [CrossRef]
24. Matsui, M. Linear Cryptanalysis Method for DES Cipher. In *Advances in Cryptology—EUROCRYPT 1993, Proceedings of the Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, 23–27 May 1993, Proceedings*; Helleseth, T., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1993, Volume 765; pp. 386–397. [CrossRef]
25. Ashur, T.; Dhooghe, S. MARVELlous: A STARK-Friendly Family of Cryptographic Primitives. Paper 2018/1098. *Cryptol. Eprint Arch.* 2018. Available online: https://eprint.iacr.org/2018/1098 (accessed on 17 June 2023).
26. Beyne, T.; Canteaut, A.; Dinur, I.; Eichlseder, M.; Leander, G.; Leurent, G.; Naya-Plasencia, M.; Perrin, L.; Sasaki, Y.; Todo, Y.; et al. Out of Oddity - New Cryptanalytic Techniques Against Symmetric Primitives Optimized for Integrity Proof Systems. In *Advances in Cryptology—CRYPTO 2020, Proceedings of the 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, 17–21 August 2020, Proceedings, Part III*; Micciancio, D., Ristenpart, T., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2020, Volume 12172; pp. 299–328. [CrossRef]
27. Eichlseder, M.; Grassi, L.; Lüftenegger, R.; Øygarden, M.; Rechberger, C.; Schofnegger, M.; Wang, Q. An Algebraic Attack on Ciphers with Low-Degree Round Functions: Application to Full MiMC. In *Advances in Cryptology—ASIACRYPT 2020, Proceedings of the 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, 7–11 December 2020, Proceedings, Part I*; Moriai, S., Wang, H., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2020; Volume 12491, pp. 477–506. [CrossRef]
28. Bouvier, C.; Canteaut, A.; Perrin, L. On the algebraic degree of iterated power functions. *Des. Codes Cryptogr.* **2023**, *91*, 997–1033. [CrossRef]
29. Cui, J.; Hu, K.; Wang, M.; Wei, P. On the Field-Based Division Property: Applications to MiMC, Feistel MiMC and GMiMC. In *Advances in Cryptology—ASIACRYPT 2022, Proceedings of the 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, 5–9 December 2022, Proceedings, Part III*; Agrawal, S., Lin, D., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2022; Volume 13793; pp. 241–270. [CrossRef]
30. Liu, F.; Anand, R.; Wang, L.; Meier, W.; Isobe, T. Coefficient Grouping: Breaking Chaghri and More. In *Advances in Cryptology—EUROCRYPT 2023, Proceedings of the 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, 23–27 April 2023, Proceedings, Part IV*; Hazay, C., Stam, M., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2023, Volume 14007; pp. 287–317. [CrossRef]
31. Grassi, L.; Rechberger, C.; Rotaru, D.; Scholl, P.; Smart, N.P. MPC-Friendly Symmetric Key Primitives. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S., Eds.; ACM: New York, NY, USA, 2016; pp. 430–443. [CrossRef]
32. Lai, X. Higher order derivatives and differential cryptanalysis. In *Communications and Cryptography: Two Sides of One Tapestry*; Springer: Berlin/Heidelberg, Germany, 1994; pp. 227–233.
33. Knudsen, L.R. Truncated and Higher Order Differentials. In *Proceedings of the Fast Software Encryption: Second International Workshop, Leuven, Belgium, 14–16 December 1994, Proceedings*; Lecture Notes in Computer Science; Preneel, B., Ed.; Springer: Berlin/Heidelberg, Germany, 1994; Volume 1008, pp. 196–211. [CrossRef]
34. Jakobsen, T.; Knudsen, L.R. The Interpolation Attack on Block Ciphers. In *FSE '97, Proceedings of the Fast Software Encryption, 4th International Workshop, Haifa, Israel, 20–22 January 1997, Proceedings*; Biham, E., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1997; Volume 1267, pp. 28–40. [CrossRef]
35. Bariant, A.; Bouvier, C.; Leurent, G.; Perrin, L. Algebraic Attacks against Some Arithmetization-Oriented Primitives. *IACR Trans. Symmetric Cryptol.* **2022**, *2022*, 73–101. [CrossRef]
36. Nagell, T. Euler's Criterion and Legendre's Symbol. In *Introduction to Number Theory*; Wiley: Hoboken, NJ, USA, 1951; p. 144.
37. von zur Gathen, J.; Gerhard, J. *Modern Computer Algebra*, 3rd ed.; Cambridge University Press: Cambridge, UK, 2013.
38. Buchberger, B. A theoretical basis for the reduction of polynomials to canonical forms. *SIGSAM Bull.* **1976**, *10*, 19–29. [CrossRef]
39. Faugere, J.C. A new efficient algorithm for computing Gröbner bases (F4). *J. Pure Appl. Algebra* **1999**, *139*, 61–88. [CrossRef]
40. Faugere, J.C. A new efficient algorithm for computing Gröbner bases without reduction to zero (F 5). In Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation, Lille, France, 7–10 July 2002; pp. 75–83.
41. Faugère, J.; Gianni, P.M.; Lazard, D.; Mora, T. Efficient Computation of Zero-Dimensional Gröbner Bases by Change of Ordering. *J. Symb. Comput.* **1993**, *16*, 329–344. [CrossRef]
42. Bettale, L.; Faugère, J.; Perret, L. Solving polynomial systems over finite fields: Improved analysis of the hybrid approach. In *ISSAC'12, Proceedings of the International Symposium on Symbolic and Algebraic Computation, Grenoble, France, 22–25 July 2012*; van der Hoeven, J., van Hoeij, M., Eds.; ACM: New York, NY, USA, 2012; pp. 67–74. [CrossRef]
43. Bardet, M.; Faugère, J.; Salvy, B. On the complexity of the F5 Gröbner basis algorithm. *J. Symb. Comput.* **2015**, *70*, 49–70. [CrossRef]
44. Bertoni, G.; Daemen, J.; Peeters, M.; Van Assche, G. Sponge functions. In Proceedings of the ECRYPT Hash Workshop, Barcelona, Spain, 24–25 May 2007; Volume 2007.

45. Bertoni, G.; Daemen, J.; Peeters, M.; Assche, G.V. On the Indifferentiability of the Sponge Construction. In *Advances in Cryptology—EUROCRYPT 2008, Proceedings of the 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, 13–17 April 2008, Proceedings*; Smart, N.P., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2008; Volume 4965, pp. 181–197. [CrossRef]
46. Brent, R.P.; Zimmermann, P. An $O(M(n)\log n)$ Algorithm for the Jacobi Symbol. In *Algorithmic Number Theory, Proceedings of the 9th International Symposium, ANTS-IX, Nancy, France, 19–23 July 2010, Proceedings*; Hanrot, G., Morain, F., Thomé, E., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6197, pp. 83–95. [CrossRef]