IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 0, No. 0, pp. 0–0. DOI:10.46586/tosc.v0.i0.0-0

Algebraic Attacks on RAIN and AIM Using Equivalent Representations

Fukang Liu[1], Mohammad Mahzoun[2], Morten Øygarden[3], Willi Meier[4]

[1] Tokyo Institute of Technology, Tokyo, Japan
liu.f.ad@m.titech.ac.jp
[2] Eindhoven University of Technology, Netherlands
mail@mahzoun.me
[3] Simula UiB, Bergen, Norway
morten.oygarden@simula.no
[4] University of Applied Sciences and Arts Northwestern Switzerland (FHNW), Windisch, Switzerland
willimeier48@gmail.com

Abstract. Designing novel symmetric-key primitives for advanced protocols like secure multiparty computation (MPC), fully homomorphic encryption (FHE) and zero-knowledge proof systems (ZK), has been an important research topic in recent years. Many such existing primitives adopt quite different design strategies from conventional block ciphers. Notable features include that many of these ciphers are defined over a large finite field, and that a power map is commonly used to construct the nonlinear component due to its efficiency in these applications as well as its strong resistance against the differential and linear cryptanalysis. In this paper, we target the MPC-friendly ciphers AIM and RAIN used for the post-quantum signature schemes AIMer (CCS 2023 and NIST PQC Round 1 Additional Signatures) and Rainier (CCS 2022), respectively. Specifically, we can find equivalent representations of 2-round RAIN and full-round AIM, respectively, which make them vulnerable to either the polynomial method, or the crossbred algorithm, or the fast exhaustive search attack. Consequently, we can break 2-round RAIN with the 128/192/256-bit key in only $2^{111}/2^{170}/2^{225}$ bit operations. For full-round AIM with the 128/192/256-bit key, we could break them in $2^{136.2}/2^{200.7}/2^{265}$ bit operations, which are equivalent to about $2^{115}/2^{178}/2^{241}$ calls of the underlying primitives. In particular, our analysis indicates that AIM does not reach the required security levels by the NIST competition.

Keywords: RAIN · AIM · equivalent representation · overdefined system · algebraic attack

1 Introduction

Developing novel symmetric-key primitives with applications to MPC, FHE and ZK has become an important research topic for their real-world impacts. Since the block cipher LowMC [ARS+15] published at EUROCRYPT 2015, there haven been a number of such novel primitives [AGR+16, CHK+21, GLR+20, GKL+22, DKR+22, BBC+23, DEG+18, GAH+23, GLR+20, MJSC16, DGGK21, GØSW23, HKL+22, CCF+16, DGH+21, AAB+20, AMT22, SLST23, GKR+21]. Especially, some ZK-friendly hash functions like Poseidon [GKR+21] and Rescue [AAB+20] were immediately adopted in real-world blockchains for their high efficiency in this application, and some MPC-friendly primitives like LowMC [ARS+15], Rain [DKR+22] and AIM [KHS+22] have been used to build post-quantum signature schemes with the MPC-in-the-head technique [IKOS07].

While traditional symmetric-key primitives typically work over $\mathbb{F}_2$, many of the new primitives are designed over large finite fields, though some are still defined over $\mathbb{F}_2$, but with new structures. This poses new challenges in terms of developing generic or dedicated cryptanalysis techniques to understand their security. Since this has been a largely unexplored area, many of these ciphers eventually turned out to be vulnerable against new attacks against their less-understood components, like the guess-and-determine attack on FLIP [DLR16], Gröbner basis attack on Jarvis and Friday [ACG+19], various algebraic attacks on LowMC [LIM21, LSW+22, Din21a, LMSI22, BBDV20, BBVY21], linearization attack on Rasta [LSMI21], higher-order differential attacks on MiMC [EGL+20, BCD+20, BCP23] and Chaghri [LAW+23, LGB+23] over large finite fields, and algebraic attacks on Rubato over rings [GAH+23], just to name a few. After years of effort to understand the security of these novel primitives, there have been some useful cryptanalytic tools developed for designers. However, this does not imply that the designers could exclude all potential attacks for their new proposals, especially when the ciphers are adopting new and less-understood structures.

In this paper, we mainly focus on the latest MPC-friendly ciphers RAIN (CCS 2022) [DKR+22] and AIM (CCS 2023) [KHS+22]. The two ciphers are tailored to specific post-quantum signature schemes Rainier and AIMer built with the MPC-in-the-head technique, respectively. Especially, AIMer is one of NIST PQC Round 1 Additional Signatures[1] that were released in July, 2023. In the signature schemes Rainier and AIMer, the public key is a single plaintext-ciphertext pair, and the secret key is the key for this plaintext-ciphertext pair. Consequently, the security of these two signature schemes relies on the difficulty to recover the secret key of RAIN and AIM with only a single known plaintext-ciphertext pair, respectively. While there are some successful attacks [Din21a, LMSI22, BBDV20, BBVY21] on LowMC in this setting by exploiting the low-degree properties of its 3-bit S-box, RAIN and AIM are quite different from LowMC as their nonlinear components are over large finite fields and their algebraic degree is high or maximal. Therefore, the designers made very aggressive choices for the secure number of rounds for the two ciphers in order to improve the efficiency of the signature schemes. Specifically, it is claimed that 3 rounds are secure for RAIN and the designers also proposed to use 4 rounds to further increase the security margin. Moreover, by the designers' analysis, even 2 rounds of RAIN cannot be broken. For AIM, although it is different from common block ciphers, we can view it as a 2-round primitive.

Our Contributions. In spite of the high agebraic degree of the nonlinear components of RAIN and AIM, we could identify nontrivial low-degree equivalent representations for 2-round RAIN and full-round AIM, which make them vulnerable to either the polynomial method [Din21a], crossbred algorithm [JV17] or the fast exhaustive search attack [BCC+10]. The main idea is to set up a system of equations in a variable related to the secret key, i.e. they are related by a high-degree equation. After solving the system of equations to recover this variable, the secret key can be then efficiently computed, and the key-recovery attack is achieved. Specifically, we could

- break 2-round RAIN using the polynomial method or the crossbred algorithm with time complexity significantly smaller than 2^n bit operations, as shown in Table 2;
- significantly improve the naive brute force attack on AIM with the fast exhaustive search (FES) method to solve a system of low-degree equations, as shown in Table 4.

Especially, our attack on AIM has motivated the AIM team to change the used S-boxes. More details of the revised version of AIM called AIM2 can be referred to [KHSL23].

[1] https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures

Table 1: Summary of the time complexity to break 2-round RAIN and full-round AIM with an n-bit key, which is given in the number of equivalent calls of the ciphers. Moreover, the number of bit operations of 2-round RAIN and full-round AIM are both estimated as n^3, just as in [ZWY+23, KHS+22].

Target	n	128	192	256
2-Round Rain	polynomial method (Sect. 4)	2^{97}	2^{149}	2^{201}
	crossbred algorithm (Sect. 4)	2^{90}	2^{147}	2^{204}
	guess and determine [ZWY+23]	$2^{120.3}$	$2^{180.4}$	$2^{243.1}$
full-round AIM	fast exhaustive search (Sect. 5)	2^{115}	2^{178}	2^{241}
	guess-and-determine [ZWY+23]	$2^{125.7}$	$2^{186.5}$	$2^{254.4}$

Comparison with [ZWY+23]. There is a parallel work [ZWY+23] on the cryptanalysis of 2-round RAIN and full-round AIM, which will appear at ASIACRYPT 2023. The main technique in [ZWY+23] is to linearize the inverse function $y = x^{-1}$ over $\mathbb{F}_{2^n}$ when n is an even number, thus resulting in a guess-and-determine attack. By this nature, their attacks will fail for RAIN and AIM if n is an odd number, while our attacks are not affected by this change. In addition, their attacks cannot reach 3 rounds of RAIN, while our attacks have this potential if weak linear layers are used. Moreover, compared with the attacks in [ZWY+23], our attacks have a lower time complexity, as can be seen in Table 1. Especially, we found that the first two countermeasures proposed in Sect. 5.2 of [ZWY+23] for AIM, i.e., adding different constants before each S-box in the first round or using odd n, cannot prevent our attacks, which further demonstrates the advantage of our technique.

Organization. In Sect. 2, we briefly introduce the relation between polynomials over $\mathbb{F}_{2^n}$ and $\mathbb{F}_2^n$, and revisit some known techniques to solve the system of multivariate Boolean equations. Then, we show in Sect. 3 how to derive an overdefined system of quadratic Boolean equations from a special form of polynomial equations over $\mathbb{F}_{2^n}$. In Sect. 4, we will present the algebraic attacks on 2 rounds of RAIN. The fast exhaustive search attack on full-round AIM is given in Sect. 5. Finally, the paper is concluded in Sect. 6.

2 Solving Multivariate Nonlinear Boolean Equations

It is well-known that solving multivariate nonlinear equations is NP-hard, even if the equations are over $\mathbb{F}_2$, i.e. Boolean equations. In algebraic attacks on cryptographic primitives, the attackers first model the to-be-solved problem (e.g., key recovery or finding a preimage/collision) with a system of nonlinear equations, and then solve the equation system with generic techniques like Gröbner basis method [Buc65, Fau99, Fau02], XL algorithm [CKPS00], fast exhaustive search (FES) [BCC+10], crossbred algorithm [JV17] or polynomial method [LPT+17, BKW19, Din21b, Din21a]. Developing generic equation-solving techniques is always challenging. In most cases, how to model the problem with better algebraic equations is crucial to improve the algebraic attack because attackers can exploit some algebraic structures of the primitives to set up equations that can be efficiently solved.

Since our target primitives RAIN [DKR+22] and AIM [KHS+22] are both defined over the finite field $\mathbb{F}_{2^n}$, and there exists an isomorphism between $\mathbb{F}_2^n$ and $\mathbb{F}_{2^n}$, e.g., see Chapter 6 of [RHPScKK07], we only focus on Boolean equations in this paper. Specifically, each polynomial $p(y_1, \ldots, y_t)$ in the polynomial ring $\mathbb{F}_{2^n}[y_1, \ldots, y_t]$ can be written as

$$p(y_1, \ldots, y_t) = \sum_{i_1=0}^{2^n-1} \cdots \sum_{i_t=0}^{2^n-1} u_{i_1,\ldots,i_t} y_1^{i_1} \ldots y_t^{i_t},$$

where $u_{i_1,\ldots,i_t} \in \mathbb{F}_{2^n}$ is the coefficient. Given any irreducible polynomial for $\mathbb{F}_{2^n}$, the polynomial p can be equivalently described by n vectorial Boolean polynomials $p_1, \ldots, p_n$ in nt Boolean variables, and the algebraic degree of each Boolean polynomial p_i is

$$\max\{\sum_{i=1}^{t} \mathrm{Hw}(i_j) : u_{i_1,\ldots,i_t} \neq 0\},$$

where $\mathrm{Hw}(i)$ denotes the Hamming weight of the integer i, i.e. the number of 1 in the binary representation of i. Indeed, we can have the following formal definition of the algebraic degree of a polynomial in $\mathbb{F}_{2^n}[y_1, \ldots, y_t]$:

Definition 1. Let

$$p(y_1, \ldots, y_t) = \sum_{i_1=0}^{2^n-1} \cdots \sum_{i_t=0}^{2^n-1} u_{i_1,\ldots,i_t} y_1^{i_1} \ldots y_t^{i_t},$$

be a polynomial in $\mathbb{F}_{2^n}[y_1, \ldots, y_t]$ where $u_{i_1,\ldots,i_t} \in \mathbb{F}_{2^n}$ is the coefficient, the algebraic degree of p denoted by $\mathrm{Deg}(p)$ is defined as

$$\mathrm{Deg}(p) = \max\{\sum_{i=1}^{t} \mathrm{Hw}(i_j) : u_{i_1,\ldots,i_t} \neq 0\}.$$

In other words, for a system of equations defined over $\mathbb{F}_{2^n}$, we can first convert them into a system of Boolean equations with known algebraic degree, and then solve them with generic equation-solving techniques developed for Boolean equations. Let us consider m Boolean polynomials $f_1, \ldots, f_m$ in n ($n \leq m$) variables $x_1, \ldots, x_n$, and their algebraic degree is upper bounded by d. For the following system of m equations:

$$f_1(x_1, \ldots, x_n) = 0, \quad \ldots, \quad f_m(x_1, \ldots, x_n) = 0,$$

there are some known techniques to solve them and we introduce them one by one.

2.1 Fast Exhaustive Search

The fast exhaustive search method is basically based on an efficient way to evaluate a Boolean polynomial over $\{0,1\}^n$, i.e. how to efficiently compute $f_i(x_1, \ldots, x_n)$ for all $(x_1, \ldots, x_n) \in \{0,1\}^n$. Since the algebraic degree of each f_i is upper bounded by d, there are $\binom{n}{\leq d}$ possible terms in $f_i(x_1, \ldots, x_n)$ where $\binom{n}{\leq d} = \sum_{i=0}^{d} \binom{n}{i}$. Therefore, the naive polynomial evaluation would require about $2^n \cdot \binom{n}{\leq d}$ bit operations for each f_i.

In [BCC+10], the authors found that evaluating f_i over $\{0,1\}^n$ only requires $d \cdot 2^n$ bit operations after an initialization phase of complexity $\mathcal{O}(n^{2d})$ (see Theorem 1 in [BCC+10]). Indeed, according to the explanation in Section 4 of [BCC+10], the term n^{2d} comes from n^d times of evaluations of the polynomial f_i, which is itself upper bounded by n^d. This suggests the complexity of such an initialization phase is indeed about $n^d \cdot \binom{n}{\leq d}$ bit operations.

With such an efficient way to evaluate f_i, given n polynomials of algebraic degree d, the attackers can first evaluate n_1 polynomials $f_1, \ldots, f_{n_1}$ in parallel. At this phase, it is expected to generate 2^{n-n_1} candidate solutions. For each such candidate solution, it can be further checked against the remaining polynomials with the naive polynomial evaluation aided with an early-aborting strategy. In this way, the time complexity to find the solution is estimated as about $4d \cdot \log_2 n \cdot 2^n$ bit operations (see Theorem 2 and its generalization at Section 5 of [BCC+10]). The memory complexity is estimated as $n \cdot \binom{n}{\leq d}$ bits, which is used to store the polynomials. Note that the time complexity of the initialization phase

is upper bounded by $n_1 \cdot n^d \cdot \binom{n}{\leq d} \leq n \cdot n^d \cdot \binom{n}{\leq d}$ bit operations according to the above careful analysis, and this is not taken into account in $4d \cdot \log_2 n \cdot 2^n$.

At EUROCRYPT 2021, Dinur proposed a memory-efficient Möbius transform [Din21a] to evaluate a Boolean polynomial $f_i(x_1, \ldots, x_n)$ of algebraic degree d over $\{0,1\}^n$. This method does not require the costly initialization phase of time complexity $\mathcal{O}(n^{2d})$. Moreover, the time complexity and memory complexity of this method are $d \cdot 2^n$ bit operations and $n \cdot \binom{n}{\leq d}$ bits, respectively. Hence, it can be trivially applied to the fast exhaustive search and the whole time complexity is kept the same, i.e. it is still $4d \cdot \log_2 n \cdot 2^n$ bit operations. The memory complexity is upper bounded by $n \cdot n \cdot \binom{n}{\leq d}$ bits.

2.2 Polynomial Method

The polynomial method to solve n multivariate Boolean equations of algebraic degree d in n variables was first proposed at SODA 2017 [LPT+17]. There have been 3 improved variants [BKW19, Din21b, Din21a] and we focus on the latest variant [Din21a] proposed by Dinur at EUROCRYPT 2021 since it gives the accurate time complexity in bit operations. The general idea is to first find the solutions to a smaller number of polynomial equations from these n polynomial equations, and then check the solutions against the remaining equations. The technique mainly exists in how to efficiently enumerate the solutions of that small system of equations by the polynomial interpolation and evaluation. We refer the interested readers to [Din21a, LMSI22] for the detailed explanation. For a system of n Boolean equations of algebraic degree d ($d \geq 2$) in n variables, the time and memory complexity to solve it are estimated as $n^2 \cdot 2^{(1-1/2.7d)n}$ bit operations and $n^2 \cdot 2^{(1-1/1.35d)n}$ bits, respectively. In the case $d = 2$, the time and memory complexity are $n^2 \cdot 2^{0.815n}$ bit operations and $n^2 \cdot 2^{0.63n}$ bits, respectively.

Due to the relatively low time complexity of the polynomial method, it has become an important tool to evaluate the resistance against algebraic attacks, especially when it is possible to construct low-degree equations. For the downsides of this method, apart from its requirement of huge memory, it also cannot take advantage of the feature of the overdefined system of equations. Specifically, even if we have $m > n$ equations in n variables, the polynomial method will give the same time complexity as in the case of n equations.

2.3 Crossbred Algorithm

The idea behind Joux and Vitse's crossbred algorithm [JV17] for finding solutions to a polynomial system $\mathcal{F}$ of m quadratic equations $f_1 = 0, \ldots, f_m = 0$ in n variables, is to derive a set of polynomials in the ideal of $\mathcal{F}$ that is easy to solve once certain variables have been fixed. The algorithm considers three integer inputs (D, n_D, d_1). In a *preprocessing* step reminiscent of the XL algorithm [CKPS00], the original polynomials of $\mathcal{F}$ are extended to the degree $\leq D$ Macaulay matrix $\mathcal{M}_{\leq D}(\mathcal{F})$. This is the matrix whose rows are given by $\mu \cdot f$ where $f \in \mathcal{F}$ and μ is a monomial of degree $\leq D - 2$, and the columns represent all monomials of degree $\leq D$. A formal definition of the Macaulay matrix $\mathcal{M}_{\leq D}(\mathcal{F})$ can be found below:

Definition 2. [JV17] Let $\mathbb{K}$ be a field. For any integer b, let T_b be an ordered set of the monomials of $\mathbb{K}[x_1, \ldots, x_n]$ of degree smaller than or equal to b. The degree D Macaulay matrix of $\mathcal{F}$, denoted by $\mathcal{M}_{\leq D}(\mathcal{F})$, is the matrix with coefficients in $\mathbb{K}$ whose columns are indexed by T_b, whose lines are indexed by the set $\{(\mu, f_i) \mid 1 \leq i \leq m; \mu \in T_{D-\deg(f_i)}\}$, and whose coefficients are those of the products μf_i in the basis T_D.

After constructing the the Macaulay matrix $\mathcal{M}_{\leq D}(\mathcal{F})$, it is then reduced in a way such that a subsystem of polynomials whose degrees are at most d_1 in the first n_D variables. In the *linearization* step, one iterates over the possible values of the last $n - n_D$ variables,

and solves the corresponding degree-d_1 subsystem in the first n_D variables. In particular, when $d_1 = 1$, this means to solve a linear system in n_D variables once the last $n - n_D$ variables have been assigned.

Choosing parameters. In general, it is not easy to choose the optimal values of parameters (D, n_D, d_1). In practice, $d_1 = 1$ is typically chosen, and we will focus on this choice as well. To determine (D, n_D), it is often assumed that the underlying polynomial system behaves as a generic system, in that the cancellations between polynomials are minimal. This is also related to the notion of semi-regularity from [BFSY]. Under this assumption, one is able to accurately predict the rank of $\mathcal{M}_{\leq D}(\mathcal{F})$ denoted by $\mathrm{Rank}(\mathcal{F}_{\leq D}(\mathcal{F}))$. Thus, for any given D, one can choose the largest corresponding n_D by counting monomials such that there exists a linear subsystem of equations in the first n_D variables. More precisely, let $\mathrm{Mon}_{n,D}(i)$, denote the number of degree $\leq D$-monomials in n variables that have degree ≥ 2 in the first $i < n$ variables. We then expect to find $\mathcal{M}_{\leq D}(\mathcal{F}) - \mathrm{Mon}_{n,D}(i)$ equations that are linear in the first i variables. Valid choices of n_D are then any number i that yields $\geq i$ such equations. The pair (D, n_D) can then be chosen so that the running time of the preprocessing and linearization steps are minimized. Indeed, such a way to choose (D, n_D) has, for instance, been adopted in Section 3.4 of [BMSV22].

Unfortunately, the polynomial systems we will work with in Section 4.3 will contain more cancellations than generic systems. Hence the usual estimates outlined above cannot be applied directly. Instead, for small values D, we will investigate the rank of $\mathcal{M}_D(\mathcal{F})$ for these particular systems, and derive optimal choices of n_D thereafter.

3 Overdefined Systems of Boolean Equations

At ASIACRYPT 2002, Courtois et al. revealed that it was possible to construct an overdefined system of Boolean equations for the S-box of AES [CP02]. Although the algebraic attack on AES in this paper was later believed to be too optimistic or even wrong, it provides a new perspective to analyze AES with algebraic techniques. In our algebraic attacks on RAIN, we rely on a quite similar idea to set up overdefined systems of Boolean equations.

Specifically, according to [CP02], for the inverse function $y = x^{-1}$ over $\mathbb{F}_{2^n}$ where $x \neq 0$, we can derive the following 5 equations over $\mathbb{F}_{2^n}$:

$$xy = 1,\ x^2y = x,\ xy^2 = y,\ x^4y = x^3,\ xy^4 = y^3.$$

Due to the isomorphism between $\mathbb{F}_{2^n}$ and $\mathbb{F}_2^n$, given any irreducible polynomial over $\mathbb{F}_{2^n}$, these 5 equations over $\mathbb{F}_{2^n}$ can be equivalently represented as $5n$ Boolean equations of algebraic degree 2, and these $5n$ quadratic Boolean equations are linearly independent [CL04].

In the above, we mainly discussed how to generate more equations from the equation $xy = 1$ over $\mathbb{F}_{2^n}$. In [CL04], the authors also discussed how to generate more quadratic equations for different power maps over $\mathbb{F}_{2^n}$ with the Gold exponent [Gol68] and Kasami exponent [Kas71], respectively. In the following, we show how to generate more quadratic equations from a special form of equations that appear in our cryptanalysis of RAIN.

Specifically, we consider 3 polynomials $P_1(x), P_2(x), P_3(x) \in \mathbb{F}_{2^n}[x]$ such that

$$P_1(x)P_2(x) + P_3(x) = 0, \tag{1}$$

where $\mathrm{Deg}(P_1) = \mathrm{Deg}(P_2) = \mathrm{Deg}(P_3) = 1$. Then, we can generate the following 3 equations

$$\begin{aligned} P_1(x)P_2(x) + P_3(x) &= 0,\\ P_1(x)(P_2(x))^2 + P_2(x)P_3(x) &= 0,\\ (P_1(x))^2P_2(x) + P_1(x)P_3(x) &= 0. \end{aligned}$$

By the definition of the algebraic degree of a polynomial in $\mathbb{F}_{2^n}[x]$ and the property that $(x+y) = x^2 + y^2$ for $\forall x, y \in \mathbb{F}_{2^n}$, we immediately obtain that these 3 equations correspond to $3n$ Boolean equations of algebraic degree upper bounded by 2.

For convenience, these $3n$ Boolean equations are all assumed to be quadratic since for most cases of interest they are indeed quadratic. We will not discuss the sufficient conditions to make these $3n$ quadratic Boolean equations linearly independent. Instead, we are only interested in how to generate more quadratic equations and expect that they are linearly independent. In the actual attacks, we will first test whether these $3n$ quadratic equations are linearly independent by computing the rank of the coefficient matrix formed by these quadratic equations where each quadratic term is viewed as a new independent variable.

4 Algebraic Cryptanalysis of RAIN

In this section, we first briefly recall the description of the MPC-friendly cipher RAIN proposed at CCS 2022 [DKR+22]. Then, we present the algebraic cryptanalysis of 2 and 3 rounds of RAIN, respectively, though the 3-round attack eventually failed. Although the successful attack only reaches 2 rounds, the designers choose the total number rounds of RAIN as either 3 or 4. This indicates that there is a small security margin if using 3 rounds of RAIN. It should be emphasized that the designers have performed thorough analysis of RAIN, and that they also performed experiments of the Gröbner basis attack on 2-round RAIN. While they were not able to find a specific attack on 2-round RAIN, they only claim security of RAIN for more than 3 rounds. Similar to [DKR+22], we denote 2 and 3 rounds of RAIN by RAIN_2 and RAIN_3, respectively.

4.1 Description of RAIN

The r-round RAIN is depicted in Figure 1, where k is the secret key and $(c_1, \ldots, c_r)$ are the round constants. The S-box $y = S(x)$ is the inverse function over $\mathbb{F}_{2^n}$, i.e.

$$\begin{cases} y = x^{-1} = \dfrac{1}{x}, \text{ for } \forall x \in \mathbb{F}_{2^n}, x \neq 0 \\ y = 0, \text{ for } x = 0 \end{cases} \tag{2}$$

Or equivalently, the S-box is defined by the power map:

$$x \mapsto x^{2^n-2}, \text{ for } \forall x \in \mathbb{F}_{2^n}.$$

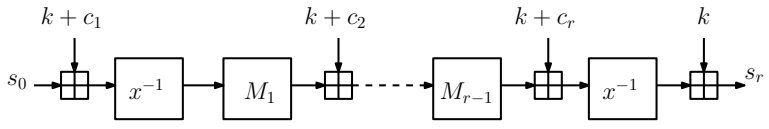

Figure 1: The r-round RAIN

For the linear layers, they are randomly generated and the designers have fixed their choices. Indeed, the binary matrix M_i can also be transformed to an $\mathbb{F}_2$-linearized polynomial by interpolation. Abusing the notation, we can write $M_i(x)$ as

$$M_i(x) = \sum_{j=0}^{n-1} a_{i,j} x^{2^j},$$

where $(a_{i,0}, \ldots, a_{i,n-1}) \in \mathbb{F}_{2^n}^n$ are known constants. In the design, it has been ensured that for each M_i, all the corresponding coefficients of the $\mathbb{F}_2$-linearized polynomial are non-zero, i.e. $a_{i,j} \neq 0$ for $j \in [0, n-1]$.

Since the security of RAIN is limited to the case when the attacker can only know one plaintext-ciphertext pair under the same key, the designers choose $r \in \{3, 4\}$. As an attacker, the goal is thus to recover the secret key k from 1 known input-output pair (s_0, s_r). Indeed, in the signature scheme Rainier built on RAIN [DKR+22], k is the secret key while (s_0, s_r) is the public key. Therefore, the above attack is directly related to the security of Rainier.

4.2 Low-degree Representation for RAIN$_2$

The designers of RAIN claim that the polynomial method [Din21a] is infeasible because the maximal algebraic degree is achieved after only one round due to the inverse function. Although the claim for the inverse function is true, we point out that it is feasible to construct a low-degree equation system to equivalently describe RAIN$_2$. Due to this low-degree representation, the polynomial method directly breaks RAIN$_2$ with the 128/192/256-bit key with only about $2^{118}/2^{172}/2^{225}$ bit operations, respectively.

Specifically, as shown in Figure 2, we introduce a variable $v_1 \in \mathbb{F}_{2^n}$ to represent the internal state after the first S-box. Given the known pair (s_0, s_2), we aim to set up a low-degree equation system only in v_1. Solving this equation system will allow us to recover v_1 and then the secret key k can be trivially recovered via:

$$k = v_1^{-1} + s_0 + c_1.$$

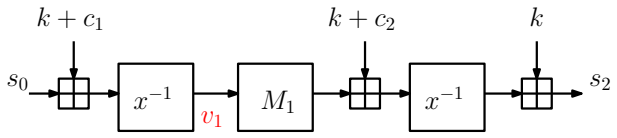

Figure 2: Illustration of RAIN$_2$

First, we consider whether $v_1 = 0$. In this case, we have $k = s_0 + c_1$ and it can be trivially verified. Similarly, we can trivially verify whether $k + s_2 = 0$. Therefore, in the following, we always assume that

$$k + c_1 + s_0 \neq 0, \quad k + s_2 \neq 0.$$

In this way, we have

$$k = \frac{1}{v_1} + s_0 + c_1$$

according to the first S-box. Then, according to M_1 and the last S-box, we have[2]

$$M_1(v_1) = \frac{1}{v_1} + s_0 + c_1 + c_2 + \frac{1}{\frac{1}{v_1} + s_0 + c_1 + s_2}.$$

For convenience, let

$$t_0 = s_0 + c_1 + c_2, \quad t_1 = s_0 + c_1 + s_2.$$

Then, t_0, t_1 are known constants and we have

$$M_1(v_1) = \frac{1}{v_1} + t_0 + \frac{1}{\frac{1}{v_1} + t_1} = \frac{1}{v_1} + t_0 + \frac{v_1}{1 + t_1 v_1}.$$

[2]Note that addition and subtraction are the same for polynomials in the polynomial ring $\mathbb{F}_{2^n}[x_1, \ldots, x_n]$.

Based on this, we further have

$$v_1(1+t_1v_1)M_1(v_1) = 1 + t_1v_1 + t_0v_1(1+t_1v_1) + v_1^2.$$

By simplifying the equation, we have

$$v_1M_1(v_1) + t_1v_1^2M_1(v_1) = 1 + t_1v_1 + t_0v_1 + t_0t_1v_1^2 + v_1^2.$$

Therefore, we can obtain n quadratic Boolean equations in v_1. Based on Dinur's algorithm for the polynomial method [Din21a], we can recover v_1 in about $n^2 \cdot 2^{0.815n}$ bit operations. Our results are summarized in Table 2.

4.3 Analysis of RAIN_2 with Low-memory Complexity

We observe that it is possible to construct an overdefined system of quadratic Boolean equations for RAIN_2, and solve it with the crossbred algorithm (Section 2.3). We focus on the choices of $d_1 = 1$, and $D = 2, 3, 4$. Note that the case $D = 2$ corresponds to the simplified crossbred algorithm used in [BDT22, WWF+21, LMSI22].

Specifically, we have

$$F(v_1) = (v_1 + t_1v_1^2)M_1(v_1) + 1 + t_1v_1 + t_0v_1 + t_0t_1v_1^2 + v_1^2 = 0. \tag{3}$$

Note that this equation is of the same form as Equation (1), hence we can derive

$$\begin{aligned} G(v_1) &= M_1(v_1)F(v_1) \\ &= (v_1 + t_1v_1^2)(M_1(v_1))^2 + (1 + t_1v_1 + t_0v_1 + t_0t_1v_1^2 + v_1^2)M_1(v_1) = 0, \quad (4) \\ H(v_1) &= (v_1 + t_1v_1^2)F(v_1) \\ &= (v_1 + t_1v_1^2)^2M_1(v_1) + (1 + t_1v_1 + t_0v_1 + t_0t_1v_1^2 + v_1^2)(v_1 + t_1v_1^2) = 0, \quad (5) \end{aligned}$$

which again correspond to $2n$ quadratic Boolean equations in v_1. Thus we have $n+2n = 3n$ Boolean equations in n variables.

Number of linearly independent equations. Recall from Section 2.3 that we are interested in understanding the number of linearly independent Boolean equations of a certain low degree D that we can form from the system $\mathcal{F}$ generated by F, G and H. This is equivalent to estimating the rank of the Macaulay matrix $\mathcal{M}_{\leq D}(\mathcal{F})$ for a sufficiently large n. We have experimentally verified that the $3n$ quadratic equations forming $\mathcal{F}$ are linearly independent, and thus

$$\mathrm{Rank}(\mathcal{M}_{\leq 2}(\mathcal{F})) = 3n. \tag{6}$$

At degree $D = 3$, we note that there are $3n^2$ possible polynomials of the form x_if_j, where x_i is a Boolean unknown variable and f_j is a polynomial in $\mathcal{F}$, but not all of these will be linearly independent. Indeed, similar to the analysis performed in [ØFRC20, Section 4], this can at least be partially understood upon examination of the extension field description of Equations (3)–(5). Denote $L = 1 + t_1v_1 + t_0v_1 + t_0t_1v_1^2 + v_1^2$, and let F, G, H, M_1 be as above. Then we have the cancellations associated with

$$\begin{aligned} M_1F + G = 0, &\qquad (v_1 + t_1v_1^2)F + H = 0, \\ (v_1 + t_1v_1^2)G + F^2 + LF = 0, &\qquad M_1H + F^2 + LF = 0, \\ (v_1 + t_1v_1^2)H + (v_1 + t_1v_1^2)^2F = 0, &\qquad M_1G + M_1^2F = 0. \end{aligned}$$

Each of these six cancellations will correspond to n sums of the form $\sum x_if_j$, which symbolically sums to zero, and hence show up as $6n$ linearly dependent rows in $\mathcal{M}_{\leq 3}(\mathcal{F})$.

In fact, experiments show that $\mathcal{M}_{\leq 3}(\mathcal{F})$ has rank $3n(n+1)-8n$, meaning that there are $2n$ more cancellations that we are unable to explain. Summing up, this means that we have

$$\mathrm{Rank}(\mathcal{M}_{\leq 3}(\mathcal{F})) = n(3n-8) + \mathrm{Rank}(\mathcal{M}_{\leq 2}(\mathcal{F})). \tag{7}$$

At degree $D=4$ we will also have to take into account the trivial syzygies in the polynomial system, i.e. cancellations of the form $f_i f_j + f_j f_i = 0$ and $f_i^2 + f_i = 0$. If these were the only cancellations in the system at degree 4 (as is e.g. the case for a semi-regular system), we would have counted the number of linearly independent equations at degree 4 as $3n\binom{n}{2} - \binom{3n+1}{2}$. However, we know from Equation (7) that there is a correction term of $8n$ at degree 3 which will likely turn into $8n^2$ at degree 4. Finally, experiments show that there is a final correction term of $17n$, which we are unable to explain. Summing up, we find the following formula for the rank of the Macaulay matrix at degree 4:

$$\mathrm{Rank}(\mathcal{M}_{\leq 4}(\mathcal{F})) = 3n\binom{n}{2} - \binom{3n+1}{2} - 8n^2 + 17n + \mathrm{Rank}(\mathcal{M}_{\leq 3}(\mathcal{F})). \tag{8}$$

Complexity evaluation for the Crossbred algorithm with $D = 2, 3, 4$. In order to choose n_D as described in Section 2.3, we first need $\mathrm{Mon}_{n,D}(i)$, i.e. the number of degree $\leq D$-monomials in n variables that have degree ≥ 2 in the first $i < n$ variables. This number is listed below for $D = 2, 3, 4$:

$$\mathrm{Mon}_{n,D}(i) = \begin{cases} \binom{i}{2} & \text{for } D=2, \\ \binom{i}{3} + (n-i)\binom{i}{2} + \mathrm{Mon}_{n,2}(i) & \text{for } D=3, \\ \binom{i}{4} + (n-i)\binom{i}{3} + \binom{n-i}{2}\binom{i}{2} + \mathrm{Mon}_{n,3}(i) & \text{for } D=4. \end{cases} \tag{9}$$

At the preprocessing step of the crossbred algorithm, we need to find a parameter n_D and eliminate $\mathrm{Mon}_{n,D}(n_D)$ monomials with Gaussian elimination. For these relatively small choices of D, the time complexity of the preprocessing step is negligible when compared to the latter linearization step. Specifically, the cost of the preprocessing step is equal to the cost to perform Gaussian elimination on the Macaulay matrix $\mathcal{M}_{\leq D}(\mathcal{F})$ of size $\left(\binom{n}{\leq D-2} \cdot 3n\right) \times \binom{n}{\leq D}$. Thus it is optimal to choose n_D as large as possible, that is, for $D = 2, 3, 4$, we define the optimal n_D as

$$n_D = n_D(\mathcal{F}) = \max\left\{ i \in \mathbb{Z}_{>0} \;\middle|\; \mathrm{Rank}(\mathcal{M}_{\leq D}(\mathcal{F})) - \mathrm{Mon}_{n,D}(i) \geq i \right\}. \tag{10}$$

This allows us to construct at least n_D equations that are linear in n_D variables after guessing $n - n_D$ variables, while at the same time keeping $n - n_D$ minimal. In this way, the time complexity to solve the target equation system is estimated as $2^{n-n_D} n_D^3$ bit operations, i.e. the time needed to solve a linear system in n_D variables 2^{n-n_D} times.

For the memory complexity, we estimate it as the memory to store all the polynomial equations. Specifically, it is estimated as

$$\left(\binom{n}{\leq D-2} \cdot m\right) \times \binom{n}{\leq D}$$

bits of memory, as the corresponding Macaulay matrix $\mathcal{M}_{\leq D}(\mathcal{F})$ has $\left(\binom{n}{\leq D-2} \cdot 3n\right)$ rows and $\binom{n}{\leq D}$ columns. The resulting time and memory complexities for the RAIN_2 parameters are given in Table 2.

Remark 1. We note that the time and memory complexity of the preprocessing step to handle the Macaulay matrix $\mathcal{M}_{\leq D}(\mathcal{F})$ can be further optimized with advanced implementation techniques. Note that eliminating $\mathrm{Mon}_{n,D}(n_D)$ many monomials can be

viewed as performing Gaussian elimination on a $\mathrm{Rank}(\mathcal{M}_{\leq D}) \times \mathrm{Mon}_{n,D}(n_D)$ submatrix of the Macaulay matrix, and hence we can consider a $\mathrm{Rank}(\mathcal{M}_{\leq D}) \times \mathrm{Mon}_{n,D}(n_D)$ submatrix. After this step, we can build each of the n_D linear equations by summing at most $\mathrm{Rank}(\mathcal{M}_{\leq D})$ polynomial equations, which means we do not need to perform the Gaussian elimination on the full Macaulay matrix $\mathcal{M}_{\leq D}(\mathcal{F})$ in order to get the desired n_D linear equations. Moreover, the memory complexity can further be reduced by using the Wiedemann algorithm for finding the kernel, as described in [BMSV22, Section 3.4]. Still, we prefer to present the numbers from the naive method for simplicity.

Table 2: Cost analysis of various methods for solving RAIN_2

Method	n	n_D	Time (bits)	Memory (bits)
Polynomial Method	128	-	2^{118}	2^{95}
	192	-	2^{172}	2^{136}
	256	-	2^{225}	2^{177}
Crossbred $D=2$	128	27	2^{115}	2^{22}
	192	33	2^{174}	2^{23}
	256	38	2^{234}	2^{25}
Crossbred $D=3$	128	30	2^{113}	2^{35}
	192	36	2^{172}	2^{37}
	256	41	2^{231}	2^{40}
Crossbred $D=4$	128	32	2^{111}	2^{45}
	192	38	2^{170}	2^{50}
	256	44	2^{228}	2^{52}

Experimental verification. Our formulas for $\mathrm{Rank}(\mathcal{M}_{\leq D}(\mathcal{F}))$ at $D \in \{2,3,4\}$ specified in Equations (6)–(8) have been experimentally verified using the built-in algorithm in MAGMA [BCP97] for computing the Hilbert series of homogenized systems $\mathcal{F}$ for smaller values of n. In particular, $D=2,3$ have been tested for n in the range of $[25,40]$, whereas $D=4$ has been tested for $n \in [29,40]$ (note that Equation (8) is negative for $n<29$).

4.4 Analysis of RAIN_3 Using the Equivalent Representation

It is natural to ask whether it is possible to extend this attack to 3 or more rounds. However, this seems infeasible. Specifically, if we work in a similar way, we have

$$\left(M_1(v_1)+\frac{1}{v_1}+t_0\right)\left(\frac{1}{v_1}+s_0+c_1+c_3+M_2^{-1}\left(\frac{1}{\frac{1}{v_1}+s_0+c_1+s_3}\right)\right)=1$$

Let

$$t_2 = s_0+c_1+c_3, \quad t_3 = s_0+c_1+s_3.$$

Then, we have

$$\begin{aligned} &\left(M_1(v_1)+\frac{1}{v_1}+t_0\right)\left(\frac{1}{v_1}+t_2+M_2^{-1}\left(\frac{v_1}{1+t_3v_1}\right)\right)=1, \\ \hookrightarrow\quad &\left(v_1M_1(v_1)+1+t_0v_1\right)\left(1+t_2v_1+v_1M_2^{-1}\left(\frac{v_1}{1+t_3v_1}\right)\right)=v_1^2. \end{aligned}$$

In this case, we cannot further expand $M_2^{-1}\left(\frac{v_1}{1+t_3v_1}\right)$. Otherwise, we could only get an equation system of algebraic degree $n-1$ in terms of v_1 because we need to multiply the term

$$\prod_{i=0}^{n-1}(1+t_3v_1)^{2^i}$$

in both sides of the equation to clear all denominators.

Another interesting observation based on the above analysis is that if there are only a few monomials in $M_1(x)$ or $M_2^{-1}(x)$, we can still construct a low-degree representation for RAIN_3. For example, if there are only ℓ monomials in $M_2^{-1}(x)$, i.e. there are only ℓ nonzero coefficients in $(a'_{2,0},\ldots,a'_{2,n-1})$ of the $\mathbb{F}_2$-linearized polynomial

$$M_2^{-1}(x)=\sum_{i=0}^{n-1}a'_{2,i}x^{2^i},$$

by clearing all the denominators, we can get n Boolean equations of algebraic degree upper bounded by $\ell+2$, which is caused by $v_1M_1(v_1)\cdot v_1M_2^{-1}\left(\frac{v_1}{1+t_3v_1}\right)$.

The reason to consider 2 cases $M_1(x)$ and $M_2^{-1}(x)$ is simple. First, the encryption and decryption of RAIN_3 has the same structure due to the inverse function. Second, we can introduce a variable to denote the input of the last S-box rather than the output of the first S-box, and the equivalent representation only in this variable still has the same form. However, the designers have fixed these polynomials and all the coefficients are nonzero, which makes the above attack infeasible.

5 Algebraic Cryptanalysis of AIM

It is found that there also exists a low-degree representation for full-round AIM. Although the designers also reported a similar observation in the Appendix of [KHS+22], they only considered the lower bound of the algebraic degree of this equivalent polynomial representation. What is worse, they only treated the polynomial method [Din21a] as the only threat for this equivalent representation, which requires more than 2^n bits of memory even for this lower bound. In this section, we perform a deep study on this equivalent representation and give its accurate algebraic degree. By using the fast exhaustive search method [BCC+10], we significantly reduce the cost to find the secret key of AIM via brute force.

5.1 Description of AIM

The general construction of AIM can be described as follows:

$$\begin{aligned} z_i &= k^{2^{e_i}-1} \text{ for } i\in[1,m-1],\\ w &= \sum_{i=1}^{m-1}B_i(z_i),\\ y &= w^{2^{e_m}-1}+k. \end{aligned}$$

In the above equations, k and y are the input and output of AIM, respectively, while z_i and w are internal states, and $B_i(x)$ is the $\mathbb{F}_2$-linearized affine polynomial, i.e.

$$B_i(x)=a_{i,n}+\sum_{j=0}^{n-1}a_{i,j}x^{2^j}.$$

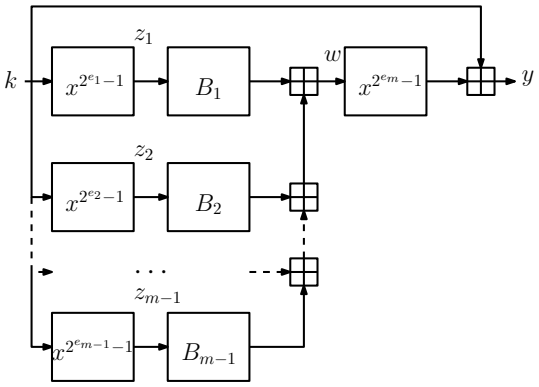

Figure 3: The general construction of AIM

The corresponding graphic illustration is given in Figure 3.

Three variants of AIM named AIM-I, AIM-II, and AIM-III are specified by the designers, as shown in Table 3.

Table 3: The three variants of AIM

Name	n	m	$(e_1, \ldots, e_m)$
AIM-I	128	3	$(3, 27, 5)$
AIM-II	192	3	$(5, 29, 7)$
AIM-III	256	4	$(3, 7, 53, 5)$

5.2 Low-degree Representation for AIM

We describe the low-degree equivalent representation for AIM in this section. As the algebraic degree of the equivalent representation is relatively high, the polynomial method cannot work efficiently. However, the fast exhaustive search [BCC+10] is still applicable and breaking AIM-I/AIM-II/AIM-III requires about $2^{115}/2^{178}/2^{241}$ equivalent calls to the corresponding primitives, respectively.

Specifically, we observe that in AIM, $m-1$ elements in the set $\{e_1, \ldots, e_m\}$ take small values, i.e. smaller than 8. In what follows, we show how to exploit this property to construct the low-degree representation for AIM.

Given the output y, we can represent k in terms of the unknown w as follows:

$$k = w^{2^{e_m}-1} + y.$$

In this way, we can further represent each $(z_i)_{1 \le i \le m-1}$ only in terms of w, as shown below:

$$z_i = (w^{2^{e_m}-1} + y)^{2^{e_i}-1}.$$

For convenience, we assume $e_1 < e_2 < \cdots < e_{m-1}$.

We first show how to compute the accurate algebraic degree for the polynomials $(z_i)_{1 \le i \le m-2}$ in terms of w. Note that

$$2^i - 1 = \sum_{j=0}^{i-1} 2^j$$

and hence we have

$$\forall a, b \in \mathbb{F}_{2^n}, \forall i \in [1, n]: \ (a+b)^{2^i-1} = \prod_{j=0}^{i-1}(a+b)^{2^j} = \prod_{j=0}^{i-1}(a^{2^j}+b^{2^j}) = \sum_{j=0}^{2^i-1} a^j b^{2^i-1-j}.$$

Therefore, we have

$$z_i = (w^{2^{e_m}-1}+y)^{2^{e_i}-1} = \sum_{j=0}^{2^{e_i}-1} y^j w^{2^{e_m-1}(2^{e_i}-1-j)}.$$

In this way, the algebraic degree of z_i in terms of w is

$$d_i = \max\left\{ \mathrm{Hw}\left(\mathcal{M}_n\left(2^{e_m-1}(2^{e_i}-1-j)\right)\right) \mid j \in [0, 2^{e_i}-1]\right\},$$

where

$$\forall a \in \mathbb{N}: \mathcal{M}_n(a) := \begin{cases} 2^n - 1 & \text{if } 2^n-1 | a \text{ and } a \geq 2^n-1, \\ a \% (2^n-1) & \text{otherwise.} \end{cases}$$

and $\mathrm{Hw}(a)$ is the hamming weight of a, i.e. the number of 1 in its binary representation. Therefore, d_i can be naively computed in time $\mathcal{O}(2^{e_i})$. Note that for $(e_i)_{1\leq i\leq m-2}$ in AIM, all of them are smaller than 8 and hence $(d_i)_{1\leq i\leq m-2}$ can be computed with time complexity of 2^8.

After computing $(d_i)_{1\leq i\leq m-2}$, we define

$$d_{\max} = \max\{d_1, \ldots, d_{m-2}\}.$$

In this way, z_{m-1} can be expressed in w of algebraic degree $d_{\max}$ due to

$$\begin{aligned} z_{m-1} &= B_{m-1}^{-1}\Big(c + w + \sum_{i=0}^{m-2} B_i(z_i)\Big) \\ &= B_{m-1}^{-1}\left(c + w + \sum_{i=0}^{m-2} B_i\Big((w^{2^{e_m}-1}+y)^{2^{e_i}-1}\Big)\right) \end{aligned}$$

In other words, the algebraic degree of the above polynomial of z_{m-1} in terms of w is $d_{\max}$. Furthermore, there is another way to establish the relation between z_{m-1} and w:

$$\begin{aligned} z_{m-1} &= (w^{2^{e_m}-1}+y)^{2^{e_{m-1}}-1}, \\ \hookrightarrow z_{m-1}(w^{2^{e_m}-1}+y) &= (w^{2^{e_m}-1}+y)^{2^{e_{m-1}}}. \end{aligned}$$

Hence, we obtain an equation only in terms of w, as shown below:

$$B_{m-1}^{-1}\left(c + w + \sum_{i=0}^{m-2} B_i\Big((w^{2^{e_m}-1}+y)^{2^{e_i}-1}\Big)\right)(w^{2^{e_m}-1}+y) = (w^{2^{e_m}-1}+y)^{2^{e_{m-1}}}. \quad (11)$$

This equation also corresponds to n Boolean equations of algebraic degree upper bounded by $d_{\max}+e_m$. These n Boolean equations can be obtained by polynomial interpolation with the recursive version of Möbius transform, e.g. as described in [Din21a]. To interpolate these n Boolean polynomials, we need to evaluate

$$B_{m-1}^{-1}\left(c + w + \sum_{i=0}^{m-2} B_i\Big((w^{2^{e_m}-1}+y)^{2^{e_i}-1}\Big)\right)(w^{2^{e_m}-1}+y) + (w^{2^{e_m}-1}+y)^{2^{e_{m-1}}}$$

for

$$T_{\text{num}} = \binom{n}{\leq d_{\max} + e_m} \tag{12}$$

different w, whose cost can be roughly estimated as T_{num} calls to AIM. After the polynomial evaluation is over, recovering the n Boolean polynomials with these T_{num} points require

$$n \cdot n \cdot \binom{n}{\leq d_{\max} + e_m}$$

bit operations.

By solving these n Boolean equations, we can first recover w and then the secret key can be trivially recovered via:

$$k = w^{2^{e_m}-1} + y.$$

Remark 2. The designers are indeed aware of this representation, but they use a lower bound on the algebraic degree of z_i in terms of w, i.e. it is simply treated as the hamming weight of $\mathcal{M}_n\left((2^{e_m}-1)2^{e_i-1}\right)$ because the monomial $x^{(2^{e_m}-1)2^{e_i-1}}$ will appear if we expand the expression [KHS+22]. Moreover, they only treat the polynomial method as a main threat for this low-degree representation, which cannot beat the naive exhaustive search because its memory complexity is much higher than 2^n bits.

5.3 Solving the n Boolean Equations of Algebraic Degree $d_{\max} + e_m$

With the memory-efficient Möbius transform [Din21a, Bou22], evaluating a Boolean polynomial in n variables of algebraic degree d over $\{0,1\}^n$ requires about $n \cdot \binom{n}{\leq d}$ bits of memory and the time is about $d \cdot 2^n$ bit operations. With this as the oracle of the fast exhaustive search method [BCC+10] to efficiently evaluate a polynomial, the time complexity to find the solution of w from n Boolean equations of algebraic degree upper bounded by $d_{\max} + e_m$ is estimated as about

$$4 \cdot (d_{\max} + e_m) \cdot \log_2 n \cdot 2^n \tag{13}$$

bit operations. The memory complexity is upper bounded by

$$n \cdot n \cdot \binom{n}{\leq d_{\max} + e_m}$$

bits.

Remark 3. We avoid the original fast exhaustive search method for its costly pre-processing phase of time complexity $\mathcal{O}(n^{2(d_{\max}+e_m)})$. However, as already stated in Sect. 2, the time complexity of the pre-processing phase for n Boolean polynomials is indeed upper bounded by

$$n \cdot n^d \cdot \binom{n}{\leq d_{\max} + e_m} \tag{14}$$

bit operations. In this sense, using the original fast exhaustive search attack indeed does not affect the whole time complexity as the cost of the pre-processing phase specified in Equation 14 is still much smaller than Equation 13 for the concrete values of $(n, d_{\max} + e_m)$ shown in Table 4.

Our results for the parameters of AIM are summarized in Table 4. Note that due to the relatively small value of $d_{\max} + e_m$ compared with n, the cost to interpolate the n Boolean polynomials is negligible, which is almost equivalent to

$$T_{\text{num}} = \binom{n}{\leq d_{\max} + e_m}$$

calls to AIM. Moreover, according to designers' analysis [KHS+22], without this low-degree equivalent representation in n variables, the naive brute force takes about 2^{149}, $2^{214.4}$ and 2^{280} bit operations for AIM-I, AIM-II and AIM-III, respectively. Hence, the fast exhaustive search attack much improves the naive brute force attack. In other words, the time complexity of our attacks is equivalent to about $2^{115}/2^{178}/2^{241}$ calls of the underlying primitives.

Table 4: Fast exhaustive search (FES) attacks on AIM

Attack Type	n	m	$(e_1, \ldots, e_m)$	$d_{\max} + e_m$	T_{num}	Time (bits)	Memory (bits)
Brute force [KHS+22]	128	3	(3, 27, 5)	–	–	2^{149}	negligible
FES				10	$2^{47.9}$	$2^{136.2}$	$2^{61.7}$
Brute force [KHS+22]	192	3	(5, 29, 7)	–	–	$2^{214.4}$	negligible
FES				14	$2^{69.3}$	$2^{200.7}$	$2^{84.3}$
Brute force [KHS+22]	256	4	(3, 7, 53, 5)	–	–	2^{280}	negligible
FES				15	$2^{79.3}$	$2^{265.0}$	$2^{95.1}$

6 Conclusion

We show that there are nontrivial low-degree equivalent representations for 2-round RAIN and full-round AIM, which can be exploited to mount effective attacks. Especially, as recovering the secret key of AIM is the underlying difficult problem of the signature scheme AIMer, which is one of the NIST Round 1 Additional Signatures, we believe that this work is meaningful to the ongoing NIST PQC competition. Moreover, while our analysis shows that 3 rounds are sufficient for RAIN, we propose to primarily use 4-round RAIN with an additional round of security margin.

Acknowledgments. We thank the reviewers of ToSC Issue (4) for improving the quality of this paper. Fukang Liu is supported by Grant-in-Aid for Research Activity Start-up (Grant No. 22K21282). The research results were also funded by the commissioned research (No. JPJ012368C05801) by National Institute of Information and Communications Technology (NICT), Japan. Morten Øygarden has been funded by the Research Council of Norway through the project qsIo2.

References

[AAB+20] Abdelrahaman Aly, Tomer Ashur, Eli Ben-Sasson, Siemen Dhooghe, and Alan Szepieniec. Design of Symmetric-Key Primitives for Advanced Cryptographic Protocols. *IACR Trans. Symmetric Cryptol.*, 2020(3):1–45, 2020.

[ACG+19] Martin R. Albrecht, Carlos Cid, Lorenzo Grassi, Dmitry Khovratovich, Reinhard Lüftenegger, Christian Rechberger, and Markus Schofnegger.

Algebraic Cryptanalysis of STARK-Friendly Designs: Application to MARVELlous and MiMC. In *ASIACRYPT (3)*, volume 11923 of *Lecture Notes in Computer Science*, pages 371–397. Springer, 2019.

[AGR+16] Martin R. Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity. In *ASIACRYPT (1)*, volume 10031 of *Lecture Notes in Computer Science*, pages 191–219, 2016.

[AMT22] Tomer Ashur, Mohammad Mahzoun, and Dilara Toprakhisar. Chaghri - A FHE-friendly Block Cipher. In *CCS*, pages 139–150. ACM, 2022.

[ARS+15] Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. In *EUROCRYPT (1)*, volume 9056 of *Lecture Notes in Computer Science*, pages 430–454. Springer, 2015.

[BBC+23] Clémence Bouvier, Pierre Briaud, Pyrros Chaidos, Léo Perrin, Robin Salen, Vesselin Velichkov, and Danny Willems. New Design Techniques for Efficient Arithmetization-Oriented Hash Functions: ttAnemoi Permutations and ttJive Compression Mode. In *CRYPTO (3)*, volume 14083 of *Lecture Notes in Computer Science*, pages 507–539. Springer, 2023.

[BBDV20] Subhadeep Banik, Khashayar Barooti, F. Betül Durak, and Serge Vaudenay. Cryptanalysis of LowMC instances using single plaintext/ciphertext pair. *IACR Trans. Symmetric Cryptol.*, 2020(4):130–146, 2020.

[BBVY21] Subhadeep Banik, Khashayar Barooti, Serge Vaudenay, and Hailun Yan. New Attacks on LowMC Instances with a Single Plaintext/Ciphertext Pair. In *ASIACRYPT (1)*, volume 13090 of *Lecture Notes in Computer Science*, pages 303–331. Springer, 2021.

[BCC+10] Charles Bouillaguet, Hsieh-Chung Chen, Chen-Mou Cheng, Tung Chou, Ruben Niederhagen, Adi Shamir, and Bo-Yin Yang. Fast Exhaustive Search for Polynomial Systems in F_2. In *CHES*, volume 6225 of *Lecture Notes in Computer Science*, pages 203–218. Springer, 2010.

[BCD+20] Tim Beyne, Anne Canteaut, Itai Dinur, Maria Eichlseder, Gregor Leander, Gaëtan Leurent, María Naya-Plasencia, Léo Perrin, Yu Sasaki, Yosuke Todo, and Friedrich Wiemer. Out of Oddity - New Cryptanalytic Techniques Against Symmetric Primitives Optimized for Integrity Proof Systems. In *CRYPTO (3)*, volume 12172 of *Lecture Notes in Computer Science*, pages 299–328. Springer, 2020.

[BCP97] Wieb Bosma, John Cannon, and Catherine Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997. Computational algebra and number theory (London, 1993).

[BCP23] Clémence Bouvier, Anne Canteaut, and Léo Perrin. On the algebraic degree of iterated power functions. *Des. Codes Cryptogr.*, 91(3):997–1033, 2023.

[BDT22] Charles Bouillaguet, Claire Delaplace, and Monika Trimoska. A Simple Deterministic Algorithm for Systems of Quadratic Polynomials over F_2. In *SOSA*, pages 285–296. SIAM, 2022.

[BFSY] Magali Bardet, Jean-Charles Faugère, Bruno Salvy, and B-Y. Yang. Asymptotic Behaviour of the Degree of Regularity of Semi-Regular Polynomial Systems.

[BKW19] Andreas Björklund, Petteri Kaski, and Ryan Williams. Solving Systems of Polynomial Equations over GF(2) by a Parity-Counting Self-Reduction. In *ICALP*, volume 132 of *LIPIcs*, pages 26:1–26:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[BMSV22] Emanuele Bellini, Rusydi H Makarim, Carlo Sanna, and Javier Verbel. An Estimator for the Hardness of the MQ Problem. In *International Conference on Cryptology in Africa*, pages 323–347. Springer, 2022.

[Bou22] Charles Bouillaguet. Boolean Polynomial Evaluation for the Masses. Cryptology ePrint Archive, Paper 2022/1412, 2022. https://eprint.iacr.org/2022/1412.

[Buc65] Bruno Buchberger. Ein algorithmus zum auffinden der basiselemente des restklassenringes nach einem nulldimensionalen polynomideal. *PhD thesis, Universitat Insbruck*, 1965.

[CCF+16] Anne Canteaut, Sergiu Carpov, Caroline Fontaine, Tancrède Lepoint, María Naya-Plasencia, Pascal Paillier, and Renaud Sirdey. Stream Ciphers: A Practical Solution for Efficient Homomorphic-Ciphertext Compression. In *FSE*, volume 9783 of *Lecture Notes in Computer Science*, pages 313–333. Springer, 2016.

[CHK+21] Jihoon Cho, Jincheol Ha, Seongkwang Kim, ByeongHak Lee, Joohee Lee, Jooyoung Lee, Dukjae Moon, and Hyojin Yoon. Transciphering Framework for Approximate Homomorphic Encryption. In *ASIACRYPT (3)*, volume 13092 of *Lecture Notes in Computer Science*, pages 640–669. Springer, 2021.

[CKPS00] Nicolas T. Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In *EUROCRYPT*, volume 1807 of *Lecture Notes in Computer Science*, pages 392–407. Springer, 2000.

[CL04] Jung Hee Cheon and Dong Hoon Lee. Resistance of S-Boxes against Algebraic Attacks. In *FSE*, volume 3017 of *Lecture Notes in Computer Science*, pages 83–94. Springer, 2004.

[CP02] Nicolas T. Courtois and Josef Pieprzyk. Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. In *ASIACRYPT*, volume 2501 of *Lecture Notes in Computer Science*, pages 267–287. Springer, 2002.

[DEG+18] Christoph Dobraunig, Maria Eichlseder, Lorenzo Grassi, Virginie Lallemand, Gregor Leander, Eik List, Florian Mendel, and Christian Rechberger. Rasta: A Cipher with Low ANDdepth and Few ANDs per Bit. In *CRYPTO (1)*, volume 10991 of *Lecture Notes in Computer Science*, pages 662–692. Springer, 2018.

[DGGK21] Christoph Dobraunig, Lorenzo Grassi, Anna Guinet, and Daniël Kuijsters. Ciminion: Symmetric Encryption Based on Toffoli-Gates over Large Finite Fields. In *EUROCRYPT (2)*, volume 12697 of *Lecture Notes in Computer Science*, pages 3–34. Springer, 2021.

[DGH+21] Christoph Dobraunig, Lorenzo Grassi, Lukas Helminger, Christian Rechberger, Markus Schofnegger, and Roman Walch. Pasta: A Case for Hybrid Homomorphic Encryption. *IACR Cryptol. ePrint Arch.*, page 731, 2021.

[Din21a] Itai Dinur. Cryptanalytic Applications of the Polynomial Method for Solving Multivariate Equation Systems over GF(2). In *EUROCRYPT (1)*, volume 12696 of *Lecture Notes in Computer Science*, pages 374–403. Springer, 2021.

[Din21b] Itai Dinur. Improved Algorithms for Solving Polynomial Systems over GF(2) by Multiple Parity-Counting. In *SODA*, pages 2550–2564. SIAM, 2021.

[DKR+22] Christoph Dobraunig, Daniel Kales, Christian Rechberger, Markus Schofnegger, and Greg Zaverucha. Shorter Signatures Based on Tailor-Made Minimalist Symmetric-Key Crypto. In *CCS*, pages 843–857. ACM, 2022.

[DLR16] Sébastien Duval, Virginie Lallemand, and Yann Rotella. Cryptanalysis of the FLIP Family of Stream Ciphers. In *CRYPTO (1)*, volume 9814 of *Lecture Notes in Computer Science*, pages 457–475. Springer, 2016.

[EGL+20] Maria Eichlseder, Lorenzo Grassi, Reinhard Lüftenegger, Morten Øygarden, Christian Rechberger, Markus Schofnegger, and Qingju Wang. An Algebraic Attack on Ciphers with Low-Degree Round Functions: Application to Full MiMC. In *ASIACRYPT (1)*, volume 12491 of *Lecture Notes in Computer Science*, pages 477–506. Springer, 2020.

[Fau99] Jean-Charles Faugère. A new efficient algorithm for computing Gröbner bases (F4). *Journal of Pure and Applied Algebra*, 139:61–88, 1999.

[Fau02] Jean-Charles Faugère. A new efficient algorithm for computing grobner bases without reduction to zero (f5). *ISSAC '02 : Proceedings of the 2002 international symposium on Symbolic and algebraic computation, New York, NY, USA*, pages 75–83, 2002.

[GAH+23] Lorenzo Grassi, Irati Manterola Ayala, Martha Norberg Hovd, Morten Øygarden, Håvard Raddum, and Qingju Wang. Cryptanalysis of Symmetric Primitives over Rings and a Key Recovery Attack on Rubato. In *CRYPTO (3)*, volume 14083 of *Lecture Notes in Computer Science*, pages 305–339. Springer, 2023.

[GKL+22] Lorenzo Grassi, Dmitry Khovratovich, Reinhard Lüftenegger, Christian Rechberger, Markus Schofnegger, and Roman Walch. Reinforced Concrete: A Fast Hash Function for Verifiable Computation. In *CCS*, pages 1323–1335. ACM, 2022.

[GKR+21] Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger. Poseidon: A New Hash Function for Zero-Knowledge Proof Systems. In *USENIX Security Symposium*, pages 519–535. USENIX Association, 2021.

[GLR+20] Lorenzo Grassi, Reinhard Lüftenegger, Christian Rechberger, Dragos Rotaru, and Markus Schofnegger. On a Generalization of Substitution-Permutation Networks: The HADES Design Strategy. In *EUROCRYPT (2)*, volume 12106 of *Lecture Notes in Computer Science*, pages 674–704. Springer, 2020.

[Gol68] Robert Gold. Maximal recursive sequences with 3-valued recursive cross-correlation functions (Corresp.). *IEEE Trans. Inf. Theory*, 14(1):154–156, 1968.

[GØSW23] Lorenzo Grassi, Morten Øygarden, Markus Schofnegger, and Roman Walch. From Farfalle to Megafono via Ciminion: The PRF Hydra for MPC Applications. In *EUROCRYPT (4)*, volume 14007 of *Lecture Notes in Computer Science*, pages 255–286. Springer, 2023.

[HKL+22] Jincheol Ha, Seongkwang Kim, ByeongHak Lee, Jooyoung Lee, and Mincheol Son. Rubato: Noisy Ciphers for Approximate Homomorphic Encryption. In *EUROCRYPT (1)*, volume 13275 of *Lecture Notes in Computer Science*, pages 581–610. Springer, 2022.

[IKOS07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In *STOC*, pages 21–30. ACM, 2007.

[JV17] Antoine Joux and Vanessa Vitse. A Crossbred Algorithm for Solving Boolean Polynomial Systems. In *NuTMiC*, volume 10737 of *Lecture Notes in Computer Science*, pages 3–21. Springer, 2017.

[Kas71] T. Kasami. The weight enumerators for several classes of subcodes of the 2nd order binary Reed-Muller codes. *Information and Control*, 18(4):369–394, 1971.

[KHS+22] Seongkwang Kim, Jincheol Ha, Mincheol Son, Byeonghak Lee, Dukjae Moon, Joohee Lee, Sangyub Lee, Jihoon Kwon, Jihoon Cho, Hyojin Yoon, and Jooyoung Lee. AIM: Symmetric Primitive for Shorter Signatures with Stronger Security (Full Version). Cryptology ePrint Archive, Paper 2022/1387, 2022. https://eprint.iacr.org/2022/1387.

[KHSL23] Seongkwang Kim, Jincheol Ha, Mincheol Son, and Byeonghak Lee. Mitigation on the AIM Cryptanalysis. Cryptology ePrint Archive, Paper 2023/1474, 2023. https://eprint.iacr.org/2023/1474.

[LAW+23] Fukang Liu, Ravi Anand, Libo Wang, Willi Meier, and Takanori Isobe. Coefficient Grouping: Breaking Chaghri and More. In *EUROCRYPT (4)*, volume 14007 of *Lecture Notes in Computer Science*, pages 287–317. Springer, 2023.

[LGB+23] Fukang Liu, Lorenzo Grassi, Clémence Bouvier, Willi Meier, and Takanori Isobe. Coefficient Grouping for Complex Affine Layers. In *CRYPTO (3)*, volume 14083 of *Lecture Notes in Computer Science*, pages 540–572. Springer, 2023.

[LIM21] Fukang Liu, Takanori Isobe, and Willi Meier. Cryptanalysis of Full LowMC and LowMC-M with Algebraic Techniques. In *CRYPTO (3)*, volume 12827 of *Lecture Notes in Computer Science*, pages 368–401. Springer, 2021.

[LMSI22] Fukang Liu, Willi Meier, Santanu Sarkar, and Takanori Isobe. New Low-Memory Algebraic Attacks on LowMC in the Picnic Setting. *IACR Trans. Symmetric Cryptol.*, 2022(3):102–122, 2022.

[LPT+17] Daniel Lokshtanov, Ramamohan Paturi, Suguru Tamaki, R. Ryan Williams, and Huacheng Yu. Beating Brute Force for Systems of Polynomial Equations over Finite Fields. In *SODA*, pages 2190–2202. SIAM, 2017.

[LSMI21] Fukang Liu, Santanu Sarkar, Willi Meier, and Takanori Isobe. Algebraic Attacks on Rasta and Dasta Using Low-Degree Equations. In *ASIACRYPT (1)*, volume 13090 of *Lecture Notes in Computer Science*, pages 214–240. Springer, 2021.

[LSW+22] Fukang Liu, Santanu Sarkar, Gaoli Wang, Willi Meier, and Takanori Isobe. Algebraic Meet-in-the-Middle Attack on LowMC. In *ASIACRYPT (1)*, volume 13791 of *Lecture Notes in Computer Science*, pages 225–255. Springer, 2022.

[MJSC16] Pierrick Méaux, Anthony Journault, François-Xavier Standaert, and Claude Carlet. Towards Stream Ciphers for Efficient FHE with Low-Noise Ciphertexts. In *EUROCRYPT (1)*, volume 9665 of *Lecture Notes in Computer Science*, pages 311–343. Springer, 2016.

[ØFRC20] Morten Øygarden, Patrick Felke, Håvard Raddum, and Carlos Cid. Cryptanalysis of the Multivariate Encryption Scheme EFLASH. In *Topics in Cryptology–CT-RSA 2020: The Cryptographers' Track at the RSA Conference 2020, San Francisco, CA, USA, February 24–28, 2020, Proceedings*, pages 85–105. Springer, 2020.

[RHPScKK07] Francisco Rodríguez-Henríquez, Arturo Díaz Pérez, Nazar Abbas Saqib, and Çetin Kaya Koç. *Cryptographic Algorithms on Reconfigurable Hardware*. Springer US, Boston, MA, 2007.

[SLST23] Alan Szepieniec, Alexander Lemmens, Jan Ferdinand Sauer, and Bobbin Threadbare. The Tip5 Hash Function for Recursive STARKs. *IACR Cryptol. ePrint Arch.*, page 107, 2023.

[WWF+21] Congming Wei, Chenhao Wu, Ximing Fu, Xiaoyang Dong, Kai He, Jue Hong, and Xiaoyun Wang. Preimage Attacks on 4-Round Keccak by Solving Multivariate Quadratic Systems. In *ICISC*, volume 13218 of *Lecture Notes in Computer Science*, pages 195–216. Springer, 2021.

[ZWY+23] Kaiyi Zhang, Qingju Wang, Yu Yu, Chun Guo, and Hongrui Cui. Algebraic Attacks on Round-Reduced RAIN and Full AIM-III. Cryptology ePrint Archive, Paper 2023/1397, 2023. https://eprint.iacr.org/2023/1397.