A Better Bivariate Resultant Attack on Round-Reduced Poseidon

Antoine Bak[1,2] and Maël Hostettler[3]

[1] Inria, Paris, France
[2] DGA, Paris, France
[3] LIX, École polytechnique, IP Paris, Palaiseau, France

Abstract

Poseidon is one of the most popular *arithmetization-oriented* (AO) hash function, due to its good performances both in evaluation and in Zero-knowledge proof protocols. It is for instance used in the Plonky3 library, and has been considered for use in the Ethereum protocol.

The security of Arithmetization-oriented hash functions is commonly evaluated through the CICO-k problem, which consists in controlling simultaneously k coordinates in the input and output of the permutation. This problem is in particular relevant to finding preimages in sponge or compression mode and solving zero-test problems. Depending on the size of the underlying field, different values of k may be relevant. In this paper, we focus on the case of $k = 2$, that was the subject of the recent bounty program by the Ethereum foundation. We denote by R_F and R_P the number of full and partial rounds of the permutation respectively.

In this setting, one can model the CICO-2 problem as a bivariate system $P(X, Y) = Q(X, Y) = 0$ where the polynomials have total degree $\delta = d^{R_F+R_P}$. The best known methods for solving bivariate systems are algorithms for computing bivariate resultants. Over a generic system with coefficients over a finite field, the best algorithms achieve an asymptotic bit complexity that is linear in $\delta^{2+\epsilon} \log(q)^{1+\epsilon}$, which is close to optimal, given that the input and output of the algorithm have bit size $\delta^2 \log(q)$.

However, the polynomial systems that stem from Poseidon are more structured, leading to a resultant that has degree $D_I = d^{2R_F+R_P}$, which is much less than what one would expect from a random bivariate system of degree $d^{R_F+R_P}$. This fact has already been exploited in a previous work that used an evaluation-interpolation approach to compute the bivariate resultant in time that is quasi-linear in $d^{3R_F+2R_P}$.

In this work, we exploit this fact by adapting another practical bivariate resultant algorithm to the special setting where the degree of the resultant of the equations is much lower than δ^2. By doing a careful analysis of the algorithm for systems with such property, we show that under some heuristics, the CICO-2 problem on Poseidon can be solved in time that is quasi-linear in $D_I \delta^{1-1/\omega}$, where $2 \leq \omega < 2.38$ is the theoretical exponent of matrix multiplication. We validate our approach by implementing our attack on reduced versions of the Poseidon permutation, and show a practical speedup compared to the previous approaches.

Keywords: Poseidon, CICO problem, Resultant

1 Introduction

Arithmetization-oriented hash functions play an important role in the security and efficiency of modern Zero-Knowledge proof protocols. Those hash functions are typically designed

Emails: antoine.bak@inria.fr (Antoine Bak), hostettler@lix.polytechnique.fr (Maël Hostettler)

This work is licensed under a "CC BY 4.0" license.
Date of this document: 2026-11-20.

so that a preimage of the hash function can be proven efficiently within the proof system. One of the ways of achieving this is to us round functions that are equivalent to low degree functions over finite fields of size $\geq 2^{30}$. Among them, Poseidon [GKR+21] and its variants [GKS23, GKK+26] are very popular choices due to their efficiency both in plain evaluation and in the proof protocol. They are for instance used in the Plonky3 library, and the Ethereum foundation recently organized a cryptanalysis initiative[4] on Poseidon with bounties that reward attacks on reduced versions of the Poseidon hash function.

The security of an Arithmetization-oriented permutation $P\colon \mathbb{F}_q^t \to \mathbb{F}_q^t$ is commonly evaluated through the CICO-k problem. This problem consists in finding an input $\mathbf{x}$ and output $\mathbf{y} = P(\mathbf{x})$ such that the first k coordinates of $\mathbf{x}$ and $\mathbf{y}$ equal 0. For a generic permutation, this problem can be solved in time q^k. Thus, solving this problem in time less than q^k yields a non-generic property of the permutation. In this work, we will focus on the special case of the CICO-2 problem.

The Poseidon CICO-2 problem. The CICO-2 problem has recently received some attention due to the Ethereum cryptanalysis initiative, where the organizers asked for solutions to CICO-2 and Zero-test problems on reduced versions of the Poseidon permutation. As the round function of Poseidon has degree d, taking an input of Poseidon linear in some variables X, Y, one can get output polynomials $P(X,Y), Q(X,Y)$ of degree $\delta = d^{R_F+R_P}$ such that a common solution to these two polynomials yields a solution to the CICO-2 problem. This paper will focus on how to solve such polynomial system.

A common method to solve systems of 2 equations in 2 variables consists in computing bivariate resultants. This was the approach followed by the authors of [BBB+26] during the 2025 bounty program of the Ethereum foundation. The authors made use of an evaluation-interpolation approach. In general, such an approach takes time $\tilde{\mathcal{O}}(\delta^3)$ for polynomials of degree δ. However, the authors observed that the resultant of P and Q has degree $D_I = d^{2R_F+R_P} < \left(d^{R_F+R_P}\right)^2$, which allows them to reduce the number of evaluation points in their evaluation-interpolation procedure. This allows them to reduce the time complexity of the algorithm to $\tilde{\mathcal{O}}(\delta \cdot D_I) = \tilde{\mathcal{O}}(d^{3R_F+2R_P})$ instead of the generic complexity bound $\tilde{\mathcal{O}}(d^{3(R_F+R_P)})$.

An alternative method for solving a CICO-2 problem consists in taking an input of the form $(X, y, 0 \ldots, 0)$ where $y \in \mathbb{F}_p$ is guessed randomly, and canceling one output coordinate [SV26] or computing the GCD of two output coordinates [BBB+26] that are univariate polynomials in X. As the probability that the second coordinate also equals 0, or that the GCD is nontrivial grows with $1/p$, such method has complexity $\tilde{\mathcal{O}}(p \cdot d^{R_F+R_P})$.

From a designer's point of view, finding a security argument against such attacks requires a plausible lower bound on the complexity of bivariate solving. Based on the fact that the polynomials P, Q are in two variables and have total degree $\delta = d^{R_F+R_P}$, the designers of Poseidon(2)b [GKK+26] conjectured that a CICO-2 attack has time complexity lower-bounded by $\delta^2 = d^{2(R_F+R_P)}$, which corresponds to the number of coefficients of P and Q.

Our contribution. In this work, we adapt another algorithm for bivariate resultant computation for solving Poseidon CICO-2 systems. The Villard algorithm [Vil18] computes the resultant of two generic polynomials of degree δ in time $\tilde{\mathcal{O}}(\delta^{3-1/\omega})$. Similarly to [BBB+26], our optimization relies on the fact that the output resultant has degree $D_I \lll \delta^2$ to reduce some parameter in the algorithm, and end up with an attack with complexity $\tilde{\mathcal{O}}(D_I \delta^{1-1/\omega}) = \tilde{\mathcal{O}}(d^{(3-1/\omega)R_F+(2-1/\omega)R_P})$. As long as $R_P > (\omega-1)R_F$, this improved bound even beats the lower bound $d^{2(R_F+R_P)}$ that the authors of [GKK+26] used in their security argument against bivariate resultant attacks.

[4]The program details can be found at www.poseidon-initiative.info

To assess the correctness of our attack, we had to make a detailed analysis of the Villard algorithm in the case where the input polynomials span an ideal of degree strictly less than δ^2. In particular, we show that analyzing the behavior of the Villard algorithm on the Poseidon systems amounts to analyzing the fraction descriptions of the bottom-left submatrix of some matrix $F^{-1}(Y)U_{\text{bot}}(Y)$, where $F(Y)$ is related to polynomials $f_s = u_s P + v_s Q$ that are of minimal degree among polynomials in $I = \langle P, Q\rangle$, and $U_{\text{bot}}(Y)$ is related to the u_s, v_s. With additional heuristics and an experimental study of the degree of the $U_{\text{bot}}(Y)$ matrix, we get convincing arguments for the validity of our approach. This validity is backed up by a successful implementation of the attack on reduced versions.

Note that while those results are stated for Poseidon, they could be adapted to other hash functions that contain nonlinear layer with only one coordinate affected by a nonlinear operation, such as Hades [GLR+20], Neptune [GOPS22] or GMiMC [AGP+19].

Related works. The CICO-k problem on Poseidon has already been studied both in the $k = 1$ and in the $k > 1$ case. For the CICO-1 problem, the authors of [BBLP22] used an algorithm that finds the roots in the field $\mathbb{F}_p$ of a polynomial of degree δ in time $\tilde{\mathcal{O}}(\delta \log(p))$ to get an attack whose complexity is quasi-linear in $d^{R_F+R_P}$. These results have recently been improved in the case where the prime p is smooth using Tangent Graeffe Transforms [ZSVD25], though it does not change the quasi-linear behavior in $d^{R_F+R_P}$.

For CICO problems with $k > 1$, the best attacks rely on Gröbner bases. The authors of [ABM23] observed that the solving degree of systems of equations stemming from Poseidon CICO problems was lower than what it would be for a random system. Thus, for some sets of parameters, the bottleneck shifts from the Gröbner basis computation step to finding an univariate polynomial in the ideal. A subsequent work [BBB+25] provided a method allowing an efficient computation of a Gröbner basis, and a proof that the degree D_I of some ideal associated to the CICO-k problem on Poseidon is upper bounded by $d^{kR_F+R_P}$. Putting those results together, we can solve the CICO-k problem in time D_I^ω, where ω is the multiplication matrix exponent. The complexity of algebraic attacks on Poseidon, without logarithmic factors, for different CICO-k problems can be found in Table 1.

A complementary direction to the works on system solving methods consists in finding techniques to skip rounds, and thus reduce the degree of the equations and of the ideal they span. The first work providing round-skipping techniques for Poseidon was [BBLP22], that skips the first full round for CICO-1 problems, with at least 3 branches. This result has then been generalized in [BBB+25] to larger CICO problems.

A recent paper [MG26] showed that, for Poseidon2(b) (which uses matrices that are not MDS), more rounds can be skipped than what is possible to do with MDS ones. On very wide states (e.g. $t = 24$ branches), this allows them to skip up to the first 9 rounds. When the matrix is MDS, it is still possible to do better than skipping one full round using the degree annihilation technique [SV26]. This technique uses the available degrees of freedom to reduce the degree of the inputs of the first partial round S-boxes, leading to lower equation degrees. We will also adapt this technique in the CICO-2 setting, for our attacks on the Ethereum bounties challenges.

Finally, we mention some recent works that provide attacks on full-round Poseidon provided the attacker can choose the linear layer of the permutation in a way that follows the specification of Poseidon. In [Vit26], the author provides a method that generates an MDS matrix that allows to skip enough round to get practical CICO-1, CICO-2 attacks on the full permutation. In [Jo26], the matrix is chosen in order to obtain collisions on the full permutation used in compression mode.

Outline. We describe the Poseidon hash function, its variants, and some properties of partial-round based primitives in Section 2. As our paper is about solving bivariate

Table 1: Best algebraic attacks for different CICO sizes (with no round-skips).

CICO-k	Method	Complexity ($\tilde{\mathcal{O}}$)	Source
1	Univariate solving	$d^{R_F+R_P}$	[BBLP22, ZSVD25]
2	Guess & Solve/GCD	$p \cdot d^{R_F+R_P}$	[BBB+26, SV26]
	Bivariate resultants	$d^{3R_F+2R_P}$	[BBB+26]
		$\geq d^{2(R_F+R_P)}$	[GKK+26]
		$d^{(3-1/\omega)R_F+(2-1/\omega)R_P}$	This work
$k \geq 3$	Gröbner bases	$d^{\omega(kR_F+R_P)}$	[BBB+25]

systems, we recall some results on bivariate solving and resultant computation in Section 3. Then, we adapt one of those algorithms, the Villard algorithm, and analyze it in the case of Poseidon in Section 4. Finally, we compare our results to previous ones in Section 5, and conclude in Section 6.

2 The Poseidon hash function and its properties

2.1 Poseidon(-2B)

In the following, we denote as t the number of branches, $M \in \mathbb{F}^{t\times t}$ an MDS matrix, and d the smallest integer ≥ 3 that is coprime with $p-1$.

The Poseidon hash function [GKR+21] and its variants [GKS23, GKK+26] rely on the Partial round strategy. More precisely, they use two kinds of rounds:

- Full rounds $\mathbf{x} \mapsto M \cdot S_{\text{full}}(\mathbf{x} + c^{(i)})$, where
$$S_{\text{full}}(x_1, \ldots, x_t) = (x_1^d, \ldots, x_t^d) \ .$$
- Partial rounds $\mathbf{x} \mapsto M \cdot S_{\text{part}}(\mathbf{x} + c^{(i)})$, where
$$S_{\text{part}}(x_1, \ldots, x_t) = (x_1^d, x_2, \ldots, x_t) \ .$$

The Poseidon [GKR+21] permutation with R_F full rounds and R_P partial rounds is defined as the composition of $R_F/2$ full rounds, R_P partial rounds, and then $R_F/2$ full rounds. The number of full round is fixed to $R_F = 8$, and the number of partial rounds depends on the targeted security level, the field size, and the number of branches.

The Poseidon2 [GKS23] hash function is very similar to Poseidon, except that it uses other non-MDS matrices that lead to better performances. In particular, they use different matrices depending on whether the round is a full (M_E) or a partial (M_I) round. Contrary to Poseidon, the Poseidon2 permutation starts by applying a linear layer M_E before the first round.

PoseidonB and Poseidon2B [GKK+26] are binary versions of the corresponding instance of Poseidon. In that case, the S-box exponent is fixed to $d = 7$. PoseidonB still uses a single MDS matrix, but contrary to Poseidon, it starts with a linear layer. On the other hand, Poseidon2B uses more optimized matrices that are not MDS. To compensate for the loss in security that may be caused by this lack of MDS property, the number of full rounds of Poseidon2B is raised to $R_F = 10$.

2.2 An upper bound on the degree of the ideal

Evaluating Poseidon on an input on of the form $(X, Y, 0, \ldots, 0)$, one gets output equations $P(X,Y), Q(X,Y)$ of total degree $\delta = d^{R_F+R_P}$. The Bézout theorem implies that a zero-dimensional system of two equations of degree δ spans an ideal of degree $D_I \leq \delta^2$, and

this bound is achieved for two generic polynomials of total degree δ. However, in the case of Poseidon, it has been observed in [BBB+26] that the degree of the resultant in the CICO-2 problem for Poseidon equals $d^{2R_F+R_P}$, which is far below the generic bound.

It has been shown in [BBB+25] that one can model the CICO-k problem on Poseidon with an ideal whose degree is upper bounded by $d^{kR_F+R_P}$. In the case of $k = 2$, this explains the observations from [BBB+26].

2.3 Other primitives with partial rounds.

While our work primarily focuses on Poseidon and its variants, our method could also find applications on other hash functions and block ciphers using the Partial round strategy. We give here the candidate primitives, as well as the corresponding degree of equations δ and degree of the ideal D_I.

- Key recovery on the Hades block cipher [GLR+20] with $t = 2$ branches based on one plaintext-ciphertext pair: in that case, it has been proven in [Ste24] that

$$D_I = d^{2R_F+R_P}, \delta = d^{R_F+R_P} \ .$$

- The Neptune [GOPS22] hash function: For the CICO-2 problem, we get

$$D_I \leq 4^{2R_F} d^{R_P}, \delta = 4^{R_F} d^{R_P} \ .$$

  For preimages on some t-to-2 compression mode, we get

$$D_I \leq 4^{R_F} d^{R_P - t + 2}, \delta = 4^{R_{f,2}} d^{R_P - t + 2} \ .$$

  Both upper bounds on D_I have been proven in [BBB+25].

- The function GMiMCHash based on $\text{GMiMC}_{\text{erf}}$, $\text{GMiMC}_{\text{crf}}$ [AGP+19]: For the CICO-2 problem in the crf case, it has been claimed in a recent talk [Roy26] that

$$\delta = D_I = d^{R-t} \ .$$

A similar analysis to the one made in this paper for Poseidon is likely to yield attacks on these primitives with a time complexity that is quasi-linear in $D_I \cdot \delta^{1-1/\omega}$. However, as our primary focus is on Poseidon and its variants, we leave as a future work checking that our bivariate resultant computation procedure actually works on these primitives (one would have to establish a result analogous to Conjecture 2), and assessing the impact of this attack on their security margin.

3 Algorithms for bivariate solving

In this section, we recall important results on solving methods for bivariate polynomial systems, and focus in particular on resultant computation.

3.1 Notations

In the following, we denote by P, Q two polynomials of respective total degrees n, m. When $n = m$, we denote $\delta = n = m$. We denote by $\mathbb{F}$ a field and by $\overline{\mathbb{F}}$ its algebraic closure; unless stated otherwise, the complexities of the algorithms are evaluated in number of field operations. $M(n)$ denotes the time complexity of multiplying two polynomials in $\mathbb{F}[X]$ of degree n. In particular, $M(n) = \mathcal{O}(n \log(n) \log\log(n))$. We make use of the soft-O notation, i.e. $f(n) = \tilde{\mathcal{O}}(g(n))$ if $f(n) = \mathcal{O}(g(n) \log(g(n))^k)$ for some $k \geq 0$.

3.2 Sylvester matrix and resultants

Definition 1 (Sylvester matrix and resultant)**.** Let $P, Q \in \mathbb{F}[X]$ such that $\deg(P) = n, \deg(Q) = m$ and $P = \sum_{i=0}^{n} p_i X^i, Q = \sum_{i=0}^{m} q_i X^i$. We define the Sylvester matrix of P and Q as

$$S(P,Q) = (s_{ij})_{1 \leq i,j \leq n+m} \ ,$$

$$s_{ij} = \begin{cases} p_{n+i-j} \ , \ 1 \leq i \leq m \ , \ 0 \leq j-i \leq n \ , \\ q_{2m+i-j} \ , \ m+1 \leq i \leq n+m \ , \ m \leq j-i \leq 2m \ , \\ 0 \text{ else.} \end{cases}$$

The resultant of P and Q is defined as the determinant of the Sylvester matrix

$$\mathrm{Res}(P,Q) = \det(S(P,Q)) \ .$$

Example 1. Let $P = 3X^2 + 2X + 1, Q = 7X^3 + 6X^2 + 5X + 4$, then:

$$S(P,Q) = \begin{pmatrix} 3 & 2 & 1 & 0 & 0 \\ 0 & 3 & 2 & 1 & 0 \\ 0 & 0 & 3 & 2 & 1 \\ 7 & 6 & 5 & 4 & 0 \\ 0 & 7 & 6 & 5 & 4 \end{pmatrix} \ .$$

The main motivation for studying resultants is the following result:

Theorem 1 (cf. Chapter 3, §5, Proposition 8 of [CLOL07])**.** *For $P, Q \in \mathbb{F}[X]$:*

$$\mathrm{Res}(P,Q) = 0 \text{ if and only if } \gcd(P,Q) \neq 1 \ .$$

That is, the resultant of P, Q cancels out whenever P and Q share a common root in the algebraic closure of $\mathbb{F}$ denoted as $\overline{\mathbb{F}}$.

3.3 Application to bivariate solving.

In the case where $P, Q \in \mathbb{F}[X, Y]$, we can consider P and Q as polynomials in X with coefficients over the ring $\mathbb{F}[Y]$. The resultant of P and Q seen as polynomials in X (denoted as $\mathrm{Res}_X(P,Q) \in \mathbb{F}[Y]$) is polynomial in the coefficients of P, Q. Then whenever $y \in \mathbb{F}$ is a root of $\mathrm{Res}_X(P,Q)$, we know by the previous theorem that $P(X,y)$ and $Q(X,y)$ share a common root. This leads to the following strategy for solving bivariate systems:

1. Compute the resultant of P, Q in X: $R(Y) = \mathrm{Res}_X(P,Q)$.
2. Find the roots of R in $\mathbb{F}$.
3. For all y root of R, compute $G(X) = \gcd(P(X,y), Q(X,y)) \neq 1$. For all $x \in \mathbb{F}$ such that $G(x) = 0$, (x,y) is a solution of the system.

The most involved step in that procedure is the first one. We call the computation of $\mathrm{Res}_X(P,Q)$ for $P, Q \in \mathbb{F}[X,Y]$ the *bivariate resultant* computation. In what follows, we recall the main algorithms for this task in the case where P and Q have the same total degree δ. In that case, we have that $\deg(\mathrm{Res}_X(P,Q)) \leq \delta^2$.

Evaluation-interpolation. The most straightforward way of computing the bivariate resultant of P and Q is the evaluation-interpolation method. The technique is as follows:

1. Evaluate P and Q at δ^2+1 values of Y: $P(X,y_i)$ and $Q(X,y_i)$, for $0 \leq i \leq \delta^2$.
2. For all $0 \leq i \leq \delta^2$, compute the resultant $r_{y_i} = \mathrm{Res}(P(X,y_i), Q(X,y_i))$. Note that $r_{y_i} = R(y_i)$ where $R(Y) = \mathrm{Res}_X(P,Q)$.
3. Interpolate $R(Y)$, using its values over $\delta^2+1 \geq \deg(R)+1$ points r_{y_i}.

Using points y_i in geometric progression, one can perform evaluation and interpolation [BS05] of polynomials of degree less than n over $n+1$ points in

$$\mathcal{O}(M(n)) \ .$$

Applying this result to the i-th coefficient of $P, p_i(Y)$ and $Q, q_i(Y)$ for $0 \leq i \leq \delta$, we perform the first step in time

$$\mathcal{O}(M(\delta^2)\delta) \ .$$

The third step is an interpolation of a polynomial of degree $\leq \delta^2$ and thus takes time $\mathcal{O}(M(\delta^2))$. For the second step, computing the resultant of two univariate polynomials of degree δ can be done in time $\mathcal{O}(M(\delta)\log(\delta))$ using the half-gcd algorithm. As we iterate over δ^2+1 values of y, the second step takes time

$$\mathcal{O}(\delta^2 M(\delta)\log(\delta)) = \tilde{\mathcal{O}}(\delta^3) \ .$$

This step is the bottleneck of the algorithm. Memory complexity in

$$\mathcal{O}(\delta^2)$$

can be achieved by doing steps 1 and 2 of the algorithm on batches of size δ. Indeed, storing the $P(X,y_i), Q(X,y_i)$ where i ranges over an interval of size δ takes memory $\mathcal{O}(\delta^2)$, and storing the points r_{y_i} never requires to store more than δ^2+1 field elements.

The Villard algorithm (generic systems) [Vil18]. Under some genericity assumption over the input polynomials, we can use the Villard algorithm to compute the bivariate resultant in time

$$\mathcal{O}(M(\delta^{3-1/\omega})\log(\delta)) = \tilde{\mathcal{O}}(\delta^{3-1/\omega}), \text{ and memory } \mathcal{O}(\delta^{2+1/\omega}) \ .$$

Here we only provide a high-level overview of the algorithm, and an analysis of its time complexity without the logarithmic factors. A more detailed presentation of the algorithm in the case of Poseidon will be made in Section 4.1. Let m be some parameter to be chosen later, the algorithm can then be summarized as follows:

1. Evaluate P, Q over $2\lceil \delta^2/m \rceil + 1$ points $y_0, \ldots, y_{2\lceil \delta^2/m \rceil}$.
2. For all $y_i, 0 \leq i \leq 2\lceil \delta^2/m \rceil$, compute the top-right block of the inverse of the Sylvester matrix:
$$S_{\mathrm{inv},i}^{\mathrm{top-right}} = \Pi_{\mathrm{top}} S^{-1}(P(X,y_i), Q(X,y_i))\Pi_{\mathrm{right}} \ .$$
3. These matrices give the power series of the top-right matrix block of the inverse of $S_X(P,Q)$:
$$S_{\mathrm{inv}}^{\mathrm{top-right}}(Y) = \Pi_{\mathrm{top}} S_X^{-1}(P(X,Y), Q(X,Y))\Pi_{\mathrm{right}} \ ,$$
modulo $\prod_{0 \leq i \leq 2\lceil \delta^2/m \rceil}(Y - y_i)$. From this power series, one can recover a fraction representation $N(Y), D(Y)$ of degree $\leq \lceil \delta^2/m \rceil$ such that:
$$S_{\mathrm{inv}}^{\mathrm{top-right}}(Y) = N(Y)D(Y)^{-1} \ .$$

4. Finally, we compute the determinant of $D(Y)$. Under some genericity assumption on P, Q, we have that:

$$\det(D(Y)) = \lambda \cdot \mathrm{Res}_X(P, Q) \ , \ \lambda \in \mathbb{F}^\times \ .$$

If needed, the constant λ can be recovered e.g. by evaluating $\mathrm{Res}(P(X, y), Q(X, y))$ and $\det(D(y))$ on y such that $\mathrm{Res}(P(X, y), Q(X, y)) \neq 0$.

Step 1 and 2 of the algorithm have complexity $\tilde{\mathcal{O}}(\delta^3/m)$, whereas step 3 and 4 have complexity $\tilde{\mathcal{O}}(m^{\omega-1}\delta^2)$. Thus, taking $m = \tilde{\Theta}(\delta^{1/\omega})$ gives optimal complexity $\tilde{\mathcal{O}}(\delta^{3-1/\omega})$.

Lecerf and Van der Hoeven algorithm (finite fields) [VDHL21]. We quickly mention an algorithm computing bivariate resultants for generic polynomials over finite fields $\mathbb{F}_q$. This algorithm computes the bivariate resultant in a number of bit operations

$$\mathcal{O}(\delta^{2+\epsilon} \log(q)^{1+\epsilon}) \ ,$$

for all $\epsilon > 0$. This is essentially optimal as the input polynomials and the output resultant have bit size $\Theta(\delta^2 \log(q))$. To the best of our knowledge, there is no available implementation of this algorithm, and the δ^ϵ hides a large factor due to the use of the Kedlaya-Umans algorithm [KU08], which makes the procedure unpractical for realistic values of δ.

3.4 Dependence on the degree of the ideal

The resultant of two generic polynomials of total degree n and m has degree $n \cdot m$. However, it may happen (for instance in the case of systems coming from Poseidon), that this degree is strictly lower. The following theorem relates the degree of the resultant of P and Q with a well-known invariant of the ideal $I = \langle P, Q \rangle$, namely the degree of the ideal D_I defined as $D_I = \dim_{\mathbb{F}}(\mathbb{F}[X, Y]/I)$.

Theorem 2 (A consequence of Proposition 5 in [BKM05])**.** *Let $P, Q \in \mathbb{F}[X, Y], I = \langle P, Q \rangle$ a zero-dimensional ideal, and D_I its degree. Let assume that the leading coefficients of P and Q in X $p_n(Y), q_m(Y)$ are coprime, then we have that*

$$\deg(\mathrm{Res}_X(P(X, Y), Q(X, Y))) = D_I \ .$$

Using the fact $D_I \ll \delta^2$, the authors of [BBB+26] adapted the evaluation-interpolation algorithm to get a complexity with a better dependence on D_I. That is, instead of evaluating the resultant $\mathrm{Res}(P(X, y_i), Q(X, y_i))$ over $\delta^2 + 1$ points, they evaluate it over only $D_I + 1$ points. As the resultant $\mathrm{Res}_X(P, Q)$ has degree D_I, this number of points is sufficient for recovering the full resultant, leading to a time complexity in

$$\mathcal{O}(D_I \cdot M(\delta) \log(\delta)) = \tilde{\mathcal{O}}(D_I \delta) \ .$$

In the case where the system stems from Poseidon, memory complexity can be improved to

$$\mathcal{O}(D_I) \ ,$$

instead of δ^2. To do so, we do not construct the polynomials P and Q by evaluating Poseidon on the symbolic variables (X, Y). Instead, for each evaluation point y_i, we evaluate Poseidon on (X, y_i), where only X is a symbolic variable. Each evaluation has time complexity $\mathcal{O}(M(\delta))$ and memory complexity $\mathcal{O}(\delta)$. Thus, we never store more than a constant number of univariate polynomials of degree δ and the $D_I + 1$ evaluation points r_{y_i}, which yields the announced memory complexity.

The aim of the present article is to provide a similar adaptation of the Villard algorithm to the case $D_I < \delta^2$.

4 Study of the Villard algorithm on Poseidon

In this section, we study the behavior of the Villard algorithm on systems coming from the Poseidon hash function, and in particular when the degree of the ideal D_I is much lower than what one would expect from Bézout bound (δ^2). We first describe explicitly the modifications we make on the Villard algorithm to take into account the lower degree of the ideal. These modifications amount to using a lower number of evaluation points than in the description of the Villard algorithm given in Section 3.3, decreasing from $\mathcal{O}(\delta^2/m)$ to $\mathcal{O}(D_I/m)$.

Then, we justify the correctness of the algorithm by making a detailed study of the fraction reconstruction step. Section 4.2 is devoted to the study of the structure of the bottom-left submatrix of the inverse of the Sylvester matrix $S_X(P,Q)$. It is shown in that section that this submatrix is the bottom-left submatrix of the fraction $F(Y)^{-1}U_{\text{bot}}(Y)$, $F(Y)$ being related to polynomials $f_s(X,Y) \in \langle P,Q\rangle$ of minimal leading monomials for the *grevlex* ordering, and $U_{\text{bot}}(Y)$ is related to some polynomials $u_s(X,Y), v_s(X,Y)$ such that $u_sP + v_sQ = f_s$.

Using this structural result, we state some heuristics in Section 4.3 under which our modified version of the Villard algorithm returns the resultant using $\mathcal{O}(D_I/m+(\deg(U_{\text{bot}})-\delta))$ points. The rationale behind our heuristics is that they are valid with high probability if $F(Y)$ and $U_{\text{bot}}(Y)$ are replaced with random matrices with the same degree profiles.

Finally, we make experiments to study the degree of the $U_{\text{bot}}(Y)$ matrix in Section 4.4. Those experiments allow us to make Conjecture 2 under which the degree of $U_{\text{bot}}(Y)$ is upper bounded by $\sqrt{2D_I} + \delta$. Using the heuristics made on the previous subsection, the number of data points needed collapses to $\mathcal{O}(D_I/m)$, which validates the number of evaluation points used in our modified version of the Villard algorithm.

Possible extension to generic bivariate systems with low D_I. While our primary focus is on the Poseidon hash function, it could be natural to ask if our methods allows better complexity for solving bivariate ideals that are generic under the same constraint on the degree of the resultant of P and Q. The results from Section 4.2 are still valid, and we could make the same heuristics as in Section 4.3 to bound the necessary evaluation points by $\mathcal{O}(D_I/m + (\deg(U_{\text{bot}}) - \delta))$.

Outside of proving that the heuristics made hold for a generic bivariate system, making the modified Villard algorithm work on generic systems would also require to establish a bound on the degree of $U_{\text{bot}}(Y)$. Finding such a bound of the form $\mathcal{O}(\delta + D_I/\delta^{1/\omega})$ and proving the heuristics from Section 4.3 for generic systems would be sufficient to prove the following conjecture.

Conjecture 1 (The modified Villard algorithm when $D_I < \delta^2$)**.** *Let $\mathcal{X} \subset \overline{\mathbb{F}}^{2\binom{\delta+2}{2}}$ the Zariski-closed set defined as*

$$\mathcal{X} = \left\{ ((p_{ij})_{i+j\leq\delta}, (q_{ij})_{i+j\leq\delta}) \mid p_{\delta 0} = q_{\delta 0} = 1,\ \deg(\text{Res}_X(P,Q)) \leq D \right\},$$

where $P(X,Y) = \sum_{i+j\leq\delta} p_{ij}X^iY^j, Q(X,Y) = \sum_{i+j\leq\delta} q_{ij}X^iY^j$.

We conjecture that whenever $\mathcal{X} \neq \emptyset$, there exists a Zariski open set $\mathcal{U} \subset \overline{\mathbb{F}}^{2\binom{\delta+2}{2}}$ such that $\dim(\mathcal{X} \cap \mathcal{U}^C) < \dim(\mathcal{X})$ and for all polynomials P, Q whose coefficients lie in $\mathcal{U} \cap \mathcal{X}$, Algorithm 1 returns $\text{Res}_X(P,Q)$ up to a nonzero constant.

A direct implication of this conjecture would be that a generic affine system conditioned on $D_{\langle P,Q\rangle} \leq D$ can be solved in time

$$\tilde{\mathcal{O}}(\max(\delta^2, D\cdot\delta^{1-1/\omega})), \text{ and memory } \tilde{\mathcal{O}}(\max(\delta^2, D\cdot\delta^{1/\omega})).$$

Note that in both cases, the δ^2 term is due to the size of the input polynomials and the need for a first multipoint evaluation. Similarly to the improvement in memory complexity presented in Section 3.4, the case where the system stems from Poseidon allows for lower time and memory complexity. Indeed, multipoint evaluation can be performed efficiently by evaluating symbolically the Poseidon permutation instead of constructing the polynomials P and Q and then evaluating them. This allows to remove the δ^2 term in both time and memory complexity. More details will be provided in the next subsection.

At the moment, the best algorithm to solve bivariate systems with that condition on the degree of the ideal comes from [BBB+26]. As long as the coefficients of X^δ of P and Q are nonzero, their adapted evaluation-interpolation algorithm computes the resultant of P and Q and solves the bivariate system in time $\tilde{\mathcal{O}}(D \cdot \delta)$ and memory $\tilde{\mathcal{O}}(\delta^2)$. If Conjecture 1 is true, Algorithm 1 improves time complexity by a factor $\delta^{1/\omega}$ when $D > \delta^{1+1/\omega}$ and by a factor D/δ when $D \leq \delta^{1+1/\omega}$.

4.1 Variant of the Villard algorithm

Our algorithm mostly follows the original Villard algorithm [Vil18]. The main difference between the original algorithm and ours lies in the number of evaluation points, where we choose to take $2\lceil D_I/m \rceil + 4\lceil \sqrt{2D_I} \rceil + 2m + 1$ instead of $4\delta\lceil \delta/m \rceil + 1$ points. We will see later that in the case where $\delta^2 > D_I/\delta^{1/\omega}$, the choice of the submatrices we extract is crucial to reduce the number of evaluation points required by the algorithm, whereas the Villard algorithm would still work under other choices of submatrices. Here, we chose the submatrices $\Pi_{\text{bot}} S(P,Q)^{-1} \Pi_{\text{left}}$, where $\Pi_{\text{bot}} = (0 \ \dots \ 0 \ I_m) \in \mathbb{F}^{m \times 2\delta}, \Pi_{\text{left}} = (I_m \ 0 \ \dots \ 0)^T \in \mathbb{F}^{2\delta \times m}$ are respectively the projections on the m bottom rows and the m leftmost columns.

In this subsection, we only present the algorithm and do a complexity analysis. The study of its correctness, and in particular checking that our number of evaluation points is sufficient for recovering a matrix fraction is done in the next subsections.

Polynomial matrix algorithms. The Villard algorithm makes use of the following procedures:

- The FractionReconstruction procedure is an algorithm that, given matrices $H_i = H(y_i), 1 \leq i \leq n_{\text{points}}$, and such that there exists matrices $\tilde{N}(Y), \tilde{D}(Y)$ such that $\deg(n_{ij}), \deg(d_{ij}) < n_{\text{points}}/2$ and $H(Y) = \tilde{D}(Y)^{-1}\tilde{N}(Y)$, computes matrices $N(Y), D(Y)$ such that $H(Y) = D(Y)^{-1}N(Y)$.

  When the points y_i are in geometric progression, it has been shown in [BBHN26, Lemma 2] that the fraction reconstruction can be performed in time

  $$\mathcal{O}(m^\omega n_{\text{points}} \log(n_{\text{points}}) + m^2 M(n_{\text{points}}) \log(n_{\text{points}})) \ .$$

  Moreover, the returned denominator function is in Popov form.

- The Polydet procedure computes the determinant of a polynomial matrix in Popov form whose coefficients have degree at most d. Using Theorem 1.2 from [NP21], this can be performed in time

  $$\mathcal{O}(m^\omega M(d) \log(d)) \ .$$

Complexity analysis. We first evaluate the complexity of the steps after the step where we evaluate the polynomials P and Q over n_{points} points y_i. The complexity of this first evaluation step will depend on whether we are in a setting where we were given two polynomials P, Q with all their coefficients, or if we are trying to attack a Poseidon instance. Thus, the analysis of this step is delayed to the next paragraph.

Algorithm 1 The modified version of the Villard algorithm

Require: $\deg(P) = \deg(Q) = \delta, \deg(\mathrm{Res}_X(P,Q)) = D_I$
Ensure: $R(Y) = \lambda \cdot \mathrm{Res}_X(P,Q)$ for some $\lambda \neq 0$
  $m = \delta^{1/\omega}$
  $n_{\text{points}} = 2\lceil D_I/m \rceil + 4\lceil \sqrt{2D_I} \rceil + 2m + 1$
  Evaluate $P(X, y_i), Q(X, y_i)$ for $1 \leq i \leq n_{\text{points}}$ ▷ geometric progression
  for $1 \leq i \leq n_{\text{points}}$ **do**
    $H_i = \Pi_{\text{bot}} S(P(X, y_i), Q(X, y_i))^{-1} \Pi_{\text{left}}$
  end for
  $N(Y), D(Y) = \mathsf{FractionReconstruction}(H_i, 1 \leq i \leq n_{\text{points}})$
  $R(Y) = \mathsf{Polydet}(D(Y))$
  return $R(Y)$

The bottom-left submatrix H_i of the Sylvester matrix $S(P(X, y_i), Q(X, y_i))$ can be computed similarly as in [Vil18], in time

$$\mathcal{O}(M(\delta)\log(\delta)) \ .$$

Doing it for all of the n_{points} evaluation points, this step has complexity

$$\mathcal{O}(n_{\text{points}} \cdot M(\delta)\log(\delta)) \ .$$

For the FractionReconstruction and PolyDet steps, we apply the results from the previous paragraph on a $m \times m$ matrix with n_{points} evaluation points for FractionReconstruction, and on a $m \times m$ matrix of degree at most $n_{\text{points}}/2$ for PolyDet, and get a complexity:

$$\mathcal{O}(m^\omega \cdot M(n_{\text{points}})\log(n_{\text{points}})) \ .$$

Putting everything together with $m = \delta^{1/\omega}, n_{\text{points}} = 2\lceil D_I/\delta^{1/\omega} \rceil + 4\lceil \sqrt{2D_I} \rceil + 2m + 1$, we have, as long as $D_I \geq \delta$, the computation of bottom-left submatrices H_i in time

$$\mathcal{O}(M(\delta)\log(\delta) n_{\text{points}}) = \mathcal{O}(M(\delta)\log(\delta) \cdot D_I/\delta^{1/\omega}) \ ,$$

and the FractionReconstruction and Polydet steps in time

$$\mathcal{O}(m^\omega M(n_{\text{points}})\log(n_{\text{points}})) = \mathcal{O}(\delta \cdot M(D_I/\delta^{1/\omega})\log(D_I/\delta^{1/\omega})) \ .$$

As $\delta \leq D_I \leq \delta^2$, both estimates amount to

$$\mathcal{O}\left(\delta^{1-1/\omega} M(D_I)\log(D_I)\right) \ . \tag{1}$$

Evaluating the $P(X, y_i), Q(X, y_i)$. If we are given two polynomials $P(X,Y), Q(X,Y)$ of degree δ, we evaluate them over n_{points} points in geometric progression as follows: for each coefficient of P and Q seen as polynomials in $(\mathbb{F}[Y])[X]$, we evaluate the coefficient $p_j(Y), q_j(Y)$ over the y_i using the multipoint evaluation procedure for points in geometric progression mentioned in Section 3.3.

This way, the complexity of that step becomes $\mathcal{O}(\delta \cdot \max(M(\delta), M(n_{\text{points}})))$, which becomes the bottleneck when $D_I \cdot \delta^{1-1/\omega} \ll \delta^2$. Having a time complexity at least δ^2 is of course expected as the input polynomials already have a number of coefficients growing with δ^2.

However, in the case of Poseidon, one does not need to generate the full polynomials P and Q: the evaluation of $P(X, y_i), Q(X, y_i)$ corresponds to evaluating the Poseidon permutation with one coordinate set to the symbolic variable X and another one set to y_i.

Such an evaluation can be performed in time $\mathcal{O}(M(\delta))$, and evaluating over n_{points} points takes time only

$$\mathcal{O}\left(n_{\text{points}} \cdot M(\delta)\right) = \mathcal{O}\left(M(\delta) \cdot D_I/\delta^{1/\omega}\right) . \tag{2}$$

This trick allows us to remove the δ^2 term for the complexity of attacks on Poseidon, and thus to get a complexity that is sub-quadratic in d^{R_P}.

This trick also improves memory complexity: for each evaluation point, we only have to generate two univariate polynomials of degree δ. Then, the main for loop requires storing n_{points} matrices of size $m \times m$. Thus, memory complexity of the algorithm is:

$$\mathcal{O}(n_{\text{points}} \cdot m^2) = \mathcal{O}(D_I \cdot \delta^{1/\omega}) . \tag{3}$$

4.2 Structure of the bottom-left submatrix

In this section, we study the structure of the bottom-left block of the matrix $S_X(P,Q)^{-1}$. In particular, we relate this submatrix to the polynomials $f_s \in I$ whose leading monomials lie in the bordering in Y of I, and to some u_s, v_s such that $f_s = u_s P + v_s Q$.

Definition 2 (Shifted Popov form [BLV99])**.** Let $P \in \mathbb{F}[X]^{n\times n}$ a polynomial matrix, $\mathbf{s} \in \mathbb{Z}^n$. We say that P is in $\mathbf{s}$-Popov form if the following properties are verified:

1. for all $1 \leq i < j \leq n$, $\deg(p_{ii}) + s_i > \deg(p_{ij}) + s_j$.
2. for all $1 \leq j \leq i \leq n$, $\deg(p_{ii}) + s_i \geq \deg(p_{ij}) + s_j$.
3. for all $i \neq j$, $\deg(p_{ii}) > \deg(p_{ji})$.
4. All p_{ii} have leading coefficient 1.

Moreover for a given $M \in \mathbb{F}[X]^{n\times n}$, there exists a matrix $U \in \mathbb{F}[X]^{n\times n}$ such that $\det(U) \in \mathbb{F} \setminus \{0\}$ and $U \cdot M$ is in $\mathbf{s}$-shifted Popov form.

In what follows, we study the properties of the shifted Popov form of $S_X(P,Q)$ with shifts $(2\delta, \ldots, 2, 1)$. In particular, we prove the following theorem that relates that shifted Popov form with the border of the ideal for the *grevlex* ordering.

Definition 3 (Initial segment and bordering)**.** Let I be an ideal, $\prec$ a monomial ordering, $\mathrm{LM}_\prec(P)$ the leading monomial of P with respect to that ordering. We define the initial segment of I as

$$\mathrm{in}_\prec(I) = \{t = X^i Y^j \mid \exists f \in I,\ \mathrm{LM}_\prec(f) = t\} .$$

Then, we define the bordering in Y of I as

$$\mathcal{F}_Y(I) = \{Y \cdot t \in \mathrm{in}_\prec(I) \mid t \notin \mathrm{in}_\prec(I)\} .$$

Theorem 3 (Structure of the shifted Popov form)**.** *Let I be 0-dimensional and the X-leading coefficients of P and Q coprime. Let $\mathcal{F}_Y(I)$ the bordering in Y of I for the* grevlex *ordering $\prec$ ($X \prec Y$). Then the Popov form of $S_X(P,Q)$ with shifts $(2\delta, \ldots, 2, 1)$ has the following structure:*

$$U(Y)S_X(P,Q) = \begin{pmatrix} I & H(Y) \\ 0 & F(Y) \end{pmatrix} ,$$

where the i-th row of $F(Y)$ represents a polynomial in I whose leading monomial is equal to some $X^{k-i}Y^j \in \mathcal{F}_Y(I)$, with k the number of rows of F.

Proof. We exploit a link between left multiplication of $S_X(P,Q)$ by a vector and elements in $I = \langle P, Q\rangle$. Let $(\tilde{u}(Y) \mid\mid \tilde{v}(Y)) \in \mathbb{F}^{2\delta}$, and $\tilde{r}(Y) = (\tilde{u}(Y) \mid\mid \tilde{v}(Y)) \cdot S_X(P,Q) \in \mathbb{F}^{2\delta}$. If we denote by $u(X,Y) = \sum_{i=0}^{\delta-1} \tilde{u}_{\delta-i}(Y)X^i, v(X,Y) = \sum_{i=0}^{\delta-1} \tilde{v}_{\delta-i}(Y)X^i, r(X,Y) = \sum_{i=0}^{\delta-1} \tilde{r}_{\delta-i}(Y)X^i$, then it is easy to see that:

$$r(X,Y) = u \cdot P + v \cdot Q \in I \ .$$

In particular, the rows of the Popov form of $S_X(P,Q)$ represent polynomials $r_i(X,Y) = u_i(X,Y)P + v_i(X,Y)Q$ with $\deg_X(u_i), \deg_X(v_i) \leq \delta - 1$.

Thus, we have that $r_i(X,Y) \in I$. Moreover, because of 1 and 2 from Definition 2, we have that $\mathrm{LM}_\prec(r_i) = X^{2\delta-i}Y^{\deg(r_i)+i-2\delta}$. Using Theorem 2, we get that the determinant of the Popov form has degree D_I. Thus:

$$\sum_{i=1}^{2\delta} (\deg(r_i) + i - 2\delta) = D_I \ .$$

Knowing that $r_i \in I$, we can consider the lowest j such that $X^{2\delta-i}Y^j \in \mathrm{in}_\prec(I)$ to get that either $X^{2\delta-i} \in \mathrm{in}_\prec(I)$, or there exists an element $f_i \in I$ such that $\mathrm{LM}_\prec(f_i) = X^{2\delta-i}Y^{\deg(f_i)+i-2\delta} \in \mathcal{F}_Y(I)$ and $\deg(f_i) \leq \deg(r_i)$. Moreover, computing the area under the Gröbner stairs, we get that:

$$\sum_{\substack{1\leq i\leq 2\delta \\ X^{2\delta-i}\notin \mathrm{in}_\prec(I)}} (\deg(f_i) + i - 2\delta) = D_I \ .$$

This implies that when $X^{2\delta-i} \notin \mathrm{in}_\prec(I)$, $\deg(f_i) = \deg(r_i)$, and when $X^{2\delta-i} \in \mathrm{in}_\prec(I)$, $\mathrm{LM}_\prec(r_i) = X^{2\delta-i}$. This gives exactly the expected structure for the Popov form. □

Remark 1. This result is very close to [Laz85, Theorem 4] except that we replace the *lex* ordering with the *grevlex* ordering and the Hermite normal form with the shifted-Popov form with shifts $(2\delta, \ldots, 1)$. In particular, our proof mostly followed the same steps as Lazard's proof.

Now we define the stability assumption and recall classical results on it:

Definition 4 (Stability)**.** Let I an ideal, and $\prec$ a *grevlex* ordering ($X \prec Y$). We say that I verifies *stability* if for all monomial t, $X \cdot t \in \mathrm{in}_\prec(I)$ implies that $Y \cdot t \in \mathrm{in}_\prec(I)$.

Lemma 1 (Size of the border $\mathcal{F}_Y(I)$)**.** *Under the stability of the ideal I, the size of the border $\mathcal{F}_Y(I)$ is lower bounded by $\sqrt{D_I/2}$.*

Proof. We make use of the fact that under the stability of I, the bordering of I has the following form:

$$\mathcal{F}_Y(I) = \left\{ Y^{s-i}X^j \;\middle|\; 0 \leq i < s, \sum_{k=0}^{i-1} w_k \leq j < \sum_{k=0}^{i} w_k \right\} \ ,$$

where all $w_i \geq 1$. For a proof of it, see Lemma 2 in Appendix B.

Using that result, the ideal degree D_I is the number of monomials strictly under the border, that is:

$$\sum_{i=0}^{s-1} (s-i) \cdot w_i \ .$$

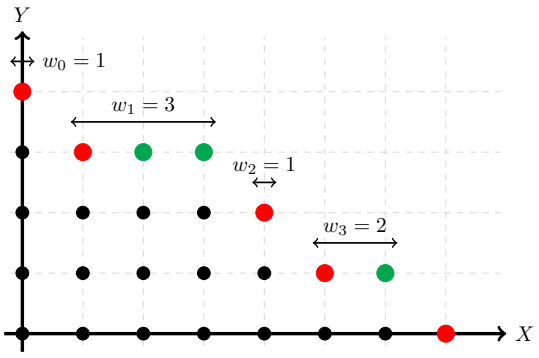


Figure 1: Example of stable stairs. Monomials in red represent leading monomials of elements in the Gröbner basis, monomials in green represent monomials that are in the bordering but not in the Gröbner basis.

Now we upper bound s as follows:

$$s^2/2 \leq \sum_{i=0}^{s-1}(s-i) \leq \sum_{i=0}^{s-1}(s-i)\cdot w_i = D_I \ ,$$

and thus

$$s \leq \sqrt{2D_I} \ .$$

Now the size of the border equals $\sum_{i=0}^{s-1} w_i$, which can be lower bounded as:

$$\sqrt{2D_I}\sum_{i=0}^{s-1} w_i \geq s \cdot \sum_{i=0}^{s-1} w_i \geq \sum_{i=0}^{s-1}(s-i)\cdot w_i = D_I \ ,$$

which yields $|\mathcal{F}_Y(I)| = \sum_{i=0}^{s-1} w_i \geq \sqrt{D_I/2}$.

□

A consequence of this result is that under the stability condition, as long as the blocking factor verifies $m < \sqrt{D_I/2}$, the bottom-left submatrix will be a submatrix of $F(Y)^{-1}U_{\text{bot}}(Y)$. Note that stability can always be guaranteed through a generic change of coordinate.

Theorem 4 (cf. Th. 15.20 [Eiz95] and Prop. 2.3 [MS05])**.** *Let* $\text{Char}(\mathbb{F}) > D_I$ *and* I *an ideal over* $\mathbb{F}[X,Y]$*. Then, there exists a Zariski-open subset* $U \subset \text{GL}_2(\overline{\mathbb{F}})$ *such that for all* $M \in U$*, the ideal*

$$M \cdot I = \{P(M\cdot(X,Y)) \mid P \in I\} \ ,$$

verifies the stability assumption.

We can now describe the structure of $S_X(P,Q)^{-1}$, and of its bottom-left submatrix. Let

$$U(Y) = \begin{pmatrix} U_{\text{top}}(Y) \\ U_{\text{bot}}(Y) \end{pmatrix} \ ,$$

with $U_{\text{top}} \in \mathbb{F}^{(2\delta - |\mathcal{F}_Y(I)|)\times 2\delta}, U_{\text{bot}} \in \mathbb{F}^{|\mathcal{F}_Y(I)|\times 2\delta}$. Then, we get that:

$$S_X(P,Q)^{-1} = \begin{pmatrix} I & H(Y) \\ 0 & F(Y) \end{pmatrix}^{-1} U(Y) = \begin{pmatrix} U_{\text{top}}(Y) - H(Y)F(Y)^{-1}U_{\text{bot}}(Y) \\ F(Y)^{-1}U_{\text{bot}}(Y) \end{pmatrix} \ .$$

Structure of U_{bot}. If we let $f_s(X,Y)$ be the polynomial in the border represented in the s-th row of $F(Y)$, we can interpret the s-th row of $U_{\text{bot}}(Y)$ as follows: writing it as $(\tilde{u}_s(Y) \mid\mid \tilde{v}_s(Y))$, where $\tilde{u}_s(Y), \tilde{v}_s(Y)$ are both length-δ vectors, we can interpret them as polynomials $u_s(X,Y), v_s(X,Y)$ such that:

$$f_s(X,Y) = u_s P + v_s Q \ .$$

Note that $\deg_X(u_s), \deg_X(v_s) \leq \delta - 1$, but no such restriction exists for their total degree.

Fraction description in the Villard algorithm. When we take the bottom-left block of size $m = \delta^{1/\omega}$ ($< \sqrt{D_I/2}$) of $S_X(P,Q)^{-1}$, we get the bottom-left block of $F(Y)^{-1}U_{\text{bot}}(Y)$, that is:

$$\Pi_{\text{bot}}^{(|\mathcal{F}_Y(I)|)} F(Y)^{-1} U_{\text{bot}} \Pi_{\text{left}} \ .$$

The study of this bottom-left projection will be the subject of the next two subsections.

4.3 Heuristics made

In this subsection, we use the results from previous subsections to analyze Algorithm 1, and make a few heuristics under which the algorithm returns the resultant up to a nonzero constant. While those heuristics stay unproven, even under genericity assumptions on P and Q, the success of our practical implementations suggest that those are valid.

The FractionReconstruction step. The FractionReconstruction step of Algorithm 1 amounts to a fraction reconstruction on the bottom-left projection of $F(Y)^{-1}U_{\text{bot}}(Y)$, that is a fraction reconstruction of

$$\Pi_{\text{bot}}^{(|\mathcal{F}_Y(I)|)} F(Y)^{-1} U_{\text{bot}}(Y) \Pi_{\text{left}} \ ,$$

with $F(Y), U_{\text{bot}}(Y)$ as defined in the previous subsection, and $\Pi_{\text{bot}}^{(|\mathcal{F}_Y(I)|)} = (0 \ \dots \ 0 \ I_m) \in \mathbb{F}^{m \times |\mathcal{F}_Y(I)|}$.

One way of studying it is to start with the matrix $\Pi_{\text{bot}}^{(|\mathcal{F}_Y(I)|)} F(Y)^{-1}$. The matrix $F(Y)$ is in Popov form, and has diagonal coefficients of degree > 0. Thus, its projection is a strictly proper fraction, and has a minimal fraction description $\Pi_{\text{bot}} F^{-1}(Y) = \tilde{D}(Y)^{-1}\tilde{N}(Y)$ such that the sum of the row degrees of $\tilde{D}(Y)$ and the sum of the row degrees of $\tilde{N}(Y)$ are both upper bounded by D_I. As stated here, the condition on the sum of degrees on the rows does not imply any condition on the maximum degree over all rows. For instance, one could have $m-1$ rows of degree 1 and 1 row of degree $D_I - m + 1$. Thus, we will need the following heuristic:

Heuristic 1. *There exists a fraction description of $\Pi_{\text{bot}} F^{-1}(Y) = \tilde{D}(Y)^{-1}\tilde{N}(Y)$ such that the coefficients of $\tilde{N}, \tilde{D}$ verify the following upper bounds:*

$$\deg(\tilde{n}_{ij}), \deg(\tilde{d}_{ij}) \leq D_I/m + \sqrt{2D_I} \ .$$

While this heuristic stays unproven in the case of a matrix F generated by taking the shifted Popov form of the Sylvester matrix, we were able to prove it for a generic matrix $F(Y)$ having the same degree profile as one that we would obtain for a system verifying *stability*. For more details, see Appendix C.1.

Then, we move on to the degree of $U_{\text{bot}}(Y)\Pi_{\text{left}}$. The degree of this matrix is related to the total degree of the u_s, v_s of degree in X upper bounded by $\delta - 1$, such that $f_s = u_s P + v_s Q$. Taking the m leftmost coefficients of $U_{\text{bot}}(Y)$, one gets the coefficients of degree $\delta - 1, \dots, \delta - m$ in X of u_s. The degree in Y of those coefficients is upper bounded by $\deg(u_s) - \delta + m$.

A fraction description of the bottom-left block can be written as

$$\Pi_{\text{bot}}^{(|\mathcal{F}_Y(I)|)} F(Y)^{-1} U_{\text{bot}}(Y) \Pi_{\text{left}} = \tilde{D}(Y)^{-1} \tilde{N}(Y) \cdot U_{\text{bot}}(Y) \Pi_{\text{left}} \ .$$

Under Heuristic 1, the degree of its denominator is at most $D_I/m + \sqrt{2D_I}$, and the degree of its numerator at most $D_I/m + \sqrt{2D_I} + (\deg(u_s) - \delta + m)$. Thus we can recover it using the FractionReconstruction procedure from

$$n_{\text{points}} = 2 \left\lceil D_I/m + \sqrt{2D_I} \right\rceil + 2(\deg(u_s) - \delta + m) + 1$$

points. The study of the degree of u_s is delayed to the next subsection, where we will conjecture that $\deg(u_s) \leq \delta - 1 + \sqrt{2D_I}$. This additional conjecture (Conjecture 2) justifies the choice of n_{points} made in Section 4.1.

Remark 2. We observe that contrary to the original Villard algorithm, the choice of the submatrix matters. For instance, taking the bottom-right block instead of the bottom-left block would lead to consider the matrix $U_{\text{bot}}(Y)\Pi_{\text{right}}$, which gives the terms of degree $0, \ldots, m-1$ in X of v_s. As those terms are expected to have degree $\deg(v_s)$ in Y, with $\deg(v_s) \sim \delta$, the fraction would have a numerator with degree bigger than δ. Thus, we would get a complexity growing at least with δ^2.

This is in contrast to our algorithm that has a complexity that grows only with $D_I \delta^{1-1/\omega}$, which may be less than δ^2 when D_I is small.

The PolyDet step. Using Theorem 3 and the fact that $\det(U(Y)) \in \mathbb{F} \setminus \{0\}$, we have that

$$\det(F(Y)) = \lambda \cdot \text{Res}_X(P, Q) \ , \ \lambda \in \mathbb{F} \setminus \{0\} \ .$$

However, the PolyDet step does not compute the determinant of $F(Y)$, but the determinant of the denominator $D(Y)$ of the bottom-left projection $\Pi_{\text{bot}}^{(|\mathcal{F}_Y(I)|)} F(Y)^{-1} U_{\text{bot}}(Y) \Pi_{\text{left}}$. Using the results from [Vil18, Section 2], we know that any irreducible fraction descriptions of projections of the form $\Pi_L M^{-1}(X) \Pi_R$ has the determinant of its denominator that divides $\det(M(X))$. Applied to the fraction description

$$D^{-1}(Y) N(Y) = \Pi_{\text{bot}} S_X(P, Q)^{-1} \Pi_{\text{left}} \ ,$$

we get that as long as $D^{-1}(Y)N(Y)$ is irreducible, $\det(D(Y))$ divides $\text{Res}_X(P, Q)$.

It seems challenging to prove that $\det(D(Y))$ actually equals $\text{Res}_X(P, Q)$ up to a nonzero constant. Thus, we make the following heuristic:

Heuristic 2. *Let an irreducible fraction description*

$$D^{-1}(Y) N(Y) = \Pi_{\text{bot}} S_X(P, Q)^{-1} \Pi_{\text{left}} \ .$$

We assume that $\det(D(Y)) = \lambda \cdot \text{Res}_X(P, Q)$ *for some constant* $\lambda \in \mathbb{F} \setminus \{0\}$.

This heuristic is justified for a generic $U_{\text{bot}}(Y)$ in Appendix C.2.

4.4 Experimental study of the degree of $U_{\text{bot}}\Pi_{\text{left}}$

We study the degree of the coefficients of the U_{bot} matrix. For this purpose, we provide an efficient way of recovering the leftmost coefficients of this matrix, as long as the Gröbner basis of the ideal has the right shape. This assumption on the shape of the ideal can be tested, and has been observed to hold as long as the number of partial rounds is large enough. This allowed us to recover the degree of $U_{\text{bot}}\Pi_{\text{left}}$ for Poseidon instances over a wide range of parameters. Those experiments lead us to make the following conjecture:

Conjecture 2 (Degree of $U_{\text{bot}}\Pi_{\text{left}}$)**.** *We conjecture that for $i \leq \lfloor \sqrt{2D_I} \rceil$ the degree of the i-th column of the U_{bot} is upper bounded by:*

$$i - 1 + g(d, R_{f1}, R_{f2}, R_P)_+ \ ,$$

where

$$g(d, R_{f1}, R_{f2}, R_P) = \left\lfloor \sqrt{2D_I} \right\rfloor - 2 \cdot d^{R_F} - (d-1)R_P \cdot d^{R_{f1}} + 1 \ .$$

Vanilla Gröbner basis. In the following, we define the Vanilla Gröbner stairs, and evaluate the size of the matrix $F(Y)$ when the ideal I has such Gröbner basis.

Definition 5 (Vanilla Gröbner stairs [VDHL18])**.** Let $I = \langle P, Q \rangle \subset \mathbb{F}[X, Y]$ of ideal degree D_I. We say that I has *vanilla Gröbner stairs* if the D_I monomials in the canonical basis of $R = \mathbb{F}[X, Y]/I$ correspond to the D_I lowest monomials with respect to the *grevlex* ordering.

It has been shown in [VDHL18, Proposition 3] that when I has vanilla Gröbner stairs, the leading monomials of the Gröbner basis elements are of the following form:

- $X^i Y^{n-i}, 0 \leq i < r$,
- $X^{i+1} Y^{n-i}, r \leq i \leq n$.

where $r \leq n$ are such that $n(n+1)/2 + n - r + 1 = D_I$. Note that $n \sim \sqrt{2D_I}$. In particular, the border of the ideal has the leading monomials of the elements of the reduced Gröbner basis, along with an additional monomial $X^r Y^{n-r}$. The number of rows of $F(Y)$ corresponds to the number of elements in the reduced Gröbner basis, plus the additional polynomial corresponding to the monomial $X^r Y^{n-r}$, minus the polynomial in the Gröbner basis with leading monomial X^{n+1}. Finally, $F(Y)$ has exactly $n + 1 \simeq \sqrt{2 \cdot D_I}$ rows.

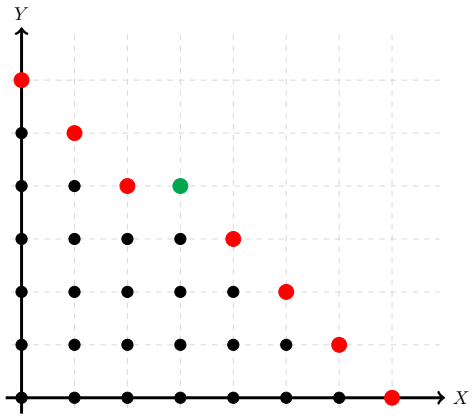


Figure 2: Vanilla stairs when $D_I = 25$ (Gröbner basis elements in red, additional element in the border in green).

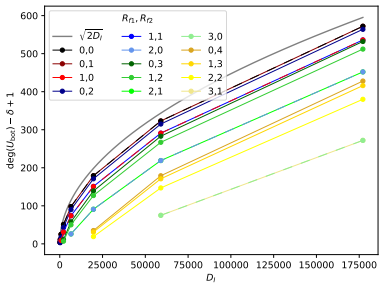
Computing the degree of $U_{\text{bot}}\Pi_{\text{left}}$ using the submatrix computation. To compute the degree of U_{bot} for large values of R_F, R_P, we exploit the efficient algorithms allowing to compute the bottom-left block of S^{-1}: more precisely, taking a block of a size corresponding to the number of rows $n + 1 \simeq \sqrt{2 \cdot D_I}$ of $F(Y)$, we get the exact block:

$$F(Y)^{-1} U_{\text{bot}}(Y) \Pi_{\text{left}} \ .$$

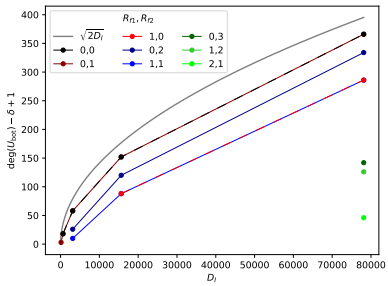
evaluated over some number of points. Using the matrix fraction reconstruction techniques, with the shifts as before, we can recover both $F(Y)$ and $U_{\text{bot}}(Y)\Pi_{\text{left}}$. This procedure takes time

$$\begin{aligned}&\tilde{\mathcal{O}}\left(n^{\omega}\cdot\left(D_I/n+\deg(U_{\text{bot}}(Y)\Pi_{\text{left}})\right)\right)\\=&\tilde{\mathcal{O}}\left(D_I^{(\omega+1)/2}+D_I^{\omega/2}\deg(U_{\text{bot}}(Y)\Pi_{\text{left}})\right)\ .\end{aligned}$$

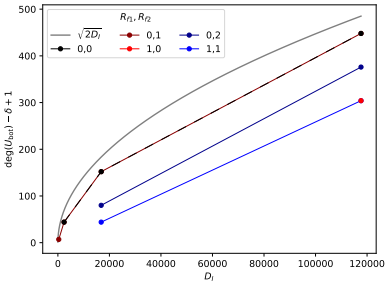
Results of the experiments. We made experiments for $d=3$ for all values of R_{f1}, R_{f2}, R_P such that $2(R_{f1}+R_{f2})+R_P \leq 11$, for $d=5$, R_{f1}, R_{f2}, R_P such that $2(R_{f1}+R_{f2})+R_P \leq 7$, and for $d=7$, R_{f1}, R_{f2}, R_P such that $2(R_{f1}+R_{f2})+R_P \leq 6$. We found that the degrees of the leftmost columns of U_{bot} matched exactly the bound from Conjecture 2 when the Vanilla stair assumption was verified. That Vanilla stair assumption was observed to hold as long as the number of partial rounds was high enough. Our results are plotted in Figure 3.



(a) Experiment for $d=3$.



(b) Experiment for $d=5$.



(c) Experiment for $d=7$.

Figure 3: Experiments on the maximum of the degree of the i-th column of U_{bot} minus $i-1$ for various sets of parameters.

5 Applications

In this section, we apply our attack on actual instances of Poseidon both in theory and in practice[5]. In Section 5.1, we evaluate the theoretical complexity of our attack on full

[5]Our code can be found at github.com/AntoineBak/PoseidonCicoTwo

instances of Poseidon, and determine the maximal number of rounds we are able to break. We also compare our attack to the previous ones in the literature. In Section 5.2, we implement our attack in a single-thread setting, to validate its correctness on reduced versions, and compare its practical timings to the evaluation-interpolation and the Gröbner basis attack. In Section 5.3, we study its multi-threaded performances and report the timings we obtained while taking part in the Ethereum bounty program.

5.1 Asymptotic comparison of methods

In this section, we compare our CICO-2 attack to the other approaches mentioned in the introduction, namely Gröbner bases, the evaluation-interpolation method for bivariate resultant computation (Eval.-Inter.), the Guess & Solve and the Guess & GCD approach.

Gröbner basis attack. For the Gröbner basis attacks, it is known from [BBB+25] that a Gröbner basis of a Poseidon CICO-k system can be computed very efficiently using some weighted graded lexicographical ordering. Thus, we focus on the change of ordering step. Here, we estimate the cost of the change of ordering as

$$\mathcal{O}(D_I^\omega) \ , \tag{4}$$

where ω is the exponent such that two $n \times n$ matrices can be multiplied in time $\mathcal{O}(n^\omega)$. This estimate corresponds to using an FGLM-like algorithm [FGHR14] based on dense linear algebra, assuming that the matrix corresponding to multiplication by some variable X_i modulo I can be computed for free. The authors of [BBB+25] also provide an estimate for computing the matrix of multiplication by X_i modulo I. Applying their formula to Poseidon, we would get $\mathcal{O}((k(d^{R_F}-1)+R_P(d^{R_{f,1}}-1))^{3-\omega} \cdot (k+R_P)^{\omega-1} \cdot D_I^\omega)$. Note that this estimate is always higher than D_I^ω. Assuming that this matrix can be computed for free, the authors of [BBB+25] also provide a more optimistic estimate from FGLM-like algorithms based on Hermite normal form computation [BNSED22]. Applied to Poseidon, this estimate gives $\tilde{\mathcal{O}}(D_I^\omega / d^{(\omega-1)R_F})$. As the assumption that the multiplication matrix can be computed for free is already quite optimistic from an attacker point of view, we think that (4) is a fair estimate for the complexity of Gröbner basis attacks.

Evaluation-interpolation attack. As established in [BBB+26], the bivariate resultant computation based on evaluation-interpolation has time complexity

$$\mathcal{O}(D_I \cdot M(\delta)\log(\delta)) \ .$$

Guess & Solve/GCD attacks. For the Guess & Solve approach [SV26], we make $\mathcal{O}(p)$ calls to a univariate solving method on polynomials of degree δ, leading to a time complexity of

$$\mathcal{O}(p \cdot M(\delta)\log(p \cdot \delta)) \ .$$

The Guess & GCD approach [BBB+26] makes $\mathcal{O}(p)$ calls to a GCD algorithm on polynomials of degree δ, leading to a time complexity in

$$\mathcal{O}(p \cdot M(\delta)\log(\delta)) \ .$$

As this estimate is better than the Guess & Solve one as p grows, we will use it in our complexity estimates.

Setting of the comparison. The value of the exponent for matrix multiplication appears both in our estimates and the estimates for Gröbner basis attacks. While the best algorithms achieve values of ω as low as $\omega < 2.38$ [CW87], this asymptotic complexity is known to hide a prohibitively large constant factor. Thus, we use the more realistic value $\omega = 2.81$ that corresponds to Strassen's algorithm [Str69] for Gröbner basis estimates and our estimates.

The comparison is made on a CICO-2 attack on an 8-branch instance over some prime field of size $\sim 2^{\lambda/2}$, where λ is the intended security level. In particular, a CICO-2 attack in time less than 2^{λ} leads to a nontrivial preimage attack on a 8-to-2 compression function built from the permutation. The number of full rounds is $R_F = 8$, and we use classical tricks to skip the first round, leading to $D_I = d^{14+R_P}, \delta = d^{7+R_P}$ (see [BBB+25, Appendix C] for more details on round skipping tricks).

Observations. We observe that for all sets of parameters, the Villard bivariate resultant algorithm breaks the highest number of rounds, outperforming for instance the best previous attack by 7 rounds on the 128-bit version with $d = 3$. We emphasize that we have not found any attack on full versions with realistic security levels (less than 256 bits) and $\omega = 2.81$. On a 384-bit version with $p \sim 2^{192}, d = 3$, we manage to solve the CICO-2 problem on a full version of Poseidon with 2^{381} field operations, whereas the other methods exceed the brute force complexity. If we use $\omega = 2$, which corresponds to a conservative choice from a designer's point of view, we start having attacks on instances with 256 bits of security. For instance, the CICO-2 problem on a permutation with 256 bits of security and S-box degree $d = 3$ can be solved in time 2^{245} when we plug $\omega = 2$ in the complexity formula. We emphasize that using $\omega = 2$ is only a conservative estimate from a designer's point of view, and does not constitute a realistic estimate.

Table 2: Complexity of attacks on the CICO-2 problem on a Poseidon2 instance with $R_F = 8, t = 8$, and p such that $2 \cdot \log_2(p)$ equals the security level. The first full round is skipped in all attacks. For each attack, the first row provides the complexity of an attack on the full round instance (with R_P in parenthesis), and the second row provides the highest R_P broken by the attack.

Security	128-bit			256-bit		
d	3	5	7	3	5	7
Gröbner	244 (41)	267 (27)	283 (22)	436 (84)	463 (57)	504 (50)
[BBB+25]	124 (14)	123 (5)	126 (2)	253 (43)	254 (25)	252 (18)
Eval.-Inter.	178 (41)	189 (27)	197 (22)	316 (84)	330 (57)	357 (50)
[BBB+26]	126 (25)	127 (14)	122 (9)	255 (65)	255 (41)	254 (32)
Guess & Solve/GCD	155 (41)	158 (27)	160 (22)	289 (84)	293 (57)	305 (50)
[BBB+26, SV26]	126 (24)	126 (14)	125 (10)	255 (63)	255 (41)	253 (32)
This work	151 (41)	162 (27)	169 (22)	265 (84)	278 (57)	300 (50)
	127 (32)	126 (18)	127 (13)	255 (80)	255 (51)	254 (40)

5.2 Practical attacks on reduced versions

In this section, we describe our practical experiments on reduced version of the Poseidon hash function. We compare the time and memory complexity of the modified Villard algorithm against the evaluation-interpolation and the Gröbner basis attacks. We did not implement the Guess & GCD and Guess & Solve approaches due to their linear dependence on p, which makes them not competitive in the single thread setting.

Choice of parameters. As we are particularly interested in the setting where $D_I \ll \delta^2$, thus $R_F \ll R_P$, we chose to fix the number of full rounds to $R_F = 2$, and to let the number of partial rounds R_P grow.

Then, we set the prime field, number of branches and exponent to $p = 2^{31} - 2^{24} + 1, t = 4, d = 3$. Note that $t = 4$ is the smallest t for which we have enough degrees of freedom to conduct a CICO-2 attack, and that such a parameter allows no round-skipping trick.

Experimental setting. The attacks were run on a Cascade Lake Intel Xeon 5218, 16 cores, 2.4GHz with 128GB of RAM. All experiments are run on a single-thread. Our implementation of the evaluation-interpolation and of the Villard algorithm are based on the PML library [tea25], and the Gröbner basis algorithms use the `Variety` function from Magma [BCP97]. Note that in practice, the bottleneck for the Gröbner basis attack comes from the change-of-ordering step, which is performed using the FGLM algorithm [FGLM93].

Table 3: Comparison of different methods for bivariate solving (Case $p = 2^{31} - 2^{24} + 1, t = 4, d = 3, R_{f1} = R_{f2} = 1$). T is the practical runtime, M is the memory consumption in Gigabytes, Th. is the theoretical time complexity according to the estimates from previous subsection.

R_P	GB			Eval.-Inter.			Villard		
	T	M (Gb)	Th.	T	M (Gb)	Th.	T	M (Gb)	Th.
4	1d3h	4.6	35	13s	0.006	30	11s	0.1	27
5		-	40	3m39s	0.01	34	1m14s	0.46	31
6		-	44	43m14s	0.017	37	7m25s	2.4	34
7		-	48	8h05m	0.046	41	51m03s	10.8	37
8		-	53	2d7h	0.22	44	5h18m	50	39

Observations. As expected, the Gröbner basis is the least competitive attack both in terms of time and memory complexity, the highest number of rounds we were able to break being $R_F = 2, R_P = 4$. This is consistent with the fact that its time complexity grows with D_I^ω and memory complexity with D_I^2.

For the resultant algorithms, the Villard algorithm outperforms the time complexity of evaluation-interpolation as long as $R_P \geq 4$, and the gap between the two grows quickly with the number of rounds, which is consistent with the better theoretical time complexity of the Villard algorithm. When $R_P = 8$, the Villard algorithm is roughly 7 times faster than evaluation-interpolation, while theoretical complexity predicts it to be $\sim 2^5 = 32$ times faster. Thus, there seems to be a constant overhead between our theoretical complexity formulas and the practical performances of the Villard algorithm. From our data, this overhead seems to be a constant close to 4 or 5.

For memory complexity, we observe that the better time complexity of the Villard algorithm comes at the cost of a much higher memory consumption than evaluation-interpolation. This memory consumption becomes the bottleneck in practice, as Villard takes 50 Gigabytes of memory when $R_P = 8$. This is again consistent with the formulas for the memory complexity of these algorithms.

5.3 Performances on Ethereum bounty program

In April 2026, the Ethereum foundation issued new cryptanalysis challenges for Poseidon. In this work, we focus on the CICO-2 and ZeroTest challenges. It has been shown in [SV26] that the ZeroTest challenges can be solved through a CICO-2 solution for the permutation. Both challenges are on Poseidon1, but the CICO-2 challenge adds an initial linear layer, whereas the ZeroTest challenge does not.

Parallelization of the Villard algorithm. Algorithm 1 features two for loops that can be parallelized very efficiently: the evaluation of the Poseidon permutation on inputs X, y_i and the computation of the bottom-left blocks of the inverse of the Sylvester matrices. If the attacker has τ threads, the time taken to perform those two loops grows in $\tilde{\mathcal{O}}(\delta D_I/(m\tau))$. On the other hand, parallelizing the FractionReconstruction and Polydet steps seems more challenging. If one does not parallelize those steps at all, their cost is still $\tilde{\mathcal{O}}(m^{\omega-1}D_I)$. Taking $m = (\delta/\tau)^{1/\omega}$ yields a time complexity:

$$\tilde{\mathcal{O}}\left(D_I\left(\frac{\delta}{\tau}\right)^{1-1/\omega}\right).$$

In that setting, the speedup with τ threads is only by a factor $\tau^{1-1/\omega}$. This is in contrast with the evaluation-interpolation, Guess & GCD [BBB+26] and the Guess & Solve [SV26] attacks, where we expect an improvement by a factor τ. To get a factor τ improvement for the Villard algorithm, one would need to get parallel versions of the FractionReconstruction and Polydet steps. This can for instance be achieved through parallel versions of the matrix multiplication algorithms: if two matrices of size m can be multiplied in time $\mathcal{O}(m^3/\tau)$, a parallelized version of Algorithm 1 can achieve time complexity $\tilde{\mathcal{O}}(D_I\delta^{2/3}/\tau)$.

Skipping rounds and adapting degree annihilation trick. The CICO-2 challenges and the ZeroTest challenge with $R_P = 8$ were made before the publication of the degree annihilation technique [SV26]. Thus, we only used the round-skipping tricks described in [BBB+25] to skip the first full round for the CICO-2 instances, and the first two full rounds for the ZeroTest challenges (due to the lack of an initial linear layer). After the publication of [SV26], we adapted their technique to our needs and skipped one partial round in the ZeroTest challenges with 9 and 10 partial rounds. Details of our version of the degree annihilation technique can be found in Appendix A.

Experimental setup. All attacks except the ZeroTest with $R_P = 10$ were performed on a single AMD EPYC 9354 processor (with 128 CPUs). The ZeroTest with $R_P = 10$ was performed in two phases: the main for loops were computed on a cluster with 552 cores, and the FractionReconstruction and PolyDet steps were performed on a H100 GPU to benefit from parallelized versions of polynomial matrix multiplications. We used the NTL [Sho] and the PML [tea25] libraries for handling polynomial matrices. The GPU optimized polynomial matrix multiplication is implemented in `CUDA`, and is based on GPU-NTT [cOS23].

Comparison to other works. We observe that the performances of our attack are close to the one of [SV26] for the ZeroTest challenges, whereas they are far worse in the case of the CICO-2 problem. This can be explained by 1. the fact the authors of [SV26] had a much bigger computational power for the CICO-2 problem than for ZeroTests (96 vs 4 or 28 GPUs), and 2. the fact that we can only skip one full round in the CICO-2 case. Indeed, our attack complexity grows in $d^{2.66R_F}$ with the number of full rounds, whereas theirs grows in d^{R_F}. One of the main practical limits of our attack compared to [SV26] is the lesser GPU-friendliness of our algorithm, that prevented us from using GPU clusters instead of CPUs. We also mention that the Ethereum foundation bounties website[6] reports CICO-2 attacks on up to 10 rounds, though the details of the attack are not public yet.

[6] www.poseidon-initiative.info

Table 4: Comparison of our work with [SV26] on the Ethereum bounties. The number r_F (resp. r_P) corresponds to the number of skipped full (resp. partial) rounds in the Poseidon instance.

Problem	R_F (r_F)	R_P (r_P)	Comp. budget	Time	Source
CICO-2	6 (1)	6 (0)	128 CPU cores	1d17h	This work
	6 (1)	7 (0)	128 CPU cores	21d5h	This work
	6 (1)	8 (0)	96 GPUs	7h	[SV26]
	6	10	No public information yet		
ZeroTest	6 (2)	7 (2)	4 GPUs	10h27m	[SV26]
	6 (2)	8 (2)	4 GPUs	2d2h	[SV26]
	6 (2)	8 (0)	128 CPU cores	3d5h	This work
	6 (2)	9 (1)	128 CPU cores	3d13h	This work
	6 (2)	10 (2)	28 GPUs	$\approx$ 2d16h†	[SV26]
	6 (2)	10 (1)	552 CPU cores + 1 GPU	$\approx$ 5d + 9h30m*	This work
	6 (2)	11 (2)	28 GPUs	$\approx$ 8d†	[SV26]
	6 (2)	12 (2)	28 GPUs	$\approx$ 24d†	[SV26]

*This experiment took place over a unhomogeneous cluster with preemptable tasks, the timing for the first part of the attack is therefore approximate.
†Time estimates are approximate as the authors lost the logs of the attacks.

6 Conclusion

Our attack improves the complexity of solving CICO-2 problems on Poseidon, both theoretically and practically. Even if our attack did not break any full-round version of Poseidon, the asymptotic complexity formula we get for solving CICO-2 problem disproves some security arguments of the designers. Indeed, the most recent variants of Poseidon (Poseidon(2)b [GKK+26]) base their security arguments against CICO-2 attacks on the fact that bivariate resultant attack needs at least to read the polynomials P, Q that are of size $d^{2(R_F+R_P)}$. Thus, they provide the following formula as a conservative lower bound on bivariate resultant attacks (assuming the first round can be skipped):

$$C_{\texttt{res}}(d, R_F, R_P) = (d^{R_F+R_P-1})^2 \ .$$

Our analysis of the Villard algorithm, with conservative assumptions from a designer's point of view ($\omega = 2$, neglecting the log factors, 1 full round skipped), would lead to a new bound, that is better when $R_P > R_F - 1$:

$$C_{\texttt{res2}}(d, R_F, R_P) = d^{2.5(R_F-1)+1.5R_P} \ .$$

Open problems. The analysis of our algorithm rests on unproven heuristics. We were able to verify them experimentally, but we would welcome further research on their proof. Indeed, we think that the justifications for random matrices and the success of the practical implementations give convincing arguments for the soundness of our approach. We leave as an open problem a formal proof of the heuristics made in Section 4.3 and of Conjecture 2.

Acknowledgment

The authors thank the reviewers for their useful comments, and the shepherd for reading the final version of the paper. The authors also thank Vincent Neiger and Léo Perrin for insightful discussions in the early stages of this work. The authors thank Télécom SudParis and the CLEPS infrastructure from the Inria of Paris for the computational resources provided for the experiments. The work was supported by the European Research Council (ERC) through the grant agreement no. 101041545 "ReSCALE", and by the French DGA.

References

[ABM23] Tomer Ashur, Thomas Buschman, and Mohammad Mahzoun. Algebraic cryptanalysis of HADES design strategy: Application to POSEIDON and poseidon2. Cryptology ePrint Archive, Paper 2023/537, 2023. URL: https://eprint.iacr.org/2023/537.

[AGP+19] Martin R Albrecht, Lorenzo Grassi, Léo Perrin, Sebastian Ramacher, Christian Rechberger, Dragos Rotaru, Arnab Roy, and Markus Schofnegger. Feistel structures for mpc, and more. In *European symposium on research in computer security*, pages 151–171. Springer, 2019.

[BBB+25] Antoine Bak, Augustin Bariant, Aurélien Boeuf, Pierre Briaud, Morten Øygarden, and Atharva Phanse. The algebraic CheapLunch: Extending FreeLunch attacks on arithmetization-oriented primitives beyond CICO-1. Cryptology ePrint Archive, Paper 2025/2040, 2025. URL: https://eprint.iacr.org/2025/2040.

[BBB+26] Antoine Bak, Augustin Bariant, Aurélien Boeuf, Maël Hostettler, and Guilhem Jazeron. Claiming bounties on small scale poseidon and poseidon2 instances using resultant-based algebraic attacks. Cryptology ePrint Archive, Paper 2026/150, 2026. URL: https://eprint.iacr.org/2026/150.

[BBHN26] Antoine Bak, Augustin Bariant, Maël Hostettler, and Vincent Neiger. Resultants meet resultant: Improving CICO-1 and CICO-2 attacks on ZK-friendly permutations. Cryptology ePrint Archive, Paper 2026/1281, 2026. URL: https://eprint.iacr.org/2026/1281.

[BBLP22] Augustin Bariant, Clémence Bouvier, Gaëtan Leurent, and Léo Perrin. Algebraic attacks against some arithmetization-oriented primitives. *IACR Transactions on Symmetric Cryptology*, 2022(3):73–101, Sep. 2022. URL: https://tosc.iacr.org/index.php/ToSC/article/view/9850, doi:10.46586/tosc.v2022.i3.73-101.

[BCP97] Wieb Bosma, John Cannon, and Catherine Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997. Computational algebra and number theory (London, 1993). URL: http://dx.doi.org/10.1006/jsco.1996.0125, doi:10.1006/jsco.1996.0125.

[BKM05] Laurent Busé, Houssam Khalil, and Bernard Mourrain. Resultant-based methods for plane curves intersection problems. In *International workshop on computer algebra in scientific computing*, pages 75–92. Springer, 2005.

[BLV99] Bernhard Beckermann, George Labahn, and Gilles Villard. Shifted normal forms of polynomial matrices. In *Proceedings of the 1999 international symposium on Symbolic and algebraic computation*, pages 189–196, 1999.

[BNSED22] Jérémy Berthomieu, Vincent Neiger, and Mohab Safey El Din. Faster change of order algorithm for gröbner bases under shape and stability assumptions. In *Proceedings of the 2022 International Symposium on Symbolic and Algebraic Computation*, pages 409–418, 2022.

[BS05] Alin Bostan and Éric Schost. Polynomial evaluation and interpolation on special sets of points. *Journal of Complexity*, 21(4):420–446, 2005.

[CLOL07] David A Cox, John B Little, Donal O'shea, and John Little. *Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra*, volume 10. Springer, 2007. Chapter 3, §5, Proposition 8.

[cOS23] Ali Şah Özcan and Erkay Savaş. Two algorithms for fast GPU implementation of NTT. Cryptology ePrint Archive, Paper 2023/1410, 2023. URL: https://eprint.iacr.org/2023/1410.

[CW87] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 1–6, 1987.

[Eiz95] D Eizenbud. Commutative algebra with a view towards algebraic geometry. *Graduate Texts in Math*, 150, 1995.

[FGHR14] Jean-Charles Faugère, Pierrick Gaudry, Louise Huot, and Guénaël Renault. Sub-cubic change of ordering for gröbner basis: a probabilistic approach. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, pages 170–177, 2014.

[FGLM93] J.C. Faugère, P. Gianni, D. Lazard, and T. Mora. Efficient computation of zero-dimensional gröbner bases by change of ordering. *Journal of Symbolic Computation*, 16(4):329–344, 1993. URL: https://www.sciencedirect.com/science/article/pii/S0747717183710515, doi:10.1006/jsco.1993.1051.

[GKK+26] Lorenzo Grassi, Dmitry Khovratovich, Katharina Koschatko, Christian Rechberger, Markus Schofnegger, Verena Schröppel, and Zhuo Wu. Poseidon (2) b. *IACR Communications in Cryptology*, 2(4), 2026.

[GKR+21] Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger. Poseidon: A new hash function for Zero-Knowledge proof systems. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 519–535, 2021.

[GKS23] Lorenzo Grassi, Dmitry Khovratovich, and Markus Schofnegger. Poseidon2: A faster version of the poseidon hash function. In *International Conference on Cryptology in Africa*, pages 177–203. Springer, 2023.

[GLR+20] Lorenzo Grassi, Reinhard Lüftenegger, Christian Rechberger, Dragos Rotaru, and Markus Schofnegger. On a generalization of substitution-permutation networks: The hades design strategy. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 674–704. Springer, 2020.

[GOPS22] Lorenzo Grassi, Silvia Onofri, Marco Pedicini, and Luca Sozzi. Invertible quadratic non-linear layers for mpc-/fhe-/zk-friendly schemes over fnp: application to poseidon. *IACR Transactions on Symmetric Cryptology*, pages 20–72, 2022.

[Jo26] Sunghyeon Jo. Midpoint reset: A full-round poseidon collision from an adaptively chosen MDS matrix. Cryptology ePrint Archive, Paper 2026/1760, 2026. URL: https://eprint.iacr.org/2026/1760.

[KU08] Kiran S Kedlaya and Christopher Umans. Fast modular composition in any characteristic. In *FOCS*, volume 8, pages 146–155, 2008.

[Laz85] Daniel Lazard. Ideal bases and primary decomposition: case of two variables. *Journal of Symbolic Computation*, 1(3):261–270, 1985.

[MG26] Simon-Philipp Merz and Àlex Rodríguez García. Skipping class: Algebraic attacks exploiting weak matrices and operation modes of poseidon2(b). Cryptology ePrint Archive, Paper 2026/306, 2026. URL: https://eprint.iacr.org/2026/306.

[MS05] Ezra Miller and Bernd Sturmfels. *Combinatorial commutative algebra*. Springer, 2005.

[NP21] Vincent Neiger and Clément Pernet. Deterministic computation of the characteristic polynomial in the time of matrix multiplication. *Journal of Complexity*, 67:101572, 2021.

[Roy26] Arnab Roy. When polynomials iterate: Structure and security in Symmetric Cryptography, 2026. Workshop on ZK, Succinct Proofs and Symmetric Cryptography. URL: https://zksc2026.secpriv.wien/static/talks/slides-arnab-roy.pdf.

[Sho] Shoup, Victor et al. NTL: A Library for Doing Number Theory. https://libntl.org/.

[Ste24] Matthias Johann Steiner. Solving degree bounds for iterated polynomial systems. *IACR Transactions on Symmetric Cryptology*, 2024(1):357–411, Mar. 2024. URL: https://tosc.iacr.org/index.php/ToSC/article/view/11410, doi:10.46586/tosc.v2024.i1.357-411.

[Str69] Volker Strassen. Gaussian elimination is not optimal. *Numer. math*, 13:354–356, 1969.

[SV26] Antonio Sanso and Giuseppe Vitto. Top gun: Degree annihilation attacks on poseidon. Cryptology ePrint Archive, Paper 2026/1254, 2026. URL: https://eprint.iacr.org/2026/1254.

[tea25] The PML team. *PML: Polynomial Matrix Library*, 2025. Version 0.5, https://github.com/vneiger/pml.

[VDHL18] Joris Van Der Hoeven and Robin Larrieu. Fast reduction of bivariate polynomials with respect to sufficiently regular gröbner bases. In *Proceedings of the 2018 acm international symposium on symbolic and algebraic computation*, pages 199–206, 2018.

[VDHL21] Joris Van Der Hoeven and Grégoire Lecerf. Fast computation of generic bivariate resultants. *Journal of Complexity*, 62:101499, 2021.

[Vil18] Gilles Villard. On computing the resultant of generic bivariate polynomials. In *Proceedings of the 2018 acm international symposium on symbolic and algebraic computation*, pages 391–398, 2018.

[Vit26] Giuseppe Vitto. Slipway: Accessing finite subspace trails in poseidon. Cryptology ePrint Archive, Paper 2026/1579, 2026. URL: https://eprint.iacr.org/2026/1579.

[ZSVD25] Ziyu Zhao, Antonio Sanso, Giuseppe Vitto, and Jintai Ding. Graeffe-based attacks on poseidon and NTT lower bounds. Cryptology ePrint Archive, Paper 2025/1916, 2025. URL: https://eprint.iacr.org/2025/1916.

A Adapting the degree annihilation technique

We provide here an adaptation of the degree annihilation technique [SV26] to our bivariate solving based setting. The original degree annihilation technique builds on the round-skipping tricks from [BBLP22, BBB+25] and exploits some additional degrees of freedom to reduce the degree of the inputs of the S-boxes of the first few partial rounds. This technique was initially described for the Zero-test challenge, which uses Poseidon instances with 6 full rounds, 16 branches, with no initial linear layer. This allows the attacker to skip the first two full rounds.

The original degree annihilation technique was tailored for a Guess & Solve solving technique. The authors devised a round-skipping technique, that given a parameter $T \in \mathbb{F}$, returns a parameterization of the input depending on the variable X that allows to skip the first 2 full rounds and the first 2 partial rounds. This way, the output equation has degree $\delta = d^{R_F+R_P-4}$ in X. On the other hand, it is likely that the output equations have very high degree in the parameter T. While this is not a problem when simply guessing the variable T and solving in X, we had to adapt it for it to work in our setting. More precisely, we adapt it to get an input depending on variables X and Y, in such a way that our trick reduces the degree of the equations in both X and Y. This adaptation comes with a cost: in the setting of the Zero-test challenge, we are only able to skip the first partial round, whereas the authors of [SV26] skipped the first 2 partial rounds.

Our adaptation. We let $M_{\text{inv},2} \in \mathbb{F}^{2\times 16}$ the matrix M^{-1} restricted to its first two rows, and $c_i^{(r)}$ the i-th coordinate of the r-th round constant. We start from vectors $a_1^{(j)}, a_2^{(k)}, b \in \mathbb{F}^{16}, 1 \le j \le 6, 1 \le k \le 4$ such that:

$$b \in \mathbb{F}^2 \times \{0\}^{14} \ , \ a_1^{(j)} \in \{0\}^2 \times \mathbb{F}^8 \times \{0\}^6 \ , \ a_2^{(k)} \in \{0\}^{10} \times \mathbb{F}^6 \ ,$$

$$M_{\text{inv},2} \cdot (b - c^{(2)}) = \left(c_1^{(1)d}, c_2^{(1)d}\right) \ ,$$

and the $\left(a_1^{(j)}\right)_j$ (resp. $\left(a_2^{(k)}\right)_k$) form a basis of $\ker(M_{\text{inv},2}) \cap (\{0\}^2 \times \mathbb{F}^8 \times \{0\}^6)$ (resp. $\ker(M_{\text{inv},2}) \cap (\{0\}^{10} \times \mathbb{F}^6)$).

This way, for all $\alpha_1^{(j)}, \alpha_2^{(k)} \in \mathbb{F}, 1 \le j \le 6, 1 \le k \le 4$ with $\alpha_1^{(1)} = \alpha_2^{(1)} = 1$, the vector

$$X^{1/d} \sum_{j=1}^{6} \alpha_1^{(j)} \cdot a_1^{(j)} + Y^{1/d} \sum_{k=1}^{4} \alpha_2^{(k)} \cdot a_2^{(k)} + b - c^{(2)}$$

allows to skip the first two rounds. That is when applying the first round in the reverse direction, we get a state that matches the CICO-2 constraint, and when applying the second round in the forward direction, we get a state that is linear in X and Y.

Applying the second and third full round to these states, we get a state that has degree 3 in X and Y and degree 9 in the $\alpha_1^{(j)}, \alpha_2^{(k)}$. Our objective is to find values of the $\alpha_1^{(j)}, \alpha_2^{(k)}$ so that the input of the S-box of the first partial round has degree 1 in X, Y. To do this, we have to write the input of that S-box as a polynomial in X, Y with coefficients in the ring $\mathbb{F}\left[\left(\alpha_1^{(j)}\right)_j, \left(\alpha_2^{(k)}\right)_k\right]$. We then have to find values of the $\alpha_i^{(j)}$ that cancel out the coefficients of degree 2 and 3 in X, Y. Setting these coefficients to 0 amounts to solving a bihomogeneous system of 7 equations in 10 variables (homogeneous in the $\left(\alpha_1^{(j)}\right)_j$ and the $\left(\alpha_2^{(k)}\right)_k$). We set $\alpha_1^{(1)} = \alpha_2^{(1)} = 1$ to dehomogenize the system, and set $\alpha_2^{(4)}$ to a random value to be able to run the procedure several time in case the system has no solution.

Solving the system. The system has the following structure:

- The remaining variables are $\alpha_1^{(j)}, 2 \leq j \leq 6$ and $\alpha_2^{(2)}, \alpha_2^{(3)}$.
- The equation that comes from the coefficient of the monomial $X^u Y^v$ has degree $3u$ in the $\alpha_1^{(j)}$ and $3v$ in the $\alpha_2^{(k)}$.

In our implementation of the degree annihilation technique, we first consider the coefficients corresponding to monomials Y^2 and Y^3. Those coefficients depend only on $\alpha_2^{(2)}$ and $\alpha_2^{(3)}$. This gives us a system of two equations in two unknowns, where the first equation has degree 6 and the second 9. Solving them yields values of $\alpha_2^{(2)}, \alpha_2^{(3)}$ so that the coefficients of Y^2 and Y^3 equal 0.

We now turn to the equations coming from the X^2, XY, X^3, X^2Y, XY^2 terms. We substitute the values of $\alpha_2^{(2)}, \alpha_2^{(3)}$ found in the previous step into those equations, that also depend on the $\alpha_1^{(j)}, 2 \leq j \leq 6$. This gives a system of 5 equations in 5 variables, with 2 equations of degree 6 and 3 equations of degree 9. Solving this system yields an input of the S-box of the first partial round that is linear in X and Y.

Implementation. We use Sagemath to generate the system, and then call Magma [BCP97] to solve it. The solving procedure takes around 15 minutes on a desktop computer. As the system may not have a solution, one may have to run the procedure with different values of $\alpha_2^{(4)}$ to find a solution.

B Proof of the shape of bordering of stable ideals

Lemma 2 (Shape of the border of stable ideals)**.** *Let $I \subset \mathbb{F}[X, Y]$ a 0-dimensional ideal verifying the stability condition. Then its border has the following form:*

$$\mathcal{F}_Y(I) = \left\{ Y^{s-i} X^j \;\middle|\; 0 \leq i < s, \sum_{k=0}^{i-1} w_k \leq j < \sum_{k=0}^{i} w_k \right\} ,$$

where all $w_i \geq 1$.

Proof. As I is 0-dimensional, the sets

$$\{i \in \mathbb{N} \mid X^i \notin \mathrm{in}_\prec(I)\} \ , \ \{i \in \mathbb{N} \mid Y^i \notin \mathrm{in}_\prec(I)\}$$

are finite. We call w (resp. s) the smallest element not in the former (resp. latter) set.

We prove by induction on i the existence of monomials $X^j Y^i, w - \sum_{k=s-i}^{s-1} w_k \leq j < w - \sum_{k=s-i+1}^{s-1} w_k$ in $\mathcal{F}_Y(I)$:

1. $X^w \in \mathrm{in}_\prec(I)$ and $X^{w-1} \notin \mathrm{in}_\prec(I)$. Thus, by stability, $X^{w-1}Y \in \mathcal{F}_Y(I)$. Taking $w - w_{s-1}$ the lowest integer such that $X^{w-w_{s-1}} Y \in \mathrm{in}_\prec(I)$, we get that $X^j Y \in \mathcal{F}_Y(I)$ for $w - w_{s-1} \leq j < w$. This yields the case $i = 1$.

2. by induction on i, we have some monomial $Y^i X^{w - \sum_{k=s-i}^{s-1} w_k} \in \mathcal{F}_Y(I)$. If $w - \sum_{k=s-i}^{s-1} w_k = 0$, then $i = s$ and we have the whole Gröbner stairs. Else, we have that $w - \sum_{k=s-i}^{s-1} w_k > 0$, and we can use the stability assumption and get that $Y^{i+1} X^{w - \sum_{k=s-i}^{s-1} w_k - 1} \in \mathrm{in}_\prec(I)$. Now we define the set

$$\{j \in \mathbb{N} \mid Y^{i+1} X^j \in \mathrm{in}_\prec(I)\} .$$

Defining $w_{s-i-1} \geq 1$ the integer such that $w - \sum_{k=s-i-1}^{s-1} w_k$ is the minimal element in the set, we get that all $Y^{i+1}X^j, w - \sum_{k=s-i-1}^{s-1} w_k \leq j < w - \sum_{k=s-i}^{s-1} w_k$ lie in $\mathrm{in}_\prec(I)$. Moreover, the elements $Y^iX^j, w - \sum_{k=s-i-1}^{s-1} w_k \leq j < w - \sum_{k=s-i}^{s-1} w_k$ do not lie in $\mathrm{in}_\prec(I)$ by minimality of $Y^iX^{w-\sum_{k=s-i}^{s-1} w_k}$ in the previous recursion step. Thus, the $Y^{i+1}X^j, w - \sum_{k=s-i-1}^{s-1} w_k \leq j < w - \sum_{k=s-i}^{s-1} w_k$ lie in $\mathcal{F}_Y(I)$.

□

C Justifications for heuristics in Section 4.3

C.1 A justification for Heuristic 1

We provide here a justification for Heuristic 1, by proving that if $F(Y)$ is replaced with a generic matrix with same degree constraints on all its coordinates, then the bounds on the degrees of $\tilde{N}, \tilde{D}$ hold. We emphasize that as $F(Y)$ is not generated randomly, but stems from polynomials in the ideal spanned by two polynomials P and Q, the matrix $F(Y)$ may have an additional structure that is not taken into account in our generic matrix model.

Additionally, we assume that the ideal I has stable stairs. Using Lemma 2, the degrees of the coordinates of $F(Y)$ have the following form:

$$\begin{cases} \deg(f_{jj}) = s - i \ , \ \sum_{k=0}^{i-1} \leq j < \sum_{k=0}^{i} w_k \ , \\ \deg(f_{ij}) \leq \deg(f_{ii}) + (j-i) \ , \ j < i \ , \\ \deg(f_{ij}) < \deg(f_{ii}) + (j-i) \ , \ j > i \ . \end{cases}$$

In this section, we will denote by d_i the degree $\deg(f_{ii})$, and by n the number of rows of $F(Y)$. The stability and shifted Popov form conditions on $F(Y)$ imply that $0 \leq d_{i+1} - d_i \leq 1, n \geq \sqrt{D_I/2}, d_n \leq \sqrt{2D_I}$. We also have that $\sum_{i=1}^{n} d_i = D_I$.

Theorem 5. *Let $F(Y)$ a matrix satisfying the degree conditions mentioned above, where the coefficients of the f_{ij} have been replaced with indeterminates. Then, there exists a nontrivial Zariski-open space on which $F(Y)$ is such that the minimal fraction description*

$$\Pi_{\mathrm{bot}} F^{-1}(Y) = \tilde{D}(Y)^{-1}\tilde{N}(Y) \ ,$$

is such that $\tilde{D}(Y), \tilde{N}(Y)$ have degree upper bounded by

$$\frac{D_I}{m} + \sqrt{2D_I} \ .$$

To prove this result, we will make use of the following lemma:

Lemma 3 (Adapted from Lemma 2.4 of [Vil18])**.** *Let $H(Y) \in \mathbb{F}(Y)^{m\times n}, H(Y) = \sum_{k\geq 0} H_k Y^{-1-k}$ a strictly proper fraction. For $\delta > 0$, we define the block Hankel matrix:*

$$\mathsf{Hk}_\delta(H) = \begin{pmatrix} H_0 & H_1 & \dots & H_{\delta-1} \\ H_1 & H_2 & \dots & H_\delta \\ \vdots & \vdots & & \vdots \\ H_{\delta-1} & H_\delta & \dots & H_{2\delta-2} \end{pmatrix} \in \mathbb{F}^{m\delta\times n\delta} \ .$$

The rank of $\mathsf{Hk}_\delta(H)$ for $\delta > 0$ is a stationary sequence, let D its limit. The limit rank D corresponds to the degree of the determinant of denominators of irreducible fraction descriptions of H. We have that $\mathrm{rk}(\mathsf{Hk}_\delta(H)) = D$ if and only if there exists fraction descriptions D_L, N_L, D_R, N_R of degree at most δ such that

$$H(Y) = D_L(Y)^{-1}N_L(Y) = N_R(Y)D_R(Y)^{-1} \ .$$

The original result was proven for square matrices. However, on can check that the proof given in [Vil18] translates to rectangular matrices.

Proof of Theorem 5. We mostly follow the steps of [Vil18, Sec. 2 and 3], and prove that there is a nontrivial Zariski-open set of matrices $F(Y)$ such that $\mathrm{rk}(\mathsf{Hk}_\delta(\Pi_{\mathrm{bot}}F(Y)^{-1})) = D_I$ for $\delta = \lceil D_I/m + \sqrt{2D_I} \rceil$. We do so by first proving that the set of matrices F verifying this condition is Zariski-open, and then prove that it is nonempty by providing an explicit example of a matrix F in that set.

A Zariski-open set. Considering the minors of size D_I of $\mathsf{Hk}_\delta(\Pi_{\mathrm{bot}}F(Y)^{-1})$, that are polynomial in the coefficients of F, it is easy to see that the set of matrices such that $\mathrm{rk}(\mathsf{Hk}_\delta(\Pi_{\mathrm{bot}}F(Y)^{-1})) \geq D_I$ is Zariski-open. Moreover, as $\deg(\det(F(Y))) = D_I$, we always have that $\mathrm{rk}(\mathsf{Hk}_\delta(\Pi_{\mathrm{bot}}F(Y)^{-1})) \leq D_I$.

A nonempty set. We provide here a matrix $F(Y)$ following the degree conditions, such that $\mathrm{rk}(\mathsf{Hk}_\delta(\Pi_{\mathrm{bot}}F(Y)^{-1})) = D_I$. The example we choose is inspired by the construction from the beginning of [Vil18, Sec. 3].

Let d_j the degree of the coefficient f_{jj} in the required bound. We let $F(Y)$ such that:

$$\begin{cases} f_{jj} = Y^{d_j} \ , \\ f_{ij} = -1 \ , \ j - i = m \ , \\ f_{ij} = 0 \ , \ \text{else.} \end{cases}$$

That is a matrix of the form

$$F(Y) = \mathrm{Diag}(Y^{d_1}, \ldots, Y^{d_n}) - \begin{pmatrix} 0 & 0 \\ I_{n-m} & 0 \end{pmatrix} \ .$$

Let $N(Y) \in \mathbb{F}[Y]^{m \times n}$ defined as

$$\begin{cases} n_{ii} = 1 \ , \ 1 \leq i \leq m \ , \\ n_{ij} = n_{ji} = 0 \ , \ 1 \leq i < j \leq m \ , \\ n_{i(j+m)} = n_{ij} \cdot Y^{d_j} \ , \ 1 \leq i \leq m, 1 \leq j \leq n-m \ . \end{cases}$$

For instance, if $m = 3, n = 11$, $N(Y)$ equals to:

$$\begin{pmatrix} 1 & 0 & 0 & Y^{d_1} & 0 & 0 & Y^{d_1+d_4} & 0 & 0 & Y^{d_1+d_4+d_7} & 0 \\ 0 & 1 & 0 & 0 & Y^{d_2} & 0 & 0 & Y^{d_2+d_5} & 0 & 0 & Y^{d_2+d_5+d_8} \\ 0 & 0 & 1 & 0 & 0 & Y^{d_3} & 0 & 0 & Y^{d_3+d_6} & 0 & 0 \end{pmatrix} \ .$$

We have that $N(Y)$ rows cancel out the $n - m$ first columns of $F(Y)$, i.e.

$$N(Y)F(Y) = (0 \ D(Y)) = D(Y)\Pi_{\mathrm{bot}} \ , D(Y) \in \mathbb{F}[Y]^{m \times m} \ .$$

Thus we have a left fraction description $\Pi_{\mathrm{bot}}F(Y)^{-1} = D(Y)^{-1}N(Y)$. As the left $m \times m$ block of $N(Y)$ is the identity matrix, $\Pi_{\mathrm{bot}}F(Y)^{-1}\Pi_{\mathrm{left}} = D(Y)^{-1}$, meaning that $\Pi_{\mathrm{bot}}F(Y)^{-1}\Pi_{\mathrm{left}}$ has left and right fraction descriptions of degree $\deg(D)$.

To evaluate the degree of $D(Y)$, we observe that

$$D(Y) = N(Y) \cdot \begin{pmatrix} 0 & \ldots & 0 \\ \vdots & & \vdots \\ 0 & \ldots & 0 \\ Y^{d_{n-m+1}} & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & Y^{d_n} \end{pmatrix} \ .$$

The m rightmost columns of $N(Y)$ give a matrix $N(Y)\Pi_{\text{right}}$ of the form:

$$C^n \cdot \text{Diag}\left(Y^{\sum_{n-km+i\leq n-m} d_{n-km+i}} \mid 1 \leq i \leq m\right) ,$$

with

$$C = \begin{pmatrix} 0 & 1 \\ I_{m-1} & 0 \end{pmatrix} .$$

This gives

$$\begin{aligned} D(Y) &= C^n \cdot \text{Diag}\left(Y^{\sum_{n-km+i\leq n-m} d_{n-km+i}} \mid 1 \leq i \leq m\right) \cdot \begin{pmatrix} Y^{d_{n-m+1}} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & Y^{d_n} \end{pmatrix} \\ &= C^n \cdot \text{Diag}\left(Y^{\sum_{n-km+i\leq n} d_{n-km+i}} \mid 1 \leq i \leq m\right) . \end{aligned}$$

Or, taking back our $m = 3, n = 11$ example:

$$\begin{aligned} D(Y) &= \begin{pmatrix} 0 & Y^{d_1+d_4+d_7} & 0 \\ 0 & 0 & Y^{d_2+d_5+d_8} \\ Y^{d_3+d_6} & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} Y^{d_9} & 0 & 0 \\ 0 & Y^{d_{10}} & 0 \\ 0 & 0 & Y^{d_{11}} \end{pmatrix} \\ &= \begin{pmatrix} 0 & Y^{d_1+d_4+d_7+d_{10}} & 0 \\ 0 & 0 & Y^{d_2+d_5+d_8+d_{11}} \\ Y^{d_3+d_6+d_9} & 0 & 0 \end{pmatrix} . \end{aligned}$$

The maximal degree of the coordinates of $D(Y)$ is the maximum of the

$$\sum_{n-km+i\leq n} d_{n-km+i} \ , \ 1 \leq i \leq m \ .$$

Upper bounding them amounts to upper bound the $\sum_{\substack{k\geq 0 \\ km+i\leq n}} d_{km+i}$ for $1 \leq i \leq m$. Using the fact that $d_i \leq d_{i+1}$ for all i, we upper bound the d_{km+i} by

$$d_{km+i} = \frac{1}{m}\sum_{l=0}^{m-1} d_{km+i} \leq \frac{1}{m}\sum_{l=0}^{m-1} d_{km+i+l} \ ,$$

when $km + i + m - 1 \leq n$, and by d_n for the last term. The sum is thus upper bounded by

$$\sum_{\substack{k\geq 0 \\ km+i\leq n}} d_{km+i} \leq \frac{1}{m}\sum_{k=1}^{n} d_k + d_n = \frac{D_I}{m} + d_n \ .$$

Using the stability assumption, we know that $d_n \leq \sqrt{2D_I}$, thus

$$\deg(D) \leq \frac{D_I}{m} + \sqrt{2D_I} \ .$$

As $\Pi_{\text{bot}}F(Y)^{-1}\Pi_{\text{left}}$ has left and right fraction descriptions of degree less than $\frac{D_I}{m} + \sqrt{2D_I}$, and that the determinant of its denominator has degree D_I, we have that

$$\text{rk}\left(\mathsf{Hk}_{\lceil D_I/m+\sqrt{2D_I}\rceil}\left(\Pi_{\text{bot}}F(Y)^{-1}\Pi_{\text{left}}\right)\right) \geq D_I \ .$$

As the matrix $\mathsf{Hk}_\delta(\Pi_{\text{bot}}F(Y)^{-1}\Pi_{\text{left}})$ is a submatrix of $\mathsf{Hk}_\delta(\Pi_{\text{bot}}F(Y)^{-1})$ for all $\delta > 0$, the rank of $\mathsf{Hk}_{\lceil D_I/m+\sqrt{2D_I}\rceil}(\Pi_{\text{bot}}F(Y)^{-1})$ is at least D_I, yielding an element in the Zariski-open set.

□

C.2 A justification for Heuristic 2

We provide here a justification for Heuristic 2 by showing that for a matrix $F(Y)$ whose bottom projection has suitable properties and a generic choice of $U_{\text{bot}}(Y)$, the denominator of the fraction $\Pi_{\text{bot}}F^{-1}(Y)U_{\text{bot}}\Pi_{\text{left}}$ has a determinant of degree D_I.

The conditions on the bottom projection of $F(Y)$ that are mentioned in the following theorem are true for a generic matrix, as shown in the previous subsection.

Theorem 6. *Let $F(Y)$ a matrix such that $\Pi_{\text{bot}}F^{-1}(Y)$ has an irreducible fraction description $\Pi_{\text{bot}}F^{-1}(Y) = D^{-1}(Y)N(Y)$ with $\deg(\det(D(Y))) = D_I$, and such that $D^{-1}(Y)N(Y)\Pi_{\text{left}}$ is also irreducible. Then, for a matrix $U_{\text{bot}}(Y)$ generic with coefficients $u_{ij}(Y), \deg(u_{ij}) \leq d_{ij}, d_{ij} \geq 0$, the fraction*

$$\Pi_{\text{bot}}F^{-1}(Y)U_{\text{bot}}\Pi_{\text{left}} = D^{-1}(Y)(N(Y)U_{\text{bot}}(Y)\Pi_{\text{left}})$$

is also irreducible.

Proof. We let a fixed matrix $F(Y)$ following the assumptions of the theorem. Let $y_1, \dots y_k \in \overline{\mathbb{F}}$ the roots of $\det(D(Y))$. The fraction $D^{-1}(Y)(N(Y)U_{\text{bot}}(Y)\Pi_{\text{left}})$ is irreducible as long as $N(y_i)U_{\text{bot}}(y_i)\Pi_{\text{left}}$ is nonsingular for all $1 \leq i \leq k$. We prove that this condition is met as long as $U_{\text{bot}}(Y)$'s coefficients lie in a nonempty Zariski-open set.

A Zariski-open set. The $N(y_i)U_{\text{bot}}(y_i)\Pi_{\text{left}}$ are nonsingular when all their determinants are nonzero. As the determinants are polynomial in the coefficients of $U_{\text{bot}}(Y)$ this condition is verified over a Zariski-open set.

A nonempty set. Taking $U_{\text{bot}}(Y) = (\Pi_{\text{left}}\ 0)$, we get that $U_{\text{bot}}(Y)\Pi_{\text{left}} = \Pi_{\text{left}}$ and $D^{-1}(Y)(N(Y)U_{\text{bot}}(Y)\Pi_{\text{left}}) = D^{-1}(Y)(N(Y)\Pi_{\text{left}})$ is irreducible by assumption. In particular the $N(y_i)\Pi_{\text{left}}$ are nonsingular for all $1 \leq i \leq k$, meaning that the set is nonempty. □