Algebraic Attacks on RAIN and AIM Using Equivalent Representations

Fukang Liu[1], Mohammad Mahzoun[2]

[1] Tokyo Institute of Technology, Tokyo, Japan
liufukangs@gmail.com
[2] Eindhoven University of Technology, Netherlands
m.mahzoun@tue.nl

Abstract. It has been an important research topic to design novel symmetric-key primitives for advanced protocols like secure multiparty computation (MPC), fully homomorphic encryption (FHE) and zero-knowledge proof systems (ZK). Many such existing primitives adopt quite different design strategies from conventional block ciphers. One notable feature is that many of these ciphers are defined over a large finite field and the power map is commonly used to construct the nonlinear component due to its strong resistance against the differential and linear cryptanalysis. In this paper, we target the MPC-friendly ciphers AIM and RAIN used for the post-quantum signature schemes AIMer (CCS 2023 and NIST PQC Round 1 Additional Signatures) and Rainer (CCS 2022), respectively. Specifically, we could find equivalent representations of the 2-round RAIN and the full-round AIM respectively, which make them vulnerable to either the polynomial method or the simplified crossbred algorithm or the fast exhaustive search attack. Consequently, we could break 2-round RAIN with the 128/192/256-bit key in only $2^{116}/2^{171}/2^{224}$ bit operations. For the full-round AIM with the 128/192/256-bit key, we could break them in $2^{136.2}/2^{200.7}/2^{265}$ bit operations, which are equivalent to about $2^{115}/2^{178}/2^{241}$ calls of the underlying primitive.

Keywords: RAIN, AIM, equivalent representation, polynomial method, fast exhaustive search

1 Description of RAIN and AIM

RAIN [DKR+22] and AIM [KHS+22] are two MPC-friendly ciphers proposed at ACM CCS 2022 and 2023, respectively. In this section, we briefly introduce the two ciphers. Note that both ciphers are defined over $\mathbb{F}_{2^n}$ and we will not emphasize this in the corresponding primitive descriptions.

1.1 Description of RAIN

The r-round RAIN [DKR+22] is depicted in Fig. 1. The S-box is the inverse function over $\mathbb{F}_{2^n}$, i.e.

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

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

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

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

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

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

Since the security of RAIN is limited to the case when the attacker can only know one plaintext-ciphertext pair under the same key, the designers choose $r \in \{3, 4\}$. According the designers' analysis, even 2-round RAIN cannot be broken. As an attacker, the goal is thus to recover the secret key k from 1 known input-output pair (s_0, s_r). Indeed, in the signature scheme Rainer, k is the secret key while (s_0, s_r) is the public key. Therefore, the above attack is directly related to the security of Rainer.

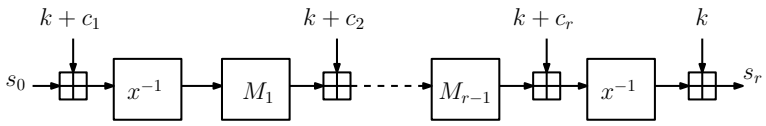

Fig. 1: The r-round RAIN

1.2 Description of AIM

The general construction of AIM [KHS+22] can be described as follows:

$$z_i = k^{2^{e_i}-1} \text{ for } i \in [1, m-1],$$
$$w = \sum_{i=1}^{m-1} B_i(z_i),$$

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

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

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

The corresponding graphic illustration is given in Fig. 2. Three variants of AIM

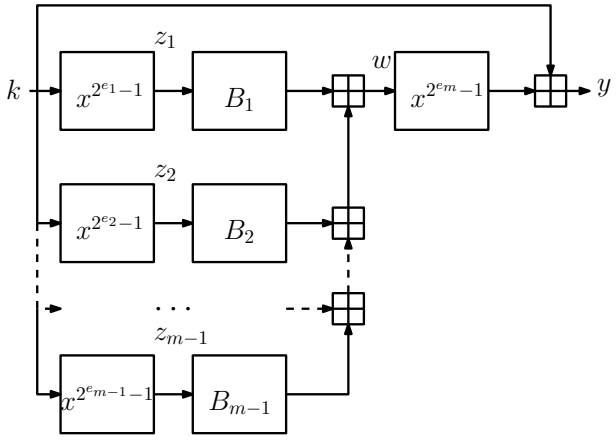

Fig. 2: The general construction of AIM

named AIM-I, AIM-II and AIM-III are specified by the designers, as shown in Table 1.

Table 1: The three variants of AIM

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

2 Equivalent Representations

The designers of RAIN claim that the polynomial method [Din21] is infeasible because the maximal algebraic degree is achieved after only one round due to the

inverse function. Although the claim for the inverse function is true, we point out that it is feasible to construct a low-degree equation system to equivalently describe the 2-round RAIN. This is obviously out of designers' expectations since even 2 rounds cannot be broken in their analysis. Due to this low-degree representation, the polynomial method works quite efficiently and we can break 2-round RAIN with the 128/192/256-bit key in only about $2^{118}/2^{171}/2^{224}$ bits, respectively.

It is found that there is also a similar problem in AIM. As the algebraic degree of the equivalent representation is relatively high, the polynomial method cannot work efficiently. However, the fast exhaustive search [BCC+10] is still applicable and breaking AIM-I/AIM-II/AIM-III requires about $2^{115}/2^{178}/2^{241}$ equivalent calls to the corresponding primitives, respectively.

2.1 Low-degree Representation for 2-Round RAIN

As shown in Fig. 3, we introduce a variable $s_1 \in \mathbb{F}_{2^n}$ to represent the internal state after the first S-box. Given the known pair (s_0, s_2), we aim to set up a low-degree equation system only in s_1. Solving this equation system will allow us to recover s_1 and then the secret key k cam be trivially recovered via:

$$k = s_1^{-1} + s_0 + c_1.$$

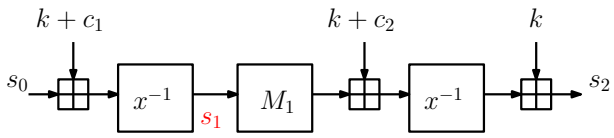

Fig. 3: The 2-round RAIN

First, we consider whether $s_1 = 0$. In this case, we have $k = s_0 + c_1$ and it can be trivially verified. Similarly, we can trivially verify whether $k + s_2 = 0$. Therefore, in the following, we always assume that

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

In this way, we have

$$k = \frac{1}{s_1} + s_0 + c_1$$

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

$$M_1(s_1) = \frac{1}{s_1} + s_0 + c_1 + c_2 + \frac{1}{\frac{1}{s_1} + s_0 + c_1 + s_2}.$$

For convenience, let

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

Then, t_0, t_1 are known constants and we have

$$M_1(s_1) = \frac{1}{s_1} + t_0 + \frac{1}{\frac{1}{s_1} + t_1} = \frac{1}{s_1} + t_0 + \frac{s_1}{1 + t_1 s_1}.$$

Based on this, we further have

$$s_1(1 + t_1 s_1) M_1(s_1) = 1 + t_1 s_1 + t_0 s_1(1 + t_1 s_1) + s_1^2.$$

By simplifying the equation, we have

$$s_1 M_1(s_1) + t_1 s_1^2 M_1(s_1) = 1 + t_1 s_1 + t_0 s_1 + t_0 t_1 s_1^2 + s_1^2.$$

Since $x \to x^{2^i}$ $(i \geq 0)$ over $\mathbb{F}_{2^n}$ corresponds to a linear vectorial Boolean function in terms of x, the above equation in terms of s_1 can be equivalently represented as n quadratic Boolean equations in s_1. Based on Dinur's algorithm for the polynomial method [Din21], we can recover s_1 in about $n^2 \cdot 2^{0.815n}$ bit operations. Our results are summarized in Table 2.

Analysis of 2-round RAIN with low-memory complexity We observe that for the 2-round RAIN, it is possible to construct an overdefined system of Boolean equations and then solve it with the simplified crossbred algorithm [BDT22, WWF+21, LMSI22].

Specifically, we have

$$(s_1 + t_1 s_1^2) M_1(s_1) = 1 + t_1 s_1 + t_0 s_1 + t_0 t_1 s_1^2 + s_1^2. \tag{2}$$

Based on this, we can derive

$$(s_1 + t_1 s_1^2)(M_1(s_1))^2 = (1 + t_1 s_1 + t_0 s_1 + t_0 t_1 s_1^2 + s_1^2) M_1(s_1), \tag{3}$$
$$(s_1 + t_1 s_1^2)^2 M_1(s_1) = (1 + t_1 s_1 + t_0 s_1 + t_0 t_1 s_1^2 + s_1^2)(s_1 + t_1 s_1^2), \tag{4}$$

which again correspond to $2n$ quadratic Boolean equations in s_1. Hence, we have $n+2n = 3n$ Boolean equations in n variables. We have experimentally verified that these $3n$ quadratic equations are linearly independent. For $n = 128/192/256$, the time complexity to solve the corresponding system of equations with the simplified crossbred algorithm is about $2^{116}/2^{176}/2^{236}$ bit operations and the memory complexity is about $2^{22}/2^{24}/2^{25}$ bits, respectively. The results are summarized in Table 2. As a future work, we will study the original crossbred algorithm [JV17] for these $3n$ equations.

Remark 1. For the simplified crossbred algorithm, we first pick u variables and eliminate all the $\binom{u}{2}$ quadratic terms in these u variables with Gaussian elimination. Then, the remaining $2n - \binom{u}{2}$ equations are linear in these u variables, i.e. by guessing the remainging $n - u$ variables, we can get $2n - \binom{u}{2}$ linear equations in u variables and solve them with Gaussian elimination. Hence, we choose the maximal u such that $2n - \binom{u}{2} - u \geq 0$.

Table 2: The results for 2-round RAIN

Method	n	Time (in bits)	Memory (in bits)
Polynomial Method	128	2^{118}	2^{91}
	192	2^{171}	2^{132}
	256	2^{224}	2^{173}
Simplified Crossbred	128	2^{116}	2^{22}
	192	2^{176}	2^{24}
	256	2^{236}	2^{25}

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

$$\left(M_1(s_1)+\frac{1}{s_1}+t_0\right)\left(\frac{1}{s_1}+s_0+c_1+c_3+M_2^{-1}\left(\frac{1}{\frac{1}{s_1}+s_0+c_1+s_3}\right)\right)=1$$

Let

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

Then, we have

$$\begin{aligned}&\left(M_1(s_1)+\frac{1}{s_1}+t_0\right)\left(\frac{1}{s_1}+t_2+M_2^{-1}\left(\frac{s_1}{1+t_3s_1}\right)\right)=1,\\ \hookrightarrow\ &\left(s_1M_1(s_1)+1+t_0s_1\right)\left(1+t_2s_1+s_1M_2^{-1}\left(\frac{s_1}{1+t_3s_1}\right)\right)=s_1^2.\end{aligned}$$

In this case, we cannot further expand $M_2^{-1}\left(\frac{s_1}{1+t_3s_1}\right)$. Otherwise, we could only get an equation system of algebraic degree $n-1$ in terms of s_1 because we need to multiply the term

$$\prod_{i=0}^{n-1}(1+t_3s_1)^{2^i}$$

in both sides of the equation to clear all denominators.

2.2 Low-degree Representation for AIM

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

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

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

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

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

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

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

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

and hence we have

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

Therefore, we have

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

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

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

where

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

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

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

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

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

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

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

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

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

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

This equation also corresponds to n Boolean equations[3] of algebraic degree upper bounded by $d_{\max}+e_m$. By solving these n Boolean equations, we can first recover w and then the secret key can be trivially recovered via:

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

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

Solving the n Boolean equations of algebraic degree $d_{\max}+e_m$. With the memory-efficient Möbius transform [Din21, Bou22], evaluating a Boolean polynomial in n variables of algebraic degree d over $\{0,1\}^n$ requires about $n\cdot\binom{n}{\leq d}$ bits of memory and the time is about $d\cdot 2^n$ bit operations. With this as the oracle of the fast exhaustive search method [BCC+10] to efficiently evaluate a polynomial, the time complexity to find the solution of w from n Boolean equations of algebraic degree upper bounded by $d_{\max}+e_m$ is estimated as about

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

[3] Note that $d_{\max}+e_m$ is very small in our attacks. We can interpolate these n polynomials of algebraic degree $d_{\max}+e_m$ with the recursive version of Möbius transform [Din21], and the corresponding time and memory are much smaller than 2^n. Of course, we first need to prepare a set of points of size $\sum_{i=0}^{d_{\max}+e_m}\binom{n}{i}$ and this phase will require us to equivalently evaluate AIM for $\sum_{i=0}^{d_{\max}+e_m}\binom{n}{i}$ different inputs, whose cost is still much smaller than 2^n.

bit operations. The memory complexity is upper bounded by

$$n \cdot n \cdot \sum_{i=0}^{d_{\max}+e_m} \binom{n}{i}$$

bits for simplicity[4].

Our results for the parameters of AIM are summarized in Table 3. Note that without this low-degree equivalent representation in n variables, the naive brute force takes about 2^{149}, $2^{214.4}$ and 2^{280} bit operations for the 128-bit, 192-bit and 256-bit security levels, respectively. Hence, the fast exhaustive search attack much improves the naive brute force attack. In other words, the time complexity of our attacks is equivalent to about $2^{115}/2^{178}/2^{241}$ calls of the underlying primitives.

Table 3: Fast exhaustive search (FES) attacks on AIM

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

3 Conclusion

We have shown that there are nontrivial low-degree equivalent representations in 2-round RAIN and full-round AIM, which can be exploited to mount effective attacks. Especially, as recovering the secret key of AIM is the underlying difficult problem of the signature scheme AIMer, which is one of the NIST Round 1 Additional Signatures, we believe that this work is meaningful.

Acknowledgment We thank Willi Meier and Morten Øygarden for discussing some ideas in this paper.

[4] In the original fast exhaustive search [BCC+10], evaluating a Boolean polynomial of degree d over $\{0,1\}^n$ requires $d \cdot 2^n$ bit operations, which is the same as the Möbius transform. However, it also requires a pre-processing phase which costs $\mathcal{O}(n^{2d})$ bit operations. This is too costly for the attack on AIM and hence we change it to the memory-efficient Möbius transform.

References

BCC+10. Charles Bouillaguet, Hsieh-Chung Chen, Chen-Mou Cheng, Tung Chou, Ruben Niederhagen, Adi Shamir, and Bo-Yin Yang. Fast exhaustive search for polynomial systems in F_2. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, volume 6225 of *Lecture Notes in Computer Science*, pages 203–218. Springer, 2010.

BDT22. Charles Bouillaguet, Claire Delaplace, and Monika Trimoska. A simple deterministic algorithm for systems of quadratic polynomials over f_2. In Karl Bringmann and Timothy Chan, editors, *5th Symposium on Simplicity in Algorithms, SOSA@SODA 2022, Virtual Conference, January 10-11, 2022*, pages 285–296. SIAM, 2022.

Bou22. Charles Bouillaguet. Boolean polynomial evaluation for the masses. Cryptology ePrint Archive, Paper 2022/1412, 2022. `https://eprint.iacr.org/2022/1412`.

Din21. Itai Dinur. Cryptanalytic applications of the polynomial method for solving multivariate equation systems over GF(2). *IACR Cryptol. ePrint Arch.*, page 578, 2021.

DKR+22. Christoph Dobraunig, Daniel Kales, Christian Rechberger, Markus Schofnegger, and Greg Zaverucha. Shorter signatures based on tailor-made minimalist symmetric-key crypto. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, pages 843–857. ACM, 2022.

JV17. Antoine Joux and Vanessa Vitse. A crossbred algorithm for solving boolean polynomial systems. In Jerzy Kaczorowski, Josef Pieprzyk, and Jacek Pomykala, editors, *Number-Theoretic Methods in Cryptology - First International Conference, NuTMiC 2017, Warsaw, Poland, September 11-13, 2017, Revised Selected Papers*, volume 10737 of *Lecture Notes in Computer Science*, pages 3–21. Springer, 2017.

KHS+22. Seongkwang Kim, Jincheol Ha, Mincheol Son, Byeonghak Lee, Dukjae Moon, Joohee Lee, Sangyub Lee, Jihoon Kwon, Jihoon Cho, Hyojin Yoon, and Jooyoung Lee. Aim: Symmetric primitive for shorter signatures with stronger security (full version). Cryptology ePrint Archive, Paper 2022/1387, 2022. `https://eprint.iacr.org/2022/1387`.

LMSI22. Fukang Liu, Willi Meier, Santanu Sarkar, and Takanori Isobe. New low-memory algebraic attacks on lowmc in the picnic setting. *IACR Trans. Symmetric Cryptol.*, 2022(3):102–122, 2022.

WWF+21. Congming Wei, Chenhao Wu, Ximing Fu, Xiaoyang Dong, Kai He, Jue Hong, and Xiaoyun Wang. Preimage attacks on 4-round keccak by solving multivariate quadratic systems. In Jong Hwan Park and Seung-Hyun Seo, editors, *Information Security and Cryptology - ICISC 2021 - 24th International Conference, Seoul, South Korea, December 1-3, 2021, Revised Selected Papers*, volume 13218 of *Lecture Notes in Computer Science*, pages 195–216. Springer, 2021.