A New Security Evaluation Method Based on Resultant for Arithmetic-Oriented Algorithms

Hong-Sen Yang

Joint with Qun-Xiong Zheng, Jing Yang, Quan-Feng Liu and Deng Tang

December 12, 2024

1. Introduction

2. Preliminaries

3. Optimized algebraic attacks against AO primitives based on resultant

4. Conclusions and Discussions

Background

High demand for privacy computing friendly symmetric primitives

- Traditional symmetric ciphers often designed and optimized for efficient software or hardware implementations.
- New primitives need consistent with many MPC/FHE/ZK-protocols to improve implementation performance.
- A lot of Arithmetic-Oriented (AO) primitives have been proposed: Rescue, `Anemoi`, JARVIS, MiMC, Poseidon, Arion, Griffin, GMiMC, ...

Background

High demand for privacy computing friendly symmetric primitives

- Traditional symmetric ciphers often designed and optimized for efficient software or hardware implementations.
- New primitives need consistent with many MPC/FHE/ZK-protocols to improve implementation performance.
- A lot of Arithmetic-Oriented (AO) primitives have been proposed: Rescue, Anemoi, Jarvis, MiMC, Poseidon, Arion, Griffin, GMiMC, ...

Characteristics of AO primitives

- Constructed over $\mathbb{F}_p$ (p is typically a large prime number).
- Mainly focus on minimizing the number of non-linear arithmetic operations.
- Use large S-boxes instead of small S-boxes.

Motivation

Cryptanalysis of AO algorithms

- Due to their native algebraic properties, algebraic attacks typically outperform other known cryptanalytic techniques against AO primitives.
- Gröbner basis are widely use to evaluate the security of AO primitives.

Motivation

Cryptanalysis of AO algorithms

- Due to their native algebraic properties, algebraic attacks typically outperform other known cryptanalytic techniques against AO primitives.
- Gröbner basis are widely use to evaluate the security of AO primitives.

Existing Problems

- The algebraic structure of an AO algorithm is not fully utilized.
- The complexity of Gröbner basis attack is not precise enough.
- The security of an AO algorithm is not well understood.

Our contributions

- Proposed a novel analysis framework for AO primitives that is much more efficient than existing ones, making the security evaluation more accurate.
- Proposed the substitution theory, effectively controlling the degree of equations when using resultants for algebraic attacks.
- Combined resultants with Lagrange interpolation, which effectively increased the number of rounds in practical attacks.

Comparison of the algebraic attacks against Rescue-Prime, Anemoi, and Jarvis

Primitives	Attacked rounds	Running time	Theoretical Complexities	References
Rescue-Prime	4	258500s	-	FSE2022
	4	885.5s	-	Sect.4
	5	-	2^{55}	FSE2022
	5	$\approx$ one day	-	Sect.4
	6	-	$2^{59.96}$	Sect.4
Anemoi	7	156348s	-	Crypto2024
	7	2968.55s	-	Sect.5
	8	38749.182s	-	Sect.5
	21	-	2^{118}	Crypto2024
	21	-	$2^{110.10}$	Sect.5
Jarvis	6	99989s	-	Asiacrypt2019
	6	368.96s	-	Sect.6
	8	455650.53s	-	Sect.6

1. Introduction
2. Preliminaries
3. Optimized algebraic attacks against AO primitives based on resultant
4. Conclusions and Discussions

The CICO problem

The so-called CICO (constrained-input constrained-output) problem, which is usually used to evaluate the security of AO algorithms, is defined as below.

Definition (CICO problem)

Let $s > 1$ be an integer and u a positive integer smaller than s. Let $F\colon \mathbb{F}_q^s \to \mathbb{F}_q^s$ be a permutation. The CICO problem of F is to find a vector $(x_1, \ldots, x_{s-u}, y_1, \ldots, y_{s-u}) \in \mathbb{F}_q^{2(s-u)}$ such that

$$F\left(x_1, \ldots, x_{s-u}, \underbrace{0, \ldots, 0}_{u}\right) = \left(y_1, \ldots, y_{s-u}, \underbrace{0, \ldots, 0}_{u}\right).$$

Resultant

Let $f(x,\boldsymbol{y}) = \sum_{i=0}^{m} f_i x^i$ and $g(x,\boldsymbol{y}) = \sum_{i=0}^{n} g_i x^i$, where $f_i, g_i \in \mathbb{F}_q[\boldsymbol{y}]$. Then

$$R(f,g,x) = \begin{vmatrix} f_m & f_{m-1} & \cdots & f_0 & & & & \\ & f_m & f_{m-1} & \cdots & f_0 & & & \\ & & \ddots & \ddots & \ddots & \ddots & & \\ & & & \ddots & \ddots & \ddots & \ddots & \\ & & & & f_m & f_{m-1} & \cdots & f_0 \\ g_n & g_{n-1} & \cdots & \cdots & g_0 & & & \\ & g_n & g_{n-1} & \cdots & \cdots & g_0 & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & & g_n & g_{n-1} & \cdots & \cdots & g_0 \end{vmatrix}.$$

The resultant is a powerful tool for solving systems of polynomial equations.

A simple example

- Consider the system of equations over $\mathbb{F}_{101}$: $\begin{cases} 5x^2 - 6xy + 5y^2 - 16 = 0 \\ 2x^2 - (1+y)x + y^2 - y - 4 = 0 \end{cases}$

A simple example

- Consider the system of equations over $\mathbb{F}_{101}$: $\begin{cases} 5x^2 - 6xy + 5y^2 - 16 = 0 \\ 2x^2 - (1+y)x + y^2 - y - 4 = 0 \end{cases}$
- Let $f(x,y) = 5x^2 - 6xy + 5y^2 - 16$ and $g(x,y) = 2x^2 - (1+y)x + y^2 - y - 4$. Then

$$R(f,g,x) = \begin{vmatrix} 5 & -6y & 5y^2-16 & 0 \\ 0 & 5 & -6y & 5y^2-16 \\ 2 & -(1+y) & y^2-y-4 & 0 \\ 0 & 2 & -(1+y) & y^2-y-4 \end{vmatrix} = 32(y-2)(y-1)(y+1)^2.$$

A simple example

- Consider the system of equations over $\mathbb{F}_{101}$: $\begin{cases} 5x^2 - 6xy + 5y^2 - 16 = 0 \\ 2x^2 - (1+y)x + y^2 - y - 4 = 0 \end{cases}$
- Let $f(x,y) = 5x^2 - 6xy + 5y^2 - 16$ and $g(x,y) = 2x^2 - (1+y)x + y^2 - y - 4$. Then

$$R(f,g,x) = \begin{vmatrix} 5 & -6y & 5y^2-16 & 0 \\ 0 & 5 & -6y & 5y^2-16 \\ 2 & -(1+y) & y^2-y-4 & 0 \\ 0 & 2 & -(1+y) & y^2-y-4 \end{vmatrix} = 32(y-2)(y-1)(y+1)^2.$$

- $\begin{cases} x = 2 \\ y = 2 \end{cases}$ or $\begin{cases} x = -1 \\ y = 1 \end{cases}$ or $\begin{cases} x = 1 \\ y = -1 \end{cases}$

Review of Rescue-Prime

Rescue-Prime is a family of AO hash functions and its round function is shown in Fig.1. The Ethereum Foundation Challenge for Rescue-Prime with $t = 3$ and $S : x \mapsto x^3$ (and so $S^{-1} : x \mapsto x^{1/3}$) is to find two pairs $(X_1, X_2), (Y_1, Y_2) \in \mathbb{F}_q^2$ satisfying $F(X_1, X_2, 0) = (Y_1, Y_2, 0)$.

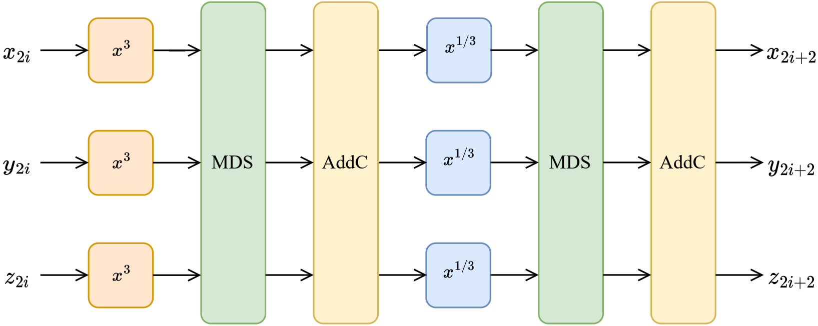


Figure 1: The i-th round function of Rescue-Prime when $t = 3$

Algebraic attack with forward modeling

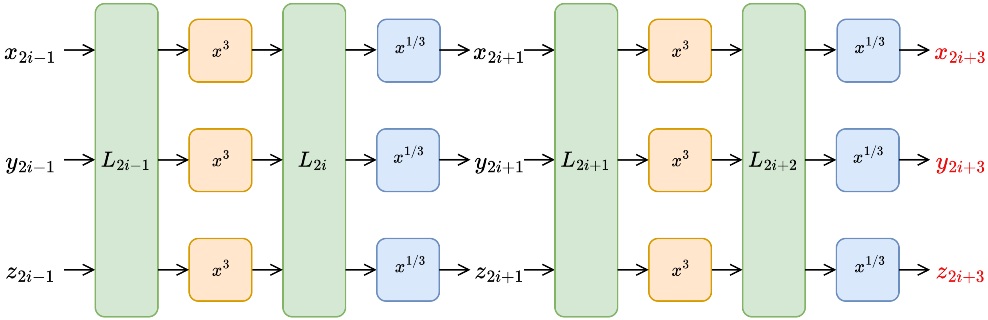


Figure 2: Forward modeling of Rescue-Prime

$$\begin{cases} f_{x_{2i+3}} := x_{2i+3}^3 - L_{2i+2,0} \circ SSS \circ L_{2i+1}(x_{2i+1}, y_{2i+1}, z_{2i+1}) = 0 \\ f_{y_{2i+3}} := y_{2i+3}^3 - L_{2i+2,1} \circ SSS \circ L_{2i+1}(x_{2i+1}, y_{2i+1}, z_{2i+1}) = 0 \\ f_{z_{2i+3}} := z_{2i+3}^3 - L_{2i+2,2} \circ SSS \circ L_{2i+1}(x_{2i+1}, y_{2i+1}, z_{2i+1}) = 0 \end{cases} \quad \text{for } i \in \{0, 1, \ldots, r-2\}.$$

Algebraic attack with forward modeling

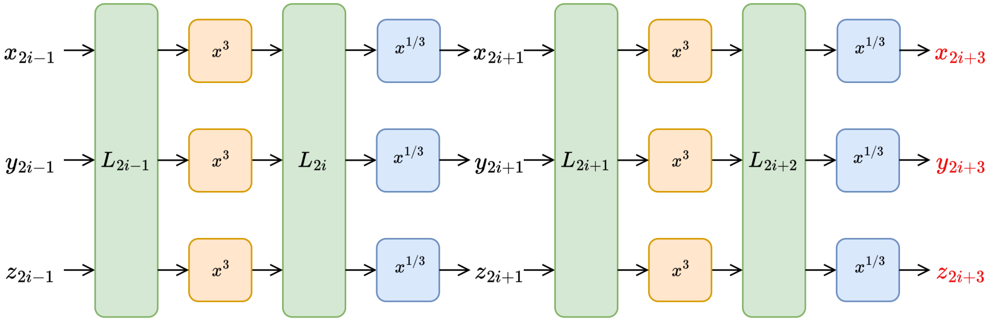


Figure 2: Forward modeling of Rescue-Prime

The output of the r-round Rescue-Prime should be of the form $(*, *, 0)$ in the CICO problem, so there is one more equation, which is

$$f_h := L_{2r-1,2}\left(x_{2r-1}, y_{2r-1}, z_{2r-1}\right) = 0.$$

Solving equations of forward modeling

$$y_1 = (-\frac{\alpha_{2,0}}{\alpha_{2,1}})^{1/3} x_1, \;\; z_1 = c \quad [\mathit{FSE}\ 2022].$$

$$\begin{cases} f_{x_3}(x_3, x_1) = 0 \\ f_{y_3}(y_3, x_1) = 0 \\ f_{z_3}(z_3, x_1) = 0 \\ \vdots \\ f_{x_{2r-1}}(x_{2r-1}, x_{2r-3}, y_{2r-3}, z_{2r-3}) = 0 \\ f_{y_{2r-1}}(y_{2r-1}, x_{2r-3}, y_{2r-3}, z_{2r-3}) = 0 \\ f_{z_{2r-1}}(z_{2r-1}, x_{2r-3}, y_{2r-3}, z_{2r-3}) = 0 \end{cases}$$

$$f_h(x_{2r-1}, y_{2r-1}, z_{2r-1}) = 0.$$

- The system has $3r-2$ equations in $3r-2$ unknowns (here we omit z_1 and y_1).
- $3r-3$ resultant operations can obtain a univariate polynomial in x_1

Cubic substitution

Using resultants for elimination will continuously increase the degree of the variables, but the special structure of this system of equations allows us to use substitution to control the degree of variables except x_1 no more than 3.

$$\begin{cases} x_{2i+3}^3 = L_{2i+2,0} \circ SSS \circ L_{2i+1}(x_{2i+1}, y_{2i+1}, z_{2i+1}) \\ y_{2i+3}^3 = L_{2i+2,1} \circ SSS \circ L_{2i+1}(x_{2i+1}, y_{2i+1}, z_{2i+1}) \\ z_{2i+3}^3 = L_{2i+2,2} \circ SSS \circ L_{2i+1}(x_{2i+1}, y_{2i+1}, z_{2i+1}) \end{cases} \quad \mathit{for}\ i \in \{0, 1, \ldots, r-2\}$$

Cubic substitution theory

A example of the cubic substitution

- After the first resultant $f_h = R(f_h, f_{z_{2r-1}}, z_{2r-1})$, by using the following substitutions in order, i.e.,

$$
\begin{aligned}
y_{2r-1}^3 &= L_{2r-2,1} \circ SSS \circ L_{2r-3}(x_{2r-3}, y_{2r-3}, z_{2r-3}), \\
x_{2r-1}^3 &= L_{2r-2,0} \circ SSS \circ L_{2r-3}(x_{2r-3}, y_{2r-3}, z_{2r-3}), \\
z_{2r-3}^3 &= L_{2r-4,2} \circ SSS \circ L_{2r-5}(x_{2r-5}, y_{2r-5}, z_{2r-5}), \\
&\cdots \\
y_3^3 &= L_{2,1} \circ SSS \circ L_1(x_1, kx_1, c), \\
x_3^3 &= L_{2,0} \circ SSS \circ L_1(x_1, kx_1, c),
\end{aligned}
$$

Then f_h can be transformed into a polynomial with the degree of each variable except x_1 less than 3.

Solving equations of forward modeling

Algorithm 1: Get the univariate polynomail for r-round Rescue-Prime

Input: $f_{x_{2i+3}}, f_{y_{2i+3}}, f_{z_{2i+3}}, f_h$ with $i \in \{0, 1, \ldots, r-2\}$.

Output: a univariate polynomial in $\mathbb{F}_q[x_1]$.

$i \leftarrow r-2$;

while $i \geq 1$ **do**

  $f_h \leftarrow R(f_h, f_{z_{2i+3}}, z_{2i+3})$;

  apply the cubic substitution to f_h;

  $f_h \leftarrow R(f_h, f_{y_{2i+3}}, y_{2i+3})$;

  apply the cubic substitution to f_h;

  $f_h \leftarrow R(f_h, f_{x_{2i+3}}, x_{2i+3})$;

  apply the cubic substitution to f_h;

  $i \leftarrow i-1$;

end

$f_h \leftarrow R(f_h, f_{z_{2i+3}}, z_3)$;

apply the cubic substitution to f_h;

$f_h \leftarrow R(f_h, f_{y_{2i+3}}, y_3)$;

apply the cubic substitution to f_h;

$f_h \leftarrow R(f_h, f_{y_{2i+3}}, x_3)$;

return f_h.

Complexity analysis (forward modeling)

Table 1: Time complexities of our attack against Rescue-Prime under forward modeling, where f_h is the output of Algorithm 1.

r	Complexity of resultants	Complexity of cubic substitutions	$\deg(f_h)$	Complexity of forward modeling
4	$2^{38.45}$	$2^{40.09}$	3^9	$\mathbf{2^{40.49}}$
5	$2^{50.17}$	$2^{52.77}$	3^{12}	$2^{52.99}$
6	$2^{61.90}$	$2^{65.47}$	3^{15}	$2^{65.58}$
7	$2^{73.62}$	$2^{76.51}$	3^{18}	$2^{76.69}$
8	$2^{85.34}$	$2^{88.10}$	3^{21}	$2^{88.30}$

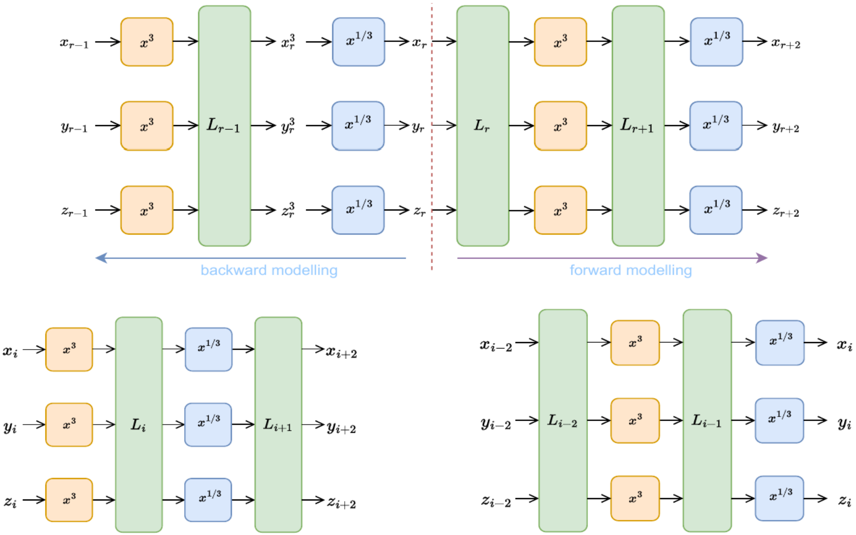


Figure 3: SFTM (start-from-the-middle) of Rescue-Prime

Eliminations of SFTM

r is odd
$$f_l = (C_{-1,2})^3 - L_{0,2}^{-1} \circ SSS \circ L_1^{-1}(x_2, y_2, z_2),$$ $$\begin{cases} f_{x_i} = x_i^3 - L_{i,0}^{-1} \circ SSS \circ L_{i+1}^{-1}(x_{i+2}, y_{i+2}, z_{i+2}) \\ f_{y_i} = y_i^3 - L_{i,1}^{-1} \circ SSS \circ L_{i+1}^{-1}(x_{i+2}, y_{i+2}, z_{i+2}) \\ f_{z_i} = z_i^3 - L_{i,2}^{-1} \circ SSS \circ L_{i+1}^{-1}(x_{i+2}, y_{i+2}, z_{i+2}) \end{cases} \text{for } i \in \{2, 4, \ldots, r-3\},$$ $$\begin{cases} f_{x_{r-1}} = x_{r-1}^3 - L_{r-1,0}^{-1}(x_r^3, y_r^3, z_r^3) \\ f_{y_{r-1}} = y_{r-1}^3 - L_{r-1,1}^{-1}(x_r^3, y_r^3, z_r^3) \\ f_{z_{r-1}} = z_{r-1}^3 - L_{r-1,2}^{-1}(x_r^3, y_r^3, z_r^3) \end{cases},$$ $$\begin{cases} f_{x_i} = x_i^3 - L_{i-1,0} \circ SSS \circ L_{i-2}(x_{i-2}, y_{i-2}, z_{i-2}) \\ f_{y_i} = y_i^3 - L_{i-1,1} \circ SSS \circ L_{i-2}(x_{i-2}, y_{i-2}, z_{i-2}) \\ f_{z_i} = z_i^3 - L_{i-1,2} \circ SSS \circ L_{i-2}(x_{i-2}, y_{i-2}, z_{i-2}) \end{cases} \text{for } i \in \{r+2, r+4, \ldots, 2r-1\},$$ $$f_h = L_{2r-1,2}\left(x_{2r-1}, y_{2r-1}, z_{2r-1}\right).$$

Resultants of SFTM

$$\begin{cases} f_l(x_2, y_2, z_2) = 0 \\ f_{x_2}(x_2, x_4, y_4, z_4) = 0 \\ f_{y_2}(y_2, x_4, y_4, z_4) = 0 \\ f_{z_2}(y_2, x_4, y_4, z_4) = 0 \\ \vdots \\ f_{x_{2r-1}}(y_{2r-1}, x_{2r-3}, y_{2r-3}, z_{2r-3}) = 0 \\ f_{y_{2r-1}}(y_{2r-1}, x_{2r-3}, y_{2r-3}, z_{2r-3}) = 0 \\ f_{z_{2r-1}}(z_{2r-1}, x_{2r-3}, y_{2r-3}, z_{2r-3}) = 0 \\ f_h(x_{2r-1}, y_{2r-1}, z_{2r-1}) = 0 \end{cases} \Longrightarrow \begin{cases} f_l(y_2, z_2) = 0 \\ f_{y_2}(y_2, x_4, y_4, z_4) = 0 \\ f_{z_2}(y_2, x_4, y_4, z_4) = 0 \\ \vdots \\ f_{x_{2r-1}}(y_{2r-1}, x_{2r-3}, y_{2r-3}, z_{2r-3}) = 0 \\ f_{y_{2r-1}}(y_{2r-1}, x_{2r-3}, y_{2r-3}, z_{2r-3}) = 0 \\ f_h(x_{2r-1}, y_{2r-1}) = 0 \end{cases}$$

Algorithm 2: Get two bivariate polynomials for r-round Rescue-Prime

Input: $f_l, f_h, f_{x_i}, f_{y_i}, f_{z_i}$ as defined in Table 3.

Output: two bivariate polynomials.

$i \leftarrow 2r - 1$;

while $i > r$ **do**

  $f_h \leftarrow R(f_h, f_{z_i}, z_i)$;

  apply the cubic substitution to f_h;

  $f_h \leftarrow R(f_h, f_{y_i}, y_i)$;

  apply the cubic substitution to f_h;

  $f_h \leftarrow R(f_h, f_{x_i}, x_i)$;

  apply the cubic substitution to f_h;

  $i \leftarrow i - 2$;

end

$i \leftarrow 2$;

while $i < r$ **do**

  $f_l \leftarrow R(f_l, f_{z_i}, z_i)$;

  apply the cubic substitution to f_l;

  $f_l \leftarrow R(f_l, f_{y_i}, y_i)$;

  apply the cubic substitution to f_l;

  $f_l \leftarrow R(f_l, f_{x_i}, x_i)$;

  apply the cubic substitution to f_l;

  $i \leftarrow i + 2$;

end

return f_h, f_l.

Solving equations of SFTM modeling

Final step

From Algorithm 2 we will get two bivariate polynomials $f_l, f_h \in \mathbb{F}_q[y_r, z_r]$ and further use them to compute $R(f_l, f_h, z_r)$ to eliminate z_r.

Solving equations of SFTM modeling

Final step

From Algorithm 2 we will get two bivariate polynomials $f_l, f_h \in \mathbb{F}_q[y_r, z_r]$ and further use them to compute $R(f_l, f_h, z_r)$ to eliminate z_r.

Memory overflow problem

However, when the number of rounds is high, the two polynomials would be quite complicated and directly computing the resultant $R(f_l, f_h, z_r)$ usually suffers from memory overflow.

Using Lagrange interpolation to compute resultant

Lagrange interpolation to compute resultant

- Assign a number of values to the variable y_r and compute $R(f_l, f_h, z_r)$ to get many interpolation pairs. (Can be parallelized)

Using Lagrange interpolation to compute resultant

Lagrange interpolation to compute resultant

- Assign a number of values to the variable y_r and compute $R(f_l, f_h, z_r)$ to get many interpolation pairs. (Can be parallelized)
- Using those pairs and the fast Lagrange interpolation to recover the univariate polynomial.

Using Lagrange interpolation to compute resultant

Lagrange interpolation to compute resultant

- Assign a number of values to the variable y_r and compute $R(f_l, f_h, z_r)$ to get many interpolation pairs. (Can be parallelized)
- Using those pairs and the fast Lagrange interpolation to recover the univariate polynomial.
- Using half-gcd to solve the univariate polynomial.

Complexity analysis (SFTM modeling)

Table 2: Time complexities of algebraic attacks under SFTM modeling against Rescue-Prime and the degrees of f_l, f_h, and f.

r	Complexity of resultants	Complexity of cubic substitutions	d_l	d_h	$\deg(f)$	Complexity of SFTM attacks
4	$2^{38.94}$	$2^{35.62}$	3^4	3^6	3^{10}	$\mathbf{2^{40.31}}$
5	$2^{38.94}$	$2^{35.62}$	3^7	3^6	3^{13}	$\mathbf{2^{48.37}}$
6	$2^{57.92}$	$2^{47.84}$	3^7	3^9	3^{16}	$2^{59.96}$
7	$2^{57.92}$	$2^{47.84}$	3^{10}	3^9	3^{19}	$2^{68.95}$
8	$2^{76.94}$	$2^{60.77}$	3^{10}	3^{12}	3^{22}	$2^{80.57}$

Experimental Results

We use the challenge parameters published by the Ethereum foundation with

$$p = 18446744073709551557 = 2^{64} - 59$$

and find a 5-round collision of Rescue-Prime under SFTM modeling which was originally thought as “hard” in the Ethereum Foundation challenge.

Table 3: Attack complexities of Rescue-Prime

r	Ethereum Foundation's time complexity	Best theoretical complexity	Time complexity of SFTM	Time complexity forward modeling	Best practical time in [1][1]	Practical time of SFTM	Practical time of forward modeling
4	$2^{37.5}$	2^{43}	$2^{43.02}$	$2^{40.49}$	258500s	2256.7s	885.5s
5	2^{45}	2^{57}	$2^{52.93}$	$2^{52.99}$	-	≈ one day	-

[1] Bariant, Augustin, et al. "Algebraic attacks against some arithmetization-oriented primitives." IACR Transactions on Symmetric Cryptology (2022): 73-101.

Table 4: Comparison with [2][2] in practical attack time of `Anemoi`

r	The attacks in [2]	Our attacks
3	$< 0.04s$	0.423s
4	0.58s	0.973s
5	30.97s	7.113s
6	2421.52s	296.568s
7	167201s	2968.55s
8	–	38749.182s

[2]Bariant, A., Boeuf, A., Lemoine, A., Ayala, I.M., Øygarden, M., Perrin, L., Raddum, H.: The algebraic freelunch efficient gröbner basis attacks against arithmetization-oriented primitives. Annual International Cryptology Conference, CRYPTO 2024.

Table 5: Comparison with [3][3] in practical attack time of JARVIS

r	Time for other resultants	Time for the final resultant	Total practical time	Time in [3]
6	11.2s	357.76s	368.96s	99989.0s
8	606.76s	455043.77s	455650.53s	–

[3] Albrecht, M.R., Cid, C., Grassi, L., Khovratovich, D., Lüftenegger, R., Rechberger, C., Schofnegger, M.: Algebraic cryptanalysis of stark-friendly designs: application to marvellous and mimc. In: Advances in Cryptology–ASIACRYPT 2019.

Summary of the resultant-based method

A novel analysis framework

- Construct a system of equations using forward modeling or SFTM modeling.

Summary of the resultant-based method

A novel analysis framework

- Construct a system of equations using forward modeling or SFTM modeling.
- Combine the resultant and the cubic substitution theory to eliminate variables in a specific order and finally get two bivariate polynomials.

Summary of the resultant-based method

A novel analysis framework

- Construct a system of equations using forward modeling or SFTM modeling.
- Combine the resultant and the cubic substitution theory to eliminate variables in a specific order and finally get two bivariate polynomials.
- Using fast Lagrange interpolation to recover the univariate polynomial.

Summary of the resultant-based method

A novel analysis framework

- Construct a system of equations using forward modeling or SFTM modeling.
- Combine the resultant and the cubic substitution theory to eliminate variables in a specific order and finally get two bivariate polynomials.
- Using fast Lagrange interpolation to recover the univariate polynomial.
- Find all the roots of the derived univariate polynomial.

Conclusions and Discussions

We present some potential weaknesses of AO algorithms that are susceptible to our attacks and give some potential improvements.

1. Using our new attack to decidie the secure number of rounds.

Conclusions and Discussions

We present some potential weaknesses of AO algorithms that are susceptible to our attacks and give some potential improvements.

1. Using our new attack to decidie the secure number of rounds.
2. Preventing variable isolation and substitution: using big S-boxes over extension finite fields.

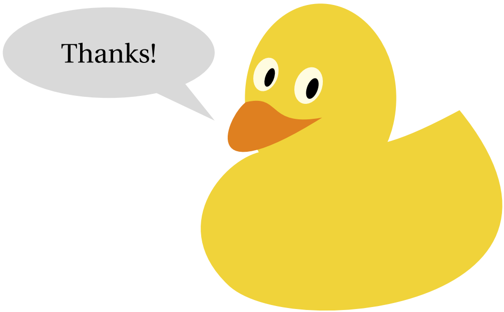
Conclusions and Discussions

We present some potential weaknesses of AO algorithms that are susceptible to our attacks and give some potential improvements.

1 Using our new attack to decidie the secure number of rounds.

2 Preventing variable isolation and substitution: using big S-boxes over extension finite fields.

3 Change addition of round constants operation $+ \longrightarrow \oplus$.


Thanks!