The ISC Int'l Journal of Information Security

ISeCure

October 2025, Volume 17, Number 3 (pp. 37–41)

http://www.isecure-journal.org

Presented at the ISCISC'2025 in TEHRAN, Iran.

Fast Exhaustive Search on AIM2 **

Arka Debnath [1,*], and Mohammad Mahzoun [2]

[1] *KU Leuven, Belgium*

[2] *3MI Labs*

ARTICLE INFO.

Keywords:
AIM2-Cryptanalysis-Fast Exhaustive Search

Type:

doi:

Abstract

This paper describes a fast exhaustive search preimage attack on AIM2, an improved version of the one-way function AIM, proposed to address algebraic vulnerabilities found in its predecessor. Our attack transforms the polynomial system describing AIM2 over $\mathbb{F}_{2^\lambda}$ to a boolean polynomial system over $\mathbb{F}_2$, allowing for an exhaustive search by guessing input bits and solving a resulting linear system. Solving the whole system is not necessary for most incorrect guesses, and use of Gray code helps optimizing the iteration over all possible guesses. Our results show that the complexity of exhaustive search on AIM2, especially AIM2-I and AIM2-III is lower than previously estimated, though still higher than that of AES.

1 Introduction

The MPC-in-the-Head (MPCitH) paradigm, introduced by Ishai *et al.* [1], offers a unique framework for constructing zero-knowledge proofs (ZKPs) by simulating multiparty computation (MPC) protocols. Recently, this approach has been utilised to develop post-quantum signature schemes whose security primarily depends on the one-way function used during key generation.

AIM, proposed by Kim *et al.* [2] is an example of such one-way functions optimised for this paradigm. AIMer, introduced in the same paper, is a signature scheme which integrates the BN++ proof system [3] with the symmetric primitive AIM. AIM is notable for its parallel structure and the use of Mersenne S-boxes, designed to exploit multiplier efficiency while maintaining robustness against algebraic attacks. However, subsequent analyses uncovered algebraic vulnerabilities in AIM [4, 5].

As an improvement [6] proposed AIM2, a new version of AIM. The key differences between AIM2 and AIM are as follows:

1. The S-box in the first round is reversed, making it significantly more challenging to generate a large set of equations compared to AIM.

2. Addition of unique constants to the inputs of the first-round S-boxes makes the inputs to the S-boxes with minimal overhead.

3. Larger exponents for some S-boxes complicate the construction of low-degree polynomial systems in approximately λ Boolean variables from a single AIM evaluation.

* Corresponding author.

**The ISCISC'2025 program committee effort is highly acknowledged for reviewing this paper.

Email addresses: arka.debnath@esat.kuleuven.be, m.mahzoun@tue.nl

ISSN: 2008-2045

Our Contribution :

We present a fast exhaustive search preimage attack on AIM2. Our attack first transforms the polynomial system of AIM2 over $\mathbb{F}_{2^\lambda}$ to a boolean polynomial system over $\mathbb{F}_2$ and guesses the input bits, which results in a linear system that can be solved in time less than the security estimate of AIM2. Also, this serves as a counterexample to the notion that adding linearly independent equations to a polynomial system reduces the solving complexity of the system.

Structure of the paper :

In Section 2, we look briefly at the structure of AIM2 and its security arguments. In Section 3, we present our attack on AIM2.

2 Related Work

2.1 Result of Fast Exhaustive Search on AIM

In [5], Fukang, Mahzoun, Øygarden, and Meier used fast exhaustive search [7] to break AIM. The versions with 128/192/256-bit security are shown to be broken with complexity $2^{115}/2^{178}/2^{241}$. The attack exploits the low degree of the non-linear operations of AIM. Similar observation was made by Markku-Juhani O. Saarinen [8] who showed that AIM does not reach the claimed security level. The results of fast exhaustive search against AIM instances for three security levels $\lambda \in \{128, 192, 256\}$ are given in Table 1.

Table 1. Fast Exhaustive Search on instances of AIM

Scheme	λ	Time	Memory	Complexity
AIM-I	128	$2^{136.2}$	$2^{61.7}$	2^{115}
AIM-III	192	$2^{200.7}$	$2^{84.3}$	2^{178}
AIM-V	256	$2^{265.0}$	$2^{95.1}$	2^{241}

2.2 Description of AIM2

Given input/output size λ and an $(m+1)$-tuple of exponents $(e_1, ..., e_m, e_*) \in Z^{m+1}$, AIM2 [6] is a function defined as AIM2 : $\{0,1\}^\lambda \times \mathbb{F}_{2^\lambda} \to \mathbb{F}_{2^\lambda}$ with the following structure:

$$\text{AIM2}(iv, pt) = \text{Mer}[e_*] \circ \text{lin}[iv] \circ \text{Mer}[e_1, ..., e_m]^{-1} \circ \text{AddConst}(pt) \oplus pt$$

Non Linear Components: AIM2 [6] uses two types of S-boxes:

1. For $x \in \mathbb{F}_{2^\lambda}$, Mersenne S-box Mer$[e]$ is defined by

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

2. Inverse Mersenne S-box Mer$[e]^{-1}$, defined by

$$\text{Mer}[e]^{-1}(x) = x^{\overline{e}}$$
$$\text{where } \overline{e} = (2^e - 1)^{-1} (\text{mod } 2^\lambda - 1)$$

The exponents e in AIM2 are selected such that

1. Mer$[e]^{-1}$ produces 3λ quadratic equations.

2. $\gcd(e, \lambda) = 1$, ensuring the inverse exponent $\overline{e}$ is well-defined.

Linear Components AIM2 includes three types of linear components:

1. Constant Addition: Fixed constants $c_1, ..., c_m$ are added to the inputs of initial S-boxes, AddConst : $\mathbb{F}_{2^\lambda} \to \mathbb{F}_{2^\lambda}^m$ is defined by

$$\text{AddConst}(x) = (x + c_1)||...||(x + c_m)$$

the constants are defined in [6].

2. Affine Layer: Same as AIM, the affine layer consists of multiplication by an $\lambda \times \lambda m$ random binary matrix A_{iv} and addition by a random constant $b_{iv} \in \mathbb{F}_2^\lambda$. The matrix A_{iv} is composed of m random invertible $\lambda \times \lambda$ submatrices $A_{iv,i}$ which are generated along with the vector b_{iv} by an extendable-output function (XOF):

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

3. Feed-forward operation: Adds the input back to the output making the entire function non-invertible. Compared to AIM, AIM2 uses non-linear layers with higher exponents, and the inputs to each non-linear function are distinct at every step [6].

Overall Structure: To summarize, AIM2 works as follows,

$$\text{AIM2}(\text{IV}, \mathcal{X}) = \mathcal{Y}$$
$$= \left(\sum_{i=1}^{\ell} \sum_{j=0}^{\lambda-1} \left(\alpha_{i,j} (\mathcal{X} + c_i)^{(2^{e_i} - 1)^{-1}} \right)^{2^j} + \alpha_\lambda \right)^{2^{e_*} - 1} + \mathcal{X},$$

where $\alpha_{i,j}, \alpha_\lambda$ values are constants derived from IV, c_i values are public constants , and $\mathcal{X}, \mathcal{Y}$ and constants are elements of $\mathbb{F}_{2^\lambda}$. The design of AIM2 is depicted in Figure 1. Recommended sets of parameters for AIM2 instances for three security levels $\lambda \in \{128, 192, 256\}$ are given in Table 2.

Table 2. Recommended sets of parameters of AIM2

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

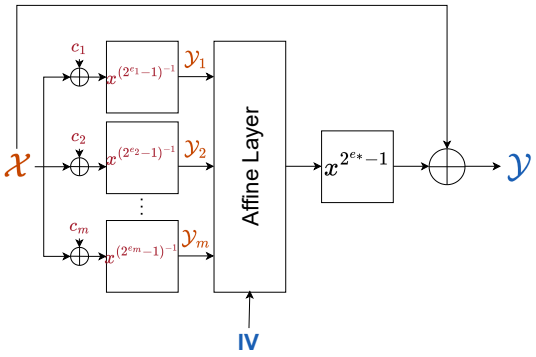

Figure 1. Design of AIM2.

2.3 Security of AIM2

The main focus in the cryptanalysis of AIM2 is against algebraic attacks, as well as the attacks that broke AIM. Different approaches to model AIM2 as a multivariate polynomial system are analysed in the case of an Algebraic attack. The summary of the complexity of algebraic attacks against AIM2 is described in Table 3.

Table 3. Summary of the complexity of algebraic attacks against AIM2 [6].

Scheme	variables	(equations, Deg)	dreg	Time (bits)
AIM2-I	λ	$(\lambda, 60)$	-	-
	2λ	$(3\lambda, 2)$	20	207.9
	3λ	$(12\lambda, 2)$	18	185.3
AIM2-III	λ	$(2\lambda, 114)$	-	-
	2λ	$(3\lambda, 2)$	30	301.9
	3λ	$(12\lambda, 2)$	26	262.4
AIM2-V	λ	$(2\lambda, 172)$	-	-
	2λ	$(\lambda, 2) + (2\lambda, 38)$	50	513.5
	3λ	$(6\lambda, 2)$	50	503.7
	4λ	$(18\lambda, 2)$	41	411.4

The AIM2 designers have analysed potential algebraic attack methods using various techniques based on the current parameters. The security model of AIM2 restricts attackers to $O(1)$ data complexity, making statistical attacks appear impractical at first glance.

3 Solving AIM2 System

The security of AIM2 is extensively analyzed against algebraic attacks. As the first step, AIM2 is modelled as an overdetermined polynomial system, and the complexity of solving that polynomial system is used to argue the security of AIM2. The reason to model AIM2 as an overdetermined polynomial system is the conventional belief that adding linearly independent equations to a polynomial system generally reduces its solving complexity [5]. While overdetermined systems are expected to be easier to solve, they are usually more complicated, and hence finding structures among the polynomials in the system is harder. We show that in the case of AIM2, using a determined polynomial system with fewer polynomials results in a simplified polynomial system that can be used to break AIM2 using exhaustive search.

3.1 Exhaustive Search on AIM2

Let $R = \mathbb{F}_{2^\lambda}[x, y_1, \ldots, y_m]$. Having an output value h, the preimage problem for AIM2 can be modeled the following polynomial system over R:

$$\begin{aligned}
(x \oplus c_1) &= y_1^{2^{e_1}-1} \\
(x \oplus c_2) &= y_2^{2^{e_2}-1} \\
&\vdots \\
(x \oplus c_m) &= y_m^{2^{e_m}-1} \\
\left(\alpha_\lambda \oplus \sum_{j=1}^{m}\sum_{i=0}^{\lambda-1} \alpha_{ji} y_j^{2^i}\right)^{2^{e_*}-1} &= (x \oplus h).
\end{aligned}$$

We can replace x with $y_1^{2^{e_1}-1} \oplus c_1$ and rewrite the equations as:

$$\begin{aligned}
\left(y_1^{2^{e_1}-1} \oplus c_1 \oplus c_2\right) &= y_2^{2^{e_2}-1} \\
&\vdots \\
\left(y_1^{2^{e_1}-1} \oplus c_1 \oplus c_m\right) &= y_m^{2^{e_m}-1} \\
\left(\alpha_\lambda \oplus \sum_{j=1}^{m}\sum_{i=0}^{\lambda-1} \alpha_{ji} y_j^{2^i}\right)^{2^{e_*}-1} &= y_1^{2^{e_1}-1} \oplus c_1 \oplus h.
\end{aligned}$$

Which can be rewritten as:

$$\begin{aligned}
\left(y_1^{2^{e_1}} y_2 \oplus y_1 y_2 (c_1 \oplus c_2)\right) &= y_1 y_2^{2^{e_2}} \\
&\vdots \\
\left(y_1^{2^{e_1}} y_m \oplus y_1 y_m (c_1 \oplus c_m)\right) &= y_1 y_m^{2^{e_m}} \\
y_1 \left(\alpha_\lambda \oplus \sum_{j=1}^{m}\sum_{i=0}^{\lambda-1} \alpha_{ji} y_j^{2^i}\right)^{2^{e_*}} & \\
= y_1^{2^{e_1}} \left(\alpha_\lambda \oplus \sum_{j=1}^{m}\sum_{i=0}^{\lambda-1} \alpha_{ji} y_j^{2^i}\right) & \\
\oplus\, y_1 \left(\alpha_\lambda \oplus \sum_{j=1}^{m}\sum_{i=0}^{\lambda-1} \alpha_{ji} y_j^{2^i}\right)(c_1 \oplus h). &
\end{aligned} \tag{1}$$

$$\tag{2}$$

which is a quadratic polynomial system over $\mathbb{F}_2$ with at most $m\lambda$ equations and $m\lambda$ variables. Note

that this introduces a spurious solution space at $y_1 = 0$. Also interestingly, i-th entry in equation 1 for $2 \leq i \leq m$ has a spurious solution at $y_i = 0$, but that does not expand to a full spurious solution unless we also have one of the following two conditions:

1. $y_1 = 0$ or,

2. $y_1 \neq 0$, and $y_i = 0$ for $2 \leq i \leq m$, and $\alpha_\lambda \oplus \sum_{i=0}^{\lambda-1} \alpha_{1i} y_1^{2^i} = 0$.

The first condition, $y_1 = 0$, implies $x = c_1$ which can be efficiently tested before starting the exhaustive search and then excluded. For AIM2-I and AIM2-III, $y_2 = 0$ implies $x = c_2$, which again can be efficiently tested and excluded before the exhaustive search. AIM2-V uses $m = 3$ with $c_2 \neq c_3$, excluding the possibility to have $y_2 = y_3 = 0$. Nevertheless, in order to enable the early-abort strategy below, one should test $x = c_2$ and $x = c_3$.

Transforming Equation 1 to a boolean polynomial system over $\mathbb{F}_2[y_{ji}], 1 \leq j \leq m, 0 \leq i < \lambda$ results in the following polynomial system:

$$\sum_{k=0}^{\lambda-1} \sum_{i=0}^{\lambda-1} \beta_{qjik} y_{1i} y_{jk} = 0 \text{ for } 0 \leq q < \lambda, 2 \leq j \leq m \tag{3}$$

Transforming Equation 2 to a boolean polynomial system over $\mathbb{F}_2[y_{ji} : 1 \leq j \leq m, 0 \leq i < \lambda]$ results in the following polynomial system:

$$\begin{aligned} &\sum_{j=2}^{m} \sum_{k=0}^{\lambda-1} \left(\sum_{i=0}^{\lambda-1} \gamma_{qjik} y_{1i} \right) y_{jk} \\ &= \sum_{i=0}^{\lambda-1} \left(\sum_{k=i+1}^{\lambda-1} \delta_{q1ik} y_{1k} \right) y_{1i} + \sum_{i=0}^{\lambda-1} \delta_{q1i1} y_{1i} \text{ for } 0 \leq q < \lambda. \end{aligned} \tag{4}$$

As a result, guessing the λ boolean variables $y_{10}, \ldots, y_{1\lambda-1}$ results in a linear system with $(m-1)\lambda$ variables and $m\lambda$ equations. Transforming equation 3 for y_{jk}'s, $0 \leq k < \lambda$ into matrix B_j and splitting equation 4 into blocks C_j and D where D represents the right-hand side of the equation, we derive the coefficient matrix of this linear system, which has the form:

$$M_{\lambda m, \lambda(m-1)} = \left(\begin{array}{cccc|c} B_2 & O & \cdots & O & 0 \\ O & B_3 & \cdots & O & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ O & O & \cdots & B_m & 0 \\ C_2 & C_3 & \cdots & C_m & D \end{array} \right), \tag{5}$$

where $B_j, C_j \in \mathbb{F}_2^{\lambda \times \lambda}$ and $D \in \mathbf{F}_2^{\lambda \times 1}$. Experimentally, each of the B_j matrices were observed to have rank $\lambda - 1$. Given the upper bound of solving a $\lambda \times \lambda$ matrix is $\left(\lambda^\omega + \lambda^2\right)$ (the LU decomposition has complexity λ^ω and back-substitution has complexity λ^2), the complexity of solving the above system is upper bounded by:

$$(m-1)\left(\lambda^\omega + \lambda^2\right).$$

At each step of the exhaustive search, the system can be updated efficiently by iterating over all 2^λ possible values of y_1 using Gray code. This way only one bit, say y_{1i}, of y_1 is changing at each iteration, and the matrix $M_{\lambda m, \lambda(m-1)}$ can be efficiently updated by adding β_{qjik} to the entry for y_{jk} in row q of B_j, γ_{qjik} to the entry for y_{jk} in row q of C_j, and $\delta_{q1i1} + \sum_{k=i+1}^{\lambda-1} \delta_{q1i1} y_{1k}$ in row q of D. Updating all B_j in this way costs $(m-1)\lambda^2$ additions and updating all C_j and D in this way costs no more than $m\lambda^2$ additions.

For each of the 2^λ guesses, the complexity of generating the matrix and solving the system is:

$$\begin{aligned} &(m-1)\left(\lambda^\omega + \lambda^2\right) + (m-1)\lambda^2 + m\lambda^2 \\ &\quad = (m-1)\left(\lambda^\omega + \lambda^2\right) + 2(m-1)\lambda^2 + \lambda^2 \\ &\quad\quad = (m-1)\left(\lambda^\omega + 3\lambda^2\right) + \lambda^2. \end{aligned}$$

However, for most attempts at the exhaustive search, it is not necessary to create all the rows of the C_i matrices. After creating all the B_i matrices for a guess of y_1, we obtain the corresponding guesses of all the y_i's, and then build the rows of the C_i matrices. If the guess y_1 was incorrect, it will not match the corresponding entry of D, and we have an inconsistency. Thus, if the guess y_1 is incorrect, then the system becomes inconsistent with high probability and the step can be aborted as soon as an inconsistency is detected.

Using the early abort method for inconsistent systems, the cost of updating B_j is $(m-1)s_{\text{abort}}$, and the cost of updating C_j and D is no more than ms_{abort}. s_{abort} is an attack parameter for the number of rows from B_i that needs to be updated during each iteration. The parameter must be large enough to detect wrong guesses within s_{abort} rows of the bottom part, allowing us to take advantage of the updates using the Gray code. At the same time, the parameter should be small enough to be able to avoid unnecessary update computations.
Thus, complexity of each step of the brute force attack is:

$$(m-1)\left(\lambda^\omega + \lambda^2 + 2s_{\text{abort}}\lambda\right) + s_{\text{abort}}\lambda,$$

Note that we stop solving the system as soon as a contradiction is found, so the $2s_{\text{abort}}\lambda$ is actually a $(1+\epsilon)s_{\text{abort}}\lambda$. We choose $s_{\text{abort}} = \log_2 \lambda$. each of the B_j matrices have rank $\lambda - 1$.

The complexities of solving different instances of AIM2, is summarised in Table 4.

Table 4. Solving complexity refers to the complexity of solving an instance of AIM2 using our exhaustive search approach. Naive search is the cost of running 2^{λ} trial executions of AIM2 as reported in [6]. Exhaustive search is the bit complexity of attack using brute force on AIM2 as reported in [6]. The complexity of solving an $n \times n$ matrix is assumed to be $\frac{n^3}{\log n}$, which is the bottleneck of the exhaustive search.

Primitive	m	Solving complexity	Naive search [6]	Exhaustive search [6]
AIM2-I	2	146.28	147.7	147
AIM2-III	2	211.9	212.9	212.3
AIM2-V	3	278.04	278.2	277.7

4 Conclusion

As shown in Table 4, the complexity of the exhaustive search on AIM2 has been overestimated in [6]. Our analysis reveals that the initially estimated complexity of 147 bits can be reduced by employing carefully optimised linear algebra techniques. Despite this improvement, the improved complexity remains higher than the complexity of an exhaustive search on AES. However, the complexity can be improved if we can devise a way to check the consistency of the linear system, leading to a more practical attack.

References

[1] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '07, page 21–30, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595936318. . URL https://doi.org/10.1145/1250790.1250794.

[2] 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 Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023*, pages 401–415. ACM, 2023. . URL https://doi.org/10.1145/3576915.3616579.

[3] Daniel Kales and Greg Zaverucha. Efficient lifting for shorter zero-knowledge proofs and post-quantum signatures. Cryptology ePrint Archive, Paper 2022/588, 2022. URL https://eprint.iacr.org/2022/588.

[4] 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. ISBN 978-981-99-8727-6.

[5] 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, December 2023. ISSN 2519-173X. .

[6] Seongkwang Kim, Jincheol Ha, Mincheol Son, and Byeonghak Lee. Efficacy and mitigation of the cryptanalysis on AIM. Cryptology ePrint Archive, Paper 2023/1474, 2023. URL https://eprint.iacr.org/2023/1474.

[7] C. Bouillaguet, H.-C. Chen, C.-M. Cheng, T. Chou, R. F. Niederhagen, A. Shamir, and B.-Y. Yang. Fast exhaustive search for polynomial systems in F_2. In *Proceedings of the 12th International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2010)*, volume 6225 of *Lecture Notes in Computer Science*, pages 203–218. Springer, 2010. .

[8] Markku-Juhani O. Saarinen. Round 1 (additional signatures) official comment: Aimer, September 2023. URL https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/BI2ilXblNy0. Google Groups post in the PQC-forum.

Arka Debnath is a PhD student at the research group COSIC at KU Leuven, Belgium. His research interest lies in the cryptanalysis of symmetric primitives.

Mohammad Mahzoun got his Ph.D. From Eindhoven University of Technology. He is currently affiliated with 3MIL-abs. His Research focuses on the analysis of algebraic ciphers.