Efficacy and Mitigation of the Cryptanalysis on AIM

Seongkwang Kim[1], Jincheol Ha[2], Mincheol Son[2], and Byeonghak Lee[1]

[1] Samsung SDS, Seoul, Korea,
{sk39.kim, byghak.lee}@samsung.com
[2] KAIST, Daejeon, Korea,
{smilecjf, encrypted.def}@kaist.ac.kr

Abstract. Recent advancements in post-quantum cryptography have highlighted signature schemes based on the MPC-in-the-Head (MPCitH) framework due to their reliance only on the one-way function of the underlying primitive. This reliance offers a diverse set of assumptions regarding the difficulty of post-quantum cryptographic problems. In this context, Kim et al. proposed AIM, an MPCitH-compatible one-way function. This function is distinguished by its large algebraic S-boxes and parallel architecture, contributing to the reduced size of signatures, as presented at CCS 2023.
However, AIM has faced several cryptanalytic challenges, which have potentially weakened its security by up to 15 bits. This paper provides a comprehensive overview of these cryptanalytic methods and proposes AIM2, an enhanced version that addresses these identified vulnerabilities. We conduct an extensive analysis of its resilience to algebraic attacks and detail the modifications in its efficiency.

1 Introduction

The MPC-in-the-Head (MPCitH) paradigm, introduced by Ishai et al. [IKOS07], represents a novel method for constructing zero-knowledge proofs (ZKPs) through the simulation of multiparty computation (MPC) protocols. This paradigm has recently been employed in the development of post-quantum signature schemes, with the security of such schemes relying solely on the one-way function utilized in their key generation process.

In this field, Kim et al. [KHS+23] proposed AIM, a one-way function optimized for MPCitH, and developed the signature scheme AIMer, which integrates the BN++ proof system [KZ22] with the symmetric primitive AIM. The AIM function is characterized by its parallel structure and Mersenne S-boxes, which are specifically designed to maximize the benefits of repeated multipliers while maintaining strong resistance to algebraic attacks. However, recent studies have identified certain algebraic vulnerabilities in AIM.

Liu et al. [LMOM23] conducted the first analysis, employing a fast exhaustive search approach on AIM. This method leverages the fact that AIM allows a low-degree system of equations in λ Boolean variables, where λ represents the security parameter. Using a fast exhaustive search algorithm [Bou22], they demonstrated a potential security degradation of up to 15 bits compared to the initial claims in [KHS+23]. The second analysis, communicated privately by Liu, devised a new low-degree equation system involving 2λ variables. Although this approach does not completely undermine AIM, it challenges the original security assertions made in [KHS+23].

A third analysis, shared by Saarinen on the PQC Forum,[3] focused on an efficient exhaustive search strategy exploiting implementational optimizations. This study proposed an unexpectedly efficient brute-force method, exploiting the fact that the input to the parallel S-boxes in AIM are all the same. The precise extent of the resulting security degradation have been unspecified.

Lastly, Zhang et al. [ZWY+23] proposed a linearization attack, which involved guessing a middle-product of the S-boxes in AIM. They claimed that this attack led to a security degradation of up to 6 bits.

1.1 Our Contribution

In this paper, we revisit the complexity estimation of the exhaustive search on AIM. As the previous estimation is so rough that the efficacy of the analyses cannot be addressed properly. Based on the new estimation, we overview the four analyses on AIM and carefully (re-)analyze the complexity if there is any ambiguity. To

[3] https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/BI2ilXblNy0

mitigate all the cryptanalysis, we propose a new version of AIM, dubbed AIM2. The main difference of AIM2 from AIM is three-fold:

1. Inverse Mersenne S-box: the S-box in the first round is placed in the opposite direction. In this way, we can make it harder to build a large number of equations compared to AIM.
2. Constant addition to the input of S-boxes: distinct constants are added to the inputs of first-round S-boxes. It differentiates the inputs of S-boxes with negligible cost.
3. Increasing exponents for S-boxes: we opt for larger exponents for some Mersenne S-boxes in order to make it harder to establish a low-degree system of equations in $\approx \lambda$ Boolean variables from a single evaluation of AIM.

We also analyze the security of AIM2 against various attacks. Finally, we implement AIMer whose symmetric primitive is replaced from AIM to AIM2, and measure how our patch affects efficiency of the resulting signature scheme.

1.2 Notation

Throughout this paper, we denote (bit-)length of AIM and AIM2 as n. Unless stated otherwise, all logarithms are to the base 2. For two vectors a and b over a finite field, their concatenation is denoted by $a\|b$. For a positive integer m, we write $[m] = \{1, \ldots, m\}$. For an integer x and a boolean vector y, $\mathsf{hw}_n(x)$ and $\mathsf{hw}(y)$ denotes the Hamming weight of $x \bmod 2^n - 1$ in its binary representation and the Hamming weight of y, respectively. For $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{F}_2^n$ and $x = (x_1, \ldots, x_n)$, monomial representation x^α means that $\prod_{i=1}^n x_i^{\alpha_i}$. For a field $\mathbb{F}$, its multiplicative group is denoted by $\mathbb{F}^\times$.

In this document, addition is usually operated on a binary field, which can be seen as bitwise exclusive-OR (XOR). When we want to emphasize this, we will write $\oplus$ to denote addition.

2 AIM and AIMer

AIM was proposed as an MPCitH-friendly symmetric primitive with high resistance to algebraic attacks [KHS+23]. AIMer is a signature scheme obtained by combining AIM with the BN++ proof system [KZ22].

Given the input/output size n and an $(\ell+1)$-tuple of exponents $(e_1, \ldots, e_\ell, e_*) \in \mathbb{Z}^{\ell+1}$,

$$\mathsf{AIM} : \{0,1\}^n \times \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$$

is defined by

$$\mathsf{AIM}(\mathsf{iv}, \mathsf{pt}) = \mathsf{Mer}[e_*] \circ \mathsf{Lin}[\mathsf{iv}] \circ \mathsf{Mer}[e_1, \ldots, e_\ell](\mathsf{pt}) \oplus \mathsf{pt}$$

where each function will be described below. The security requirement of AIM for the signature scheme AIMer is the one-wayness of AIM with respect to pt for a given public iv. See Figure 2 for the pictorial description of AIM with $\ell = 3$.

NON-LINEAR COMPONENTS. In AIM, S-boxes are exponentiation functions by Mersenne numbers over a large field. More precisely, for $x \in \mathbb{F}_{2^n}$,

$$\mathsf{Mer}[e](x) = x^{2^e - 1}$$

for some e. Note that this map is a permutation if $\gcd(e, n) = 1$. As an extension, $\mathsf{Mer}[e_1, \ldots, e_\ell] : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}^\ell$ is defined by

$$\mathsf{Mer}[e_1, \ldots, e_\ell](x) = \mathsf{Mer}[e_1](x)\| \ldots \|\mathsf{Mer}[e_\ell](x).$$

LINEAR COMPONENTS. AIM includes two types of linear components: an affine layer and feed-forward. The affine layer consists of multiplication by an $n \times \ell n$ random binary matrix A_{iv} and addition by a random constant $b_{\mathsf{iv}} \in \mathbb{F}_2^n$. The matrix

$$A_{\mathsf{iv}} = \begin{bmatrix} A_{\mathsf{iv},1} & \ldots & A_{\mathsf{iv},\ell} \end{bmatrix} \in (\mathbb{F}_2^{n \times n})^\ell$$

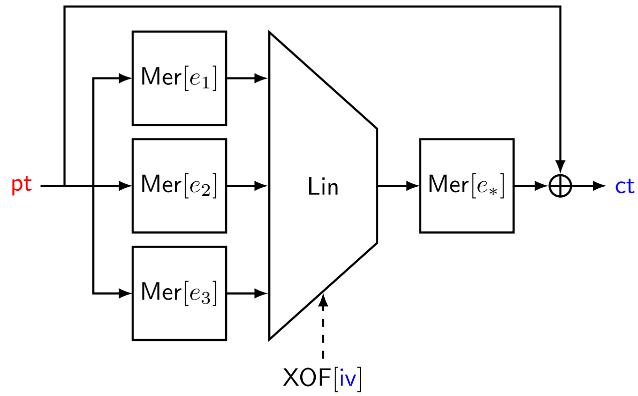

Fig. 1: The AIM-V one-way function with $\ell = 3$. The input pt (in red) is the secret key of the signature scheme, and $(\mathsf{iv}, \mathsf{ct})$ (in blue) is the corresponding public key.

is composed of ℓ random invertible matrices $A_{\mathsf{iv},i}$. The matrix A_{iv} and the vector b_{iv} are generated by an extendable-output function (XOF) with the initial vector iv. Each matrix $A_{\mathsf{iv},i}$ can be equivalently represented by a linearized polynomial $L_{\mathsf{iv},i}$ on $\mathbb{F}_{2^n}$. For $x = (x_1, \ldots, x_\ell) \in (\mathbb{F}_{2^n})^\ell$,

$$\mathsf{Lin}[\mathsf{iv}](x) = \sum_{1 \le i \le \ell} L_{\mathsf{iv},i}(x_i) \oplus b_{\mathsf{iv}}.$$

Feed-forward operation, which is addition by the input itself, makes the entire function non-invertible.

RECOMMENDED PARAMETERS. Recommended sets of parameters for $\lambda \in \{128, 192, 256\}$ are given in Table 1. The irreducible polynomials for extension fields $\mathbb{F}_{2^{128}}$, $\mathbb{F}_{2^{192}}$, and $\mathbb{F}_{2^{256}}$ are the same as those used in Rain [DKR+22].

Scheme	λ	n	ℓ	e_1	e_2	e_3	e_*
AIM-I	128	128	2	3	27	-	5
AIM-III	192	192	2	5	29	-	7
AIM-V	256	256	3	3	53	7	5

Table 1: Recommended sets of parameters of AIM.

3 Complexity Models for Algebraic Attacks

In this section, we briefly introduce some algebraic attack models and their complexities. Throughout this section, we will focus on constructing an overdetermined system of m equations in n Boolean variables where the degree of each equation is denoted as d_i for $i = 1, \ldots, m$.

3.1 XL Algorithm with Independent Equations Model

The XL algorithm [CKPS00] is a generalization of the relinearization attack [KS99]. The XL algorithm extends the system by multiplying all the monomials of degree $D - d_i$ to the equation of degree d_i, resulting in

$$S_D = \sum_{i=1}^{m} \left(\sum_{j=0}^{D-d_i} \binom{n}{j} \right)$$

equations of degrees at most D. As the extended system is of degrees at most D, at most $\sum_{i=1}^{D}\binom{n}{i}$ monomials appear in the extended system. When the number of linearly independent equations becomes greater than the number of monomials as D grows, one can solve the extended system of equations by linearization.

The complexity of the XL attack depends on the number of linearly independent equations obtained from the XL algorithm, while we may loosely upper bound the number of linearly independent equations by S_D.

Assumption 1 *All the equations obtained while running the XL algorithm are linearly independent.*

Under Assumption 1, which is in favor of the attacker, we can search for the (smallest) degree D such that

$$\sum_{i=1}^{m}\sum_{j=0}^{D-d_i}\binom{n}{j} \geq T_D \tag{1}$$

where T_D denotes the exact number of monomials appearing in the extended system of equations, which is upper bounded by $\sum_{i=1}^{D}\binom{n}{i}$. Once D is fixed, the extended system of equations can be solved by trivial linearization whose time complexity is given as $O\left(T_D^{\omega}\right)$, where the constant ω is the matrix multiplication exponent.

In the literature, Assumption 1 is not widely-used to estimate the security of a cryptosystem since it is regarded unrealistic. The equations obtained while running the XL algorithm are usually linearly dependent, and the degree D is required to be much higher than one computed under Assumption 1. Ars et al. [AFI+04] showed that the XL algorithm is in fact a redundant variant of the F_4 algorithm [Fau99]. AIM was claimed to be secure against direct algebraic attacks even if Assumption 1 is true [KHS+23].

3.2 Gröbner Basis Attack Model

The Gröbner basis attack is to solve a system of equations by computing its Gröbner basis. The attack consists of the following steps.

1. Compute a Gröbner basis in the *grevlex* (graded reverse lexicographic) order.
2. Change the order of terms to obtain a Gröbner basis in the *lex* (lexicographic) order.
3. Find a univariate polynomial in this basis and solve it.
4. Substitute this solution into the Gröbner basis and repeat Step 3.

When a system of equations has only finitely many solutions in its algebraic closure, its Gröbner basis in the *lex* order always contains a univariate polynomial. When a single variable of the polynomial is replaced by a concrete solution, the Gröbner basis still remains a Gröbner basis of the "reduced" system, allowing one to obtain a univariate polynomial again for the next variable. We refer to [SS21] for more details on Gröbner basis computation.

The security of a cryptosystem against the Gröbner basis attack is usually estimated by the complexity of the first step, which is the Gröbner basis computation in the *grevlex* order using F_4/F_5 algorithm or its variants [Fau99, Fau02]. The complexity of Gröbner basis computation can be estimated using the *degree of regularity* of the system of equations [BFS04]. Consider a system of m homogeneous equations $\{f_i(x_1,\ldots,x_n) = 0\}_{i=1}^{m}$ in n Boolean variables. Let d_i denote the degree of f_i for $i = 1, 2, \ldots, m$. If the system of equations is overdetermined, i.e., $m > n$, then the degree of regularity can be estimated by the smallest degree of the terms with non-positive coefficients appearing in the Hilbert series

$$\frac{(1+z)^n}{\prod_{i=1}^{m}(1+z^{d_i})}$$

under Assumption 2.

Assumption 2 ([Frö85]) *Almost all polynomial sequences are semi-regular.*

For nonhomogeneous equations, the degree of regularity is computed from the following Hilbert series obtained by homogenization [BFSS13]:

$$\frac{(1+z)^n}{(1-z)\prod_{i=1}^{m}(1+z^{d_i})}. \tag{2}$$

Given the degree of regularity d_{reg}, the complexity of computing a Gröbner basis of the system of equations is known to be

$$O\left(\binom{n}{d_{\mathrm{reg}}}^{\omega}\right).$$

In [KHS+23], the degree of regularity has been wrongly computed using the Hilbert series

$$\frac{1}{(1-z)^n}\prod_{i=1}^{m}(1-z^{d_i}).$$

and the complexity formula

$$O\left(\binom{n+d_{\mathrm{reg}}}{d_{\mathrm{reg}}}^{\omega}\right)$$

which gives zeroes over the algebraic closure of $\mathbb{F}_2$. As far as we check, this discrepancy leads to no significant difference of the attack complexity.

3.3 Hybrid Wiedemann XL Algorithm Model

The state-of-the-art model of solving a system of polynomial equations is to use the hybrid Wiedemann XL algorithm [BFP09, YCBC07]. This model is based on the following three techniques:

1. XL algorithm with termination at the degree of regularity (also known as the operating degree),
2. hybrid approach with the guess-and-determine attack [BFP09],
3. sparse linear system solving algorithm which is called the Wiedemann algorithm [Wie86].

Nowadays, the XL algorithm has been proved to terminate at degree d_{reg} defined by the Hilbert series (2) [YC04, YCBC07] under Assumption 2. So, the complexity of the hybrid Wiedemann XL algorithm on a system of Boolean equations is upper bounded by

$$\min_k 3\cdot 2^k\cdot\binom{n-k}{d_{\mathrm{reg}}(n,k)}^2\cdot\binom{n-k}{\max_i d_i} \tag{3}$$

where the degree of regularity $d_{\mathrm{reg}}(n,k)$ is the smallest degree of the terms with non-positive coefficients of the Hilbert series

$$\frac{(1+z)^{n-k}}{(1-z)\prod_{i=1}^{m}(1+z^{d_i})}. \tag{4}$$

3.4 Complexity Model in this Paper

In the previous sections, we introduced three complexity models for algebraic attacks (XL and Gröbner basis computation). Although the hybrid Wiedemann XL algorithm is the most widely-deployed model, we use the Gröbner basis attack model with $\omega = 2$ and hybrid approach [BFP09] since the complexity of this model lower bounds that of the hybrid Wiedemann XL model. Specifically, we use the complexity formula

$$\min_k 2^k\cdot\binom{n-k}{d_{\mathrm{reg}}(n,k)}^2 \tag{5}$$

where $d_{\mathrm{reg}}(n,k)$ is the smallest degree of the terms with non-positive coefficients of (4).

Operation	#AND	#XOR	#Gate	Note
FF Mult.	n^2	n^2	$2n^2$	[DCK+20]
FF Square	0	n	n	Underestimated
Mat-Vec Mult.	0	n^2	n^2	Fixed $n \times n$-matrix
FF Add	0	n	n	

Table 2: Summary of required number of binary gates. We compute the gate-count complexities of analyses which will be described in this section. "FF" is the abbreviation of finite field. The finite field implies $\mathbb{F}_{2^n}$, and the vector is binary vector of length n.

Scheme	Circuit	Chain: $(a) \to (b) = x^{2^a-1} \to x^{2^b-1}$	# Operations	
			FF Mult.	FF Square
AIM-I	Mer$[3, 27]$	$(1) \to (2) \to (3) \to (6) \to (12) \to (24) \to (27)$	6	26
	Mer$[5]$	$(1) \to (2) \to (4) \to (5)$	3	4
AIM-III	Mer$[5, 29]$	$(1) \to (2) \to (3) \to (5) \to (7) \to (14) \to (28) \to (29)$	7	28
	Mer$[7]$	$(1) \to (2) \to (4) \to (6) \to (7)$	4	6
AIM-V	Mer$[3, 7, 53]$	$(1) \to (2) \to (3) \to (5) \to (7) \to (12) \to (24) \to (48) \to (53)$	8	52
	Mer$[5]$	$(1) \to (2) \to (4) \to (5)$	3	4

Table 3: The addition chain used to compute each Mersenne-Sbox in AIM and the number of field operations it takes.

4 Cryptanalysis on AIM

4.1 Claimed Security of AIM

Before we address analyses on AIM proposed so far, we clarify the complexity of exhaustive search attack on AIM. The complexities have been differently described in [KHS+23] and [KCC+23]. We revisit how the complexities have been computed, and provide more accurate figures with explicitly written assumptions.

In the earliest ePrint version of [KHS+23], the S-boxes in AIM are assumed to be computed by naive square-and-multiply method. Then, each S-box Mer$[e]$ requires $2(e-1)$ multiplications over $\mathbb{F}_{2^n}$, so that we counted the numbers of finite field multiplication are 32, 38, and 64 for AIM-I, AIM-III, and AIM-V, respectively. We assumed that a single $\mathbb{F}_{2^n}$-multiplication requires $2n^2$ binary gates, and computed the gate-count complexity by multiplying 2^λ and the number of finite field multiplications. The resulted complexity was given as 2^{149}, $2^{214.4}$, and 2^{280} for AIM-I, AIM-III, and AIM-V, respectively.

While we were preparing the submission for the NIST PQC project, we found out that addition chain exponentiation technique [Knu97] can reduce the number of required finite field multiplication to evaluate AIM. We reduce it to 11/14/17 for AIM-I/III/V by utilizing the addition chain exponentiation technique. Similarly to previous paragraph, we computed the gate-count complexity by multiplying 2^λ and the number of finite field multiplications. The resulted complexity was given as $2^{146.4}$, $2^{211.9}$, and 2^{277} for AIM-I, AIM-III, and AIM-V, respectively.

Those complexity estimations in fact have omitted some non-dominant computations: finite field squaring, matrix-vector multiplication, addition and comparison of finite field elements. So, the estimations have provided a lower bound of complexity of the exhaustive search. To be accurate, we summarize the assumed gate count for each component operations in Table 2. We assume that a multiplication by a fixed matrix does not require any AND gate. We also assume that squaring of a finite field element requires n XOR gates which is obviously underestimated, but this does not affect total complexities much.

On the other hand, we found that the S-boxes in the first round (parallel structure) can be computed in a single addition chain since they have all the same inputs. It reduces the required number of multiplication as in Table 3. Overall, we summarize the number of required operations and the total cost of exhaustive search on AIM in Table 4.

Scheme	#Operations				Total Cost
	FF Mult.	FF Square	Mat-Vec Mult.	FF Add	
AIM-I	9	30	2	4	146.3
AIM-III	11	34	2	4	211.8
AIM-V	11	56	3	5	276.7

Table 4: The number of operations for each type of operation used in AIM, and the total cost of exhaustive search on AIM for each level of security. The total cost is the log of required number of binary gates.

4.2 Fast Exhaustive Search for Low-degree Algebraic System

Exhaustive search is the most basic attack for any keyed function $f_k(\cdot)$. For some given pairs (x_i, y_i) such that $f_k(x_i) = y_i$, an attacker checks whether $f_{\bar{k}}(x_i) = y_i$ or not for all i over all possible keys $\bar{k}$ in the key space. Fast exhaustive search improves concrete efficiency of exhaustive search when the keyed function can be represented by a set of low-degree polynomials.

For a degree-d system in n variables, Bouillaguet et al. proposed a fast exhaustive search with time complexity $4d\log(n)2^n$ in Boolean operations and memory complexity $O(n^{2d})$ [BCC+10]. Bouillaguet also proposed a memory-efficient version of the fast exhaustive search with the same time complexity and memory complexity $n^2 \cdot \sum_{i=0}^{d} \binom{n}{i}$ in bits [Bou22]. We refer to the original papers for more details.

Liu et al. proposed a low-degree representation of AIM, and applied the fast exhaustive search algorithm to it [LMOM23]. The low-degree representation is described as follows.

Let z be the output of Lin. Then, pt can be represented in terms of z as follows.

$$\mathsf{pt} = z^{2^{e_*}-1} + \mathsf{ct}$$

Denoting the output of $\mathsf{Mer}[e_i]$ by t_i for $i = 1, \ldots, \ell$, one has

$$t_i = \left(z^{2^{e_*}-1} + \mathsf{ct}\right)^{2^{e_i}-1}.$$

Let d_i be the degree of t_i with respect to z, and let $d_{\max} = \max_{i \neq 2} d_i$ as the exponent e_2 is the largest from $\{e_1, \ldots, e_\ell\}$ (for the sets of recommended parameters). Then, t_2 can be expressed as

$$t_2 = A_{\mathsf{iv},2}^{-1} \left(b_{\mathsf{iv}} + z + A_{\mathsf{iv},1}(t_1) + A_{\mathsf{iv},3}(t_3)\right)$$

where the last summand $A_{\mathsf{iv},3}(t_3)$ in the parentheses does not appear for AIM-I or AIM-III. Now we obtain an equation of degree at most $d_{\max} + e_*$ from $\mathsf{pt} \cdot t_2 = \mathsf{pt}^{2^{e_*}}$ as follows.

$$\left(z^{2^{e_*}-1} + \mathsf{ct}\right) \cdot A_{\mathsf{iv},2}^{-1} \left(b_{\mathsf{iv}} + z + A_{\mathsf{iv},1}(t_1) + A_{\mathsf{iv},3}(t_3)\right) = \left(z^{2^{e_*}-1} + \mathsf{ct}\right)^{2^{e_2}}$$

The degree $d_{\max} + e_*$ is known to be 10/14/15 for AIM-I, III, V, respectively.[4] As the time complexity of the fast exhaustive search is $4d(\log n)2^n$, the gate-count complexity becomes $2^{136.2}/2^{200.7}/2^{265.0}$ for AIM-I, III, V, respectively. According to the new complexity estimation of exhaustive search on AIM in Section 4.1, it degrades the security of AIM by up to 12 bits.

4.3 Possible Algebraic Vulnerability on AIM

While communicating with the authors of [LMOM23], Liu pointed out that introducing a new variable results in an easier system of equations than expected. In this section, we briefly introduce how to make such a system.

[4] The detailed computation can be found in [LMOM23].

Let a new variable $w = \mathsf{pt}^{-1}$, and let t_i be the output of $\mathsf{Mer}[e_i]$ for $i \in \{1, \ldots, \ell\}$. Then, we have

$$t_i = \mathsf{pt}^{2^{e_i}} w$$

for all $i = 1, \ldots, \ell$. Then we can establish three types of equations

$$\mathsf{pt} \cdot w = 1, \tag{6}$$

$$\mathsf{Lin}\left(\mathsf{pt}^{2^{e_1}} w, \ldots, \mathsf{pt}^{2^{e_\ell}} w\right) \cdot (\mathsf{pt} + \mathsf{ct}) = \mathsf{Lin}\left(\mathsf{pt}^{2^{e_1}} w, \ldots, \mathsf{pt}^{2^{e_\ell}} w\right)^{2^{e_*}}, \tag{7}$$

$$\mathsf{Lin}\left(\mathsf{pt}^{2^{e_1}} w, \ldots, \mathsf{pt}^{2^{e_\ell}} w\right) \cdot (1 + w \cdot \mathsf{ct}) = \mathsf{Lin}\left(\mathsf{pt}^{2^{e_1}} w, \ldots, \mathsf{pt}^{2^{e_\ell}} w\right)^{2^{e_*}} \cdot w. \tag{8}$$

Since the inverse S-box of n-bit input produces $5n$ linearly independent quadratic equations, we obtain $5n$ quadratic equations from (6). For (7) and (8), each produces n cubic equations, and multiplying pt and w results in n more cubic equations, respectively. Moreover, we have

$$\begin{aligned}
&\mathsf{Lin}\left(\mathsf{pt}^{2^{e_1}} w, \ldots, \mathsf{pt}^{2^{e_\ell}} w\right)^2 \cdot (\mathsf{pt} + \mathsf{ct}) + \mathsf{Lin}\left(\mathsf{pt}^{2^{e_1}} w, \ldots, \mathsf{pt}^{2^{e_\ell}} w\right)^2 \cdot (1 + w \cdot \mathsf{ct}) \cdot \mathsf{ct} \\
&= \mathsf{Lin}\left(\mathsf{pt}^{2^{e_1}} w, \ldots, \mathsf{pt}^{2^{e_\ell}} w\right)^{2^{e_*}+1} + \mathsf{Lin}\left(\mathsf{pt}^{2^{e_1}} w, \ldots, \mathsf{pt}^{2^{e_\ell}} w\right)^{2^{e_*}+1} \cdot w \cdot \mathsf{ct} \\
&= \mathsf{Lin}\left(\mathsf{pt}^{2^{e_1}} w, \ldots, \mathsf{pt}^{2^{e_\ell}} w\right)^{2^{e_*+1}} \cdot w
\end{aligned}$$

which produces n more cubic equations. We remark that the exponents of the Lin term in the second line and in the third line are different ($2^{e_*} + 1$ vs. 2^{e_*+1}). Overall, we have a system of $5n$ quadratic equations and $5n$ cubic equations in $2n$ Boolean variables regardless of ℓ.

Under Assumption 1 and the condition $\omega = 2$, the time complexity of the XL algorithm is $2^{124.8}/2^{157.5}/2^{188.9}$ for AIM-I/III/V, which harms the original security claim in [KHS+23]. However, this assumption is usually regarded *too strong* as all the expanded equations are not likely to be linearly independent. This assumption estimates the complexity much lower than the required amount of computation for the XL algorithm [AFI+04] in practice. If we estimate the complexity in the hybrid Gröbner basis attack model with Assumption 2 which is regarded as a more realistic assumption, the time complexity of the XL algorithm is $2^{158.3}/2^{226.5}/2^{290.2}$ for AIM-I/III/V. Those values imply all the instances are secure against the XL algorithm.

The main reason of this vulnerability is insufficient difference between S-boxes in the first round. Since the exponents are simple and similar to each other, it is possible to set a new variable from a common factor.

4.4 Efficient Exhaustive Search by Optimized Implementation

Saarinen claimed that an efficient exhaustive search attack using linear-feedback shift register (LFSR) can results in a security degradation of AIM.[5] To be accurate, he claimed that the complexity of exhaustive search of AIM's input is less than that of AES. We briefly describe the exhaustive search algorithm by Saarinen in the following.

At first, an attacker needs an LFSR corresponding to a given field $\mathbb{F}_{2^n}$. In the LFSR, moving forward (resp. backward) shifting corresponds to multiplying X (resp. X^{-1}). Iteratively shifting the element x forward and its corresponding inverse $w = x^{-1}$ backward makes the attacker can visit all the element in $\mathbb{F}_{2^n}$. For each shift after initializing $x = 1$ and $w = 1$, attacker can check whether $\mathsf{pt} = x, \mathsf{pt}^{-1} = w$ satisfies (7).

As Saarinen have not provided the explicit complexity of the exhaustive search attack, we analyze the exact gate-count complexity of the attack. Checking whether (7) holds or not costs $\ell + 1$ finite field multiplications, $(\max\{e_1, \ldots, e_\ell\} + e_*)$ finite field squarings, ℓ matrix-vector multiplication, and $\ell + 1$ field additions. Assuming Table 2, the gate-count complexity of an efficient exhaustive search using LFSR is given as $2^{145.0}$, $2^{210.2}$, and $2^{275.5}$ for AIM-I, AIM-III, AIM-V, respectively. Contrary to the claim of Saarinen, these values are still larger than the gate-count complexity of AES ($2^{143}/2^{207}/2^{272}$), while these values are slightly (< 2 bit) smaller than the security of AIM in Table 4. The number of required operations and the total cost of exhaustive search attack using LFSR are summarized in Table 5.

[5] https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/BI2ilXblNy0

Scheme	#Operations				Total Cost
	FF Mult.	FF Square	Mat-Vec Mult.	FF Add	
AIM-I	3	32	2	3	145.0
AIM-III	3	36	2	3	210.2
AIM-V	4	58	3	4	275.5

Table 5: The number of operations for each type of operation and the total cost of exhaustive search on (7) for each security level. The total cost is the log of required number of binary gates.

4.5 Linearization Attack by Guessing Intermediate Variables

Zhang et al. [ZWY+23] proposed an algebraic attack on AIM that exploits the linearity of the square function over binary extension fields and the vulnerability of AIM having the common input to multiple S-boxes. A Mersenne S-boxes in the first round, which maps as $x \mapsto x^{2^{e_i}-1}$ over $\mathbb{F}_{2^n}$ for $i = 1, \ldots, \ell$, can be decomposed as follows.

$$x^{2^{e_i}-1} = (x^d)^{s_i} \cdot x^{2^{t_i}}$$

for some positive integers d, s_i, t_i for $i = 1, \ldots, \ell$, where $d \mid 2^n - 1$. Since the power mapping $x \mapsto x^{2^{t_i}}$ is linear over $\mathbb{F}_2$, one can linearize all the Mersenne S-boxes $x \mapsto x^{2^{e_i}-1}$ for $i = 1, \ldots, \ell$ by guessing the value of x^d. As $\{x^d \,|\, x \neq 0\}$ forms a multiplicative subgroup of $\mathbb{F}_{2^n}^\times$ for $d \mid 2^n - 1$, the number of possible guesses for nonzero x^d is only $(2^n - 1)/d$.

Once the value of pt^d is guessed as α, one can represent the input to the last Mersenne S-box $\mathsf{Mer}[e_*]$ as follows, which is linear to pt.

$$\mathsf{Lin}(\alpha^{s_1}\mathsf{pt}^{2^{t_1}}, \ldots, \alpha^{s_\ell}\mathsf{pt}^{2^{t_\ell}}).$$

Since the output of the last Mersenne S-box is $(\mathsf{pt} + \mathsf{ct})$ where ct is a public value, one can obtain n quadratic equations over $\mathbb{F}_2$ from the following relation.

$$\mathsf{Lin}(\alpha^{s_1}\mathsf{pt}^{2^{t_1}}, \ldots, \alpha^{s_\ell}\mathsf{pt}^{2^{t_\ell}}) \cdot (\mathsf{pt} + \mathsf{ct}) = \mathsf{Lin}(\alpha^{s_1}\mathsf{pt}^{2^{t_1}}, \ldots, \alpha^{s_\ell}\mathsf{pt}^{2^{t_\ell}})^{2^{e_*}}. \tag{9}$$

In case of AIM-III with $n = 192$, Zhang et al. chose $d = 45$ for the most efficient attack. Then the following linear relation on pt and $\alpha = \mathsf{pt}^d$ is used to solve (9).

$$\mathsf{pt}^{2^{12}} = (\mathsf{pt}^{45})^{91} \cdot \mathsf{pt} = \alpha^{91} \cdot \mathsf{pt}.$$

Considering α as a guessed (and so fixed) value, $\gamma = 180$ linearly independent equations on pt over $\mathbb{F}_2$ are obtained from the above equation experimentally. Thus, one can express each bit of pt as a linear combination of $n - \gamma = 12$ free variables $v = \{v_j\}_{j \in [n-\gamma]}$. By changing variables of (9) from pt to v, one can rewrite the quadratic equations with fewer variables. Then, using $m = |v| + |v|(|v| - 1)/2$ linearly independent equations among them, it can be solved by (trivial) linearization.

The attack complexity consists of three parts. The first step is to find the set of free variables v from the linear relation on pt and α by the Gaussian elimination, which takes $O(n^3)$ time. Next, rewriting the quadratic equations on the input pt from (9) to ones on the free variables v takes $O(n^2|v|^2)$ times. Finally, solving the quadratic equations on v takes $O(m^3)$ time. The above steps are repeated $(2^n - 1)/d$ times according to the guessed value of α. Considering only the dominant term among them, Zhang et al. estimate the bit complexity of their attack as $T_{bit} = (2^n - 1)/d \cdot \max\{n^3, n^2|v|^2, m^3\}$. When interpreting the bit complexity in terms of the number of encryption, they divide T_{bit} by n^3 assuming the encryption takes $O(n^3)$ time in bit complexity.

Although Zhang et al. estimated the complexities of their attack by only using asymptotic complexity, we carefully re-analyze the complexity of the attack with hidden constants as it was claimed to degrade the security of AIM by a few bits.

1. The number of binary operations for Gaussian elimination is about $n(n-1) + \cdots + 2 \cdot 1 \approx n^3/3$. Some alternative algorithms for matrix multiplication have smaller exponents asymptotically (e.g., $\omega = 2.81$

or 2.37), while they have larger hidden constants in big-O notation [CH17]. For the matrix of dimension less than or equal to 256, Gaussian elimination has smaller cost than the alternative algorithms in all the parameters.
2. Rewriting the equations from (9) for the set of variables v involves approximately $2.5n^2|v|^2 + n|v|$ terms to change, which is experimentally checked in the original paper.[6] As the estimation omit the cost for rewriting itself, we also neglect it in our estimation.
3. As all the steps should have done for every guess, the estimation should be $T^*_{bit} = (2^n - 1)/d \cdot (n^3/3 + 2.5n^2|v|^2 + n|v| + m^3/3)$.
4. Contrary to our initial circuit size estimation of AIM, the circuit size of AIM is turned out to be not $O(n^3)$ but $O(\ell n^2)$. Concretely speaking, the estimated circuit size is $2^{18.3}/2^{19.8}/2^{20.7}$ for AIM-I/III/V, which are quite a few bit smaller than n^3.

Table 6 summarizes the recommended parameters and the corresponding attack complexities to each level of AIM. One can see that the attack complexities for AIM-V (resp. AIM-I, III) estimated by the dominant term is slightly greater (resp. smaller) than the exhaustive search attack complexity. Based on the new complexity estimation of the exhaustive search on AIM, this attack degrades the security by up to 2 bits. Even considering that our estimated complexity for exhaustive search has been reduced, the efficacy of the attack is slightly overestimated if the hidden constants are handled carefully.

Scheme	n	d	t_1	t_2	t_3	$\lvert v\rvert$	m	T^*_{bit}	Exhaustive Search (Sec. 4.1)
AIM-I	128	5	1	1	-	4	10	$2^{146.0}$	$2^{146.3}$
AIM-III	192	45	8	8	-	12	78	$2^{210.4}$	$2^{211.8}$
AIM-V	256	3	0	0	0	2	3	$2^{277.0}$	$2^{276.7}$

Table 6: The recommended attack parameters and the corresponding gate-count complexities of the attack proposed by Zhang et al.

5 Mitigation on the Cryptanalysis

In the presented analyses, the vulnerabilities of AIM can be described in two primary aspects: identical input values to the S-boxes and the low-degree system. The easier system by Liu, Saarinen's efficient exhaustive search, and the linearization attack employed by Zhang et al. are fundamentally premised on the observation that the S-boxes in the initial round receive identical inputs. Additionally, the fast exhaustive search attack proposed by Liu et al. is based on the system of a low-degree polynomial with a moderately sized set of variables.

This section introduces an enhanced version of AIM, dubbed AIM2. A comprehensive examination has led to an effective solution for these vulnerabilities, which simultaneously maintains a similar level of efficiency. In AIM2, different constant values are added to ensure distinct inputs for the S-boxes on the first round, thereby achieving input differentiation with minimal impact on performance. Furthermore, increasing the exponents e in $\mathsf{Mer}[e]$ substantially enlarges degree of a system with a moderate number of variables.

While these modifications effectively mitigate current known attacks, an additional safeguard is introduced in the form of the inverse Mersenne S-box. This S-box is the inverse function of a Mersenne S-box, further complicating the generation of a large amount of quadratic equations as observed in Section 4.3. The integration of the inverse Mersenne S-box, along with the feasibility of repeated multipliers, ensures that the operational overhead remains comparable to AIMer with the previous version of AIM.

[6] In the original paper, the authors upper bound the number of terms by $3n^2|v|^2 + n|v|$. But we use the ratio 2.5 as Table 2 in the original paper indicates it.

5.1 AIM2: Overall Patch

Given input/output size n and an $(\ell+1)$-tuple of exponents $(e_1, \ldots, e_\ell, e_*) \in \mathbb{Z}^{\ell+1}$, $\mathsf{AIM2} : \{0,1\}^n \times \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ is defined by

$$\mathsf{AIM2}(\mathsf{iv}, \mathsf{pt}) = \mathsf{Mer}[e_*] \circ \mathsf{Lin}[\mathsf{iv}] \circ \mathsf{Mer}[e_1, \ldots, e_\ell]^{-1} \circ \mathsf{AddConst}(\mathsf{pt}) \oplus \mathsf{pt}$$

where each function will be described below. See Figure 2 for the pictorial description of AIM2 with $\ell = 3$.

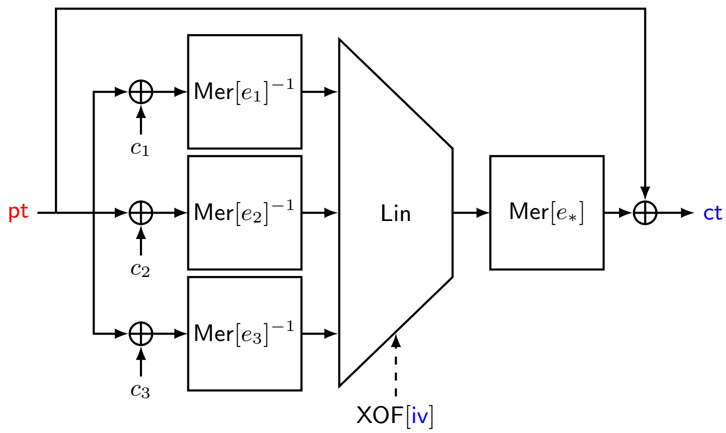

Fig. 2: The AIM2-V one-way function with $\ell = 3$. The input pt (in red) is the secret key of the signature scheme, and $(\mathsf{iv}, \mathsf{ct})$ (in blue) is the corresponding public key.

NON-LINEAR COMPONENTS. AIM2 uses two types of S-boxes: Mersenne S-box $\mathsf{Mer}[e]$, and its inverse $\mathsf{Mer}[e]^{-1}$. These two S-boxes are defined by exponentiation over a large field as follows. For $x \in \mathbb{F}_{2^n}$,

$$\begin{aligned} \mathsf{Mer}[e](x) &= x^{2^e - 1}, \\ \mathsf{Mer}[e]^{-1}(x) &= x^{\bar{e}} \quad \text{where } \bar{e} = (2^e - 1)^{-1} \pmod{2^n - 1} \end{aligned}$$

for some e. To follow the spirit of AIM, the exponents e in AIM2 are selected for $\mathsf{Mer}[e]^{-1}$ to have $3n$ quadratic equations. We remark that the exponents e are chosen such that $\gcd(e, n) = 1$, and hence the inverse exponent $\bar{e}$ is well-defined. As an extension, $\mathsf{Mer}[e_1, \ldots, e_\ell]^{-1} : \mathbb{F}_{2^n}^\ell \to \mathbb{F}_{2^n}^\ell$ is defined by

$$\mathsf{Mer}[e_1, \ldots, e_\ell]^{-1}(x_1, \ldots, x_\ell) = \mathsf{Mer}[e_1]^{-1}(x_1) \| \ldots \| \mathsf{Mer}[e_\ell]^{-1}(x_\ell).$$

LINEAR COMPONENTS. AIM2 includes three types of linear components: constant addition, an affine layer, and feed-forward. For fixed constants $c_1, \ldots, c_\ell$, $\mathsf{AddConst} : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}^\ell$ is defined by

$$\mathsf{AddConst}(x) = (x + c_1) \| \ldots \| (x + c_\ell)$$

where the constants are defined in Table 7.

The affine layer in AIM2 is exactly the same as AIM. It consists of multiplication by an $n \times \ell n$ random binary matrix A_{iv} and addition by a random constant $b_{\mathsf{iv}} \in \mathbb{F}_2^n$. The matrix

$$A_{\mathsf{iv}} = \begin{bmatrix} A_{\mathsf{iv},1} & \ldots & A_{\mathsf{iv},\ell} \end{bmatrix} \in (\mathbb{F}_2^{n \times n})^\ell$$

is composed of ℓ random invertible matrices $A_{\mathsf{iv},i}$. The matrix A_{iv} and the vector b_{iv} are generated by an extendable-output function (XOF) with the initial vector iv. Each matrix $A_{\mathsf{iv},i}$ can be equivalently represented by a linearized polynomial $L_{\mathsf{iv},i}$ over $\mathbb{F}_{2^n}$. For $x = (x_1, \ldots, x_\ell) \in (\mathbb{F}_{2^n})^\ell$,

$$\mathsf{Lin}[\mathsf{iv}](x) = \sum_{1 \le i \le \ell} L_{\mathsf{iv},i}(x_i) \oplus b_{\mathsf{iv}}.$$

AIM2-I	c_1	0x243f6a88 85a308d3 13198a2e 03707344
	c_2	0xa4093822 299f31d0 082efa98 ec4e6c89
AIM2-III	c_1	0x452821e6 38d01377 be5466cf 34e90c6c c0ac29b7 c97c50dd
	c_2	0x3f84d5b5 b5470917 9216d5d9 8979fb1b d1310ba6 98dfb5ac
AIM2-V	c_1	0x2ffd72db d01adfb7 b8e1afed 6a267e96 ba7c9045 f12c7f99 24a19947 b3916cf7
	c_2	0x0801f2e2 858efc16 636920d8 71574e69 a458fea3 f4933d7e 0d95748f 728eb658
	c_3	0x718bcd58 82154aee 7b54a41d c25a59b5 9c30d539 2af26013 c5d1b023 286085f0

Table 7: Constants $c_1, \ldots, c_\ell$ in AddConst are written in hexadecimal. These constants are taken from the numbers below the decimal point of the π ratio.

Feed-forward operation, which is addition by the input itself, makes the entire function non-invertible.

RECOMMENDED PARAMETERS. Recommended sets of parameters for $\lambda \in \{128, 192, 256\}$ are given in Table 8. The irreducible polynomials for extension fields $\mathbb{F}_{2^{128}}$, $\mathbb{F}_{2^{192}}$, and $\mathbb{F}_{2^{256}}$ are the same as those used in Rain [DKR+22].

Scheme	λ	n	ℓ	e_1	e_2	e_3	e_*
AIM2-I	128	128	2	49	91	-	3
AIM2-III	192	192	2	17	47	-	5
AIM2-V	256	256	3	11	141	7	3

Table 8: Recommended sets of parameters of AIM2.

5.2 Algebraic Attacks on AIM2

VARIOUS SYSTEMS OF AIM2. There are multiple ways of building a system of equations from an evaluation of AIM2. We can categorize them according to the number of (Boolean) variables and find the optimal choice of variables to obtain a system of the lowest degree. Since $\ell \in \{2, 3\}$ is recommended, we consider four types of systems of Boolean equations as follows.

1. Systems in n variables.
2. Systems in $2n$ variables.
3. Systems in $3n$ variables.
4. Systems in $4n$ variables (only for $\ell = 3$).

With $(\ell + 1)n$ variables, we can establish a system S_{quad} of *quadratic* equations. The variables are denoted as follows.

- x: the input of AIM2, i.e., pt
- t_i: the output of $\mathsf{Mer}[e_i]^{-1}$ for $i = 1, \ldots, \ell$
- z: the output of Lin

From $\mathsf{Mer}[e_i]^{-1}(x + c_i) = t_i$, we obtain $3n$ Boolean quadratic equations in x and t_i induced by the following relations.

$$\begin{cases} t_i(x + c_i) = t_i^{2^{e_i}}, \\ t_i(x + c_i)^2 = t_i^{2^{e_i}}(x + c_i), \\ t_i^2(x + c_i) = t_i^{2^{e_i}+1}. \end{cases}$$

When x and t_i are of higher degrees with respect to other variables, the first two relations result in $2n$ equations of degree $\deg x + \deg t_i$, while the last one results in n equations of degree $\max(\deg x + \deg t_i, 2\deg t_i)$. There are also n Boolean quadratic equations in t_i and t_j induced by the following.

$$(c_i + c_j)t_i t_j = t_i^{2^{e_i}} t_j + t_i t_j^{2^{e_j}}.$$

We note that z has the same relation with t_i with respect to x as $z = \mathsf{Mer}[e_*]^{-1}(x + \mathsf{ct})$. Using the brute-force search of quadratic equations on toy parameters, which will be described later in this section, we find that these are all the possible (linearly independent) quadratic equations on AIM2. Hence, S_{quad} consists of $3(\ell+1)n + \binom{\ell+1}{2}n$ quadratic equations.

With fewer variables, the resulting systems would have higher degrees. For example, $\mathsf{Mer}[e_i]^{-1}$ implicitly determines $3n$ quadratic equations in x and t_i as above, while t_i (resp. x) can be explicitly represented by a polynomial in x (resp. t_i). We can also explicitly represent t_i using t_j for $j \neq i$ or z as follows.

$$\begin{aligned} t_i &= \mathsf{Mer}[e_i]^{-1}\left(\mathsf{Mer}[e_j](t_j) \oplus c_i \oplus c_j\right) \\ &= \mathsf{Mer}[e_i]^{-1}\left(\mathsf{Mer}[e_*](z) \oplus \mathsf{ct}\right). \end{aligned}$$

The degree of t_i with respect to t_j (resp. z) might be greater than the degree of $\mathsf{Mer}[e_i]^{-1} \circ \mathsf{Mer}[e_j]$ (resp. $\mathsf{Mer}[e_i]^{-1} \circ \mathsf{Mer}[e_*]$) due to the constant addition, while we estimate it as the degree of the composition (without constant addition) for simplicity.

Scheme	Type	#Var	Variables	(#Eq, Deg)	Complexity		
					k	d_{reg}	Time (bits)
AIM2-I	S_1	n	t_1	$(n, 60)$	-	-	-
	S_2	$2n$	t_1, t_2	$(3n, 2)$	62	15	207.9
	S_{quad}	$3n$	x, t_1, t_2	$(12n, 2)$	0	16	185.3
AIM2-III	S_1	n	x	$(2n, 114)$	-	-	-
	S_2	$2n$	t_1, t_2	$(3n, 2)$	100	20	301.9
	S_{quad}	$3n$	x, t_1, t_2	$(12n, 2)$	0	22	262.4
AIM2-V	S_1	n	x	$(2n, 172)$	-	-	-
	S_2	$2n$	t_2, z	$(n, 2) + (2n, 38)$	253	30	513.5
	S_3	$3n$	t_1, t_2, t_3	$(6n, 2)$	2	47	503.7
	S_{quad}	$4n$	x, t_1, t_2, t_3	$(18n, 2)$	9	32	411.4

Table 9: Optimal systems of equations and their security against algebraic attacks. $(\#\mathrm{Eq}, \mathrm{Deg}) = (a, b)$ means that the system contains a equations of degree b. All the complexities are measured by (5). k is the number of guessed bits and d_{reg} is the degree of regularity.

Table 9 summarizes a system of equations of the lowest degree for each type, where such systems are denoted by $S_1, S_2, \ldots, S_{\mathsf{quad}}$ respectively, according to the number of variables. The complexities are measured by (5). For systems of equations of type S_1 in n variables, we did not compute precise complexities since the degree near $n/2$ requires the XL algorithm to use approximately 2^n monomials with time complexity close to $O(2^{2n})$.

BRUTE-FORCE SEARCH OF QUADRATIC EQUATIONS. Given an overdetermined quadratic system, algebraic attacks tend to solve the system faster when the system has more linearly independent equations. To lower bound the complexity of the algebraic attacks, we need to find all linearly independent equations. To find all such equations, we used brute-force search with the following experiment.

1. Set variables as follows.
   - x: the input of AIM2, i.e., pt
   - t_i: the output of $\mathsf{Mer}[e_i]^{-1}$ for $i = 1, \ldots, \ell$

- z: the output of $\mathsf{Mer}[e_*]^{-1}(x + \mathsf{ct})$

2. Make a generic quadratic equation with indeterminate coefficients $a_{\alpha,\beta,\gamma} \in \mathbb{F}_2$;

$$\sum_{\substack{\alpha,\gamma\in\mathbb{F}_2^n,\beta\in\mathbb{F}_2^{\ell n}\\ \mathsf{hw}_n(\alpha)+\mathsf{hw}_n(\beta)+\mathsf{hw}_n(\gamma)\leq 2}} a_{\alpha,\beta,\gamma} x^{\alpha} t_i^{\beta_i} z^{\gamma} = 0 \tag{10}$$

where $\beta = (\beta_1, \ldots, \beta_\ell)$.

3. Randomly sample $x \in \mathbb{F}_{2^n}$, and compute the corresponding t_i and z. Substitute those values to (10).
4. Repeat the previous step $O\left(\binom{(\ell+2)n}{2}\right)$ times.
5. Solve the system of linear equations with respect to $a_{\alpha,\beta,\gamma}$. The quadratic equations for the target system can be computed by substituting such $a_{\alpha,\beta,\gamma}$ to (10).

For system S_{quad}, this experiment found $12n$ quadratic equations for AIM2-I and III, and $18n$ quadratic equations for AIM2-V. For system S_2 of AIM2-I and III, it found $3n$ quadratic equations. For system S_3 of AIM2-V, it found $6n$ quadratic equations. We remark that this experiment does not consider the affine layer by introducing a redundant variable z. Although this may lead to more equations than the actual number, we checked that all the equations obtained from the experiment are linearly independent.

This experiment can be easily generalized for a general degree d. However, the generalized experiment will include *all* the equations of degree d expanded from the quadratic equations. For this reason, we opted for finding equations of a higher degree by hand rather than running the generalized experiment.

Resistance to Fast Exhaustive Search. The fast exhaustive search attacks in [BCC+10, Bou22] are infeasible if the target polynomial system is of high degree. Although the time complexity of the fast exhaustive search is claimed to be $4d\log(n)2^n$, there is a hidden preprocessing cost

$$T = \sum_{k=0}^{d-1} k\binom{n}{k}\binom{k}{\downarrow \min(d-k,k)} \geq \frac{2d}{3}2^{2d/3}\binom{n}{\lfloor 2d/3\rfloor}$$

in binary operations where $\binom{n}{\downarrow k} = \sum_{i=0}^{k}\binom{n}{i}$. One can see that $T \gg d2^n$ if $d \geq 0.341n$. Furthermore, if $d \geq n/2$, then the memory complexity will also be higher than 2^n bits.

Resistance to Linearization Attack by Guessing. The constant addition by AddConst prevents the linearization attack by Zhang et al. [ZWY+23] by making inputs to the S-boxes on the first round different. This is the simplest patch among the possible ones of AIM proposed by the authors.

Introducing New Variables Other Than S-box Outputs. As seen in Section 4.3, Liu showed that the number of quadratic equations can be increased by introducing new variables ($w = \mathsf{pt}^{-1}$) in addition to the inputs and the outputs of the S-boxes without significantly increasing the degree of the entire system of equations. We will further generalize Liu's attack, and analyze the security of AIM2 against this type of attacks. For simplicity, we write $t_{\ell+1} = z$ and $c_{\ell+1} = \mathsf{ct}$. To mount a successful attack by introducing new variables $w_i = (\mathsf{pt} + c_i)^a$ (instead of t_i) for some $i \in \{1, \ldots, \ell+1\}$, the following two conditions should hold.

1. The number of quadratic equations between x and the chosen w_i's should be greater than the number of quadratic equations between x and the corresponding t_i's.
2. The degree $\deg t_i$ of t_i with respect to x and w_i's should not be too large for the chosen i's.

We first categorize the exponent a yielding quadratic equations. We claim that the two conditions described above cannot hold simultaneously, and its theoretical and experimental justification will be given in Appendix A. From the method of counting the quadratic equations from exponential functions [NGG09], we can derive the conditions for a to yield quadratic equations as follows, where all arithmetic operations are done modulo $2^n - 1$.

– Case A: we have theoretical lower bound of $\deg(t_i)$.
  1. $\mathsf{hw}_n(a) \leq 2$.
  2. $\mathsf{hw}_n(a + 2^p) \leq 2$ for some $p \in \{0, \ldots, n-1\}$.

3. $\mathsf{hw}_n((2^k+1)a) \le 2$ for some $k \in \{1, \ldots, n/2\}$.

– Case B: we experimentally checked that the number of quadratic equations is always less than $3n$ assuming that a is not in Case A.

1. $2^r a = a + 2^p$ for some $r \in \{1, \ldots, n-1\}$ and $p \in \{0, \ldots, n-1\}$.
2. $2^r(a+2^p) = (2^k+1)a$ for some $r \in \{1, \ldots, n-1\}$, $k \in \{1, \ldots, n/2\}$ and $p \in \{0, \ldots, n-1\}$.

– Case C: we experimentally found that these cases do not contribute to algebraic cryptanalysis unless they simultaneously belong to other case(s)

1. $(2^m-1)a = 0$ for some $m \mid n$.
2. $(2^m-1)(2^k+1)a = 0$ for some $m \mid n$ and $k \in \{1, \ldots, n/2\}$.
3. $2^r a = a$ for some $r \in \{1, \ldots, n-1\}$.
4. $2^r a = (2^k+1)a$ for some $r \in \{1, \ldots, n-1\}$ and $k \in \{1, \ldots, n/2\}$.
5. $2^r(2^k+1)a = (2^{k'}+1)a$ for some $r \in \{1, \ldots, n-1\}$ and $k, k' \in \{1, \ldots, n/2\}$

– Case D: we theoretically and experimentally checked that the system either has a large degree $\deg(t_i)$ or generates a small number of quadratic equations.

1. $2^r(a+2^p) = (a+2^q)$ for some $r \in \{1, \ldots, n-1\}$ and $p, q \in \{0, \ldots, n-1\}$.
2. $(2^m-1)(a+2^p) = 0$ for some $m \mid n$ and $p \in \{0, \ldots, n-1\}$.

5.3 Other Attacks on AIM2

EFFICIENT EXHAUSTIVE SEARCH BY OPTIMIZED IMPLEMENTATION. The point of this attack is that the Mersenne S-boxes can be represented as $\mathsf{Mer}[e](x) = x^{2^e} \cdot (x^{-1})$, and x^{-1} can be efficiently iterated by an LFSR. In AIM2, the same attack is not applied because of the inverse Mersenne S-boxes. Even if an attacker iterates an intermediate state to use the same method, the attacker should evaluate at least one of Mersenne S-boxes and at least one of their inverses. For example, in AIM-I or AIM-III, one may find y by iterating y and y^{-1} such satisfies

$$\mathsf{Mer}[e_1](t_1) = x + c_1 \text{ where } \begin{cases} x := y^{2^{e_2}} \cdot y^{-1} + c_2, \\ t_* := \mathsf{Mer}[e_*]^{-1}(x + \mathsf{ct}), \\ t_1 := A_{\mathsf{iv},1}^{-1}(b_{\mathsf{iv}} + A_{\mathsf{iv},2}(y) + t_*). \end{cases}$$

We present number of field multiplications for all Mersenne Sboxes and their inverses, and the gate-count complexity AIM2 in Table 10. From this fact, we believe that this kind of attack cannot be applied to AIM2.

QUANTUM ATTACKS. For larger exponents, it will take slightly more time to compute the (inverse) Mersenne S-boxes. This leads to a slightly larger complexity of the Grover's algorithm. The complexities of quantum algebraic attacks will be changed not critically as new quadratic systems are found for AIM2. For `QuantumBooleanSolve` [FHK+17], the complexity becomes $O(2^{0.462 \cdot \ell n})$ since there are quadratic systems in ℓn Boolean variables for all the instances of AIM2. The complexity of GroverXL [BY18] is $2^{(1.1061+o(1))n}$ for AIM2-I, III and $2^{(1.3568+o(1))n}$ for AIM2-V. We remark that these attacks are not better than the Grover's algorithm.

STATISTICAL ATTACKS. As differential probability and linear probability of an S-box is the same as its inverse, most of the analysis on statistical attacks will remain unchanged except the weight of a correlation trail. Since e_1 becomes larger than $n/2$, the weight is lower bounded by $n - 2e_*$ (with the previous bound being $2(n - e_1 - e_*)$). We note that it does not imply that linear cryptanalysis is feasible since an adversary is not given a large enough number of plaintext-ciphertext pairs to mount this analysis.

Scheme	Circuit	#Operations			Total Cost
		FF Mult.	FF Square	Mat-Vec Mult.	
AIM2-I	Mer[3] / Mer$[3]^{-1}$	2 / 8	2 / 126	-	-
	Mer[49] / Mer$[49]^{-1}$	7 / 11	48 / 127	-	-
	Mer[91] / Mer$[91]^{-1}$	9 / 14	90 / 127	-	-
	AIM2(iv, $\cdot$)	14	256	2	147
AIM2-III	Mer[5] / Mer$[5]^{-1}$	3 / 9	4 / 190	-	
	Mer[17] / Mer$[17]^{-1}$	5 / 11	16 / 191	-	
	Mer[47] / Mer$[47]^{-1}$	8 / 11	46 / 191	-	
	AIM2(iv, $\cdot$)	15	386	2	212.3
AIM2-V	Mer[3] / Mer$[3]^{-1}$	2 / 10	2 / 255	-	-
	Mer[7] / Mer$[7]^{-1}$	4 / 11	6 / 255	-	-
	Mer[11] / Mer$[11]^{-1}$	5 / 10	10 / 255	-	-
	Mer[141] / Mer$[141]^{-1}$	10 / 10	140 / 253	-	-
	AIM2(iv, $\cdot$)	23	765	3	277.7

Table 10: The number of operations for each type of operation and the total cost of exhaustive search on AIM2 components. We have assumed that a single AIM2 evaluation requires a field multiplication, a power-of-two exponentiation, a Mersenne S-box evaluation, and $\ell - 1$ of inverse Mersenne S-box evaluations, with all the different e_i's. The total cost is the log of required number of binary gates.

5.4 Effect on Efficiency

The main feature of AIM is to fully utilize the repeated multipliers in BN++ when proving an AIM instance. Although the S-boxes on the first round are replaced by inverse Mersenne S-boxes, the structure of AIM2 still remains unchanged, so the signature size will be unchanged as well.

In AIMer, for every input share $[\![x]\!]$ of an S-box, the prover and the verifier should compute $[\![x]\!]^{2^e}$. For a larger exponent e, this computation takes more time. From our experiment, signing and verification of the AIMer with AIM2 is expected to cost more time by up to 11%. Table 11 shows the detailed benchmark results on signing time of AIMer and AIMer with AIM2.

References

[AFI+04] Gwénolé Ars, Jean-Charles Faugère, Hideki Imai, Mitsuru Kawazoe, and Makoto Sugita. Comparison Between XL and Gröbner Basis Algorithms. In Pil Joong Lee, editor, *Advances in Cryptology - ASIACRYPT 2004*, pages 338–353, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[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 $\mathbb{F}_2$. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010*, pages 203–218, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[BFP09] Luk Bettale, Jean-Charles Faugere, and Ludovic Perret. Hybrid Approach for Solving Multivariate Systems over Finite Fields. *Journal of Mathematical Cryptology*, 3(3):177–197, 2009.

[BFS04] Magali Bardet, Jean-Charles Faugere, and Bruno Salvy. On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations. In *Proceedings of the International Conference on Polynomial System Solving*, pages 71–74, 2004.

[BFSS13] Magali Bardet, Jean-Charles Faugère, Bruno Salvy, and Pierre-Jean Spaenlehauer. On the complexity of solving quadratic Boolean systems. *Journal of Complexity*, 29(1):53–75, 2013.

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

[BY18] Daniel J. Bernstein and Bo-Yin Yang. Asymptotically Faster Quantum Algorithms to Solve Multivariate Quadratic Equations. In *PQCrypto 2018*, pages 487–506. Springer, 2018.

Level	Parameter	Sign of AIMer (ms)	Sign of AIMer + AIM2 (ms)	ratio
L1	PARAM_1	0.59	0.59	1.00
	PARAM_2	1.36	1.38	1.01
	PARAM_3	4.42	4.64	1.05
	PARAM_4	22.29	23.92	1.07
L3	PARAM_1	1.36	1.42	1.03
	PARAM_2	3.48	3.64	1.05
	PARAM_3	11.01	11.57	1.05
	PARAM_4	53.38	57.87	1.08
L5	PARAM_1	2.45	2.69	1.10
	PARAM_2	6.29	7.01	1.11
	PARAM_3	19.56	21.68	1.11
	PARAM_4	95.65	105.36	1.10

Table 11: The performance comparison between AIMer and AIMer with AIM2. Both signing times were measured in Intel Xeon E5-1650 v3 @ 3.50 GHz with 128 GB RAM, TurboBoost and Hyper-threading disabled, and gcc 7.5.0 with -O3 option.

[CH17] Murat Cenk and M. Anwar Hasan. On the arithmetic complexity of Strassen-like matrix multiplications. *Journal of Symbolic Computation*, 80:484–501, 2017.

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

[DCK+20] Jintai Ding, Ming-Shing Chen, Matthias Kannwischer, Jacques Patarin, Albrecht Petzoldt, Dieter Schmidt, and Bo-Yin Yang. Rainbow. Technical report, National Institute of Standards and Technology, 2020, 2020. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

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

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

[Fau99] Jean-Charles Faugére. A new efficient algorithm for computing Gröbner bases (F4). *Journal of Pure and Applied Algebra*, 139(1):61–88, 1999.

[Fau02] Jean Charles Faugère. A New Efficient Algorithm for Computing Gröbner Bases without Reduction to Zero (F5). In *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation*, ISSAC '02, page 75–83, New York, NY, USA, 2002. Association for Computing Machinery.

[FHK+17] Jean-Charles Faugère, Kelsey Horan, Delaram Kahrobaei, Marc Kaplan, Elham Kashefi, and Ludovic Perret. Fast Quantum Algorithm for Solving Multivariate Quadratic Equations. Cryptology ePrint Archive, Paper 2017/1236, 2017. https://eprint.iacr.org/2017/1236.

[Frö85] Ralf Fröberg. An Inequality for Hilbert Series of Graded Algebras. *MATHEMATICA SCANDINAVICA*, 56, Dec. 1985.

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

[KCC+23] Seongkwang Kim, Jihoon Cho, Mingyu Cho, Jincheol Ha, Jihoon Kwon, Byeonghak Lee, Joohee Lee, Jooyoung Lee, Sangyub Lee, Dukjae Moon, Mincheol Son, and Hyojin Yoon. AIMer. *Submission to the NIST's Standardization of Additional Digital Signature Schemes*, 2023. https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures.

[KHS+23] 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. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, CCS '23, page 401–415, New York, NY, USA, 2023. Association for Computing Machinery.

[Knu97] Donald E. Knuth. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Addison-Wesley, Boston, third edition, 1997.

[KS99] Aviad Kipnis and Adi Shamir. Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization. In *CRYPTO '99*, pages 19–30. Springer, 1999.

[KZ22] Daniel Kales and Greg Zaverucha. Efficient Lifting for Shorter Zero-Knowledge Proofs and Post-Quantum Signatures. Cryptology ePrint Archive, Paper 2022/588, 2022. https://eprint.iacr.org/2022/588.

[LMOM23] Fukang Liu, Mohammad Mahzoun, Morten Øygarden, and Willi Meier. Algebraic Attacks on RAIN and AIM Using Equivalent Representations. *IACR Transactions on Symmetric Cryptology*, 2023(4):166–186, Dec. 2023.

[NGG09] Yassir Nawaz, Kishan Chand Gupta, and Guang Gong. Algebraic Immunity of S-Boxes Based on Power Mappings: Analysis and Construction. *IEEE Transactions on Information Theory*, 55(9):4263–4273, 2009.

[SS21] Jan Ferdinand Sauer and Alan Szepieneic. SoK: Gröbner Basis Algorithms for Arithmetization Oriented Ciphers. Cryptology ePrint Archive, Paper 2021/870, 2021. https://eprint.iacr.org/2021/870.

[Wie86] D. Wiedemann. Solving sparse linear equations over finite fields. *IEEE Transactions on Information Theory*, 32(1):54–62, 1986.

[YC04] Bo-Yin Yang and Jiun-Ming Chen. Theoretical Analysis of XL over Small Fields. In Huaxiong Wang, Josef Pieprzyk, and Vijay Varadharajan, editors, *Information Security and Privacy*, pages 277–288, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[YCBC07] Bo-Yin Yang, Owen Chia-Hsin Chen, Daniel J. Bernstein, and Jiun-Ming Chen. Analysis of QUAD. In Alex Biryukov, editor, *Fast Software Encryption*, pages 290–308, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[ZWY+23] Kaiyi Zhang, Qingju Wang, Yu Yu, Chun Guo, and Hongrui Cui. Algebraic Attacks on Round-Reduced Rain and Full AIM-III. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology – ASIACRYPT 2023*, pages 285–310, Singapore, 2023. Springer Nature Singapore.

A More Details on New Variables Other Than S-Box Outputs

A.1 How to Count the Number of Quadratic Equation

Before the main analysis, we briefly introduce how to enumerate the number of quadratic equations which was introduced in [NGG09]. Suppose we have a power function $y = x^a$ in $\mathbb{F}_{2^n}$. Arithmetic operations on the exponent is done in $\mathbb{Z}_{2^n-1}$. In this section, arithmetic operations involving exponents are done in modulo $2^n - 1$. In $\mathbb{Z}_{2^n-1}$, multiplying by 2 is equivalent to bit-wise circular left shift. As x^{2^i} in characteristic-2 fields is linear over $\mathbb{F}_2$, x^a and $x^{2^i a}$ are equivalent up to linear mapping. If $(2^m - 1)a = 0$ for some $m|n$, we call a to be *m-cyclic*.

Discarding equivalent exponents ($a \sim 2^i a$), quadratic equations between x and y are basically generated from three types of monomial: y, yx^{2^p}, and y^{2^k+1}. If those monomials can be represented only in x variables (e.g., $yx^{2^p} = x^{a+2^p}$) and has degree less than or equal to 2 (which is represented by Hamming weight of the exponent), the system has following quadratic equations.

1. $\mathsf{hw}_n(a) \le 2$: $y = x^a$
2. $\mathsf{hw}_n(a + 2^p) \le 2$ for some $p \in \{0, \dots, n-1\}$: $x^{2^p} y = x^{a+2^p}$
3. $\mathsf{hw}_n((2^k + 1)a) \le 2$ for some $k \in \{1, \dots, n/2\}$: $y^{2^k+1} = x^{(2^k+1)a}$

We note that the domain of k is less than or equal to $n/2$ in the third case since $(2^k + 1)a$ and $(2^{n-k} + 1)a$ are in the same coset (under the linear equivalence). Sometimes, it generates quadratic equations if at least two of a, $a + 2^p$, and $(2^p + 1)a$ are in the same coset as follows.

1. $2^r a = a$ for some $r \in \{1, \dots, n-1\}$: $y^{2^r} = y$
2. $2^r a = a + 2^p$ for some $r \in \{1, \dots, n-1\}$ and $p \in \{0, \dots, n-1\}$: $y^{2^r} = yx^{2^p}$
3. $2^r a = (2^k + 1)a$ for some $r \in \{1, \dots, n-1\}$ and $k \in \{1, \dots, n/2\}$: $y^{2^r} = y^{2^k+1}$
4. $2^r(a + 2^p) = a + 2^q$ for some $r \in \{1, \dots, n-1\}$ and $p, q \in \{0, \dots, n-1\}$: $(yx^{2^p})^{2^r} = yx^{2^q}$
5. $2^r(a + 2^p) = (2^k + 1)a$ for some $r \in \{1, \dots, n-1\}$, $k \in \{1, \dots, n/2\}$ and $p \in \{0, \dots, n-1\}$: $(yx^{2^p})^{2^r} = y^{2^k+1}$
6. $2^r(2^k + 1)a = (2^{k'} + 1)a$ for some $r \in \{1, \dots, n-1\}$ and $k, k' \in \{1, \dots, n/2\}$: $(y^{2^k+1})^{2^r} = y^{2^{k'}+1}$

If one of a, $a+2^p$, and $(2^p+1)a$ is m-cyclic, quadratic equations between y itself can be generated as follows.

1. $(2^m - 1)a = 0$ for some $m \mid n$: $y^{2^m} = y$
2. $(2^m - 1)(a + 2^p) = 0$ for some $m \mid n$ and $p \in \{0, \dots, n-1\}$: $(yx^{2^p})^{2^m} = yx^{2^p}$
3. $(2^m - 1)(2^k + 1)a = 0$ for some $m \mid n$ and $k \in \{1, \dots, n/2\}$: $(y^{2^k+1})^{2^m} = y^{2^k+1}$

A.2 Detailed Analysis of AIM2

In this section, we provide detailed analysis of setting new variables other than S-box outputs which was described in Section 4.3. For each S-boxes, we either lower bound of $\deg(t_i)$ or upper bound the number of quadratic equation by $3n$. We believe that upper bounding the number of quadratic equations by $3n$ is sufficient to prevent unknown attack since (inverse) Mersenne S-boxes already generates $3n$ quadratic equations. Setting a new variable other than S-box outputs which generates less than or equal to $3n$ quadratic equations seems to have no benefit compared to setting S-box outputs to be new variables.

Recall that we categorized the exponent a for the new variable $w_i = (\mathsf{pt} + c_i)^a$ as follows. The following categorization is different from above, and it is categorized by how we handled the case.

- Case A: we have theoretical lower bound of $\deg(t_i)$.
  1. $\mathsf{hw}_n(a) \le 2$.
  2. $\mathsf{hw}_n(a + 2^p) \le 2$ for some $p \in \{0, \dots, n-1\}$.
  3. $\mathsf{hw}_n((2^k + 1)a) \le 2$ for some $k \in \{1, \dots, n/2\}$.
- Case B: we experimentally checked that the number of quadratic equations is always less than $3n$ assuming that a is not in Case A.

1. $2^r a = a + 2^p$ for some $r \in \{1, \ldots, n-1\}$ and $p \in \{0, \ldots, n-1\}$.
2. $2^r(a + 2^p) = (2^k + 1)a$ for some $r \in \{1, \ldots, n-1\}$, $k \in \{1, \ldots, n/2\}$ and $p \in \{0, \ldots, n-1\}$.

– Case C: the quadratic equations in these cases consist of only y-variables. We found that these cases do not contribute to algebraic cryptanalysis unless they simultaneously belong to other case(s).
  1. $(2^m - 1)a = 0$ for some $m \mid n$.
  2. $(2^m - 1)(2^k + 1)a = 0$ for some $m \mid n$ and $k \in \{1, \ldots, n/2\}$.
  3. $2^r a = a$ for some $r \in \{1, \ldots, n-1\}$.
  4. $2^r a = (2^k + 1)a$ for some $r \in \{1, \ldots, n-1\}$ and $k \in \{1, \ldots, n/2\}$.
  5. $2^r(2^k + 1)a = (2^{k'} + 1)a$ for some $r \in \{1, \ldots, n-1\}$ and $k, k' \in \{1, \ldots, n/2\}$

– Case D: we theoretically checked that the system either has a large degree $\deg(t_i)$ or generates a small number of quadratic equations, under experimentally verified assumptions.
  1. $2^r(a + 2^p) = (a + 2^q)$ for some $r \in \{1, \ldots, n-1\}$ and $p, q \in \{0, \ldots, n-1\}$.
  2. $(2^m - 1)(a + 2^p) = 0$ for some $m \mid n$ and $p \in \{0, \ldots, n-1\}$.

In the following, we give analysis for each case. As our analyses given below consider a single S-box case, we use simpler notation without constant addition as follows.

– x: the input of S-box
– y: the new variable $y = x^a$
– t: the output of S-box, $t = x^{\bar{e}}$, $\bar{e} = (2^e - 1)^{-1} \bmod (2^n - 1)$ for some $e \in \{e_1, \ldots, e_\ell, e_*\}$

CASE A. **(A-1 and A-2)** We want to show that t should be of at least certain degree with respect to x and y when a is one of the following types:

– $a = -1$;
– $a = 2^p + 1$, where $p \in \{2, ..., n-1\}$;
– $a = 2^p - 1$, where $p \in \{2, ..., n-1\}$;
– $a = 2^p + 2^q - 1$, where $p, q \in \{2, ..., n-1\}$, $p \neq q$;

Define

$$D_{\min,a} := \min_u \left\{ \mathsf{hw}_n(u) + \mathsf{hw}_n(\bar{e} - a \cdot u) \right.$$

and

$$D_{\min} := \min_a \{D_{\min,a}\}.$$

$D_{\min}$ is the lower bound of the degree of t with respect to x and y by

$$t = y^u \cdot x^{\bar{e} - a \cdot u}.$$

At first, suppose $a = 2^p + 2^q - 1$ for some $p, q \in \{2, ..., n-1\}$ where $p \neq q$. By the definition, we have

$$D_{\min,2^p+2^q-1} = \min_u \left\{ \mathsf{hw}_n(u) + \mathsf{hw}_n(\bar{e} - (2^p + 2^q - 1) \cdot u) \right. .$$

By using the fact $\mathsf{hw}_n(x) + \mathsf{hw}_n(y) \geq \mathsf{hw}_n(x + y)$, we have

$$\begin{aligned} &2 \cdot \mathsf{hw}_n(u) + \mathsf{hw}_n(\bar{e} - (2^p + 2^q - 1) \cdot u) \\ &= \mathsf{hw}_n(2^p \cdot u) + \mathsf{hw}_n(2^q \cdot u) + \mathsf{hw}_n(\bar{e} - (2^p + 2^q - 1) \cdot u) \\ &\geq \mathsf{hw}_n(\bar{e} + u), \end{aligned}$$

and it implies that

$$D_{\min,2^p+2^q-1} \geq \min_u \left\{ \max \left\{ \mathsf{hw}_n(u), \mathsf{hw}_n(\bar{e} + u) - \mathsf{hw}_n(u) \right. \right\}. \tag{11}$$

Now we want to lower bound $\mathsf{hw}_n(\bar{e}+u)$ for arbitrary u. For an integer j, define

$$\mathsf{NumSeg}_n(j) := \left\{ i \in \{0,...,n-1\} : 2 \mid (2^i \cdot j \bmod (2^n-1)), 4 \nmid (2^i \cdot j \bmod (2^n-1)) \right.$$

which counts the number of connected "1" segments in the n-bit binary representation of j allowing bitwise rotation. Then, for an integer j and $h \in \{0,\ldots,n-1\}$,

$$\mathsf{NumSeg}_n(j+2^h) \geq \mathsf{NumSeg}_n(j) - 1,$$

so we get

$$\mathsf{hw}_n(\bar{e}+u) \geq \mathsf{NumSeg}_n(\bar{e}+u) \geq \mathsf{NumSeg}_n(\bar{e}) - \mathsf{hw}_n(u).$$

Together with (11), we have

$$D_{\min,2^p+2^q-1} \geq \min_u \left\{ \max\left\{\mathsf{hw}_n(u), \mathsf{NumSeg}_n(\bar{e}) - 2\cdot\mathsf{hw}_n(u) \right.\right\} \geq \lceil \mathsf{NumSeg}_n(\bar{e})/3 \rceil.$$

Similarly, we have

$$\begin{aligned}
D_{\min,2^p-1} &\geq \min_u \left\{ \max\left\{\mathsf{hw}_n(u), \mathsf{hw}_n(\bar{e}+u) \right.\right\} \geq \lceil \mathsf{NumSeg}_n(\bar{e})/2 \rceil,\\
D_{\min,2^p+1} &\geq \min_u \left\{ \max\left\{\mathsf{hw}_n(u), \mathsf{hw}_n(\bar{e}) - \mathsf{hw}_n(u) \right.\right\} \geq \lceil \mathsf{hw}_n(\bar{e})/2 \rceil,\\
D_{\min,-1} &\geq \min_u \left\{\mathsf{hw}_n(u) + \mathsf{hw}_n(\bar{e}+u)\right\} \geq \lceil \mathsf{NumSeg}_n(\bar{e}) \rceil,
\end{aligned}$$

and overall, we get following lower bound:

$$D_{\min} \geq \lceil \mathsf{NumSeg}_n(\bar{e})/3 \rceil. \tag{12}$$

(A-3) Suppose $\mathsf{hw}_n((2^k+1)a) = 2^p + 2^q$ for some $p, q \in \{0,\ldots,n-1\}$, $p \neq q$. Then

$$\begin{aligned}
D_{\min,a} &= \min\{\mathsf{hw}_n(u) + \mathsf{hw}_n(v) : \bar{e} = au + v\}\\
&= \min\{\mathsf{hw}_n(u) + \mathsf{hw}_n(v) : (2^k+1)\bar{e} = (2^p+2^q)u + (2^k+1)v\}\\
&= \min\left\{\frac{1}{2}(\mathsf{hw}_n(2^p u) + \mathsf{hw}_n(2^q u) + \mathsf{hw}_n(2^k v) + \mathsf{hw}_n(v)) : (2^k+1)\bar{e} = (2^p+2^q)u + (2^k+1)v\right\}\\
&\geq \min\left\{\frac{\mathsf{hw}_n((2^k+1)\bar{e})}{2}\right\}
\end{aligned}$$

Therefore,

$$D_{\min} \geq \min_k \{\mathsf{hw}_n((2^k+1)\bar{e})\}/2 \tag{13}$$

Case B. If a is in Case B, there exists either

- $r \in \{1,\ldots,n-1\}$ such that $\gcd(2^r-1, 2^n-1) = 1$ and $\mathsf{hw}_n((2^r-1)a) = 1$ or
- $r, s \in \{1,\ldots,n-1\}$ such that $\gcd(2^r+2^s-1, 2^n-1) = 1$ and $\mathsf{hw}_n((2^r+2^s-1)a) = 1$.

Then, we can count the number of equations for each r or (r,s), while check a is in Case A. As a result, at least for $n \in \{128, 192, 256\}$, the corresponding a all belong to Case A or produce $3n$ or fewer quadratic equations.

Case C. The quadratic equation from a in Case C consists of only y-variables. For example, if a satisfies $2^r(2^k+1)a = (2^{k'}+1)a$, then we get

$$y^{2^{k+r}+2^r} = y^{2^{k'}+1}.$$

This kind of equations cannot contribute to solve the whole system since it only reduces the number of candidates of y, not x. Therefore, we ignored this case.

CASE D. Although the exponent a is in Case D but not in Case A and B, we experimentally checked that the system from $y = x^a$ has large $\deg(t)$ or $3n$ or fewer quadratic equations. Recall that the system with $y = x^a$ is equivalent (up to linear mapping) to the system with $y = x^{2^i a}$ for some i.

(D-1) Let $(2^r - 1)a = 2^p - 1$ for some $r \in \{1, \ldots, n-1\}$ and $p \in \{0, \ldots, n-1\}$. Since $r = 1$ or $r = n-1$ or $p \in \{0, 1\}$ implies that this type of a is covered in Case A or B, let $1 < r < n-1$ and $p > 1$. For a to exist, it should be $\gcd(r, n) \mid p$.

- Suppose a also satisfies $(2^m - 1)a = 0$ for some $m > 0$. Then, $(2^m - 1)(2^r - 1)a = (2^p - 1)(2^m - 1) = 0$, which is a contradiction. Therefore, $y = x^a$ does not imply quadratic equations from the condition $(2^m - 1)a = 0$.
- Suppose a also satisfies $(2^m - 1)(2^k + 1)a = 0$ for some $m \mid n$ and $k \in \{1, \ldots, n-1\}$ and $(2^m - 1)(2^k + 1) \neq 0$. Then, $(2^m - 1)(2^k + 1)(2^r - 1)a = (2^k + 1)(2^p - 1)(2^m - 1) = 0$, which implies $k = p = n/2$. Therefore, we have $(2^m - 1)(2^{n/2} + 1)a = 0$ and it means that we get at most n more equations from
$$y^{2^{n/2}+1} = y^{2^{n/2+m}+2^m}. \tag{14}$$
- Suppose a also satisfies $(2^k - 2^s + 1)a = 0$ for some k, s such that $(2^k - 2^s + 1) \neq 0$. Then, $(2^k - 2^s + 1)(2^r - 1)a = (2^k - 2^s + 1)(2^p - 1) = 0$, which implies $p = n/2$ and $(k, s) = (n/2 + 1, n/2)$ or $(k, s) = (n/2 - 1, n - 1)$. Since the case satisfying $(2^{n/2} + 1)a = 0$ and $(2^{n/2-1} + 2^n - 2^{n-1})a = 0$ are covered in Case A, a cannot satisfy such condition without satisfying Case A.
- Suppose a also satisfies $(2^q + 2^k - 2^s - 1)a = 0$ for some q, k, s such that $(2^q + 2^k - 2^s - 1) \neq 0$ and $q < k$. Then, $(2^q + 2^k - 2^s - 1)(2^r - 1)a = (2^q + 2^k - 2^s - 1)(2^p - 1) = 0$, which implies one of the following.
  - $q = 1, k = n/2 - 1, s = n - 1, p = n/2$. Since $2 + 2^{n/2-1} - 2^{n-1} - 1 = 2^{n-1} + 2^{n/2-1}$, this case is covered in Case A.
  - $q = 2, k = n/2, s = 1, p = n/2$. Since $4 + 2^{n/2} - 2 - 1 = 2^{n/2} + 1$, this case is covered in Case A.
  - $k = q + n/2, s = n/2, p = n/2$. In other word, $(2^q + 2^{q+n/2} - 2^{n/2} - 1)a = (2^q - 1)(2^{n/2} + 1)a = 0$. Since $\gcd(2^q - 1, 2^n - 1) = 2^{\gcd(q,n)} - 1$, one can get at most n equations same as in (14).
- Suppose a also satisfies $2^m(2^s - 1)a = 2^q - 1$ for some m, s, q. Then, $2^m(2^s - 1)(2^r - 1)a = (2^r - 1)(2^q - 1) = 2^m(2^s - 1)(2^p - 1)$, which implies one of the following.
  - $m = 0, p = r, q = s$. It means that $(2^r - 1)a = 2^r - 1$ or equivalently, $(2^m - 1)a = (2^m - 1)$ for $m = \gcd(n, r)$. Then, we get $\frac{n-m}{2m} \cdot n$ equations from
$$y^{im}x = yx^{im}, \text{ for } i = 1, \ldots, \left\lfloor \frac{n}{2m} \right\rfloor.$$
  - $m = 0, p = s, q = r$. It means that $(2^r - 1)a = 2^s - 1$ and $(2^s - 1)a = 2^r - 1$, and it only holds when $r - s$ which become exactly same condition in above.
  - $r = n/2 \pm 1, p = \pm 2$. It means that
$$a = (2^{n/2 \pm 1} - 1)^{-1}(2^{\pm 2} - 1) = 2^{n/2 \pm 1} + 1,$$
and such a is covered by Case A.
  - $r = n/2 \pm 1, p = n/2$. It means that
$$(2^{n/2} + 1)a = (2^{n/2} + 1)(2^{n/2 \pm 1})^{-1}(2^{n/2} - 1) = 0,$$
and such a is covered by Case A.
  - $s = n/2 \pm 1, q = \pm 2$. It means that
$$a = 2^{-m}(2^{n/2 \pm 1} - 1)^{-1}(2^{\pm 2} - 1) = 2^{-m}(2^{n/2 \pm 1} + 1),$$
and such a is covered by Case A.
  - $s = n/2 \pm 1, q = n/2$. It means that
$$2^m(2^{n/2} + 1)a = (2^{n/2} + 1)(2^{n/2 \pm 1})^{-1}(2^{n/2} - 1) = 0,$$
and such a is covered by Case A.

- $r = n/2 \pm 1, p = \mp 2$, or $s = n/2 \pm 1, q = \mp 2$. We counted the number of all quadratic equations for each a of this form and experimentally checked that $y = x^a$ implies $1.5n$ equations for all $n \in \{128, 192, 256\}$.

In summary, if $(2^r - 1)a = 2^p - 1$ for some $1 < r < n - 1$ and $p > 1$, one of the following events happen:

– if a is also in Case A, we have theoretic lower bound of $\deg(t)$;
– if a is also in Case B but not in Case A, we experimentally checked that the number of quadratic equations is always less than $3n$;
– if $p = r$ and $\gcd(r, n) = m < n/2$, $y = x^a$ produces $\frac{n-m}{2m} \cdot n$ equations which implies that less than or equal to $3n$ quadratic equations are generated when $m \geq n/7$;
– if $p = r = n/2$, this case generates $1.5n$ equations;
– if $p = n/2$ and r does not satisfy above conditions, one has at most $2n$ equations;
– otherwise, one has n equations.

Therefore, to have more than $3n$ equations, a should satisfy $(2^m - 1)a = 2^m - 1$ where $m \mid n$ and $m < n/7$. Let $\bar{e} = au + v$. Then,

$$(2^m - 1)\bar{e} = (2^m - 1)(u + v) \Rightarrow u + v = \bar{e} + \frac{2^n - 1}{2^m - 1} \cdot b$$

for some $0 \leq b \leq 2^m - 1$. Therefore

$$D_{\min} \geq \mathsf{hw}_n(u) + \mathsf{hw}_n(v) \geq \mathsf{hw}_n(u + v) \geq \min_b \left\{ \mathsf{hw}_n \left(\bar{e} + \frac{2^n - 1}{2^m - 1} \cdot b \right) \right\} \tag{15}$$

(D-2) Let $(2^m - 1)(a + 2^p) = 0$ for some $m \mid n$ and $p \in \{0, \ldots, n - 1\}$, and let $\bar{e} = au + v$ for some u, v. Discarding equivalent exponents, let $p = 0$. In this case, $a + 1$ is m-cyclic, which implies the binary representation of $a + 1$ is the concatenation of n/m number of a length-m string.

We divide this case into three subcases: $2 \leq m \leq n/4$, $m = n/3$, and $m = n/2$. For the latter two cases, we utilize the brute-force result of toy examples since the number of candidates of a is too many. Let $a = \frac{2^n - 1}{2^m - 1} \cdot b - 1$ for some $0 \leq b < 2^m - 1$. In toy examples ($n = 16, 24, 32, 48$), we found that the number of quadratic equations from $y = x^a$ is no more than n unless $\mathsf{hw}_n(b) = 1$ by brute-force searching b. If $\mathsf{hw}_n(b) = 1$, then

$$\mathsf{hw}_n(a + 1) = \mathsf{hw}_n \left(\frac{2^n - 1}{2^m - 1} \cdot b \right) = \frac{n}{m} \cdot \mathsf{hw}_n(b) = \frac{n}{m}.$$

We will use this fact to bound the degree of t when $m = n/2$ or $n/3$.

– Suppose that $2 \leq m \leq n/4$. We will show that the number of quadratic equations are less than $3n$.
  - Since a cannot be cyclic, there is no $m'|n$ with $(2^{m'} - 1)a = 0$. Similarly, there is no $k \in \{1, \ldots, n/2\}$ or $p \in \{1, \ldots, n - 1\}$ with $(2^{m'} - 1)(2^k + 1)a = 0$ or $(2^{m'} - 1)(a + 2^p) = 0$.
  - Suppose $2^r a = a$ for some $1 \leq r < n$. It implies that

$$\begin{aligned} & 2^r \cdot \frac{2^n - 1}{2^m - 1} \cdot b - 2^r = \frac{2^n - 1}{2^m - 1} \cdot b - 1 \\ \iff \quad & \frac{2^n - 1}{2^m - 1} \cdot b(2^r - 1) = 2^r - 1 \end{aligned}$$

    where $2^r - 1$ cannot be cyclic for $1 < r < n$.
  - Suppose $2^r a = (2^k + 1)a$ for some $1 \leq r < n$ and $1 \leq k \leq n/2$. It means that

$$\frac{2^n - 1}{2^m - 1} \cdot b(2^k - 2^r - 1) = 2^k - 2^r - 1.$$

    Since $2^k - 2^r - 1$ is nonzero and cannot be m-cyclic for $m \leq n/4$, this condition does not generate any quadratic equation.

- Suppose $2^r(2^k+1)a = (2^{k'}+1)a$ for some $1 \le r < n$ and $1 \le k, k' \le n/2$. It means that

$$\frac{2^n-1}{2^m-1} \cdot b(2^r(2^k+1) - (2^{k'}+1)) = 2^r(2^k+1) - (2^{k'}+1).$$

  Let $2^r(2^k+1) = 2^{i_1} + 2^{i_2}$ with $i_1 > i_2$. We will check the form of $(2^{i_1}+2^{i_2}) - (2^{k'}+1)$ by dividing into four cases.
  * If $i_1 > i_2 > k$, then $(2^{i_1}+2^{i_2}) - (2^{k'}+1)$ is nonzero and could be $n/3$-cyclic but not lower.
  * If $i_1 \ge k \ge i_2$, then it could be $n/2$-cyclic but not lower. It cannot be zero since $0 \le k \ne k' \le n/2$.
  * If $k > i_1 > i_2$ and $i_2 > 0$, then it is nonzero and could be $n/3$-cyclic but not lower.
  * If $k > i_1 > i_2$ and $i_2 = 0$, then it is nonzero and acyclic.

  Since $\frac{2^n-1}{2^m-1} \cdot b(2^r(2^k+1) - (2^{k'}+1))$ is m-cyclic where $m \le n/4$, this condition does not produce any quadratic equation.
- Suppose $2^r(a+2^p) = a + 2^q$ for some $1 \le r < n$ and $0 \le p \ne q < n$. It means that

$$\frac{2^n-1}{2^m-1} \cdot b(1-2^r) = 2^r(2^p-1) - (2^q-1).$$

  Without loss of generality, we can assume that $p > q$. Up to circular shift, we can rewrite $2^r(2^p-1) - (2^q-1)$ by $2^{r_1}(2^p-1) - 2^{r_2}(2^q-1)$ where $0 < r_2 + q \le r_1 + p < n-1$ or $r_1 + p = n-1$. We will check the form of $2^{r_1}(2^p-1) - 2^{r_2}(2^q-1)$ by dividing into four cases.
  * If $0 < r_2 + q = r_1 + p < n-1$, then it is acyclic.
  * If $0 < r_2 + q < r_1 + p < n-1$, then it is nonzero and could be $n/2$-cyclic if $r_1 \le r_2$ or $n/3$-cyclic if $r_1 > r_2$ but not lower.
  * If $r_1 + p = n-1$ and $r_2 + q \le n-1$, then it is nonzero and is acyclic if $r_2 + q = n-1$ or could be $n/2$-cyclic if $r_1 \le r_2$ or $n/3$-cyclic if $r_1 > r_2$ but not lower.
  * Suppose $r_2 + q \ge n$. Let $2^{r_2}(2^q-1) = 2^{n-q_1}(2^{q_1}-1) + (2^{q_2}-1)$ where $q_1 + q_2 = q$, $1 \le q_1 \le q < p$, and $n - q_1 > q_2 \ge 1$. Then, it is nonzero and could be $n/3$-cyclic but not lower.

  Since $\frac{2^n-1}{2^m-1} \cdot b(1-2^r)$ is m-cyclic where $m \le n/4$, this condition does not produce any quadratic equation.

– Suppose that $m = n/3$ (and $n = 192$). We found that only $b = 1$ and $b = 2^{m-1}$ induce more than $3n$ quadratic equations, provided that $\mathsf{hw}_n(b) = 1$.
  - For $b = 1$, $a = 2^{2m} + 2^m$. Let $\bar{e} = ua + v$. Then

$$2 \cdot \mathsf{hw}_n(u) + \mathsf{hw}_n(v) = \mathsf{hw}_n(2^{2m}u) + \mathsf{hw}_n(2^m u) + \mathsf{hw}_n(\bar{e} - ua) \ge \mathsf{hw}_n(\bar{e}),$$

    so that

$$\mathsf{hw}_n(u) + \mathsf{hw}_n(v) \ge \max\{\mathsf{hw}_n(\bar{e}) - \mathsf{hw}_n(u), \mathsf{hw}_n(u)\} \ge \lceil \mathsf{hw}_n(\bar{e})/2 \rceil.$$

  - For $b = 2^{m-1}$, $a^{-1} = 2^{2m} + 2^m$. Let $\bar{e} = ua + v$, then $u = a^{-1}\bar{e} - a^{-1}v$. Similarly, we have

$$\mathsf{hw}_n(u) + 2 \cdot \mathsf{hw}_n(v) = \mathsf{hw}_n(a^{-1}\bar{e} - a^{-1}v) + \mathsf{hw}_n(2^{2m}v) + \mathsf{hw}_n(2^m v) \ge \mathsf{hw}_n(a^{-1}\bar{e}) = \mathsf{hw}_n((2^{2m} + 2^m)\,\bar{e})$$

    so that

$$\mathsf{hw}_n(u) + \mathsf{hw}_n(v) \ge \max\left\{\mathsf{hw}_n((2^{2m} + 2^m)\,\bar{e}) - \mathsf{hw}_n(v), \mathsf{hw}_n(v)\right. \ge \lceil \mathsf{hw}_n((2^{2m} + 2^m)\,\bar{e})/2 \rceil.$$

– Suppose that $m = n/2$. Let $\bar{e} = ua + v$. We will lower bound $\mathsf{hw}_n(u) + \mathsf{hw}_n(v)$. Then,

$$\begin{aligned} \mathsf{hw}_n(v) &= \mathsf{hw}_n(\bar{e} - ua) = \mathsf{hw}_n(\bar{e} + u - u(a+1)) \\ &\ge \mathsf{hw}_n(\bar{e} + u) - \mathsf{hw}_n(u(a+1)) \\ &\ge \mathsf{NumSeg}(\bar{e}) - \mathsf{hw}_n(u) - \mathsf{hw}_n(a+1)\mathsf{hw}_n(u). \end{aligned}$$

  Therefore,

$$\begin{aligned} \mathsf{hw}_n(u) + \mathsf{hw}_n(v) &\ge \max\{\mathsf{NumSeg}(\bar{e}) - \mathsf{hw}_n(a+1)\mathsf{hw}_n(u), \mathsf{hw}_n(u)\} \\ &\ge \lceil \mathsf{NumSeg}(\bar{e})/(\mathsf{hw}_n(a+1)+1) \rceil = \lceil \mathsf{NumSeg}(\bar{e})/3 \rceil. \end{aligned}$$

In summary, if $(2^m-1)(a+1)=0$ for some $m \mid n$, (or equivalently, if $a=\frac{2^n-1}{2^m-1}\cdot b-1$ for some $0 \le b < 2^m-1$) we have one of the following:

- if $\mathsf{hw}_n(b) \neq 1$, it is believed that there are fewer than n equations (and it is checked in toy parameters).
- if a is also in Case A, we have theoretic lower bound of $\deg(t)$;
- if a is also in Case B but not in Case A, we experimentally checked that the number of quadratic equations is always less than $3n$;
- if $2 \le m \le n/4$, it produces at most $n-m$ equations;
- if $n=192, m=n/3$,
$$D_{\min} \geq \min\left\{\lceil \mathsf{hw}_n(\bar{e})/2\rceil, \lceil \mathsf{hw}_n((2^{2m}+2^m)\,\bar{e})/2\rceil \right. ; \tag{16}$$
- if $m=n/2$,
$$D_{\min} \geq \lceil \mathsf{NumSeg}(\bar{e})/3\rceil. \tag{17}$$

LOWER BOUNDS OF THE DEGREES OF THE INDUCED SYSTEM. Since the largest degree reaching while running a Gröbner basis computation algorithm or the XL algorithm (also known as *solving degree* [DS13]) should be larger than or equal to the degree of the system, we can lower bound the security of AIM2 against Liu's attack. Table 12 summarizes the lower bound of time complexity (from (5)) of Case A and D and the bound of $D_{\min}$ for each exponents (from (12), (13), (15), (16), and (17)). We only considered the case of replacing some variables in S_{quad}, since otherwise we would get a system with a lot higher degree.

Scheme	$(e_1, D_{\min})$	$(e_2, D_{\min})$	$(e_3, D_{\min})$	$(e_*, D_{\min})$	Complexity		
					k	sd	Time (bits)
AIM2-I	$(49,16)$	$(91,15)$	-	$(3,15)$	0	≥ 15	176.2
AIM2-III	$(17,17)$	$(47,17)$	-	$(5,26)$	0	≥ 17	214.4
AIM2-V	$(11,31)$	$(141,23)$	$(7,25)$	$(3,29)$	0	≥ 23	310.4

Table 12: Lower bounds of the degrees of the system for Case A and D. $(e_i, D_{\min}) = (e, d)$ means that there is no such f with $\deg(f) < d$ where $t_i = \mathsf{Mer}[e_i]^{-1}(\mathsf{pt}) = f(\mathsf{pt}, w_i)$ and $w_i = (\mathsf{pt}+c_i)^a$ for some integer a, while there exists degree 2 polynomial $g(\mathsf{pt}, w) = 0$. All the complexities are measured by (5). k is the number of guessed bits and sd is the solving degree, which is larger than at least one of $D_{\min}$.