STARK Friendly Hash – Survey and Recommendation

Eli Ben-Sasson Lior Goldberg David Levit

StarkWare Industries*

Abstract

A report on the selection process of the STARK friendly hash (SFH) function for standardization by the Ethereum Foundation. The outcome of this process, described here, is our recommendation to use the Rescue function over a prime field of size $\approx 2^{61}$ in sponge mode with 12 field elements per state.

With an Appendix by Jean-Charles Faugère and Ludovic Perret of CryptoNext Security.

1 Executive Summary

On July 2, 2018, the Ethereum Foundation gave StarkWare a 2-year milestone-based grant to select a STARK friendly hash (SFH) function, to be used in combination with transparent and plausibly post-quantum secure proof systems within blockchains, and release an open source efficient STARK system for it. Under the grant agreement, StarkWare committed to *surveying and benchmarking a shortlist of STARK-friendly hash function candidates (giving thorough consideration to MiMC), discussing and getting feedback on the various candidates and deciding on a single STARK-friendly hash function around which StarkWare will build optimized tooling.* This report answers this requirement.

Timeline A quick survey of pre-existing hash functions led to conclude that standard functions like SHA2/3 will require exorbitant proving time and that new constructions should be solicited and collected. A list of relatively new SFH candidates was collected including MiMC [AGRRT16] (which existed prior to this grant), GMiMC [AGPRRRRS19], HadesMiMC [GKKRRS19] and MARVELlous [AABDS19] – the latter two projects received funding from this grant. A set of specific members for consideration was curated (see Section 3), and their STARK complexity and computational efficiency were measured (see Section 6). Simultaneously, these candidates were exposed to public and scientific examination via three different efforts:

- A public SFH challenge for attacking candidate constructions at various security levels between 45 and 256 bits of security.
- Algebraic cryptanalysis was applied to all candidates by the CryptoNext Security (CNS) team, arguably the world-wide leading experts in this area [FP19; Fau20].
- Symmetric cryptanalysis was applied to the constructions on two separate occasions by two teams of expert cryptographers [Can+20; Bey+20; BCLNPPW20].

*Send inquiries to `info@starkware.co`.

Recommendations Based on the results of the security analysis described in Section 4 StarkWare "pruned" the list of candidates, focusing on two of the original contenders: the MiMC construction and the MARVELlous family. STARK arithmetization considerations led us to prefer constructions over small prime fields rather than large fields and/or binary ones, as explained in Section 5. **Consequently, we recommend a particular version of the Rescue hash function – a member of the MARVELlous family – over a prime field of size $\approx 2^{61}$ as the SFH function for standardization by the Ethereum Foundation. This particular version of Rescue is denoted as Rescue_{122} in Section 3.**

We stress that like all new cryptographic constructions, the level of trust placed in the security of Rescue_{122} and the amount of value that should be staked on its security should depend on its tenure and ability to withhold further security scrutiny. Therefore, we recommend a moderated pace of adoption and usage over the next 12 months, alongside continued support for researching the security of this SFH function. Any party staking significant value on the security of Rescue_{122} must take into consideration the potential impact of security flaws in it that may be discovered with time.

Our choice of Rescue_{122} is to a large extent the result of the short and aggressive timeline decided upon for this project. Our choice does not imply that other SFH candidates and/or members of the MiMC, GMiMC, HadesMiMC and MARVELlous are inadequate. Despite security concerns regarding certain variants of the SFH candidates [ACGKLRS19; Bon19; EGLRSW20; KR20; Bey+20], all of the families considered plausibly contain parameter settings that make them secure hash functions, and some of these variants are more efficient computationally and have lower (i.e., better) STARK complexity than Rescue_{122}. However, suggesting such variants and studying them properly would have exceeded the time given to the project.

Finally, we note that the aggressive timeline of the project was shorter than commonly recommend for soliciting and selecting new cryptographic primitives. Therefore, we hope that the Ethereum Foundation and other blockchain and ZKP communities will view Rescue_{122} as a preliminary time-limited SFH construction, one meant to serve until the community concludes a longer process of selecting an SFH function that will improve on our current selection.

Overview of report Section 2 surveys the project timeline in detail. Section 3 describes the shortlist of specific SFH candidates examined by the public and by expert cryptographers. Section 4 discusses the results of cryptanalytic efforts carried out on the SFH candidates and our pruning of the list of candidates based on it. Section 5 explains how we decided on the field type and size for the SFH recommendation. Section 6 surveys the STARK complexity and computational efficiency of new and pre-existing hash functions. Finally, Section 7 integrates the information from previous sections and explains our final recommendation.

2 SFH project timeline

We recount the evolution of the project over time, explaining how we reached our recommendation (see Fig. 1).

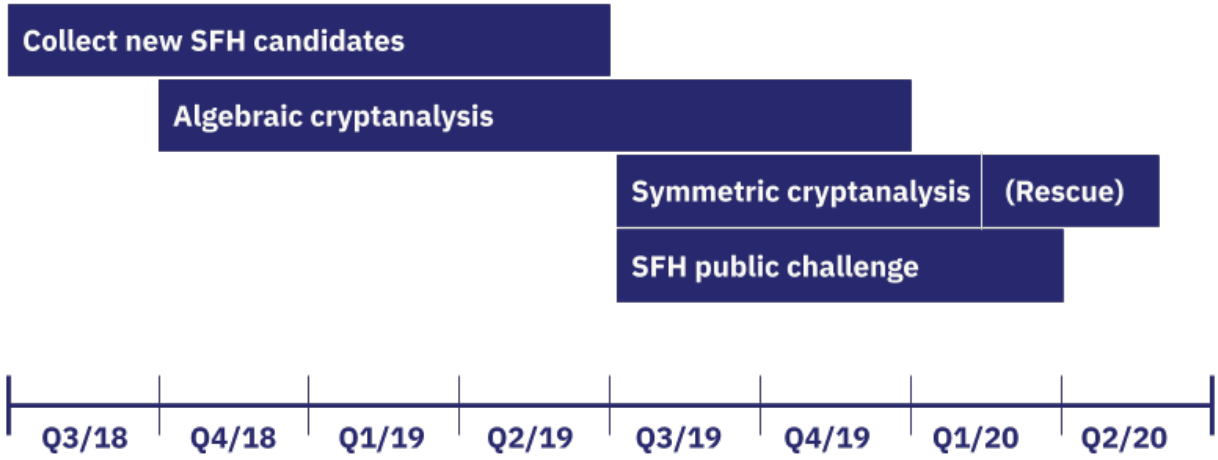

Figure 1: Timeline of SFH project.

- **July 2, 2018:** The 2-year grant agreement between StarkWare and the Ethereum Foundation is signed.
- **Q3/2018:** The Computer Security and Industrial Cryptography (COSIC) group at KU Leuven is contracted to curate a list of existing SFH candidates. It is quickly realized that, with the exclusion of MiMC [AGRRT16], the STARK complexity of existing hash functions (most notably, standard ones like SHA2/3) is prohibitively large. An alternative approach is agreed upon between StarkWare and the Ethereum Foundation: soliciting the construction of *new* SFH candidates.
- **Q3/2018–Q2/2019** New constructions of SFH candidates are collected. Several, like Pedersen[1] and MiMC, have existed before the project started. Others appeared independently during this time, like GMiMC [AGPRRRRS19]. Finally, two separate teams received support under the Ethereum Foundation grant to speedily advance the design of new families of primitives, leading to:
  1. **MARVELlous** – a family which includes Jarvis and Vision (binary fields) and Pepper and Rescue (prime fields), designed by a team of researchers from COSIC [AD18; AABDS19].
  2. **HadesMiMC** – a family which includes Starkad (binary fields) and Poseidon (prime fields), designed by a team of researchers from the IAIK (Institute for Applied Information Processing and Communication at the Graz University of Technology) [GLRRS20; GKKRRS19].

[1]The Pedersen hash is quantum-susceptible, thus unsuitable for the EF goals of post-quantum secure constructions. However, it serves as a good benchmark for SFH functions.

- **Q4/2018–Q4/2019** To further understand the algebraic security of the SFH candidates, StarkWare contracted the CryptoNext Security (CNS) company led by Professors Jean-Charles Faugère and Ludovic Perret, who are members of the POLSYS project-team (Solvers for Algebraic Systems and Applications) of the Laboratoire d'Informatique de Paris 6 (LIP6), headed by Prof. Faugère. They examined the algebraic security of the above mentioned SFH candidates and submitted a series of reports to StarkWare and the Ethereum Foundation [FP19; Fau20], attached as Appendices A and B at the end of this report.

- **Q3/2019-Q1/2020** By the end of Q2/2019 it was clear that the list of SFH contenders should include MiMC, GMiMC, the MARVELlous family and the HadesMiMC family. Dr. Tomer Ashur from KU Leuven and Cryptomeria B.V. was contracted by StarkWare to help fix specific parameters to all constructions in a manner that would be comparable among different families and yet agreeable to the co-authors of those constructions. To foster examination of the constructions, a set of public challenges has been posted online, engaging the public in attempts to break the various candidates. A set of smart contracts have been deployed on Ethereum that reward the first finder of a collision in one of the constructions. The challenges cover several security levels – from 45 bits to 256 bits of security. At time of writing, a few of the 45-bit challenges have been broken (as anticipated) but none of the challenges at higher security levels (80-bits and above).

- **Nov 2019** A STARK-friendly hash selection committee, led by Prof. Anne Canteaut of INRIA Paris and including ten expert symmetric cryptographers convened in Paris for three days (November 11-13, 2019) to conduct cryptanalysis to the SFH candidates. The report submitted by this team [Can+20] (cf. [Bey+20]) recommended using the MARVELlous family, pending (i) further algebraic cryptanalysis of a specific question – the constrained input, constrained output (CICO) problem – and (ii) targeted cryptanalytic examination of Rescue.

- **Q2/2020** The CryptoNext Security company was contracted again to perform the needed algebraic analysis of the CICO problem for Rescue [Fau20] (cf. Appendix B). Additionally, a dedicated cryptanalytic effort targeted at Rescue was performed by a team of experts convened by the cryptosolutions company, led by Prof. Gregor Leander and Dr. Friedrich Wiemer [BCLNPPW20]. Both reports reached favorable results regarding the security of Rescue, leading to StarkWare recommending it for standardization as the SFH function for the Ethereum Foundation.

3 Specific SFH candidates considered

We assume familiarity with the candidate SFH families – MiMC, GMiMC, HadesMiMC and MARVELlous (see [Bey+20] for a concise description). Each of these can be instantiated in a variety of ways, based on the security level, choice of field, capacity and rate of sponge constructions, etc. For cryptanalytic purposes, all these parameters must be fixed. Dr. Tomer Ashur of Cryptomeria B.V. curated a list of specific parameters that was agreeable to the authors of the various SFH candidates. This list was used in the public SFH challenge phase and in the algebraic and symmetric cryptanalytic studies mentioned above. The specific candidates at the security level of ≈ 120 bits were the following constructions:

- **MiMC** Instantiated as a sponge construction using the MiMC$-2n/n$ Feistel construction described in [AGRRT16, Section 2.1] over the prime field of size $2^{253} + 2^{199} + 1$ with 320 rounds. We denote this specific construction by MiMC_{126} since its conjectured security is at least 126 bits (subscripts reflect conjectured security also below).

- **Other** All other constructions were instantiated over fields of size $\approx 2^{64}$: the binary field used in Starkad and Vision was of size 2^{63} (using a standard basis over GF(2), defined by the irreducible polynomial $x^{63}+x+1$) and the prime field used by GMiMC, Poseidon and Rescue was of size $p = 2^{61} + 20 \times 2^{32} + 1$; elements of this field require 62 bits to be written, but later on, when considering security, we only assume the field size is at least 2^{61}. All constructions were in sponge mode with a state of 12 field elements, with a capacity of 4 field elements and rate of 8 field elements. Regarding the number of rounds:

  - **Starkad** was instantiated over the binary field with 8 full rounds and 43 partial rounds (total of 51 rounds). Denote this specific version by $\mathsf{Starkad}_{126}$.
  - **Poseidon** was instantiated over the prime field with 8 full rounds and 40 partial rounds (total of 48 rounds). Denote this specific version by $\mathsf{Poseidon}_{122}$.
  - **Vision** was instantiated over the binary field with 10 rounds, denoted Vision_{126}.
  - **Rescue** was instantiated over the prime field with 10 rounds, denoted Rescue_{122}.
  - **GMiMC** using the expanding round function (erf) version described in [AGPRRRRS19, Section 2.1.2] was instantiated with the prime field and 101 rounds. We denote this version by GMiMC_{122}.

4 Security

The confidence we have in the security of a symmetric cryptographic primitive scales (roughly) as a function of the time that has lapsed since its invention, and the amount of usage it receives and value it supports (the "Lindy Effect") and the quantity of public scrutiny it has received from cryptographers. According to these criteria, standard hash functions like SHA2, SHA3 and hash functions derived securely from block-ciphers like AES/Rijndael are deemed most secure. Closely following in terms of security are constructions that, though not selected as standards by national and international bodies, have been reviewed publicly for a large duration of time, and have been deployed in various settings. This family includes contenders for standardization like the SHA3 finalists, Blake, Grøstl, JH and Skein. (Jumping ahead, all of these constructions are extremely efficient but have prohibitively large STARK complexity.)

Next, we summarize our understanding of the security of the new SFH candidates. Our analysis is based entirely on the external sources written by expert cryptographers, namely, Professors Faugère and Perret for algebraic cryptanalysis [Appendices A and B], the reports of Prof. Canteaut's team [Can+20] and the cryptosolutions team [BCLNPPW20], and the published papers [ACGKLRS19; Bon19; Bey+20; KR20; EGLRSW20].

MiMC_{126} Among the SFH candidates, MiMC has received the largest amount of scrutiny due to its tenure and simplicity. To date, we are aware of two attacks on variants of it: the first, appearing in [Bon19], applies only to block-ciphers using the Feistel construction, but not the hash version, of which MiMC_{126} is a specific case. The second, recent, attack, applies to binary fields, and, to a

lesser extent, to extension fields [EGLRSW20] but not to the large prime field underlying MiMC_{126}. **Therefore, we consider MiMC_{126} to be secure enough for consideration as the EF SFH.**

GMiMC_{122}, $\mathsf{Starkad}_{126}$ and $\mathsf{Poseidon}_{122}$ To date, several papers have revealed weaknesses in certain variants of the GMiMC family and the HadesMiMC family (to which $\mathsf{Starkad}_{126}$ and $\mathsf{Poseidon}_{122}$ belong) [Bon19; KR20; Bey+20]. The recent [Bey+20], which was written by the members of the SFH cryptanalytic committee (and is in line with their report [Can+20]) concludes by saying:

> *Our analysis of STARK-friendly primitives clearly shows that the concrete instances of GMiMC and HadesMiMC proposed in the StarkWare challenges present several major weaknesses, independently from the choice of the underlying finite field. At a first glance, the third contender involved in the challenges, namely Vision for the binary field and Rescue for the prime fields, seems more resistant to the cryptanalytic techniques we have used against the other two primitives. This seems rather expected since Vision and Rescue follow a more classical substitution permutation network (SPN) construction with full Sbox layers; for similar parameters, they include a larger number of Sboxes which may prevent them from the unsuitable behaviors we have exhibited on the other primitives.* [Bey+20, Section 6]

Consequently, we removed GMiMC_{122}, $\mathsf{Starkad}_{126}$ and $\mathsf{Poseidon}_{122}$ from further consideration as the EF SFH. This does not mean the families GMiMC and HadesMiMC are insecure. Rather, our removal only implies that given the short timeline of our project and the desire for high security we would not feel comfortable recommending use of the very specific constructions GMiMC_{122}, $\mathsf{Starkad}_{126}$ and $\mathsf{Poseidon}_{122}$.

Vision_{126} and Rescue_{122} A precursor to Vision, named Jarvis, was shown to be insecure by both the algebraic cryptanalysis report [FP19] and an external publication [ACGKLRS19], and this attack line could also be mounted on the unpublished precursor to Rescue, named Pepper (see [AD20] for a discussion of it). However, to date we are unaware of attacks on Vision_{126} and Rescue_{122} (as echoed in the report cited above). Furthermore, the dedicated algebraic cryptanalysis applied to the CICO problem for Rescue_{122} (see Appendix B) and the dedicated cryptanalytic effort targeted at it led by the cryptosolutions team [BCLNPPW20] have not revealed any weaknesses in Rescue_{122}. To quote from the final report on Rescue by the latter team:

> *We considered several attack vectors and did not find attacks that pose a direct risk to the current design. Time was limited and we cannot exclude that other attack vectors or variants of the ones we considered lead to a full break of Rescue.* [BCLNPPW20, Section 1]

Based on this, we find Vision_{126} and Rescue_{122} to be secure enough for consideration as the EF SFH, alongside MiMC_{126}.

5 Choice of finite field

Our recommendation takes into account the (i) type of field – prime vs. binary and (ii) ease of creating inputs to the hash function. Both aspects are discussed next.

Field type — binary vs. prime Binary fields have simpler arithmetic and are ideally suited for mapping bit-strings to field elements. Prime fields require arithmetic that is slightly more complicated and the mapping between bit strings and field elements is more problematic. However, prime fields offer an advantage in the context of STARK constructions: The evaluation domain of an algebraic execution trace can be picked to be a multiplicative subgroup of size precisely 2^k whereas no multiplicative sub-group of a binary field can be of such a size (but for the trivial group, that contains only the identity element). This means that over certain prime fields one may encode each column of the algebraic execution trace as a mapping from a group of size 2^k to the field, and then carry out quasi-linear time interpolation and multi-point evaluation via FFT and iFFT algorithms. Furthermore, the Fast RS IOPP (FRI) protocol [BBHR18] underlying STARK efficiency requires groups of size 2^k, and the same group used for arithmetization and encoding of the algebraic execution trace (AET), defined later in Definition 1, can be used in the FRI protocol. In the case of binary fields, the FRI protocol is carried out over an additive subgroup whereas the encoding of the AET is done over a subset that "embeds" a group of size $2^k - 1$ in a manner that is not as clean from an algebraic viewpoint (see [BBHR19] for details). **Due to this consideration, we find constructions over prime fields to be more efficient and focus our attention on these constructions[2].** Among the SFH contenders, the relevant ones are MiMC_{126} and Rescue_{122} (GMiMC_{122} and $\mathsf{Poseidon}_{122}$ have been excluded from the discussion for security reasons, as explained in Section 4).

Ease of creating inputs for the hash function Future uses of the SFH function will create STARK proofs that involve claims of computational integrity in addition to assertions about the correct invocation of the SFH. This is already the case for all instances of deployed ZKPs, like the Sprout and Sapling circuits of Zcash and StarkWare's StarkEx system. Natural encoding via R1CS and AIR systems use elements of an ambient field to describe the computation whose integrity is being proved. Moving from an encoding in one field to a different encoding is costly. In this respect, MiMC_{126} with its state consisting of just two field elements suffers from a disatvantage with respect to other constructions: it is not clear how to efficiently add new inputs to the hash function, since the internal state of the hash is comprised of a small number of large field elements. In contrast, all other SFH candidates are instantiated with a state comprised of many elements in smaller fields and thus can incorporate new inputs without necessarily moving between field representations.

To summarize, sponge constructions using a state with several field elements from a small field, like $\mathsf{GMiMC}_{122}, \mathsf{Starkad}_{126}, \mathsf{Poseidon}_{122}, \mathsf{Vision}_{126}$ and Rescue_{122}, are better suited to deal with general computational integrity statements than constructions, like MiMC_{126}, whose state holds a small number of large field elements.

[2] Another reason to prefer prime over binary fields is the current lack of native support in the Ethereum Virtual Machine for binary field multiplication, which would entail high gas costs for operations over binary fields.

6 STARK complexity and hash function efficiency

In this section we compare the computational efficiency and the proving complexity of the different SFH candidates (and pre-existing hash functions).

6.1 A simple STARK complexity measure

When discussing STARK complexity we focus on prover complexity because it is the major bottleneck in this project, and also because verification time and proof size scale poly-logarithmically with proof size and are thus less affected by the efficiency of different constructions. In particular, verification time of under 10ms and a proof size of at most 200KB for STARK soundness error of at most 2^{-80} — as defined in the grant agreement — are attainable for all hash functions mentioned in this section.

We start by defining a simple metric to gauge the "STARK friendliness" of different cryptographic primitives. Before reaching this metric in Definition 2 we need a preliminary definition. Later on we shall use this metric to compare different SFH candidates (and other hash functions). We limit ourselves to the minimal requirements and refer the interested reader to prior research on post-quantum secure scalable and transparent proof systems in, e.g., [AHIV17; BBHR19; BCGGRS19; COS20].

Definition 1 (Constraint systems and satisfiability)**.** *Let $\mathbb{F}$ be a finite field and $\mathsf{w}, \mathsf{t}, d$ be integers. An* $(\mathbb{F}, \mathsf{w}, \mathsf{t}, d)$-constraint *is a multivariate polynomial C of total degree d over variables $X[i,j], i \in \{1,\ldots,\mathsf{t}\}, j \in \{1,\ldots,\mathsf{w}\}$.*

An $(\mathbb{F}, \mathsf{w}, \mathsf{t}, d)$-constraint system S is a set of $(\mathbb{F}, \mathsf{w}, \mathsf{t}, d')$-constraints for $d' \leq d$, where d, the degree of S, is the maximal degree of a constraint in S.

An $(\mathbb{F}, \mathsf{w}, \mathsf{t})$-algebraic execution trace *(AET) is an array A with t rows and w columns, each entry of which is an element of $\mathbb{F}$. The i-th row represents the* state *of a computation at time i and the j-th column tracks the state of an* algebraic register *over time. We denote the i,j entry of an AET A by $\mathsf{A}[i,j]$. We say that A* satisfies *S if and only if for every constraint $C(X[i_1,j_i],\ldots,X[i_k,j_k])$ in S we have $C(\mathsf{A}[i_1,j_1],\ldots,\mathsf{A}[i_k,j_k]) = 0$.*

With this definition in hand we proceed to define STARK complexity for purposes of measuring different SFH candidates.

Definition 2 (STARK complexity)**.** *Let $f : \mathbb{F}^m \to \mathbb{F}^n$ be a function. We say that an $(\mathbb{F}, \mathsf{w}, \mathsf{t}, d)$-constraint system S* implements *f if all of the following conditions hold:*

- **i/o mapping** *There exist $n+m$ distinct indices $I_1,\ldots,I_n,O_1,\ldots,O_m \in [\mathsf{t}] \times [\mathsf{w}]$ called the* input *and* output *indices. Informally, we expect an $(\mathbb{F}, \mathsf{t}, \mathsf{w})$-AET A to contain the input and output of f in these locations.*
- **completeness** *For every input $x = (x_1,\ldots,x_n) \in \mathbb{F}^n$ and output $y = (y_1,\ldots,y_m) = f(x) \in \mathbb{F}^m$, there exists an $(\mathbb{F}, \mathsf{t}, \mathsf{w})$-AET A_x that satisfies S and furthermore $\mathsf{A}_x[I_i] = x_i$ for all $i \in [n]$ and $\mathsf{A}_x[O_j] = y_j$ for all $j \in [m]$.*
- **efficiency** *There exists an efficient algorithm that, given $x \in \mathbb{F}^n$, will produce an AET A_x that satisfies the completeness condition above.*

- **soundness** *Given* $\mathsf{A} \in \mathbb{F}^{\mathsf{t}\times\mathsf{w}}$ *denote* $x(\mathsf{A}) = (\mathsf{A}[I_1], \ldots, \mathsf{A}[I_n])$ *and* $y(\mathsf{A}) = (\mathsf{A}[O_1], \ldots, \mathsf{A}[O_m])$. *If* $y(\mathsf{A}) \neq f(x(\mathsf{A}))$ *then* A *does not satisfy* S.

The STARK complexity *of* S *is now defined as*

$$\mathsf{c}(S) := \lceil \log_2 |\mathbb{F}|/64 \rceil^2 \cdot (d + \mathsf{w}) \cdot \mathsf{t} \log_2 \mathsf{t} \tag{1}$$

and the STARK complexity of f*, denoted* $\mathsf{c}(f)$*, is the minimal STARK complexity of* S *taken over all* S *that implements* f.

A few remarks are in place regarding the definition above.

- A trivial optimization is to convert a $\mathsf{t} \times \mathsf{w}$ table to a different one with $\mathsf{t}' = 1$ and $\mathsf{w}' = \mathsf{t} \cdot \mathsf{w}$. While this slightly reduces the complexity measure in Eq. (1), it also increases verifier complexity, not discussed here. The exact choices of t and w appearing in Table 1 reflect a setting that we believe optimizes the tradeoff between proving and verification time (details omitted). Readers interested in optimizing t and w in various settings are referred to [BBHR19], where the tradeoff is explained.

- Practically speaking, for all but trivial functions we cannot determine $\mathsf{c}(f)$ exactly because proving lower bounds on $\mathsf{c}(f)$ for nontrivial functions is beyond our reach. Thus, we can only rely on ever decreasing upper bounds on $\mathsf{c}(f)$. Indeed, since we first examined the question of STARK compelxity, a number of new techniques like holography [COS20; CHMMVW20] and lookup tables [GW20] may lead to better (lower) complexity measures for various functions. Relatedly, notice that determining the exact finite field over which to define the constraint system could drastically affect complexity. For practical reasons we limited our attention to a few "natural" candidates for the different hash functions.

- We intetionally left the notion of "efficiency" undefined in Definition 2. Asymptotically, it could be defined in the usual way, as polynomial running time. Practically, the exact polynomial time for this item did not enter our considerations and estimates

- The actual STARK proving time also depends on the rate chosen to encode the AET. Since its effect on all hash functions is similar, for the sake of comparing SFH candidates it can be omitted.

6.2 STARK complexity

Table 1 summarizes the STARK complexity of various hash functions according to Eq. (1) for the statement requested by the Ethereum Foundation in the grant agreement. The statement refers to a hash-chain of length 10^5, i.e., involving 10^5 invocations of the hash function, and colors reflect STARK efficiency (green is good, red is bad).

Explanation of Table 1 The hash functions are sorted by families. We start with the standard functions SHA2 and SHA3 which did not have STARK complexity as their design goal, leading to large complexity measure c. The subscripts in all hash function names designate the number of bits of security that they claim. For SHA2, the estimate over the 64-bit binary field is taken from [BBHR19] and that paper also gives the estimates for the AES-based hash functions which are

Family	Name	$\mathbb{F}$	t / invocation	w	d	c / 10^5 invocations
Standard	$\mathsf{SHA2}_{128}$	64-bit prime	1000	20	2	5.6×10^{10}
		$\mathrm{GF}(2^{64})$	3762	56	11	7.2×10^{11}
	$\mathsf{SHA3}_{128}$	$\mathrm{GF}(2^{64})$	1536	25	2	1.1×10^{11}
AES-DM	AES_{64}	$\mathrm{GF}(2^{64})$	48	62	8	7.5×10^{9}
	$\mathsf{Rijndael}_{80}$	$\mathrm{GF}(2^{64})$	58	68	8	9.9×10^{9}
Algebraic Sponge	$\mathsf{Pedersen}_{128}$	256-bit prime	128	16	2	8.7×10^{10}
	MiMC_{126}	253-bit prime	320	2	3	6.4×10^{10}
	GMiMC_{122}	61-bit prime	101	1	3	9.4×10^{8}
	$\mathsf{Starkad}_{126}$	$\mathrm{GF}(2^{63})$	10	14	3	3.4×10^{8}
	$\mathsf{Poseidon}_{122}$	61-bit prime	8	17	3	3.1×10^{8}
	Rescue_{122}	61-bit prime	10	12	3	3×10^{8}
	Vision_{126}	$\mathrm{GF}(2^{63})$	20	12	6	7.5×10^{8}
			40	12	4	1.4×10^{9}

Table 1: The STARK complexity (c) of various hash functions for 100,000 invocations, computed according to Eq. (1). See Section 6.2 for explanations.

built using the Davies–Meyer (DM) construction; we denote this family by AES-DM (see [BBHR19, Figure 4] for details). The family of algebraic functions, used in sponge mode, includes pre-existing functions like Pedersen (which is not post-quantum secure) and MiMC as well as the new SFH candidates. In the table, the third column ($\mathbb{F}$) is the field over which the function is defined (notice that for SHA2 we provide two options). The fourth column specifies the length of the AET for a single invocation and the next column specifies the number of columns (see the discussion following Definition 2 for potential tradeoffs between t and w). The final column gives the STARK complexity c according to Eq. (1) for 100,000 invocations of the hash function.

6.3 Efficiency as hash functions

One of the most important parameters in the selection of standard hash functions (like SHA2/3) is their efficiency and speed. In the context of this project efficiency considerations take second place compared with STARK complexity (c) and security but are nevertheless important to consider. Table 2 approximate the efficiency of the various SFH candidates on current CPUs, using the number of 64-bit multiplication operations as a rough estimate. We stress that we have not dedicated considerable efforts to optimizing any of these constructions. The number of rounds is denoted by R; For the HadesMiMC members ($\mathsf{Starkad}_{126}$ and $\mathsf{Poseidon}_{122}$) we use R_f for full rounds and R_p for partial rounds. We explain the formulas used in the fifth column below:

- Rescue_{122}**:** Each round involves two steps. Each step involves a linear transformation, costing m^2 multiplications, one step requires a cubing operation (2 multiplications) per state variable and the other step requires a single cube-root operation per state variable (involving 68 multiplications).
- MiMC_{126}: We use the $2n/n$ Feistel construction described at the end of Section 2.1 of [AGRRT16]. This involves 320 rounds over a state of size 2. Each field multiplication over the large ($\approx$ 256-bit) field is approximated by 16 field operations over a 64-bit field.

SFH	$\log \lvert\mathbb{F}\rvert$	Rounds R	State m	# 64-bit multiplications	Total
Rescue_{122}	61	10	12	$(2m^2 + 70m) \times R$	11280
MiMC_{126}	256	320	2	$16 \times 2 \times R$	10240
Vision_{126}	63	10	12	$(2m^2 + 8m + 2(\log \lvert\mathbb{F}\rvert + 3m)) \times R$	5820
$\mathsf{Starkad}_{126}$	63	$R_f = 8$, $R_p = 43$	12	$(m^2 + 2m) \times R_f + (2m + 2) \times R_p$	2462
$\mathsf{Poseidon}_{122}$	61	$R_f = 8$, $R_p = 40$	12	$(m^2 + 2m) \times R_f + (2m + 2) \times R_p$	2384
GMiMC_{122}	61	101	12	$2 \times R$	202

Table 2: The estimated efficiency of new SFH candidates, sorted in decreasing order (lower is better). See Section 6.3 for explanations.

- Vision_{126}: Each round of Vision involves two steps. In each step we apply inversion for each state variable. However, using Montgomery inversion this amounts to a single inversion ($\log |F|$ multiplications) and 3 additional multiplications per state variable. Additionally, we need to apply a linear transformation (m^2 multiplications per step) and a linearized polynomial of degree 4 (4 multiplications per state variable), or its inverse, which we assume amounts to 4 multiplications as well.

- $\mathsf{Starkad}_{126}$ and $\mathsf{Poseidon}_{122}$: Each full round requires one cubing per state variable (2 multiplicatoins) and one linear transformation on all m state variables (m^2 multiplications). Each partial round involves only cubing a single variable and adding a linear combination of other state variables, totaling $2m + 2$ multiplications per (partial) round.

- GMiMC_{122}: Each round of GMiMC involves only a single cubing (2 multiplications), leading to extreme efficiency (10× better than all other constructions discussed above).

7 Rescue_{122} as the recommended SFH for the EF

We end the report by summarizing the reasoning that led us to recommend Rescue_{122} as the SFH candidate for standartization by the Ethereum Foundation.

Previous standard constructions like SHA2/3 have exhorbitant STARK complexity (see Table 1). Among newer constructions, we focus on constructions over prime fields, as explained in Section 5. While the MiMC hash family is simplest and oldest, the reliance of MiMC_{126} on a large prime field makes its STARK complexity worse than other candidates and its ability to incorporate new inputs in an efficient algebraic manner makes it unsuitable for standartization. The security analysis described in Section 4 suggests avoiding GMiMC_{122} and $\mathsf{Poseidon}_{122}$. This leaves us with Rescue_{122} as the only remaining choice, by way of elimination.

Rescue is a new construction and thus has not received as much scrutiny as MiMC. However, its similarity to the familiar SPN construction (used by AES) and the preliminary algebraic and symmetric cryptanalysis results indicate it seems suitable for use. In terms of efficiency, Rescue_{122} lags behind $\mathsf{Poseidon}_{122}$ but not by a large factor, and when considering STARK complexity, Rescue_{122} is slightly better (i.e., has smaller c) than $\mathsf{Poseidon}_{122}$.

We close by pointing out that like all new cryptographic constructions, more scrutiny is essential for improving confidence in the security of Rescue_{122}. We therefore strongly recommend to the

Ethereum Foundation to consider further funding of public challenges and cryptanalytic research focused on Rescue_{122}, as well as on the large families of GMiMC, HadesMiMC and MARVELlous.

References

[AABDS19] Abdelrahaman Aly, Tomer Ashur, Eli Ben-Sasson, Siemen Dhooghe, and Alan Szepieniec. "Efficient Symmetric Primitives for Advanced Cryptographic Protocols (A Marvellous Contribution)". In: *IACR Cryptol. ePrint Arch.* 2019 (2019), p. 426. URL: https://eprint.iacr.org/2019/426.

[ACGKLRS19] Martin R. Albrecht, Carlos Cid, Lorenzo Grassi, Dmitry Khovratovich, Reinhard Lüftenegger, Christian Rechberger, and Markus Schofnegger. "Algebraic Cryptanalysis of STARK-Friendly Designs: Application to MARVELlous and MiMC". In: *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part III*. Ed. by Steven D. Galbraith and Shiho Moriai. Vol. 11923. Lecture Notes in Computer Science. Springer, 2019, pp. 371–397. DOI: 10.1007/978-3-030-34618-8_13. URL: https://doi.org/10.1007/978-3-030-34618-8_13.

[AD18] Tomer Ashur and Siemen Dhooghe. "MARVELlous: a STARK-Friendly Family of Cryptographic Primitives". In: *IACR Cryptol. ePrint Arch.* 2018 (2018), p. 1098. URL: https://eprint.iacr.org/2018/1098.

[AD20] Tomer Ashur and Siemen Dhooghe. "Prelude to Marvellous (With the Designers' Commentary, Two Bonus Tracks, and a Foretold Prophecy)". In: *IACR Cryptol. ePrint Arch.* 2020 (2020), p. 568. URL: https://eprint.iacr.org/2020/568.

[AGPRRRRS19] 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: *Computer Security - ESORICS 2019 - 24th European Symposium on Research in Computer Security, Luxembourg, September 23-27, 2019, Proceedings, Part II*. Ed. by Kazue Sako, Steve Schneider, and Peter Y. A. Ryan. Vol. 11736. Lecture Notes in Computer Science. Springer, 2019, pp. 151–171. DOI: 10.1007/978-3-030-29962-0_8. URL: https://doi.org/10.1007/978-3-030-29962-0_8.

[AGRRT16] Martin Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. "MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity". In: *Advances in Cryptology – ASIACRYPT 2016*. Ed. by Jung Hee Cheon and Tsuyoshi Takagi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 191–219. ISBN: 978-3-662-53887-6.

[AHIV17] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. "Ligero: Lightweight Sublinear Arguments Without a Trusted Setup". In: *Proceedings of the 24th ACM Conference on Computer and Communications Security*. CCS '17. 2017, pp. 2087–2104.

[BBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. "Fast Reed-Solomon Interactive Oracle Proofs of Proximity". In: *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*. Ed. by Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella. Vol. 107. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018, 14:1–14:17. DOI: 10.4230/LIPIcs.ICALP.2018.14. URL: https://doi.org/10.4230/LIPIcs.ICALP.2018.14.

[BBHR19] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. "Scalable Zero Knowledge with No Trusted Setup". In: *Proceedings of the 39th Annual International Cryptology Conference*. CRYPTO '19. 2019, pp. 733–764.

[BCGGRS19] Eli Ben-Sasson, Alessandro Chiesa, Lior Goldberg, Tom Gur, Michael Riabzev, and Nicholas Spooner. "Linear-Size Constant-Query IOPs for Delegating Computation". In: *Proceedings of the 17th Theory of Cryptography Conference*. TCC '19. 2019.

[BCLNPPW20] Tim Beyne, Anne Canteaut, Gregor Leander, Mara Naya-Plasencia, Léo Perrin, and Friedrich Wiemer. *On the security of the Rescue hash function*. Cryptology ePrint Archive, Report 2020/820. https://eprint.iacr.org/2020/820. 2020.

[Bey+20] Tim Beyne et al. "Out of Oddity - New Cryptanalytic Techniques against Symmetric Primitives Optimized for Integrity Proof Systems". In: *IACR Cryptol. ePrint Arch.* 2020 (2020), p. 188. URL: https://eprint.iacr.org/2020/188.

[Bon19] Xavier Bonnetain. *Collisions on Feistel-MiMC and univariate GMiMC*. Cryptology ePrint Archive, Report 2019/951. https://eprint.iacr.org/2019/951. 2019.

[CHMMVW20] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas Ward. "Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS". In: *Proceedings of the 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT '20. 2020.

[COS20] Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. "Fractal: Post-Quantum and Transparent Recursive Proofs from Holography". In: *Proceedings of the 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT '20. 2020.

[Can+20] Anne Canteaut et al. "Report on the Security of STARK-friendly Hash Functions (Version 2.0)". working paper or preprint. June 2020. URL: https://hal.inria.fr/hal-02883253.

[EGLRSW20] Maria Eichlseder, Lorenzo Grassi, Reinhard Lftenegger, Morten ygarden, Christian Rechberger, Markus Schofnegger, and Qingju Wang. *An Algebraic Attack on Ciphers with Low-Degree Round Functions: Application to Full MiMC*. Cryptology ePrint Archive, Report 2020/182. https://eprint.iacr.org/2020/182. 2020.

[FP19] Jean-Charles Faugère and Ludovic Perret. *Algebraic Attacks against STARK-Friendly Ciphers*. Version 1.2. Appearing as Appendix A to this report. 2019.

[Fau20] Jean-Charles Faugère. *Rescue CICO Problem*. Appearing as Appendix B to this report. 2020.

[GKKRRS19] Lorenzo Grassi, Daniel Kales, Dmitry Khovratovich, Arnab Roy, Christian Rechberger, and Markus Schofnegger. "Starkad and Poseidon: New Hash Functions for Zero Knowledge Proof Systems". In: *IACR Cryptol. ePrint Arch.* 2019 (2019), p. 458. URL: https://eprint.iacr.org/2019/458.

[GLRRS20] Lorenzo Grassi, Reinhard Lüftenegger, Christian Rechberger, Dragos Rotaru, and Markus Schofnegger. "On a Generalization of Substitution-Permutation Networks: The HADES Design Strategy". In: *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part II*. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12106. Lecture Notes in Computer Science. Springer, 2020, pp. 674–704. DOI: 10.1007/978-3-030-45724-2_23. URL: https://doi.org/10.1007/978-3-030-45724-2_23.

[GW20] Ariel Gabizon and Zachary J. Williamson. “plookup: A simplified polynomial protocol for lookup tables”. In: *IACR Cryptol. ePrint Arch.* 2020 (2020), p. 315. URL: https://eprint.iacr.org/2020/315.

[KR20] Nathan Keller and Asaf Rosemarin. “Mind the Middle Layer: The HADES Design Strategy Revisited”. In: *IACR Cryptol. ePrint Arch.* 2020 (2020), p. 179. URL: https://eprint.iacr.org/2020/179.

A CryptoNext Security Report

Algebraic Attacks against STARK-Friendly Ciphers (Version 1.2)

Jean-Charles Faugère & Ludovic Perret

CryptoNext Security
https://cryptonext-security.com/
contact@cryptonext-security.com

Abstract. This report presents a security analysis of the so-called STARK-friendly ciphers : Jarvis, MiMC, Pepper, Vision, Rescue, Poseidon and Starkad. These primitives are currently candidates for being the default cipher in STARK applications. The goal of this report is to asses the security of these primitives with respect to some algebraic attacks. This report has been prepared at the demand of Starkware Industries. The analysis has been performed during two periods : from November 2018 to September 2020 and in February 2020 (for an additional assessment on Rescue).

Table of Contents

1 Introduction

The rise of practical applications of advanced cryptographic protocols such as homomorphic encryption, Multi-Party Computation (MPC) or Zero-Knowledge (ZK) proof system emphasized the needs to design symmetric primitives that are optimized with respect to specific metrics related to the applications considered.

In particular, the ZK-STARK proof system [10] offers zero-knowledge proofs in which verification scales exponentially faster than data size. ZK-STARK proposed then a viable solution to the scalability problem in cryptocurrencies. In particular, the ZK-STARK technology is expected to be deployed on top of the Ethereum[1] blockchain in 2020.

The security and efficiency of ZK-STARK proof systems rely, in particular but crucially, on the security of a hash function. However, it turns that standard constructions – such as SHA2 [42] or the Davies–Meyer hash function based on Rijndael [20] – are too costly for the ZK-STARK technology. Regarding the potential applications of the such technology, it is of primary importance to design secure hash functions optimized for this ZK-proof system.

As formalized in [5], this leads to revisit the classical design rationale of block ciphers and hash functions based on such block ciphers. Following the terminology of [5], we shall call *Arithmetization-Oriented (AO) ciphers* this new design paradigm. In contrast to traditional block ciphers design, the main attacks to consider for AO ciphers are not of statistical nature but of algebraic nature.

The goal of this report is to investigate the security of STARK-friendly ([2,5,3,32]) primitives against algebraic attacks. More precisely, we have considered :

Jarvis [5]	The first generation of block cipher optimized for STARK-proofs
MiMC [2]	A lightweight block cipher well suited for the STARK technology
Pepper/Rescue/ Vision [3]	Second generation of AO ciphers that fix some weaknesses of Jarvis
Poseidon/Starkad [32]	Latest design of STARK-friendly hash functions

Table 1. List of STARK-friendly primitives considered in this report.

1.1 Overview of the Results

This report presents a security analysis of AO ciphers listed in Table 1 against algebraic attacks. An algebraic attack has typically two steps :

- *modelling* the cipher as a set of algebraic equations and,
- *solving* the non-linear equations to recover a secret of the cipher (secret-key or pre-image in our context). In this document, we use a Gröbner basis [14,13] (Section 2.1) for the solving step.

It is well known that algebraic attacks against modern block ciphers have had – so far – a limited impact on the security. This can be explained by at least two facts. First, it is challenging to perform practical experiments on sufficiently relevant reduced version of the

[1] https://medium.com/starkware/stark-friendly-hash-tire-kicking-8087e8d9a246

ciphers [18]. Secondly, it is difficult to derive precise complexity bounds on the complexity of such attacks [6]; in particular on the solving step.

However, as emphasised in [5], the issue is central for STARK-friendly ciphers:

> *"Consequently, the question of security against Gröbner basis attacks is the crucial concern raised by arithmetization-oriented ciphers, and no such proposal is complete without explicitly addressing it".*

In this report, we present new results on the algebraic cryptanalysis of STARK-friendly ciphers (those listed in Table 1). We emphasize that a follow-up report considered [23] a broader set of attacks against AO ciphers; including the ciphers listed in Table 1.

1.1.1 Methodology. Starkware Industries provided a description and a SAGE ([43]) reference code of the primitive to analyse. We then implemented the primitive in MAGMA [12] an/or MAPLE [39] and always compared the output of the function with the reference SAGE implementation.

	Jarvis	Jarvis-v2	Pepper-v1	Pepper-v2	Rescue	Vision	MiMC	Poseidon Starkad
Sage (ref)	✘	✘	✘	✘	✘	✘	✘	✘
Maple	✘	✘	✘	✘	✘	✘	✘	
Magma	✘	✘	✘	✘	✘		✘	✘

Table 2. Summary of the codes developed.

In addition, we developed a new dedicated C program to attack some of the primitives.

1.1.2 How to Fix the Number of Rounds. Given a security level s, the main question addressed in this work is to derive a minimal condition on the number N of rounds of STARK-friendly ciphers to reach a security of s bits. To do so, we developed a common methodology to attack STARK-friendly ciphers.

Let $D(N)$ be the the number of solutions of the algebraic system modelling a STARK-friendly cipher with N rounds. We present an algebraic attack whose complexity is polynomial in the number of solutions, i.e.

$$O\big(D(N)^{\omega}\big), \text{where } \omega \text{ is 1 or 2.}$$

This leads to the following methodology for deriving the minimal number of rounds.

1. **Estimate the number $D(N)$ of solutions**. To do so, we performed practical experiments and used theoretical bounds such as the *Bézout bound* (Definition 1) and the so-called *multi-homogeneous Bézout bound* (Definition 2, [35,38,22]). The former bound is especially interesting since it takes advantage of the structure and leads to tighter complexity results. To our knowledge, this is the first time that such *multi-homogeneous Bézout bounds* are used cryptography.

2. **Choose** N such that :

$$D(N)^\omega > 2^s.$$

Thanks to fast linear algebra techniques (Section 2.2.2, [25,24]), we can always assume that $\omega = 2$. We will see that for some primitives ω can be even chosen equal 1.

In the following tables, we summarize the value of $D(N)$ for `STARK`-friendly ciphers, the number of rounds N initially suggested by the designers to reach a security level of 128 bits and the number of rounds $N^\star$ derived from our new attacks.

1.2 Algebraic Key-Recovery

Here, we provide the results obtained for an algebraic key-recovery against various `AO` block ciphers.

	MiMC	`Jarvis`	`Jarvis-v2`	`Pepper-v1`	`Pepper-v2`	`Rescue`
Field	$\mathbb{F}_{2^n}$	$\mathbb{F}_{2^n}$	$\mathbb{F}_{2^n}$	$\mathbb{F}_p$	$\mathbb{F}_p$	$\mathbb{F}_p$
D	3^N	$(N+1)4^N$	$(5N+3)4^{N-2}$	$4\,N\,3^{N-1}$	3^{N+1}	$3^{(N+1)m}$
N	82	10	10			32
N^*	≥ 81	≥ 62	≥ 62	≥ 77	≥ 80	$\geq 20(m=2)$

Table 3. Minimal number of rounds for `MiMC`, `Jarvis`, `Pepper` and `Rescue`.

	`Vision`		
Field	$\mathbb{F}_{2^n}$		
m	2	3	4
D	$\approx \frac{15}{256}2^{6N}$	$\approx \frac{4615N}{589824}(2^6\,3^2)^N$	$\approx 0.0022(2^{10}3^2)^N$
$N^\star$	12	≥ 8	≥ 6

Table 4. Minimal number of rounds for `Vision`.

1.3 Algebraic Pre-Image

Here, we describe the results obtained for an algebraic pre-image on the permutation induced by `AO` ciphers. In this context, we consider the so-called *Constrained-Input Constrained-Output* (`CICO`) problem introduced in [32]. The `CICO` problem is then capturing a pre-image attack against the permutation defining a sponge hash function [11].

Let perm : $\mathbb{K}^m \longrightarrow \mathbb{K}^m$ be a permutation. The `CICO` problem considered here is defined as follows:

CICO(k)
Find $(x_{k+1}, \ldots, x_t) \in \mathbb{K}^{m-k}$ such that
$\mathbf{x} = (0, \ldots, 0, x_{k+1}, \ldots, x_t) \in \mathbb{K}^m$ // Constrained input
$\mathbf{y} = \text{perm}_{\text{Hades}}(\mathbf{x})$
$y_1 = \cdots = y_k = 0$ // Constrained output

Fig. 1. The CICO problem.

CICO *Problem on* Poseidon *and* Starkad. The results are summarized in Table 5.

Name	Security level [32]	Field	n	R_F	R_P	t	Complexity of our attack	Cost
Poseidon-256	2^{128}	$\mathbb{F}_p$	256	8	84	6	$3^{R_F+R_P-1}$	$2^{144.2}$
Starkad-252	2^{126}	$\mathbb{F}_{2^{63}}$	63	12	39	24	$3^{2(4\,R_F+R_P-8)+2}$	$2^{253.6}$

Table 5. Complexity of our attack against Poseidon and Starkad. n is the bit size of the field, R_F (resp. R_P) refers to to the numbers of full (resp. plain) rounds and t is the number of field elements (parameters of Poseidon and Starkad defined in Section 4.4).

CICO *Problem on* Rescue. In the particular case of Rescue, we considered algebraic key-recovery on the block cipher (Table 3) as well as algebraic pre-image against the Rescue permutation (Table 6).

k	1	2	> 2
Complexity for Rescue	$3^{(N-1)m+k}$	$3^{(N-1)m+k}$	$3^{2\left((N-1)m+k\right)}$

Table 6. Complexity of our attack against Rescue. k is the CICO parameter, N is the number of rounds and m is the number of field elements (parameters of Rescue are defined in Section 4.3.2).

This yields, in particular, for the 128-bit challenge parameters defined for the STARK-Friendly Hash (SFH) Challenge[2]:

Name	Field	$n = \log_2(p)$	N	m	Exhaustive search	Algebraic attack
Rescue128a	$\mathbb{F}_p$	125	16	4	2^{250}	$2^{98.3}$
Rescue128b	$\mathbb{F}_p$	253	22	12	2^{253}	$2^{401.3}$
Rescue128c	$\mathbb{F}_p$	125	10	12	2^{250}	$2^{174.3}$
Rescue128d	$\mathbb{F}_p$	61	10	12	2^{244}	2^{355}
Rescue128e	$\mathbb{F}_p$	253	10	11	2^{253}	$2^{158.5}$

Table 7. Challenge parameters for the Rescue-based hash function.

[2] https://starkware.co/hash-challenge/

1.4 Organisation of the Report

After this introduction, the paper is organized as follows. In Section 2, we introduce the algebraic tools used to analyse STARK-friendly ciphers such as Gröbner bases (Section 2.1) and multi-homogeneous systems (Section 2.3). In Section 3, we briefly review known results in the algebraic cryptanalysis of block ciphers. Section 4 describes the primitives analysed in this report (the ciphers listed in Table 1). We then describe key-recovery attacks against Jarvis (Section 5), MiMC (Section 6), Pepper (Section 7), Rescue (Section 8) and Vision (Section 9). Finally, we present pre-image attacks in Section 10 : against Starkad/Poseidon (Section 10.1) and Rescue (Section 10.2).

This report has been prepared at the demand of Starkware Industries. The analysis has been performed during two periods (see Section 11 an history of the document) : from November 2018 to September 2019 and then in March 2020 (the latest period was dedicated to investigate the CICO problem on Rescue(Section 10.2)). This final document is a compilation of several intermediate reports. The goal of this final report is to present the complexity of our attacks and to detail the main ideas. On the other hand, some proofs are only sketched and few technical details are omitted.

2 Polynomial System System Solving and Gröbner Bases

A central task in the algebraic cryptanalysis of STARK-friendly primitives is to solve a system of non-linear equations over a finite field $\mathbb{F}_q$ (with $q = p^s$, p prime and $s > 0$). This classical NP-hard problem [31] is defined as follows:

Polynomial System Solving over a Finite Field ($\texttt{PoSSo}_q$)
Input. $p_1(x_1, \ldots, x_n), \ldots, p_m(x_1, \ldots, x_n) \in \mathbb{F}_q[x_1, \ldots, x_n]$.
Goal. Find – if any – a vector $(z_1, \ldots, z_n) \in \mathbb{F}_q^n$ such that:

$$p_1(z_1, \ldots, z_n) = 0, \ldots, p_m(z_1, \ldots, z_n) = 0.$$

In this work, we will use Gröbner bases [14,13] for solving the instances of $\texttt{PoSSo}_q$ arising in our algebraic attacks. The rational is that we have efficient algorithms, such as the F_4 and F_5 algorithms [26,27]), as well as software (typically, Magma and Fgb [12,28]) to compute these bases and perform large scale experiments. In addition, a rich set of tools from computer algebra/algebraic geometry allow to understand the complexity of computing such bases. Finally, Gröbner bases algorithms turn to be quite flexible for taking advantage of *structured systems* that naturally appear in algebraic cryptanalysis. All these features will be illustrated in this report.

Before that, we briefly review some basic facts about Gröbner bases, recall basic complexity results and finally introduce the concept of multi-homogeneous systems.

2.1 Gröbner Bases

A *monomial* in $\mathbb{F}_q[x_1, \ldots, x_n]$ is a power product of the variables, i.e. an element of the form $x_1^{\alpha_1} \cdots x_n^{\alpha_n}$. The *leading monomial* of $p \in \mathbb{F}_q[x_1, \ldots, x_n]$ – denoted by $\mathrm{LM}(g, \prec)$ – is the largest monomial w.r.t. some admissible monomial ordering $\prec$ among the monomials of f.

Remark 1. LEX and DRL are two widely used examples of monomial orderings. Given $\alpha = (\alpha_1, \ldots, \alpha_n)$ and $\beta = (\beta_1, \ldots, \beta_n) \in \mathbb{N}^n$, they are defined as follows:

- $x_1^{\alpha_1} \cdots x_n^{\alpha_n} \prec_{\text{LEX}} x_1^{\beta_1} \cdots x_n^{\beta_n}$ if the first left-most non-zero entry of $\beta - \alpha$ is positive.
- $x_1^{\alpha_1} \cdots x_n^{\alpha_n} \prec_{\text{DRL}} x_1^{\beta_1} \cdots x_n^{\beta_n}$ if $\sum_{i=1}^n \alpha_i < \sum_{i=1}^n \beta_i$, or $\sum_{i=1}^n \alpha_i = \sum_{i=1}^n \beta_i$ and the right-most non-zero entry of $\beta - \alpha$ is negative.

We can now define Gröbner basis [14,13].

Definition 2.1 (Gröbner basis). *Let $\mathcal{I} = \langle p_1, \ldots, p_m \rangle \subset \mathbb{F}_q[x_1, \ldots, x_n]$ be a polynomial ideal. A subset $G \subset \mathcal{I}$ is a Gröbner basis – w.r.t. an admissible monomial ordering $\prec$ – of $\mathcal{I}$ if:*

$$\forall p \in \mathcal{I}, \text{ there exists } g \in G \text{ such that } \mathrm{LM}(g, \prec) \text{ divides } \mathrm{LM}(p, \prec).$$

It is clear from Definition 2.1, that the notion of Gröbner bases depends on a admissible monomial ordering. Gröbner bases have different computational and algorithmic properties with respect to the monomial ordering considered. Typically, `LEX`-Gröbner bases (i.e. Gröbner bases w.r.t. the lexicographical ordering) allow to eliminate variables whilst `DRL` is a more computational-friendly ordering as we will see.

In algebraic cryptanalysis, we are usually not interested in the Gröbner basis itself but rather in the set of solutions, or *variety*.

Definition 2.2 (Variety). *Let $\mathbb{F}_q \subset \mathbb{L}$ and $\mathcal{I} = \langle p_1, \ldots, p_m \rangle \subset \mathbb{F}_q[x_1, \ldots, x_n]$ be an ideal. We denote by*

$$V_{\mathbb{L}}(\mathcal{I}) = V_{\mathbb{L}}(p_1, \ldots, p_m) = \left\{ \mathbf{z} = (z_1, \ldots, z_n) \in \mathbb{L}^n \mid p_i(\mathbf{z}) = 0, \forall i, 1 \leq i \leq m \right\},$$

the $\mathbb{L}$-variety associated to $\mathcal{I}$, i.e. the common zeroes over $\mathbb{L}^n$ of $p_1, \ldots, p_m$. When $\mathbb{L} = \overline{\mathbb{F}_q}$, we simply denote $V(\mathcal{I}) = V_{\overline{\mathbb{F}_q}}(\mathcal{I})$. We shall say that the ideal $\mathcal{I} \subset \mathbb{F}_q[x_1, \ldots, x_n]$ is zero-dimensional if $\mid V(\mathcal{I}) \mid < \infty$.

Let $\mathcal{I} \subset \mathbb{F}_q[x_1, \ldots, x_n]$ be an ideal. A Gröbner basis allows to explicitly compute in the quotient space $\mathbb{F}_q[x_1, \ldots, x_n]/\mathcal{I}$. In the zero-dimensional case, this quotient is a finitely generated vector-space.

Proposition 1. *Let $\mathcal{I} = \langle p_1, \ldots, p_m \rangle \subset \mathbb{F}_q[x_1, \ldots, x_n]$ be a zero-dimensional ideal and $G \subset \mathcal{I}$ be a Gröbner basis of $\mathcal{I}$ w.r.t. an admissible monomial ordering $\prec$. $\mathbb{F}_q[x_1, \ldots, x_n]/\mathcal{I}$ is a finite-dimensional vector space generated by :*

$$\left\{ \mathbf{x}_\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n} \mid \mathbf{x}^\alpha \notin \left\langle \mathrm{LM}(g, \prec) \mid g \in G \right\rangle \right\}.$$

A `LEX`-Gröbner basis permits to eliminate variables.

Theorem 1 (Elimination theorem). *Let $\mathcal{I} = \langle p_1, \ldots, p_m \rangle \subset \mathbb{F}_q[x_1, \ldots, x_n]$ be a polynomial ideal and $G \subset \mathcal{I}$ be a `LEX`-Gröbner basis of $\mathcal{I}$. It holds that:*

$$\forall \ell, 0 \leq \ell < n, G_\ell = G \cap \mathbb{F}_q[x_{\ell+1}, \ldots, x_n],$$

is a `LEX`-Gröbner basis of $\mathcal{I} \cap \mathbb{F}_q[x_{\ell+1}, \ldots, x_n]$.

In the zero-dimensional case, the variety can be then easily computed from a `LEX`-Gröbner basis. Indeed, we can deduce from the elimination property (Theorem 1) that a `LEX`-Gröbner

basis of a zero-dimensional ideal has always the following triangular shape:

$$\begin{cases} h_n(x_n) \\ h_{n-1}^{(1)}(x_n, x_{n-1}) \\ \dots \\ h_{n-1}^{(k_1)}(x_n, x_{n-1}) \\ h_{n-2}^{(1)}(x_n, x_{n-1}, x_{n-2}) \\ \vdots \end{cases}$$

To compute the variety from a LEX-Gröbner basis, we then simply have to successively eliminate variables by computing zeroes of univariate polynomials and back-substituting the results.

2.2 Zero-Dimensional Solving

From a practical point of view, computing (directly) a LEX-Gröbner basis is usually slower that computing a Gröbner basis w.r.t. another monomial ordering. On the other hand, it is known that computing Gröbner bases w.r.t. to a degree reverse lexicographical (DRL-Gröbner bases) is much faster in practice.

The FLGM algorithm [29] permits – in the zero-dimensional case – to efficiently solve this issue. This algorithm use the knowledge of a Gröbner basis computed for a given order to construct a Gröbner for another order. The complexity of FGLM is polynomial in the number of solutions of the considered ideal. This suggests the following strategy for computing the solutions of a zero-dimensional system $p_1 = 0, \dots, p_m = 0$.

1. Compute a DRL-Gröbner basis $G_{\texttt{DRL}}$ of $\mathcal{I} = \langle p_1, \dots, p_m \rangle \subset \mathbb{F}_q[x_1, \dots, x_n]$.
2. Compute a LEX-Gröbner basis of $\mathcal{I}$ using FGLM on $G_{\texttt{DRL}}$.

Fig. 2. Zero-dim solving steps.

This is for instance the default strategy used in the MAGMA computer algebra system. We detail now the complexity of each part.

2.2.1 Complexity of Computing a DRL-Gröbner Bases.

The complexity of computing a DRL-Gröbner bases will be related to the quantity defined below.

Definition 2.3 (Degree of regularity). *Let $p_1, \dots, p_m \in \mathbb{F}_q[x_1, \dots, x_n]$ be homogeneous polynomials. We shall call degree of regularity of $p_1, \dots, p_m$, denoted by $D_{\text{reg}}(p_1, \dots, p_m)$, the smallest integer $D_0 \geq 0$ such that the polynomials of degree D_0 in $\mathcal{I} = \langle p_1, \dots, p_m \rangle$ generate – as a $\mathbb{F}_q$ vector space – the set $\texttt{Monomials}_q(n, D_0)$ of all monomials of degree D_0 in n variables over $\mathbb{F}_q$, i.e.*

$$D_{\text{reg}}(p_1, \dots, p_m) = \min \left\{ D_0 \geq 0 \mid \dim_{\mathbb{F}_q}\left(\{p \in \mathcal{I} \mid \deg(p) = D_0\}\right) = \#\texttt{Monomials}_q(n, D_0) \right\}.$$

Remark 2. The degree of regularity can be generalized to non-homogeneous polynomials $\mathbf{p} = (p_1, \ldots, p_m) \in \mathbb{F}_q[x_1, \ldots, x_n]^m$ by considering the degree of regularity of the homogeneous components of highest degrees $\mathbf{p}^{\mathrm{H}} = (p_1^{\mathrm{H}}, \ldots, p_m^{\mathrm{H}}) \in \mathbb{F}_q[x_1, \ldots, x_n]^m$. Namely :

$$D_{\mathrm{reg}}(p_1, \ldots, p_m) = D_{\mathrm{reg}}(p_1^{\mathrm{H}}, \ldots, p_m^{\mathrm{H}}).$$

Once this notion fixed, we can rather easily establish an upper bound on the cost of computing a DRL-Gröbner basis [37,33,6,9,8].

Theorem 2. *Let $\omega, 2 \leq \omega \leq 3$ be the linear algebra constant and $p_1, \ldots, p_m \in \mathbb{F}_q[x_1, \ldots, x_n]$ such that $\mathbf{p}^{\mathrm{H}} = (p_1^{\mathrm{H}}, \ldots, p_m^{\mathrm{H}})$ is a zero-dimensional system. Let then $D_{\mathrm{reg}} = D_{\mathrm{reg}}(p_1, \ldots, p_m)$ and $\prec$ be a total degree monomial ordering. We can compute a Gröbner basis of $\langle p_1, \ldots, p_m \rangle$ with respect to $\prec$ in*

$$O\left(m \cdot \binom{n + D_{\mathrm{reg}}}{D_{\mathrm{reg}}}^{\omega}\right) \text{ arithmetic operations over } \mathbb{F}_q. \tag{1}$$

with $2 \leq \omega < 2.3728639$ the linear algebra constant [44,30].

The complexity of computing a DRL-Gröbner basis is then exponential in the degree of regularity. Unfortunately, this degree of regularity is difficult to compute in general; as difficult as computing the Gröbner basis. Fortunately, there is a particular class of systems for which this degree can be computed efficiently : (*regular* and) *semi-regular sequences*.

The notion of regular sequences is classical [37,36] but holds only for $m \leq n$. Semi-regular sequences, that we recall below, have been introduced in [6,9] for over-defined systems. In essence, the algebraic systems encountered in the cryptographic context are naturally over-defined due to the field equations motivating then such a notion.

Definition 2.4. *We assume that $m > n$ and $q > 2$. Let $p_1, \ldots, p_m \in \mathbb{F}_q[x_1, \ldots, x_n]$ be homogeneous polynomials of degrees $d_1, \ldots, d_m$ respectively. This sequence is semi-regular if:*

1. $\langle p_1, \ldots, p_m \rangle \neq \mathbb{F}_q[x_1, \ldots, x_n]$,
2. *for all $i, 1 \leq i \leq m$ and $g \in \mathbb{F}_q[x_1, \ldots, x_n]$:*

$$\deg(g \cdot p_i) < D_{\mathrm{reg}}(p_1, \ldots, p_m) \text{ and } g \cdot p_i \in \langle p_1, \ldots, p_{i-1} \rangle \Rightarrow g \in \langle p_1, \ldots, p_{i-1} \rangle.$$

Now, let $p_1, \ldots, p_m \in \mathbb{F}_q[x_1, \ldots, x_n]$ be polynomials of degrees $d_1, \ldots, d_m$ respectively. We shall say that the sequence $p_1, \ldots, p_m$ is semi-regular if the sequence $p_1^{\mathrm{H}}, \ldots, p_m^{\mathrm{H}}$ is semi-regular.

Regular sequences are defined almost as in Definition 2.4. The only difference is that $m \leq n$ and the second condition is simply: $g \cdot p_i \in \langle p_1, \ldots, p_{i-1} \rangle \Rightarrow g \in \langle p_1, \ldots, p_{i-1} \rangle$. Definition 2.4 of semi-regular sequences requires to be adapted for the Boolean polynomial ring $\mathbb{F}_2[x_1, \ldots, x_n]/\langle x_i^2 - x_i \rangle_{1 \leq i \leq n}$ [6,9] to take into account the field equations.

Remark 3. Note that semi-regular sequences can be also defined from a more algorithmic point of view. Semi-regular sequences can be defined as the sequences for which all the matrices generated during a Gröbner basis computation with F_5 are of maximal possible rank [27].

The degree of regularity can be computed explicitly for semi-regular sequences [6,7]. It is derived from a particular coefficient in a power series.

Definition 2.5. *Let $S(z) = \sum_{k\geq 0} c_k z^k \in \mathbb{N}[[z]]$ be a formal power series. We define the index* $\mathrm{Ind}(S)$ *as the first k such that $c_k \leq 0$ (if no such k exist, then* $\mathrm{Ind}(S) = \infty$*). We denote by:*

$$[S(z)]_+ = \sum_{k=0}^{\mathrm{Ind}(S)-1} c_k z^k, \text{ the series truncated at } \mathrm{Ind}(S).$$

We have then:

Property 1. A sequence $p_1, \ldots, p_m \in \mathbb{F}_q[x_1, \ldots, x_n]$ of respective degrees $d_1, \ldots, d_m$ is semi-regular if and only if its Hilbert series is given by:

$$\mathrm{HS}_q(t) = \left[\frac{\prod_{i=1}^{m}(1-z^{d_i})}{(1-z)^n}\right]_+, \text{ if } m > n \text{ and } q > 2, \text{ and } \mathrm{HS}_2(t) = \left[\frac{(1+z)^n}{\prod_{i=1}^{m}(1+z^{d_i})}\right]_+, \text{ if } q = 2.$$

The degree of regularity of a semi-regular sequence $p_1, \ldots, p_m \in \mathbb{F}_q[x_1, \ldots, x_n]$ is given by $1 + \deg\left(\mathrm{HS}_q(t)\right)$.

Remark 4 (Macaulay bound). For regular sequences ($m \leq n$), the degree of regularity is given by the so-called *Macaulay bound* [37,33]:

$$\sum_{i=1}^{m}(d_i - 1) + 1. \tag{2}$$

For quadratic polynomials, the bound is $n + 1$.

2.2.2 Change of Ordering. We now consider the last step in the zero-dim solving process. To date, FLGM [29] is the fastest technique for the change of ordering. We recall below the complexity of the initial FLGM.

Theorem 3. *Let $\mathcal{I} \subset \mathbb{F}_q[x_1, \ldots, x_n]$ be a zero-dimensional ideal and $G_{\prec_{\mathrm{old}}}$ be a $G_{\prec_{\mathrm{old}}}$-Gröbner basis of $\mathcal{I}$ (w.r.t. to an admissible monomial ordering $\prec_{\mathrm{old}}$).* FGLM *[29] permits to compute a $\prec_{\mathrm{old}}$-Gröbner basis $G_{\prec_{\mathrm{new}}}$ of $\mathcal{I}$ knowing $G_{\prec_{\mathrm{old}}}$ in*

$$\mathcal{O}(n \cdot D^3),$$

with D being the dimension of the $\mathbb{F}_q$-vector space $\mathbb{F}_q[x_1, \ldots, x_n]/\mathcal{I}$ (which is equivalent to the number of zeroes of $\mathcal{I}$ counted with their multiplicities).

The complexity of the version described in [29] can be improved using fast linear algebra techniques [25,24]. These lead to a version of FGLM whose complexity is :

$$\mathcal{O}(n \cdot D^\omega), \text{with } 2 \leq \omega < 2.3728639.$$

In this report, we can always assume that ω is at most two. For some `STARK`-friendly primitives, the cost of the change of ordering can be further reduced to:

$$\mathcal{O}(n \cdot D).$$

This is linear in the number of solutions.

In any case, it is clear that the value of D is crucial for the complexity of FGLM and solving $\mathtt{PoSSo}_q$. We recall below a classical bound, known as *Bézout bound*, on the number of solutions.

Definition 1 (Bézout bound). *Let $\mathcal{I} = \langle p_1, \ldots, p_n \rangle \subset \mathbb{F}_q[x_1, \ldots, x_n]$ be a zero-dimensional ideal and $d_1, \ldots, d_n$ be the degrees of $p_1, \ldots, p_m$ respectively. We have :*

$$\#V(\mathcal{I}) \leq \prod_{i=1}^{n} d_i.$$

In the case of STARK-friendly primitives, the change of ordering is the dominant part of the solving process. It is then of primary importance to find tighter bounds on the number of solutions.

2.3 Multi-Homogeneous Systems

The systems arising in algebraic cryptanalysis behave usually differently than semi-regular systems (Definition 2.4). In this report, we shall see that the systems derived from many STARK-friendly ciphers have a multi-homogeneous structure [35,38,22].

Definition 2.6. *Let $\mathbf{p} = (p_1, \ldots, p_m) \in \mathbb{F}_q[x_1, \ldots, x_n]^m$ be homogeneous polynomials. We consider a partition $\{\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(\ell)}\}$ of the variables $\mathbf{X} = \{x_1, \ldots, x_n\}$.*

- *We shall say that $\mathbf{p}$ is multi-homogeneous if the polynomials p_i are homogeneous w.r.t. the $\mathbf{X}^{(j)}$'s.*
- *The finite sequence, denoted by $\mathrm{mdeg}(f)$, of degrees with respect to each block is called the multi-degree of f.*
- *Let $f \in \mathbb{K}[\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(\ell)}]$ be a multi-homogeneous polynomial of multi-degree $(d_1, \ldots, d_\ell)$. It holds that:*

$$\forall (\alpha_1, \ldots, \alpha_\ell) \in (\mathbb{F}_q^*)^\ell = f\left(\alpha_1 \mathbf{X}^{(1)}, \ldots, \alpha_\ell \mathbf{X}^{(\ell)}\right) = \left(\prod_{i=1}^{\ell} \alpha_i^{d_i}\right) f\left(\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(\ell)}\right).$$

In the case of multi-homogeneous systems, we have a dedicated version of the Bézout bound (Definition 1).

Definition 2 (Multi-homogeneous Bézout bound). *Let $\mathbf{p} = (p_1, \ldots, p_n) \in \mathbb{F}_q[\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(\ell)}]^n$ be a set of multi-homogeneous polynomials of multi-degrees $\mathrm{mdeg}(f_i) = (d_{i,1}, \ldots, d_{i,\ell})$. Then, the number of zeroes of $\mathbf{p}$ is bounded from above by the coefficient of $\alpha_1^{|\mathbf{X}^{(1)}|} \cdot \alpha_2^{|\mathbf{X}^{(2)}|} \cdots \alpha_\ell^{|\mathbf{X}^{(\ell)}|}$ in the polynomial :*

$$\prod_{i=1}^{n} \left(d_{i,1} + d_{i,2} + \cdots + d_{i,\ell}\right).$$

The multi-homogeneous Bézout bound is a central tool that will be used to derive tighter complexity bounds of our algebraic attacks.

Remark 5. Note that the problem of finding an optimal multi-homogeneous structure minimizing the multi-homogeneous Bézout bound is NP-Hard.

3 Algebraic Cryptanalysis of Block Ciphers

Algebraic cryptanalysis of block ciphers has been popularized by [19]. The authors demonstrated that key-recovery on AES [20] can be reduced to the problem of solving a system of quadratic equations over $\mathbb{F}_2$.

Given a pair of message/ciphertext, the basic idea of such modelling is to introduce intermediate variables for each round as well as variables corresponding to the secret-key. In AES, the Sbox $S_{\texttt{AES}} : \mathbb{F}_{2^8} \to \mathbb{F}_{2^8}$ is the inverse function in $\mathbb{F}_{2^8}$. That is:

$$S_{\texttt{AES}}(X) = \begin{cases} 1/X, \text{ if } X \neq 0 \\ 0, \text{otherwise} \end{cases}$$

If $X \neq 0$, then $X \times S_{\texttt{AES}}(X) = 1$. This yields 8 quadratic equations over $\mathbb{F}_2$.

Initially, it was claimed in [19] that the complexity of solving the whole key-recovery system modelling AES

> *"does not grow exponentially with the number of rounds".*

This claim was proved to be false in a series of papers, e.g. [17,21,4].

Despite this false claims on the complexity, the results of [19] motivated to further investigate the behaviour of algebraic attacks against block ciphers. We recall below a selection of these results.

Alternative modelling. In [40], the authors introduced the so-called *Big Encryption Standard* (BES). This is an *embedding* of AES over $\mathbb{F}_{2^8}$ leading to an alternative algebraic modelling of AES with multivariate polynomials over $\mathbb{F}_{2^8}$. these equations are of higher degree than in the previous modelling [19]. In the case of BES, the SBox is represented as

$$S_{\texttt{BES}}(X) = X^{2^8-2} = X^{254}.$$

Practical experiments. In order to understand the practical behaviour of algebraic attacks, [18] proposed a general framework to perform such attacks on so-called *small scale variants* of AES. Even for such small scale variants, it turns to be difficult to perform relevant practical experiments.

A Gröbner basis for AES*.* In [16], the authors proved a rather surprising result. The algebraic equations $G_{\texttt{AES}}$ describing a key-recovery in AES already forms a zero-dimensional Gröbner basis for a suitable monomial ordering (the degree-lexicographical ordering). To do so, [16] considered the BES-style modelling of AES over $\mathbb{F}_{2^8}$ [40].

As explained in Section 2.2.1, the cost of computing the variety is the cost of the change of ordering. Thus, the complexity of this approach is essentially polynomial in number of solutions of the algebraic system over $\mathbb{F}_{2^8}$. A bound on the number of solutions can be easily deduced from the shape of the leading terms of $G_{\texttt{AES}}$. In the case of AES128, the number of solutions D (counted with multiplicities) is:

$$D = 254^{200} \approx 2^{1598}. \tag{3}$$

Remark that 254 is the degree of the polynomial representation of the inverse function in $\mathbb{F}_{2^8} = \mathbb{F}_{256}$. Note also that the cost of the change of ordering is polynomial in (3). Thus, the attack has no direct impact on the security of AES. Still, this approach provides a first rigorous (upper) bound on the complexity of an algebraic attack against AES. It also emphasizes the importance of the number of solutions D in the cost of algebraic attacks against block ciphers.

Block Ciphers Sensitive to Gröbner bases. The idea of [16] has been be generalized in [15] to a large class of block ciphers. In particular, the authors introduces two parametrized families of block ciphers – `Curry` (SPN) and `Flurry` (Feistel) – that have a sound design strategy against linear and differential attacks but which can be modelled by very simple polynomial equations. Then, they generalized the approach described in the previous paragraph for `AES` on `Curry` and `Flurry`. This demonstrates the existence of block ciphers resistant against differential and linear cryptanalysis but vulnerable against Gröbner basis attacks.

4 `STARK`-Friendly Ciphers

We briefly describes here the `STARK`-friendly ciphers that have been considered in this report. Namely:

`Jarvis` [5]	The first generation of block cipher optimized for `STARK`-proofs
`MiMC` [2]	A lightweight block cipher well suited for the `STARK` technology
`Pepper`/`Rescue`/ `Vision` [3]	A second generation of `AO` ciphers fixing some weaknesses of `Jarvis`
`Starkad`/`Poseidon` [32]	Latest design of `STARK`-friendly hash functions

We review here the main components of these primitives and refer to the initial papers for details regarding the design rational. As mentioned in the introduction, we describe the ciphers with the specification provided by `Starkware` Industries.

4.1 `Jarvis`

The state of `AES` is a matrix composed by elements in $\mathbb{F}_{2^8}$ transformed through a certain number of rounds with operations in $\mathbb{F}_{2^8}$ or in $\mathbb{F}_2$. `Jarvis` is a generalization of `AES`, or more precisely of `BES`, whose state is reduced to a single field element of $\mathbb{F}_{2^n}$ and involves few simple operations in $\mathbb{F}_{2^n}$. It borrows to `AES`, the use of the inverse as a non-linear component as well as *quasi-linear* polynomials for the linear layer. We recall that $L(X) \in \mathbb{F}_{2^n}[X]$ is a quasi-linear polynomial if it is defined as follows :

$$L(X) = c_{-1} + \sum_{i=0}^{n-1} c_i X^{2^i}.$$

In particular, `Jarvis` uses a special polynomial $A(X) \in \mathbb{F}_{2^n}[X]$ chosen such that:

$$A(X) = C\big(B^{-1}(X)\big) \tag{4}$$

where $C(X) \in \mathbb{F}_{2^n}[X]$ and $B(X) \in \mathbb{F}_{2^n}[X]$ quasi-linear polynomials of degree 4 which define a permutation.

Algorithm 1 `Jarvis`

Input: plaintext $M \in \mathbb{F}_{2^n}$ and a secret-key $K_0 \in \mathbb{F}_{2^n}$
$K_1, \ldots, K_N \leftarrow$ `KeyExpand`(K_0)
$S_0 \leftarrow M + K_0$
for r **from** 1 **to** N **do**
    $S_r \leftarrow A(1/S_{r-1}) + K_r$ (*with* $A(X) \in \mathbb{F}_{2^n}[X]$ *being defined as in* (4))
end for

Algorithm 2 `KeyExpand`

Input: $K_0 \in \mathbb{F}_{2^n}$
Internal constant: $C_0 \in \mathbb{F}_{2^n}$
$C_1, \ldots, C_N \leftarrow$ `ConsGen`(C_0)
for r **from** 1 **to** N **do**
    $K_r \leftarrow 1/K_{r-1} + C_r$
end for
return $K_1, \ldots, K_N \in \mathbb{F}_{2^n}$

Algorithm 3 `ConsGen`

1: **Input:** $C_0 \in \mathbb{F}_{2^n}$
2: **Internal constants:** $a, b \in \mathbb{F}_{2^n}$
3: **for** r **from** 1 **to** N **do**
4: $\quad C_r \leftarrow a\, C_{r-1} + b$
5: **end for**
6: **return** $C_1, \ldots, C_N \in \mathbb{F}_{2^n}$

Security analysis. We give below the parameters initially recommended by the designers of `Jarvis` [5] for various security levels.

Name	n	N
`Jarvis128`	128	10
`Jarvis192`	192	12
`Jarvis256`	256	14

Recently, and in parallel of this work, the authors of [1] presented an improved algebraic attack against `Jarvis`. In particular, [1] has been able to exploit the particular structure of the linear layer, i.e. the polynomial A as defined in (4), to improve the algebraic modelling of `Jarvis`. They then rely on generic bounds on the degree of regularity and number of solutions (Section 2) to derive new complexity bounds on the cost of an algebraic attack.

Name	n	N	New security bound [1]
`Jarvis128`	128	10	72
`Jarvis192`	192	12	85
`Jarvis256`	256	14	92

Table 8. Complexity of the attack of [1] (assuming $\omega = 2$).

The authors [1] also mentioned that at least $N = 18$ rounds are necessary to reach a security level of 128 bit according to this new attack. With our new algebraic attack, we show that number of rounds should be, at least, $N = 62$ to reach this security level (Table 3).

Finally, the authors [1] observed that their algebraic attack behaves better in practice than expected. In particular, they observed that the number of solutions $D(N) \approx 2 \times 5^N$. In Theorem 5.3, we prove that $D(N) = (N+1)4^N$.

4.2 `MiMC`

The `MiMC` family operates on finite field $\mathbb{K}$ that can be a binary extension $\mathbb{F}_{2^n}$ or a prime field $\mathbb{F}_p$. In this report, we have considered the (SPN) version of `MiMC` where the non-linear component is the cubic Sbox (i.e. $d = 3$).

Algorithm 4 `MiMC`

Input: plaintext $M \in \mathbb{K}$ and a secret-key $K \in \mathbb{K}$
Internal constants: $C_1, \ldots, C_N \in K$
$S_1 \leftarrow \big(M + (K + C_1)\big)^d$
for r **from** 1 **to** N **do**
    $S_r \leftarrow \big(S_{r-1} + (K + C_r)\big)^d$
end for
return S_N

The internal constants $C_1, \ldots, C_N \in \mathbb{K}$ are chosen randomly.

Remark 6. We provide in Appendix A some reference codes for implementing Algorithm 4 in Sage, Magma and Maple.

Number of Rounds. According to the designers [2], the number of rounds N required to reach a security level s is equal to :

$$N = \left\lceil \frac{s}{\log_2 3} \right\rceil.$$

More recently, the authors of [1] quickly analysed `MiMC` against algebraic attacks. In particular, they observed than a natural algebraic modelling of `MiMC` has a number of solutions $D(N) \approx 3^N$. We also prove this result in Theorem 5.3.

4.3 `Pepper/Rescue/Vision`

In [5], the authors set the basis on the design of `AO` ciphers. This leads to the proposal of two new block ciphers `Vision` and `Rescue` optimized for the `STARK`-technology. These are SPN block ciphers operating on fields of odd and even characteristic, respectively. In contrast to `Jarvis` (Section 4.1), the state space in `Vision` and `Rescue` are composed of more than one element.

The goal of [5] was also to fix initial weaknesses quickly identified on `Jarvis`. In this process, we also analysed intermediate versions such as `Jarvis`-v2 (one dimensional version of `Vision`) and `Pepper` (evolution of `Jarvis` that removed the linear layer and uses two Sboxes).

4.3.1 `Pepper`**.** The first version of `Pepper`-v1 operates on a prime field $\mathbb{K} = \mathbb{F}_p$. We set $d = 3, c = (2p-1)/3$ and $N' = 2N$.

Algorithm 5 `Pepper`-v1

Input: plaintext $M \in \mathbb{K}$ and a secret-key $K_0 \in \mathbb{K}$
$K_1, \ldots, K_{N'} \leftarrow$ `KeyExpand`(K_0)
$S_0 \leftarrow M$
for r **from** 1 **to** N **do**
    $S_r \leftarrow\leftarrow \big(S_{r-1}^d + K_{2r-1}\big)^c + K_{2r}$
end for
return S_N

Algorithm 6 `KeyExpand`

Input: $K_0 \in \mathbb{K}$
Internal constant: $C_0 \in \mathbb{K}$
$C_1, \ldots, C_{N'} \leftarrow$ `ConsGen`(C_0)
for r **from** 1 **to** N' **do**
    $K_r \leftarrow 1/K_{r-1} + C_r$
end for
return $K_1, \ldots, K_{N'} \in \mathbb{K}$

Algorithm 7 `ConsGen`

Input: $C_0 \in \mathbb{K}$
Internal constants: $a, b \in \mathbb{F}_{2^n}$
for r **from** 1 **to** N' **do**
    $C_r \leftarrow a\, C_{r-1} + b$
end for
return $C_1, \ldots, C_{N'} \in \mathbb{K}$

The second version, denoted `Pepper`-v2, simplifies the key expansion and add a key addition on the message.

Algorithm 8 `Pepper-v2`

Input: plaintext $M \in \mathbb{K}$ and a secret-key $K_0 \in \mathbb{K}$
$K_1, \ldots, K_{N'} \leftarrow$ `KeyExpand`(K_0)
$S_0 \leftarrow M + K_0$
for r **from** 1 **to** N **do**
    $S_r \leftarrow\leftarrow \left(S_{r-1}^d + K_{2r-1}\right)^c + K_{2r}$
end for
return S_N

Algorithm 9 `KeyExpand`

Input: $K_0 \in \mathbb{K}$
Internal constants: $a, b \in \mathbb{F}_{2^n}$
for r **from** 1 **to** N' **do**
    $K_r \leftarrow a\, K_{r-1} + b$
end for
return $K_1, \ldots, K_{N'} \in \mathbb{K}$

4.3.2 `Rescue`**.** In this case, the finite field in $\mathbb{F}_p$. The Sboxes are the power maps X^d and X^c, where d is the smallest prime such $\gcd((p-1), d) = 1$. In most cases, we can set $d = 3$ and $c = (2p-1)/3$. This is the situation analysed in this report. Finally, we set $N' = 2N$.

Algorithm 10 `Rescue`

Input: plaintext $M \in \mathbb{F}_p^m$ and a secret-key $K_0 \in \mathbb{F}_p^m$
$K_1, \ldots, K_{N'} \leftarrow$ `KeyExpand`(K_0)
$S_0 \leftarrow M + K_0$
for r **from** 1 **to** N **do**
    $S \leftarrow {}^T\left[S_{r-1}[1]^d, \ldots, S_{r-1}[m]^d\right]$
    $S' \leftarrow \mathcal{M}S + K_{2r-1}$
    $S'' \leftarrow {}^T\left[S'[1]^c, \ldots, S'[m]^c\right]$
    $S_r \leftarrow \mathcal{M}S'' + K_{2r}$
end for
return S_N

Algorithm 11 `KeyExpand`

Input: $K_0 \in \mathbb{F}_p^m$
Internal constants: $a, b \in \mathbb{F}_p$
$B \leftarrow {}^T[b, \ldots, b]$
for r **from** 1 **to** N' **do**
    $K_r \leftarrow a\, K_{r-1} + b$
end for
return $K_1, \ldots, K_{N'} \in \mathbb{F}_p^m$

Number of Rounds. According to [5], the number of rounds N required to reach a security level $m \cdot \log_2(p)$ is equal to :

$$\max\left(10, 2\left\lceil \frac{\log_2(p)}{4} \right\rceil\right).$$

4.3.3 Hash Functions. Once the secret-key is fixed in `Rescue`, then Algorithm 10 defines a permutation, denoted as:

$$\text{perm}_{\text{Rescue}} : \mathbb{F}_q^m \longrightarrow \mathbb{F}_q^m. \tag{5}$$

As explained in [5], $\text{perm}_{\text{Rescue}}$ can be precessed into a hash function thanks to the sponge construction [11]. In this case, the value of the secret-key is fixed to a random value. The state of a sponge function is defined as $m = r + c$ bits, where r and c are called the *rate* and the *capacity* of the sponge, respectively. In particular, we focused our attention the following 128-bit challenge parameters[3].

4.3.4 `Vision`**.** This block cipher [5] operates on the finite field $\mathbb{F}_{2^{n/m}}$. The non-linear component is the inverse function in $\mathbb{F}_{2^{n/m}}$ and it uses a quasi-linear permutation $B(X) \in \mathbb{F}_{2^{n/m}}[X]$ of degree d (d is usually 4). Finally, the state is also mixed thanks to an invertible matrix $\mathcal{M} \in \mathbb{F}_{2^{n/m}}^{m \times m}$. Finally, we set $N' = 2N$.

[3] https://starkware.co/hash-challenge/

Name	Field	$n = \log_2(p)$	N	m	c
`Rescue128a`	$\mathbb{F}_p$	125	16	4	2
`Rescue128b`	$\mathbb{F}_p$	253	22	12	1
`Rescue128c`	$\mathbb{F}_p$	125	10	12	2
`Rescue128d`	$\mathbb{F}_p$	61	10	12	4
`Rescue128e`	$\mathbb{F}_p$	253	10	11	1

Table 9. Challenge parameters for the `Rescue`-based hash function.

Algorithm 12 `Vision`(N, d, m, n)

Input: plaintext $M \in \mathbb{F}_{2^{n/m}}^m$ and a secret-key $K_0 \in \mathbb{F}_{2^{n/m}}^m$
$K_1, \ldots, K_{N'} \leftarrow$ `KeyExpand`(K_0)
$S_0 \leftarrow M + C_0$
for r **from** 1 **to** $N - 1$ **do**
  $S \leftarrow^T [B(1/S_{r-1}[1]), \ldots, B(1/S_{r-1}[m])]$
  $S \leftarrow \mathcal{M}S + K_{2r-1}$
  $S \leftarrow^T [B^{-1}(1/S[1]), \ldots, B^{-1}(1/S[m])]$
  $S_r \leftarrow \mathcal{M}S + K_{2r}$
end for
$S \leftarrow^T [1/S_{N-1}[1], \ldots, 1/S_{N-1}[m]]$
$S_N \leftarrow \mathcal{M}S + K_{2N-1}$
return S_N

Algorithm 13 `KeyExpand`

Input: $K_0 \in \mathbb{F}_{2^{n/m}}^m$
Internal constants: $a, b \in \mathbb{F}_{2^{n/m}}^m$
$B \leftarrow^T [b, \ldots, b]$
for r **from** 1 **to** N' **do**
  $K_r \leftarrow a\, K_{r-1} + B$
end for
return $K_1, \ldots, K_{N'} \in \mathbb{F}_{2^{n/m}}^m$

Number of Rounds. According to [5], the number of rounds N required to reach a security level n is equal to :

$$\max\left(10, 2\left\lceil \frac{n}{5.5m} \right\rceil\right).$$

Variant of `Jarvis`*.* We will have `Jarvis`$-$v2$(N, n) =$ `Vision`$(N, d, 1, n)$.

4.4 `Poseidon` and `Starkad`

To date, `Poseidon` and `Starkad` this are the last primitives especially optimized for `STARK` applications. `Starkad` and `Poseidon` [32] are hash functions that operate on a binary field and prime field respectively. More precisely, `Starkad` and `Poseidon` are constructed from a permutation and transformed into a hash function thanks to the sponge construction [32].
To describe the permutations, we use the following notations :

- $\mathbb{K}$ a finite field
- MDS is $t \times t$ matrix,
- $C[\ldots]$ are round constants
- P is an extra parameter and is usually equal to 1.
- d is the degree of the S-BOX and is usually equal to 3.
- The field is chosen so that $x \longrightarrow x^d$ is a *permutation.*

Algorithm 14 `Starkad`/`Poseidon`(R_F, R_P, t, d, P)

Input: $\mathbf{s} \in \mathbb{K}^t$
$R_f \leftarrow R_F/2$
for r **from** 1 **to** R_f **do**
  $\mathbf{s}[i] \leftarrow \left(\mathbf{s}[i] + C[t\,(r-1)+i]\right)^d$ for $i = 1, \ldots, t$
  $\mathbf{s} \leftarrow \mathbf{s} \times \mathrm{MDS}$
end for
for r **from** 1 **to** R_P **do**
  $\mathbf{s}[i] \leftarrow \left(\mathbf{s}[i] + C[t\,(R_f + r - 1) + i]\right)^d$ for $i = 1, \ldots, P$
  $\mathbf{s}[i] \leftarrow \mathbf{s}[i] + C[t\,(R_f + r - 1) + i]$ for $i = (P+1), \ldots, t$
  $\mathbf{s} \leftarrow \mathbf{s} \times \mathrm{MDS}$
end for
for r **from** 1 **to** $R_f - 1$ **do do**
  $\mathbf{s}[i] \leftarrow \left(\mathbf{s}[i] + C[t\,(R_f + R_P + r - 1) + i]\right)^d$ for $i = 1, \ldots, t$
  $\mathbf{s} \leftarrow \mathbf{s} \times \mathrm{MDS}$
  $\mathbf{s}[i] \leftarrow \left(\mathbf{s}[i] + C[t\,(R_F + R_P + r - 1) + i]\right)^d$ for $i = 1, \ldots, t$
end for
Output : $\mathbf{s}$

Remark 7. A reference code for Algorithm 14 is provided in Appendix B.

We report some typical values (S-Box is $x \longrightarrow x^3$). We also give another value so that $n\,k \approx 256$. To simplify the description we introduce the following full round and partial round

Name	Security	Field	n	R_F	R_P	t
`Poseidon-256`	2^{128}	$\mathbb{F}_p$	256	8	84	6
`STARK-252`	2^{126}	$\mathbb{F}_{2^{63}}$	63	12	39	24
Sage Reference Code	2^{128}?	$\mathbb{F}_{2^{63}}$	63	6	42	24

Table 10. Parameters for `Poseidon` and `Starkad`.

functions:

Algorithm 15 $R_r(s)$

Input: $\mathbf{s} \in \mathbb{K}^t$
$\mathbf{s}[i] \leftarrow (\mathbf{s}[i] + C[t\,(r-1)+i])^d$ for $i = 1, \ldots, t$
Output : $\mathbf{s} \times \mathrm{MDS}$

Algorithm 16 $R'_r(s)$

Input: $\mathbf{s} \in \mathbb{K}^t$
$\mathbf{s}[i] \leftarrow \left(\mathbf{s}[i] + C[t\,(r-1)+i]\right)^d$ for $i = 1, \ldots, P$
$\mathbf{s}[i] \leftarrow \mathbf{s}[i] + C[t\,(r-1)+i]$ for $i = (P+1), \ldots, t$
Output : $\mathbf{s} \times \mathrm{MDS}$

Algorithm 17 $L(s)$

Input: $s \in \mathbb{K}^t$
$\mathbf{s}[i] \leftarrow \left(s[i] + C[t\,(r-1)+i]\right)^d$ for $i = 1, \ldots, P$
$\mathbf{s}[i] \leftarrow s[i] + C[t\,(r-1)+i]$ for $i = (P+1), \ldots, t$
Output : $s \times \mathrm{MDS}$

In the following, we always define $R_f = R_F/2$. The permutation function `perm` in `Starkad` and `Poseidon` is defined by:

$$\text{perm}_{\text{Hades}}(\mathbf{s}) = L(\mathbf{s}''') \text{ where } \begin{cases} \mathbf{s}' = R_{R_f} \circ \cdots \circ R_1(\mathbf{s}) \\ \mathbf{s}'' = R'_{R_f+R_P} \circ \cdots \circ R'_{R_f+1}(\mathbf{s}') \\ \mathbf{s}''' = R_{R_F+R_P-1} \circ \cdots \circ R_{R_f+R_P+1}(\mathbf{s}'') \end{cases} \tag{6}$$

We then have four maps:

$$\begin{aligned} \text{perm}_{\text{Hades}} &: \mathbb{K}^t \longrightarrow \mathbb{K}^t \\ R_r &: \mathbb{K}^t \longrightarrow \mathbb{K}^t \\ R'_r &: \mathbb{K}^t \longrightarrow \mathbb{K}^t \\ L &: \mathbb{K}^t \longrightarrow \mathbb{K}^t \end{aligned}$$

5 Algebraic Cryptanalysis of `Jarvis`

We present in this part an improved algebraic attack against `Jarvis` (Section 4.1). Our modelling is given in Section 5.1. Then, we show that the equations modelling `Jarvis` have a multi-homogeneous structure (Section 2.3). This allows to derive the bound on the number of solutions given in Table 3 (Theorem 5.3). We then present experimental results in Section 5.3 and derive an experimental upper bound on the maximal degree reached in the computation of DRL-Gröbner basis; leading the complexity results obtained in Section 5.4.

5.1 Algebraic Modelling

We describe in this part the algebraic modelling of `Jarvis`. To this end, we introduce several extra variables. Namely:

- $s_0, s_1, \ldots, s_{N-1}$ corresponding to intermediate state variables,
- a variable k_0 corresponding to the master-key and variables $k_1, \ldots, k_N$ corresponding to the sub-keys.

According to Algorithm 1 and the very definition of the linear layer $A(X) \in \mathbb{F}_{2^n}[X]$ (as in (4)), we have for each round r:

$$s_r = A(1/s_{r-1}) + k_r = C\big(B^{-1}(1/s_{r-1})\big) + k_r. \tag{7}$$

For each $r, 1 \leq r \leq N$, we introduce a new variable

$$u_{r-1} = B^{-1}\big(1/s_{r-1}\big). \tag{8}$$

Combining (8) with (7), we get:

$$s_r = C(u_{r-1}) + k_r.$$

Finally, (8) is equivalent to:

$$B(u_{r-1}) = 1/s_{r-1}$$

By taking the numerator, this leads to:

$$B(u_{r-1})\, s_{r-1} - 1 = 0.$$

Let $K_0 \in \mathbb{F}_{2^n}$ and $\big(M, C = \texttt{Jarvis}(M, K_0)\big) \in \mathbb{F}_{2^n} \times \mathbb{F}_{2^n}$ be a pair message/ciphertext encrypted under the secret-key K_0. Our key-recovery system for `Jarvis` is as follows:

$$\mathcal{S}_{\texttt{Jarvis}}(M,C) = \begin{cases} s_0 + k_0 + M, \\ B(u_0)\, s_0 + 1, \\ C(u_0) + s_1 + k_1, \\ \cdots \\ B(u_{r-1})\, s_{r-1} + 1, \\ C(u_{r-1}) + s_r + k_r, \\ \cdots \\ B(u_{N-1})\, s_{N-1} + 1, \\ C(u_{N-1}) + C + k_N \end{cases} \quad (9) \qquad \mathcal{K}_{\texttt{Jarvis}} = \begin{cases} k_1\, k_0 + k_0\, C_0 + 1, \\ \vdots \\ k_r\, k_{r-1} + k_{r-1}\, C_r + 1, \\ \vdots \\ k_N\, k_{N-1} + k_{N-1}\, C_N + 1, \end{cases}$$

According to Section 4.1, the polynomials $B, C \in \mathbb{F}_{2^n}[X]$ are of degree 4, and constants $C_1, \ldots, C_N \in \mathbb{F}_{2^n}$ are known and explicitly given. From now on, we denote by $\mathrm{Sys}_{\texttt{Jarvis}}(M,C)$ the key-recovery system for `Jarvis` composed by the equations of $\mathcal{S}_{\texttt{Jarvis}}(M,C)$ and from $\mathcal{K}_{\texttt{Jarvis}}$.

Lemma 1. *The key-recovery system $\mathcal{S}_{\texttt{Jarvis}}(M,C)$ has $3N+1$ equations over $\mathbb{F}_{2^n}$ (N of degree 2, N of degree 4, N of degree 5 and one linear equation) and $3N+1$ variables.*

Example 1. Let's consider $\mathrm{Sys}_{\texttt{Jarvis}}(M,C)$ for $\mathbb{F}_{2^n} = \mathbb{F}_{2^{31}}$, $N = 3$ and with ($M = 539787998, C = 1389316418$).

$s_0 + k_0 + 539787998,$

$s_0\, {u_0}^4 + 283125965\, s_0\, {u_0}^2 + 1450602990\, s_0\, u_0 + 424342659\, s_0 + 1,$
${u_0}^4 + 1781139933\, {u_0}^2 + k_1 + s_1 + 498270819\, u_0 + 761232961,$

$s_1\, {u_1}^4 + 283125965\, s_1\, {u_1}^2 + 1450602990\, s_1\, u_1 + 424342659\, s_1 + 1, {u_1}^4 + 1781139933\, {u_1}^2 + k_2 + s_2 + 498270819\, u_1 + 761232961,$

$s_2\, {u_2}^4 + 283125965\, s_2\, {u_2}^2 + 1450602990\, s_2\, u_2 + 424342659\, s_2 + 1,$
${u_2}^4 + 1781139933\, {u_2}^2 + k_3 + 498270819\, u_2 + 2140157699,$

$k_0\, k_1 + 1824939501\, k_0 + 1,$
$k_1\, k_2 + 1240992103\, k_1 + 1,$
$k_2\, k_3 + 1828068674\, k_2 + 1.$

5.2 Bounding the Number of Solutions

As emphasized in Sections 2 and 3, the number of solutions $D(N)$ of $\mathrm{Sys}_{\texttt{Jarvis}}(M,C)$ is a key quantity to understand the complexity of an algebraic attack. The Bézout bound (Definition 1, Section 2.2.2) on $\mathrm{Sys}_{\texttt{Jarvis}}(M,C)$ yields :

$$D(N) \leq 5^N \times 4^N \times 2^N = 40^N.$$

However, this bound it not tight since it does not take into account the structure of the equations.

To estimate $D(N)$ more accurately, we first consider a variant of $\mathcal{S}_{\texttt{Jarvis}}(M, C)$ where we keep the variable s_{N-1}. The system is:

$$\mathcal{S}^h_{\texttt{Jarvis}}(M) = \begin{cases} s_0 + k_0 + M, \\ B(u_0)\, s_0 + 1, \\ C(u_0) + s_1 + k_1, \\ \cdots \\ B(u_{r-1})\, s_{r-1} + 1, \\ C(u_{r-1}) + s_r + k_r, \\ \cdots \\ B(u_{N-1})\, s_{N-1} + 1, \\ C(u_{N-1}) + s_N + k_N \end{cases} \tag{10}$$

We denote by $\mathrm{Sys}^h_{\texttt{Jarvis}}(M)$ the corresponding *homogenized system* (that it, $\mathcal{S}^h_{\texttt{Jarvis}}(M) + \mathcal{K}_{\texttt{Jarvis}}$.). This system has a multi-homogeneous structure (Definition 2.6, Section 2.3).

Theorem 4. *The system* $\mathrm{Sys}^h_{\texttt{Jarvis}}(M) \subset \mathbb{F}_{2^n}[s_0, s_1, \ldots, s_N, k_0, k_1, \ldots, k_N, u_0, \ldots, u_{N-1}]$ *is multi-homogeneous and its multi-homogeneous Bézout bound is:*

$$(N+1)4^N.$$

Proof. $\mathrm{Sys}^h_{\texttt{Jarvis}}(M)$ is multi-homogeneous w.r.t. to following partition of the variables:

$$V = \big[[s_0], [u_0], [s_1], [u_1], \ldots, [s_{N-1}], [u_{N-1}], [k_0], \ldots, [k_N]\big].$$

Let $s = \#\mathrm{Sys}^h_{\texttt{Jarvis}}(M)$ be the number of polynomials and $n_h = 3N + 1$ be the number of variables. We build the matrix:

$$M = [\deg_{V_j}(f_i)]_{1 \le i \le s, 1 \le j \le n_h} \tag{11}$$

where V_j is the j-th variable in the list V and f_i is the i-th equation in the system $\mathrm{Sys}^h_{\texttt{Jarvis}}(M)$.

Example 2. In the case of Example 1, the matrix (11) is as follows:

$$M = \left[\begin{array}{cccccc|cc} 1 & & & & & & 1 & \\ 1 & 4 & & & & & & \\ & 4 & 1 & & & & 1 & \\ & & 1 & 4 & & & & \\ & & & 4 & 1 & & 1 & \\ & & & & 1 & 4 & & \\ & & & & & 4 & & 1 \\ \hline & & & & & & 1 & 1 \\ & & & & & & 1 & 1 \\ & & & & & & & 1 \; 1 \end{array}\right]$$

To ease the notations, we denote by $x_1, \ldots, x_{n_h}$ be variables of $\mathcal{S}_{\texttt{Jarvis}}(M, C)$. Let then :

$$W = M \times [x_1, \ldots, x_{n_h}]^T.$$

The Bézout multi-homogeneous bound (Definition 2, Section 2) is the coefficient of $V_1 V_2 \cdots V_{n_h}$ in the polynomial

$$P_N = \prod_{i=1}^{s} W_i.$$

The proof is by induction that this coefficient if $(N+1)4^N$.

Base case. When $N = 1$, the polynomial is:

$$P_1 = (k_0 + s_0)(s_0 + 4u_0)(4u_0 + k_1)(k_0 + k_1).$$

By expanding this expression, we get: $P_1 = k_0^2\, k_1\, s_0 + 4\, k_0^2\, k_1\, u_0 + 4\, k_0^2\, s_0\, u_0 + 16\, k_0^2\, u_0^2 + k_0\, k_1^2\, s_0 + 4\, k_0\, k_1^2\, u_0 + k_0\, k_1\, s_0^2 + \boxed{8\, k_0\, k_1\, s_0\, u_0} + 16\, k_0\, k_1\, u_0^2 + 4\, k_0\, s_0^2\, u_0 + 16\, k_0\, s_0\, u_0^2 + k_1^2\, s_0^2 + 4\, k_1^2\, s_0\, u_0 + 4\, k_1\, s_0^2\, u_0 + 16\, k_1\, s_0\, u_0^2$ and the bound is 8.

Induction. We assume that the bound is $N4^{N-1}$ for $N-1$ rounds. Then it is easy to check that:

$$P_N = P_{N-1}(s_{N-1} + 4u_{N-1})(4u_{N-1} + k_N)(k_{N-1} + k_N)$$

We expand the polynomial starting with $(k_{N-1} + k_N)$.

1. We begin with k_{N-1}. Since the next products are $(k_0+k_1)(k_1+k_2)\cdots(k_{N-2}+k_{N-1})$ so we obtain successively $k_{N-2}, \ldots k_1$, k_0. Next we have $(k_0 + s_0)(s_0 + 4\,u_0)(4\,u_0 + k_1 + s_1)(s_1 + 4\,u_1)\cdots$. Thus, we deduce (in this order): s_0, $4u_0$, s_1, $4u_1, \ldots$. In total, the coefficient is 4^N.
2. We now begin with k_N. From the product $(s_{N-1} + 4u_{N-1})\ (4u_{N-1} + k_N)$ we obtain $4u_{N-1}$ and s_{N-1} so we have $4u_{N-1}s_{N-1}k_N P_{N-1}$. Hence, by induction the coefficient is $4N\, 4^{N-1} = N4^N$.

By mixing the two cases, we have $4^N + N4^N = (N+1)4^N$. □

in 4.1, the polynomials $B, C \in \mathbb{F}_{2^n}[X]$ are
We can generalize the proof for any polynomials $B, C \in \mathbb{F}_{2^n}[X]$ as in (4) (Section 4.1) such that $\deg(B) = d_B$ and $\deg(C) = d_C$.

Theorem 5. *Let* $\mathrm{Sys}^h_{\mathtt{Jarvis}}(M)$ *be the key-recovery system generated from polynomials* $B(X)/C(X)$ *of degree* d_B/d_C*. The Bézout multi-homogeneous bound is equal to*

$$B = \begin{cases} \frac{d_B^{N+1} - d_C^{N+1}}{d_B - d_C} & \text{when } d_B \neq d_C \\ (N+1)d^N & \text{when } d = d_B = d_C \end{cases}$$

5.3 Maximal Degree in DRL Gröbner Basis

In this part, we present experiment results showing that the number of solutions derived in Theorem indeed holds in practice.

Experimental Fact 1. *The maximal degree occurring in the computation of a* DRL*-Gröbner basis of* $\mathcal{S}_{\mathtt{Jarvis}}(M, C)$ *is*

$$\max(5, N+2). \tag{12}$$

Note that for any value of N, $\mathcal{S}_{\mathtt{Jarvis}}(M, C)$ contains equations of degree 5 (Lemma 1). So, the maximal degree is always at least 5.

Remark 8. The experimental bound (12) is much better than the Macaulay bound (Remark 4) for $\mathcal{S}_{\mathtt{Jarvis}}(M, C)$:

$$8N + 1.$$

We also provide some the timings on the two steps in zero-dim solving process (Section 2.2). In what follows, we denote by:

– T_{DRL}, the time for computing a DRL-Gröbner basis,
– $D_{\max}$, the maximum degree reached in the computation of a DRL-Gröbner basis
– T_{FGLM}, the time for the change of ordering,

In Table 11, we present the experimental results obtained on `Jarvis` using the Fgb software. Most of the experiments can also be done with the Magma computer algebra system. The only

N	n	#Sol	T_{DRL}	$D_{\max}$	T_{FGLM}	Density FGLM
2	31	24	0.0s	5	0.0s	52.9%
3	31	160	0.03s	5	0.01s	31.3%
4	31	896	1.85s	6	0.89s	26.0%
5	31	4608	169.7s	7	153.7s	35.1%
6	31	22528	23027s	8	17639s	37.6%

Table 11. Algebraic cryptanalysis of `Jarvis` with Fgb.

main difference is the implementation of the FGLM algorithm which can be the bottleneck in Magma.

N	n	#Sol	T_{DRL}	T_{FGLM}
2	31	24	0.02s	0.00s
3	31	160	0.14s	0.28s
4	31	896	9.35s	56.6s
5	31	4608	962.6s	5464.2s

Table 12. Algebraic cryptanalysis of `Jarvis` with Magma.

5.4 Complexity Analysis

According to the experiments of Section 5.3, we can assume that the maximum degree reached in the computation of a DRL-Gröbner basis of $\mathcal{S}_{\texttt{Jarvis}}(M, C)$ is $\max(5, N+2)$ (Exp. Fact 1). According to Theorem 2, the cost of computing a DRL-Gröbner basis is then:

$$T_1 = O\left(\binom{4N+3}{N+2}^{\omega}\right).$$

This out of the scope of this report, but we demonstrated that the change of ordering (Section 2.2.2) can be done in :

$$T_2 = O\left((N+1)4^N\right).$$

This is linear in the number of solutions $\mathcal{S}_{\texttt{Jarvis}}(M, C)$ (Theorem 5.3). The number of rounds in Table 3 is obtained by assuming that the complexity of solving is essentially equivalent to T_2. From now, we will always used this approach.

6 Algebraic Cryptanalysis of MiMC

6.1 A First Simple Algebraic Attack

Let $K \in \mathbb{K}$ and $\big(M, C = \texttt{MiMC}(M, K)\big) \in \mathbb{K} \times \mathbb{K}$ be a pair message/ciphertext encrypted under the secret-key K. The algebraic modelling of MiMC (Section 4.2) is rather straightforward.

We need to introduce:

- variables $s_0, s_1, \ldots, s_{N-1}$ corresponding to the intermediate states, and
- a variable k corresponding to the master-key.

The key-recovery system is then as follows:

$$\mathcal{S}_{\texttt{MiMC}}(M, C) = \begin{cases} s_0 = (M + k + C_1)^d, \\ \quad \vdots \\ s_r = (S_{r-1} + k + C_r)^d, \\ \quad \vdots \\ C = k + S_{N-1} \end{cases}$$

Recall that there is no key schedule in MiMC. Thus :

Lemma 2. *The key-recovery system $\mathcal{S}_{\texttt{MiMC}}(M, C)$ has $N+1$ equations over $\mathbb{K}$ (N of degree d in one linear equation) and $N+1$ variables.*

Example 3. We provide below the equations of the system $\mathcal{S}_{\texttt{MiMC}}(M, C)$ for $N = 3, d = 3$ and $\mathbb{K} = \mathbb{F}_{2^{31}}$.

$$\begin{cases} k^3 + 18\,k^2 + 260\,k + s_0 + 4680, \\ k^3 + s_0\,k^2 + {s_0}^2 k + {s_0}^3 + 573341058\,k^2 + 573341058\,{s_0}^2 \\ \quad + 1561922980\,k + 1561922980\,s_0 + s_1 + 1910775587, \\ k^3 + s_1\,k^2 + {s_1}^2 k + {s_1}^3 + 54203621\,k^2 + 54203621\,{s_1}^2 \\ \quad + 21629643\,k + 21629643\,s_1 + s_2 + 869220636, \\ s_2 + k + 2100078585 \end{cases}$$

We have 4 variables k, s_0, s_1, s_2 and 4 equations (3 cubic equations and 1 linear equation).

The following result on the number of solutions of $\mathcal{S}_{\texttt{MiMC}}(M, C)$ is obvious :

Theorem 6. *Let d be the degree of* SBox *used in* MiMC *(Algorithm 4, Section). The Bézout bound for $\mathcal{S}_{\texttt{MiMC}}(M, C)$ is:*

$$d^N.$$

Similarly than in the previous section, we can assume that the change of ordering can be done in linear time. This explains the number of rounds derived in Table 3.

6.2 Improved Algebraic Attacks

For MiMC, we also considered variants of the attack considered in Section 6.1.

6.2.1 Several Pairs. We consider the case when we have $\ell > 1$ pairs of message/ciphertext (instead of just one in the Section 6.1). For each pair, we need to introduce N new variables corresponding to the intermediate states (for each pair message/ciphertext). On the other other hand, we can keep the same variable k for the secret-key. All in all, we have then $\ell\,(N+1)$ equations and $\ell\,N+1$ variables. The algebraic system is then slightly overdetermined.

Remark 9. For Gröbner basis the more equations we have the best it is. More precisely, when m is the number of equations and n the number of variables, the complexity is exponential when $m \approx n$ but sub-exponential when $m \gg n$ (under some algebraic assumptions).

Conclusion. We perform some experiments. In comparison to the attack of Section 6.1, we can only reduce by one the maximal degree occurring in the Gröbner basis computation. Hence, since we increase a lot the number of variables there is no gain in practice.

6.2.2 Splitting the Equations. We assume here that the field $\mathbb{K} = \mathbb{F}_{2^n}$. The idea is to split the system $\mathcal{S}_{\texttt{MiMC}}(M, C)$ over the base field $\mathbb{F}_2$. In addition, we can add the field equations. We have then $(N+1)n + n$ equations of degree 2 in $\mathbb{F}_2$ and $(N+1)n$ variables. This has to be compared with the direct attack (Section 6.1)that has mush less variables, but whose equations are of degree 3.

We present below some practical results. In particular, we report the maximal degree $D_{\max}$ reached during the computation of a DRL-Gröbner basis. In this context, the system has very few solutions making the cost of the change of ordering negligible (in contrast to the direct algebraic attack of Section 6.1 that has an exponential number of variables). The cost of this attack is then exponential in $D_{\max}$. The last column essentially corresponds to the time for computing a DRL-Gröbner basis.

N	n	$D_{\max}$	CPU Time
3	7	3	0.00s
3	17	3	2.75s
3	27	3	23.67s
3	50	3	177.45s
5	7	3	0.11s
5	10	3	1.41s
5	15	3	34.22s
5	17	3	80.49s
10	10	3	26.44s
15	10	3	1179.57s
6	10	3	4.2s
6	11	> 3	
6	12	> 3	
6	15	> 3	
8	15	> 3	
10	15	> 3	

– Experiments are difficult to conduct (due to the large number of variables). Hence, only partial experimental results are available.
– Surprisingly when n (say ≤ 10) is small the complexity does not seem to depend on the number of rounds.

- When $n > 10$ then there is a change in the complexity (maximal degree in the computation). Consequently the computation is probably exponential.
- The attack is not very efficient in practice, however it is difficult to evaluate precisely the *theoretical complexity.*

Thus, since it is difficult to evaluate precisely the complexity, it is recommended to take $\mathbb{K} = \mathbb{F}_{\mathbf{p}}$ where p is a prime number to avoid this attack.

7 Algebraic Cryptanalysis of Pepper

Let $K_0 \in \mathbb{K}^m$ and $\big(M, C = \texttt{Pepper}-\text{v}i(M, K_0)\big) \in \mathbb{K}^m \times \mathbb{F}_q^m$ be a pair message/ciphertext encrypted under the secret-key K_0. We denote by $\text{Sys}_{\texttt{Pepper}-\text{v}i}(M, C)$ the corresponding algebraic system.
In this case, we introduce:

- $s_0, s_1, \ldots, s_{N-1}$ corresponding to intermediate state variables,
- a variable k_0 corresponding to the master-key and variables $k_1, \ldots, k_{N'}$ corresponding to the sub-keys.

For Pepper−v1, the system $\text{Sys}_{\texttt{Pepper}-\text{v1}}(M, C)$ is as follows:

$$\text{S}_{\texttt{Pepper}-\text{v1}}(M, C) \begin{cases} S_1^3 - M^3 - k_1, \\ \quad \vdots \\ S_r^3 - k_{2r+1} - (S_{r-1} + k_{2r})^3, \\ \quad \vdots \\ C = k_{2N} + S_N \end{cases} \qquad \mathcal{K}_{\texttt{Pepper}} \begin{cases} k_1\, k_0 + C_0\, k_0 - 1, \\ \quad \cdots \\ k_{2r-1}\, k_{2r} + C_{2r-1}\, k_{2r-1} - 1, \\ k_{2r}\, k_{2r+1} + C_{2r+1}\, k_{2r} - 1, \\ \quad \cdots \\ k_{2N}\, k_{2N-1} + C_{2N}\, k_9 - 1 \end{cases}$$

We that that the C_r are known constants. The system corresponding $\text{Sys}_{\texttt{Pepper}-\text{v2}}(M, C)$ is essentially similar. The main difference is that the equations corresponding to the key scheduling are linear.

Theorem 7. *For* Pepper−v1(N)*, the Bézout multi-homogeneous bound is*

$$(d+1)\, N\, d^{N-1}.$$

For Pepper−v2(N)*, the Bézout bound is* d^{N+1}*.*

8 Algebraic Cryptanalysis of Rescue

8.1 Analysis of the Block Cipher

Let $K_0 \in \mathbb{F}_q^m$ and $\big(M, C = \texttt{Rescue}(M, K_0)\big) \in \mathbb{F}_q^m \times \mathbb{F}_q^m$ be a pair message/ciphertext encrypted under the secret-key K_0. We denote by $\text{Sys}_{\texttt{Rescue}}(M, C)$ the corresponding algebraic system.

Example 4. We consider $N = 3, m = 2, p = 65519$, $M = \begin{bmatrix} 13612 \\ 47071 \end{bmatrix}$, $C = \begin{bmatrix} 55389 \\ 56921 \end{bmatrix}$, $a = 20801$, $b = 275$, and $k = \begin{bmatrix} 64238 \\ 8057 \end{bmatrix}$ In this case, $\mathrm{Sys}_{\texttt{Rescue}}(M, C)$ is as follows:

$[{s_{0,1}}^3 - k_0 - 13612, {s_{0,2}}^3 - k_1 + 18448,$
${s_{1,1}}^3 + 11\ (-11\, s_{0,1} - 132\, s_{0,2} + 20801\, k_0 + 275)^3 + 132\ (12\, s_{0,1} + 133\, s_{0,2} + 20801\, k_1 + 275)^3 + 5875\, k_0 - 20397, {s_{1,2}}^3 -$
$12\ (-11\, s_{0,1} - 132\, s_{0,2} + 20801\, k_0 + 275)^3 - 133\ (12\, s_{0,1} + 133\, s_{0,2} + 20801\, k_1 + 275)^3 + 5875\, k_1 - 20397, {s_{2,1}}^3 +$
$11\ (-11\, s_{1,1} - 132\, s_{1,2} - 12940\, k_0 - 22772)^3 + 132\ (12\, s_{1,1} + 133\, s_{1,2} - 12940\, k_1 - 22772)^3 + 12888\, k_0 - 22273, {s_{2,2}}^3 -$
$12\ (-11\, s_{1,1} - 132\, s_{1,2} - 12940\, k_0 - 22772)^3 - 133\ (12\, s_{1,1} + 133\, s_{1,2} - 12940\, k_1 - 22772)^3 + 12888\, k_1 - 22273, -11$
$(-11\, s_{2,1} - 132\, s_{2,2} + 20460\, k_0 + 16099)^3 - 132\ (12 s_{2,1} + 133\, s_{2,2} + 20460\, k_1 + 16099)^3 - 22964\, k_0 + 18095,$
$12\ (-11\, s_{2,1} - 132\, s_{2,2} + 20460\, k_0 + 16099)^3 + 133\ (12\, s_{2,1} + 133\, s_{2,2} + 20460\, k_1 + 16099)^3 - 22964\, k_1 + 16563].$

The following result on the number of solutions of $\mathcal{S}_{\texttt{Rescue}}(M, C)$ is obvious :

Theorem 8. *Let d be the degree of the first* SBox *used in* `Rescue` *(Algorithm 10, Section 4.3.2). The Bézout bound for $\mathcal{S}_{\texttt{Rescue}}(M, C)$ is:*

$$d^{m\cdot(N+1)}. \tag{13}$$

Proof. We have :

- $N + 1$ rounds,
- m equations of degree d at each round.

So, $\mathcal{S}_{\texttt{Rescue}}(M, C)$ has in total $m(N + 1)$ equations of degree d. The result follows from Definition 1. □

The result in Table 3 is obtained by assuming that the change of ordering is quadratic in the Bézout bound (13).

8.2 Analysis of the Hash Function

As explained in [5]), `Rescue` (Section 4.3) can be transformed into a hash function thanks to the sponge construction. In this part, we consider the security of `Rescue`-based hash function against an algebraic attacks. To do so, we consider the so-called *Constrained-Input Constrained-Output* (`CICO`) problem. This problem, introduced in [32], is related to the the problem of finding a pre-image of the hash function.

Assuming that the secret-key is fixed in `Rescue`, then Algorithm 10 (Section 4.3) defines a permutation. Let $\mathrm{perm}_{\mathrm{Rescue}} : \mathbb{F}_q^m \longrightarrow \mathbb{F}_q^m$ be the corresponding permutation.

In this context, the `CICO` problem is as follows:

`CICO` (k)
Find $x_{k+1}, \ldots, x_t \in \mathbb{K}^{t-k}$ such that
$\mathbf{x} = (0, \ldots, 0, x_{k+1}, \ldots, x_t)$ // Constrained input
$\mathbf{y} = \mathrm{perm}_{\mathrm{Rescue}}(\mathbf{x})$
$y_1 = \cdots = y_k = 0$ // Constrained output

9 Algebraic Cryptanalysis of Vision

9.1 Algebraic Modelling

Let $K_0 \in \mathbb{F}_{2^{n/m}}$ and $\big(M, C = \texttt{Vision}(M, K_0)\big) \in \mathbb{F}_{2^{n/m}} \times \mathbb{F}_{2^{n/m}}$ be a pair message/ciphertext encrypted under the secret-key K_0. We denote by $\mathrm{Sys}_{\texttt{Vision}}(M, C)$ the algebraic system modelling Rescue (Algorithm 12) where the variables are:

- $s_{0,1}, \ldots, s_{0,m}, s_{1,1}, \ldots, s_{1,m}, \ldots, s_{N-1,1}, \ldots, s_{N-1,m}$ corresponding to intermediate state variables,
- additional variables $u_{0,1}, \ldots, u_{0,m}, u_{1,1}, \ldots, u_{1,m}, \ldots, u_{N-1,1}, \ldots, u_{N-1,m}$ corresponding to intermediate state variables,
- variables $k_{0,1}, \ldots, k_{0,m}$ corresponding to the master-key.

We give below a pseudo-code for generating the equations of $\mathrm{Sys}_{\texttt{Vision}}(M, C)$.

Input: $\big(M, C = \texttt{Vision}(M, K_0)\big) \in \mathbb{F}_{2^{n/m}} \times \mathbb{F}_{2^{n/m}}$
$K := {}^T[k_{0,1}, \ldots, k_{0,m}]$
$S_0 := M + K_0$
Eqs := []
$K_1, \ldots, K_{2N} := \texttt{KeyExpand}(K_0)$ // K_i depends on $\{k_{0,i}\}$
for r **from** 1 **to** $N-1$ **do**
  Eqs := Eqs $\cup$ [Numerator$\big(B(1/S_{r-1}[i])u_{r,i}\big) \mid i = 1, \ldots, m]$
  $S := \mathcal{M}\,{}^T[u_{r,1}, \ldots, u_{r,m}] + K_{2r-1}$
  Eqs := Eqs $\cup$ $[B(s_{r,i})S[i] - 1 \mid i = 1, \ldots, m]$
  $S_r := \mathcal{M}\,{}^T[s_{r,1}, \ldots, s_{r,m}] + K_{2r}$
EndFor
Eqs := Eqs $\cup$ $[u_{N,i}S_{N-1}[i] - 1 \mid i = 1, \ldots, m]$
Eqs := Eqs $\cup$ $[u_{N,i} + K_{2N-1}[i] - C[i] \mid i = 1, \ldots, m]$

Fig. 3. MAGMA pseudo-code for $\mathrm{Sys}_{\texttt{Vision}}(M, C)$.

9.2 Bounding the Number of Solutions

Similarly to the algebraic attack against Jarvis (Section 5), the algebraic system $\mathrm{Sys}_{\texttt{Vision}}(M, C)$ (Figure 3) has a multi-homogeneous structure (Definition 2.6, Section 2.3). In particular, we can derive a precise bound on the number of solutions of $\mathrm{Sys}_{\texttt{Vision}}(M, C)$ for $m = 2, 3, 4$. Remark that the technique can be generalized to any m.

Theorem 9. *Let d be the degree of B (Algorithm 12). $\mathrm{Sys}_{\texttt{Vision}}(M, C)$ has a multi-homogeneous structure. Let then $B_{N,m}(d)$ be the multi-homogeneous Bézout bound of $\texttt{Vision}(N, d, m, n)$ (with the modelling of Figure 3). We have :*

$$B_{N,2}(d) = \frac{2\left(-\left(3\,d^2 + 12\,d + 3\right)N + (11\,d + 1)\,(d - 1)\right)\left(d^2\right)^N + \left(14\,d^2 + 11\,d + 2\right)\left(4\,d^2\right)^N}{18\,d^4},$$

$$B_{N,3}(d) = -\frac{\left(-249\,d^3 + 33\,d^2 + 9\,d + 47\right)\left(d^3\right)^N}{64\,d^6} + \frac{\left(51\,d^3 + 77\,d^2 + 29\,d + 3\right)(N+1)\left(9\,d^3\right)^N}{144\,d^6},$$
$$-\frac{\left(201\,d^3 + 655\,d^2 + 295\,d + 33\right)\left(9\,d^3\right)^N}{576\,d^6} + \frac{\left(7\,d^3 + 9\,d^2 + 9\,d + 7\right)(N+1)\left(d^3\right)^N}{16d^6}.$$

$$B_{N,4}(d) \approx \frac{\left(377\,d^2 + 231\,d + 22\right)\left(66\,d^2 + 63\,d + 11\right)\left(36\,d^4\right)^N}{61250\,d^8}.$$

In the table below, we report experimental results on the exact number of solutions of $\mathrm{Sys}_{\texttt{Vision}}(M, C)$ for $m = 2$. We also compute the bound derived in Theorem 9. We can notice that the bound is perfectly sharp.

(N, m, d)	Exact Value	$B_{N,2}(d)$
$(1, 2, 4)$	4	4
$(2, 2, 4)$	233	233
$(3, 2, 4)$	15072	15072
$(2, 2, 2)$	65	65
$(3, 2, 2)$	1096	1096
$(4, 2, 2)$	17968	17968
$(2, 2, 1)$	20	20
$(3, 2, 1)$	90	90
$(4, 2, 1)$	376	376
$(5, 2, 1)$	1526	1526

It is easy to obtain an asymptotic estimates:

$$B_{N,2}(d) \approx \frac{\left(14\,d^2 + 11\,d + 2\right)\left(4\,d^2\right)^n}{18\,d^4}.$$

In particular when $d = \deg(B) = 4$, we deduce the value given in Table 4:

$$B_{N,2}(4) \approx \frac{15}{256}\,64^N = \frac{15}{256}\,2^{6^N}.$$

In the next table, we provide experimental results for $m = 3$. In this case, The bound of Theorem 9 is tight but no longer exact.

(N, m, d)	Exact Value	$B_{N,3}(d)$
$(1, 3, 4)$	8	8
$(2, 3, 2)$	667	679
$(2, 3, 4)$	4645	4693
$(2, 3, 1)$	109	112
$(3, 3, 1)$	1626	1752

For $m = 3$, we get:

$$B_{3,N}(d) \approx \frac{(3\,d + 1)\left(4\,(d + 1)\,(17\,d + 3)\,n + d^2 - 116\,d - 21\right)\left(9\,d^3\right)^n}{576\,d^6}.$$

In particular when $d = 4$, we get:

$$B_{3,N}(d) \approx \frac{(18460\,n - 6097)\,576^n}{2359296}.$$

10 Algebraic Pre-Image Attacks

In this part, we consider the `CICO` problem (Figure 1, [32]) against `Starkad`, `Poseidon` (Section 10.1) and `Rescue` (Section 10.2). This is a preimage attack against the permutations underlying these ciphers.

10.1 The CICO Problem for Starkad and Poseidon

Let $\text{perm}_{\text{Hades}} : \mathbb{K}^t \longrightarrow \mathbb{K}^t$ be the permutation defined as in (6) (Section 4.4). In this context, the CICO problem is defined as follows:

$\texttt{CICO}_{\text{Hades}}\ (k)$
Find $(x_{k+1}, \ldots, x_t) \in \mathbb{K}^{t-k}$ such that
$\mathbf{x} = (0, \ldots, 0, x_{k+1}, \ldots, x_t) \in \mathbb{K}^t$ // Constrained input
$\mathbf{y} = \text{perm}_{\text{Hades}}(\mathbf{x})$
$y_1 = \cdots = y_k = 0$ // Constrained output

Remark 10. We assume that $\mathbb{K}$ is of size n. We then fix $n\,k$ bits of the input and the output of the permutation function. In practice, we will have $n\,k \approx 256$ bits.

In the CICO problem, we want to find $\mathbf{x} = (0, \ldots, 0, x_{k+1}, \ldots, x_t)$ such that:

$$\text{perm}_{\text{Hades}}(\mathbf{x}) = (0, \ldots, 0, \star, \ldots, \star)$$

It is equivalent to find $\mathbf{z} = (z_1, \ldots, z_k)$ such that $L(z_1, \ldots, z_k) = (0, \ldots, 0, \star, \ldots, \star)$ and $\mathbf{z} = \text{perm}'_{\text{Hades}}(\mathbf{x})$ where the $\text{perm}'_{\text{Hades}}$ function is:

$$\text{perm}'_{\text{Hades}}(\mathbf{s}) = \mathbf{s}''' \text{ where } \begin{cases} \mathbf{s}' = R_{R_f} \circ \cdots \circ R_1(\mathbf{s}) \\ \mathbf{s}'' = R'_{R_f+R_P} \circ \cdots \circ R'_{R_f+1}(\mathbf{s}') \\ \mathbf{s}''' = R_{R_F+R_P-1} \circ \cdots \circ R_{R_f+R_P+1}(\mathbf{s}'') \end{cases}$$

Since there is no matrix multiplication, we can recover explicitly the values of $z_1, \ldots, z_k$. We then define a variant of the CICO problem by:

$\texttt{CICO}'_{\text{Hades}}\big(k, (a_1, \ldots, a_k) \in \mathbb{K}^k\big)$
Find: $(x_{k+1}, \ldots, x_t) \in \mathbb{K}^{t-k}$ such that
$\mathbf{x} = (0, \ldots, 0, x_{k+1}, \ldots, x_t) \in \mathbb{K}^t$ // Constrained input
$\mathbf{y} = \text{perm}'_{\text{Hades}}(\mathbf{x})$
$y_1 = a_1, \ldots, y_k = a_k$ // Constrained output

10.1.1 Algebraic Modelling of the CICO Problem – Starkad and Poseidon.

Let $\mathbf{a} = (a_1, \ldots, a_k) \in \mathbb{K}^k$. The algebraic system $\mathcal{S}_{\texttt{CICO}'_{\text{Hades}}}(\mathbf{a}) \subset \mathbb{K}[x_{k+1}, \ldots, x_t]$ corresponding to the $\texttt{CICO}'_{\text{Hades}}$ problem is given below :

$$\mathcal{S}_{\texttt{CICO}'_{\text{Hades}}}(\mathbf{a}) = \begin{cases} y_1 = a_1 \\ \quad\vdots \\ y_k = a_k \end{cases} \text{ where } \mathbf{y} = \text{perm}'_{\text{Hades}}\big(0, \ldots, 0, x_{k+1}, \ldots, x_t\big).$$

Hence, we have k equations and $t-k$ variables. For practical parameters (Table 10), we see that $t-k$ is much larger than k. Hence the system $\mathcal{S}_{\texttt{CICO}'_{\text{Hades}}}(\mathbf{a})$ is under-defined. As a consequence, we can fix arbitrarily some values of the input $\mathbf{x}$.

By randomly fixing $r_{2k+1}, \ldots, r_t \in \mathbb{K}$, we define a square algebraic system of k equations and k variables:

$$\mathcal{S}_{\texttt{CICO}'_{\text{Hades}}}(\mathbf{a}) = \begin{cases} y_1 = a_1 \\ \quad \vdots \\ y_k = a_k \end{cases} \text{where } y = \text{perm}'_{\text{Hades}}\big((0, \ldots, 0, x_{k+1}, \ldots, x_{2k}, r_{2k+1}, \ldots, r_t)\big).$$

Remark 11. The complexity of solving $\mathcal{S}_{\texttt{CICO}'_{\text{Hades}}}(\mathbf{a})$ does not depend on t. The attack is more efficient when the field (n) is big.

10.1.2 Practical Experiments We present here experimental results on the complexity of solving $\mathcal{S}_{\texttt{CICO}'_{\text{Hades}}}(\mathbf{a})$. For the experiments, we took $d = 2$ and a prime field $\mathbb{K} = \mathbb{F}_p$. We have done the experiments for $k = 1, 2, 3$ (we also checked that the results are the same when $\mathbb{K} = \mathbb{F}_{2^n}$ and $d = 3$). In each case, we compute a Gröbner basis of $\mathcal{S}_{\texttt{CICO}'_{\text{Hades}}}(\mathbf{a})$ w.r.t. a lexicographical monomial ordering.

We report below the degree of the non linear leading terms in the Gröbner basis.

R_F	R_P	t	degree
2	0	5	x_3^2, x_4^2
2	1	5	x_3^2, x_4^4
2	2	5	x_3^2, x_4^8
4	0	5	x_3^2, x_4^{32}
4	1	5	x_3^2, x_4^{64}
4	2	5	x_3^2, x_4^{128}
6	0	5	x_3^2, x_4^{512}
6	1	5	x_3^2, x_4^{1024}

$k = 2$.

R_F	R_P	t	degree
2	0	6	x_4^2, x_5^2, x_6^2
2	1	6	x_4^2, x_5^2, x_6^4
2	2	6	x_4^2, x_5^2, x_6^8
4	0	6	x_4^2, x_5^2, x_6^{128}
4	1	6	x_4^2, x_5^2, x_6^{256}
4	2	6	x_4^2, x_5^2, x_6^{512}
6	0	6	x_4^2, x_5^2, x_6^{8192}

$k = 3$.

R_F	R_P	t	degree
2	0	4	x_2^2
2	1	4	x_2^4
2	2	4	x_2^8
4	0	4	x_2^8
4	1	4	x_2^{16}
4	2	4	x_2^{32}
6	0	4	x_2^{32}
6	1	4	x_2^{64}
10	0	4	x_2^{512}
11	1	4	x_2^{1024}
14	0	4	x_2^{8192}

$k = 1$.

10.1.3 Bounding the Number of Solutions `Starkad` and `Poseidon`. We can prove the following results:

Theorem 10. *The degree of the univariate polynomial of the lexicographical Gröbner basis of* $\mathcal{S}_{\texttt{CICO}'_{\text{Hades}}}(\mathbf{a})$ *is*

$$d^{k\,R_F + P\,R_P - 2\,k + 1}$$

For the permutation of `Starkad`/`Poseidon` *($d = 3$ and $P = 1$), it holds in particular that the number of solutions D is equal to:*

$$D = 3^{k\,R_F + R_P - 2\,k + 1}.$$

Remark 12. As expected, when $k \neq 1$ we see that the number of full rounds is the most important factor in the complexity.

The complexity of the attack is polynomial in the degree D of the univariate polynomial: $O(D^\omega)$. When $k = 1$ or 2, we take $\omega = 1$; when $k > 2$ then $\omega = 2$.
The complexity of the attack is then:

k	1	2	> 2
Complexity for `Starkad/Poseidon`	$3^{R_F+R_P-1}$	$3^{2\,R_F+R_P-3}$	$3^{2(k\,R_F+R_P-2\,k)+2}$

This allows to derive the security estimates of Table 6 (Section 1.3).

10.2 The `CICO` Problem for `Rescue`

Let $\text{perm}_{\texttt{Rescue}} : \mathbb{F}_q^m \longrightarrow \mathbb{F}_q^m$ be the `Rescue` permutation (Section 4.3.3). As in the previous part, we consider:

$\texttt{CICO}_{\texttt{Rescue}}(k)$
Find $(x_{k+1}, \ldots, x_t) \in \mathbb{K}^{m-k}$ such that
$\mathbf{x} = (0, \ldots, 0, x_{k+1}, \ldots, x_t) \in \mathbb{K}^m$ // Constrained input
$\mathbf{y} = \text{perm}_{\texttt{Rescue}}(\mathbf{x})$
$y_1 = \cdots = y_k = 0$ // Constrained output

This is not detailed in the report but, as in Section 10.1, we can define an algebraic set of equations $\mathcal{S}_{\texttt{CICO}_{\texttt{Rescue}}}(k)$ modelling $\texttt{CICO}_{\texttt{Rescue}}(k)$. Also, it holds that:

Theorem 11. *The number of solutions D of $\mathcal{S}_{\texttt{CICO}_{\texttt{Rescue}}}(k)$ is equal to:*

$$D = 3^{(N-1)m+k}.$$

The complexity of the attack is polynomial in the degree D of the univariate polynomial: $O(D^\omega)$. When $k = 1$ or 2, we take $\omega = 1$; when $k > 2$ then $\omega = 2$.

k	1	2	> 2
Complexity for `Rescue`	$3^{(N-1)m+k}$	$3^{(N-1)m+k}$	$3^{2\left((N-1)m+k\right)}$

This allows to derive the security estimates of Table 7 (Section 1.3).

11 History of the Report

Version	Comment	Date
	Algebraic analysis of `STARK`-friendly ciphers	11/2018-09/2019
1.0	First version of the document	09/2019
1.1	Minor typos corrected	03/2020
	Algebraic analysis of the `CICO` problem against `Rescue`	03/2020
1.2	New part (Section 10.2) dedicated to the `CICO` problem against `Rescue`. Report updated and re-organised accordingly.	04/2020

References

1. M. R. Albrecht, C. Cid, L. Grassi, D. Khovratovich, R. Lüftenegger, C. Rechberger, and M. Schofnegger. Algebraic cryptanalysis of stark-friendly designs: Application to marvellous and mimc. In S. D. Galbraith and S. Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part III*, volume 11923 of *Lecture Notes in Computer Science*, pages 371–397. Springer, 2019.
2. M. R. Albrecht, L. Grassi, C. Rechberger, A. Roy, and T. Tiessen. Mimc: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. In J. H. Cheon and T. Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 191–219, 2016.
3. A. Aly, T. Ashur, E. Ben-Sasson, S. Dhooghe, and A. Szepieniec. Design of symmetric-key primitives for advanced cryptographic protocols. Cryptology ePrint Archive, Report 2019/426, 2019. https://eprint.iacr.org/2019/426.
4. G. Ars, J. Faugère, H. Imai, M. Kawazoe, and M. Sugita. Comparison between XL and gröbner basis algorithms. In Lee [34], pages 338–353.
5. T. Ashur and S. Dhooghe. Marvellous: a stark-friendly family of cryptographic primitives. Cryptology ePrint Archive, Report 2018/1098, 2018. https://eprint.iacr.org/2018/1098.
6. M. Bardet. *Étude des systèmes algébriques surdéterminés. Applications aux codes correcteurs et à la cryptographie*. PhD thesis, Université de Paris VI, 2004.
7. M. Bardet, J.-C. Faugère, and B. Salvy. On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations. In *International Conference on Polynomial System Solving – ICPSS*, pages 71–75, 2004.
8. M. Bardet, J.-C. Faugère, and B. Salvy. On the Complexity of the F5 Gröbner basis Algorithm. *Journal of Symbolic Computation*, pages 1–24, Sept. 2014. 24 pages.
9. M. Bardet, J.-C. Faugère, B. Salvy, and B.-Y. Yang. Asymptotic behaviour of the degree of regularity of semi-regular polynomial systems. In *The Effective Methods in Algebraic Geometry Conference – MEGA 2005*, pages 1–14, 2005.
10. E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. Scalable, transparent, and post-quantum secure computational integrity. *IACR Cryptology ePrint Archive*, 2018:46, 2018.
11. G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche. On the indifferentiability of the sponge construction. In N. P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, volume 4965 of *Lecture Notes in Computer Science*, pages 181–197. Springer, 2008.
12. W. Bosma, J. J. Cannon, and C. Playoust. The Magma algebra system I: The user language. *Journal of Symbolic Computation*, 24(3-4):235–265, 1997.
13. B. Buchberger. Bruno Buchberger's PhD thesis 1965: An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal. *Journal of Symbolic Computation*, 41(3-4), 2006.
14. B. Buchberger, G. E. Collins, R. G. K. Loos, and R. Albrecht. Computer algebra symbolic and algebraic computation. *SIGSAM Bull.*, 16(4):5–5, 1982.
15. J. A. Buchmann, A. Pyshkin, and R. Weinmann. Block ciphers sensitive to gröbner basis attacks. In D. Pointcheval, editor, *Topics in Cryptology - CT-RSA 2006, The Cryptographers' Track at the RSA Conference 2006, San Jose, CA, USA, February 13-17, 2006, Proceedings*, volume 3860 of *Lecture Notes in Computer Science*, pages 313–331. Springer, 2006.
16. J. A. Buchmann, A. Pyshkin, and R. Weinmann. A zero-dimensional gröbner basis for AES-128. In M. J. B. Robshaw, editor, *Fast Software Encryption, 13th International Workshop, FSE 2006, Graz, Austria, March 15-17, 2006, Revised Selected Papers*, volume 4047 of *Lecture Notes in Computer Science*, pages 78–88. Springer, 2006.
17. C. Cid and G. Leurent. An analysis of the XSL algorithm. In B. K. Roy, editor, *Advances in Cryptology - ASIACRYPT 2005, 11th International Conference on the Theory and Application of Cryptology and Information Security, Chennai, India, December 4-8, 2005, Proceedings*, volume 3788 of *Lecture Notes in Computer Science*, pages 333–352. Springer, 2005.
18. C. Cid, S. Murphy, and M. J. B. Robshaw. Small scale variants of the AES. In H. Gilbert and H. Handschuh, editors, *Fast Software Encryption: 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Revised Selected Papers*, volume 3557 of *Lecture Notes in Computer Science*, pages 145–162. Springer, 2005.

19. N. Courtois and J. Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. In *Advances in Cryptology - ASIACRYPT 2002, 8th International Conference on the Theory and Application of Cryptology and Information Security, Queenstown, New Zealand, December 1-5, 2002, Proceedings*, pages 267–287, 2002.
20. J. Daemen and V. Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.
21. C. Diem. The xl-algorithm and a conjecture from commutative algebra. In Lee [34], pages 323–337.
22. M. S. E. Din and P. Trebuchet. Strong bi-homogeneous bézout theorem and its use in effective real algebraic geometry. *CoRR*, abs/cs/0610051, 2006.
23. A. C. (ed.), T. Beyne, I. Dinur, M. Eichlseder, G. Leander, G. Leurent, M. N. Plasencia, L. Perrin, Y. Sasaki, Y. Todo, and F. Wiemer. Report on the security of stark-friendly hash functions (version 2.0), 2020.
24. J. Faugère, P. Gaudry, L. Huot, and G. Renault. Sub-cubic change of ordering for gröbner basis: a probabilistic approach. In Nabeshima et al. [41], pages 170–177.
25. J. Faugère and C. Mou. Fast algorithm for change of ordering of zero-dimensional gröbner bases with sparse multiplication matrices. In É. Schost and I. Z. Emiris, editors, *Symbolic and Algebraic Computation, International Symposium, ISSAC 2011 (co-located with FCRC 2011), San Jose, CA, USA, June 7-11, 2011, Proceedings*, pages 115–122. ACM, 2011.
26. J.-C. Faugère. A new efficient algorithm for computing gröbner bases (F4). *Journal of Pure and Applied Algebra*, 139(1-3):61–88, 1999.
27. J.-C. Faugère. A new efficient algorithm for computing gröbner bases without reduction to zero : F5. In *ISSAC'02*, pages 75–83. ACM press, 2002.
28. J.-C. Faugère. FGb: A Library for Computing Gröbner Bases. In K. Fukuda, J. Hoeven, M. Joswig, and N. Takayama, editors, *Mathematical Software - ICMS 2010*, volume 6327 of *Lecture Notes in Computer Science*, pages 84–87, Berlin, Heidelberg, September 2010. Springer Berlin / Heidelberg.
29. J.-C. Faugère, P. M. 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.
30. F. L. Gall. Powers of tensors and fast matrix multiplication. In Nabeshima et al. [41], pages 296–303.
31. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
32. L. Grassi, D. Kales, D. Khovratovich, A. Roy, C. Rechberger, and M. Schofnegger. Starkad and poseidon: New hash functions for zero knowledge proof systems. Cryptology ePrint Archive, Report 2019/458, 2019. https://eprint.iacr.org/2019/458.
33. D. Lazard. Gröbner-bases, Gaussian elimination and resolution of systems of algebraic equations. In *Proceedings of the European Computer Algebra Conference on Computer Algebra*, volume 162 of *Lecture Notes in Computer Science*, Berlin, Heidelberg, New York, 1983. Springer Verlag.
34. P. J. Lee, editor. *Advances in Cryptology - ASIACRYPT 2004, 10th International Conference on the Theory and Application of Cryptology and Information Security, Jeju Island, Korea, December 5-9, 2004, Proceedings*, volume 3329 of *Lecture Notes in Computer Science*. Springer, 2004.
35. T. Li, Z. Lin, and F. Bai. Heuristic methods for computing the minimal multi-homogeneous bézout number. *Appl. Math. Comput.*, 146(1):237–256, Dec. 2003.
36. F. Macaulay. *The Algebraic Theory of Modular Systems*. Cambridge University Press, 1916.
37. F. S. Macaulay. On some formula in elimination. *London Mathematical Society*, 1(33):3–27, 1902.
38. G. Malajovich and K. Meer. Computing minimal multi-homogeneous bézout numbers is hard. In *STACS 2005, 22nd Annual Symposium on Theoretical Aspects of Computer Science, Stuttgart, Germany, February 24-26, 2005, Proceedings*, pages 244–255, 2005.
39. M. B. Monagan, K. O. Geddes, K. M. Heal, G. Labahn, S. M. Vorkoetter, J. McCarron, and P. DeMarco. *Maple 10 Programming Guide*. Maplesoft, Waterloo ON, Canada, 2005.
40. S. Murphy and M. J. B. Robshaw. Essential algebraic structure within the AES. In M. Yung, editor, *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, volume 2442 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2002.
41. K. Nabeshima, K. Nagasaka, F. Winkler, and Á. Szántó, editors. *International Symposium on Symbolic and Algebraic Computation, ISSAC '14, Kobe, Japan, July 23-25, 2014*. ACM, 2014.
42. F. PUB. Secure hash standard (shs). FIPS PUB 180, 4, 2012.
43. W. Stein et al. *Sage Mathematics Softwarz*. The Sage Development Team. `http://www.sagemath.org`.
44. J. von zur Gathen and J. Gerhard. *Modern Computer Algebra (3. ed)*. Cambridge University Press, 2013.

A Reference codes for MiMC

A.1 Sage code for MiMC

```
n = 129
K = GF(2**n, "a")
K.inject_variables()
constants = [.....]
def mimc_encryption(p, k, num_rounds):
    state = (p + (k + K.fetch_int(constants[0])))^3
    for i in range(1, num_rounds):
        state = (state + (k + K.fetch_int(constants[i])))^3
    state = state + k
    return state
num_rounds = ceil(n / log(3, 2))
k = K.fetch_int(0x42424242424242424242424242424242)
p = K.random_element()
c = mimc_encryption(p, k, num_rounds)

print "Plaintext:", p.integer_representation()
print "Ciphertext:", c.integer_representation()
```

A.2 Maple code MiMC

```
Nb:=80; # number of rounds
d:=3; # degree (3 by default)
Vs:=proc(i) cat('s',i): end:
encryption_symb:=proc(msg, cyp, Nb)
    local state,eqs,i;
    state:=(msg + (k + C[1]))^d;
    eqs:=[Vs(0)-state];
    state:=Vs(0);
    for i from 1 to (Nb-1) do
            state:=(state + (k + C[i+1]))^d;
        eqs:=[op(eqs),Vs(i)-state];
        state:=Vs(i);
    od:
    return [op(eqs),state+k-cyp];
end:

# Algebraic Equations:
Eqs:=encryption_symb(M,C,Nb):
```

A.3 Magma code for MiMC

```
n:=129;
d:=3;
Nb:=Ceiling(n / Log(2, 3));
```

```
K<a>:=GF(2,n);

constants:=[...];
encryption:=function(p, sk, Nb)
    state:=(p + (sk + constants[1]))^d;
    for i in [1..(Nb-1)] do
        state:=(state + (sk + constants[i+1]))^d;
    end for;
    state:=state + sk;
    return state;
end function;
sk:=Random(K);
msg:=Random(K);
cyp:=encryption(msg, sk, Nb);
```

B MAGMA code for Poseidon/Starkad

The following MAGMA code permits to:

- easily change the ground field to be $\mathbb{F}_p$, $\mathbb{F}_{2^n}$ or even $\mathbb{Q}$.
- change the degree of the equations between 2 and 3.
- try to modify P the number of S-boxes which are applied in the middle rounds (partial rounds).

```
perm:=function(input_words,dd,part)
    R_f := Z!(R_F / 2);
    state_words := Vector(F,input_words);
    for r in [1..R_f] do
        for i in [1..t] do
            state_words[i] := (state_words[i] + round_constants_field[t*(r-1)+i])^dd;
        end for;
        state_words := state_words * MDS_matrix_field;
    end for;
    for r in [1..R_P] do
        for i in [1..t] do
            state_words[i] := state_words[i] + round_constants_field[t*(R_f+r-1)+i];
        end for;
        for i in [1..part] do
            state_words[i] := (state_words[i])^dd;
        end for;
        state_words := state_words * MDS_matrix_field;
    end for;
    for r in [1..(R_f-1)] do
        for i in [1..t] do
            state_words[i] := (state_words[i] +
             round_constants_field[t*(R_f+R_P+r-1)+i])^dd;
        end for;
```

```
        state_words := state_words * MDS_matrix_field;
    end for;
    for i in [1..t] do
        state_words[i] := (state_words[i] + round_constants_field[t*(R_F+R_P-1)+i])^dd;
    end for;
    return state_words;
end function;
```

B CICO Report

Rescue
CICO Problem

Prepared at the request of StarkWare Industries

Jean-Charles Faugère

March 16,2020

Short description of Rescue

$Rescue_d(N,m,p)$

Arithmetic: finite field $\mathbb{K}=\mathbb{F}_p$, $c=\frac{1}{d} \mod (p-1)$, $N=2N$
$\mathscr{M}$ is an invertible $m \times m$ matrix.

Rescue:
Input: $M \in \mathbb{K}^m$
$S_0 = M + C_0$
for r **from** 1 **to** N **do**
$S = {}^T\left[S_{r-1}[1]^d, \ldots, S_{r-1}[m]^d\right]$
$S = \mathscr{M} S + C_{2r-1}$
$S = {}^T[S[1]^c, \ldots, S[m]^c]$
$S_r = \mathscr{M} S + C_{2r}$
Output : $S_N \in \mathbb{K}^m$

CstesGeneration:
Input: $C_0 \in \mathbb{K}^m, a \in \mathbb{K}, b \in \mathbb{K}$
$B = {}^T[b, \ldots, b]$
for r **from** 1 **to** N **do**
$C_r = a C_{r-1} + B$
Output : $C_0, \ldots, C_N$

In practice: $d = 3$ and $c = (2p-1)/3$

Rescue overview

Since we want to study the CICO problem, C_0 is no longer a secret key but a randomly-chosen vector:

1. Choose randomly $a \in \mathbb{K}, b \in \mathbb{K}$ and $C_0 \in \mathbb{K}^m$
2. Run CstesGeneration(C_0, a, b)
3. Given a message $M \in \mathbb{K}^m$ run C:=Rescue(M)
4. Send $C \in \mathbb{K}^m$

A sponge construction generates a hash function from an underlying permutation by iteratively applying it to a large state. The state of a sponge function is defined to consist of $m = r + c$ bits, where r and c are called the rate and the capacity of the sponge, respectively.

Algebraic Equations

Variables/equations

The CICO Problem

The CICO Problem (Constrained-Input Constrained-Output) is defined by:

CICO Problem(k):
Find: $x_{k+1}, \ldots, x_m \in \mathbb{K}^{m-k}$ such that
$x = [0, \ldots, 0, x_{k+1}, \ldots, x_m]$ // Constrained input
$y = \mathit{Rescue}(x)$
$y_1 = \cdots = y_k = 0$ // Constrained output

Hence if we assume that $\mathbb{K}$ is of size n we fix nk bits of the input **and** the output of the permutation function. In practice we will have $nk \approx 256$ bits.

Remark

Thanks to the last Matrix multiplication $\mathscr{M}$ at the end, we cannot remove the last round.However, we will see how to simplify the algebraic system.

CICO first example and optimizations

We take, $N = 3$, $m = 2$, $k = 1$ and $\mathscr{M} = \begin{bmatrix} -11 & -132 \\ 12 & 133 \end{bmatrix}$

We take for the input/output vectors:

$$X = [0, x_1] \text{ and } Y = [0, y_1]$$

We have the following equations:

$[{s_{0,2}}^3 - x_1 + 24825,$

$${s_{1,1}}^3 + 11\left(7615 - 132 s_{0,2}\right)^3 + 132\left(31617 + 133 s_{0,2}\right)^3 - 29116,$$
$${s_{1,2}}^3 - 12\left(7615 - 132 s_{0,2}\right)^3 - 133\left(31617 + 133 s_{0,2}\right)^3 - 21978,$$

$${s_{2,1}}^3 + 11\left(-11 s_{1,1} - 132 s_{1,2} - 15445\right)^3 + 132\left(12 s_{1,1} + 133 s_{1,2} - 26929\right)^3 + 31513,$$
$${s_{2,2}}^3 - 12\left(-11 s_{1,1} - 132 s_{1,2} - 15445\right)^3 - 133\left(12 s_{1,1} + 133 s_{1,2} - 26929\right)^3 + 27923,$$

$$-11\left(-11 s_{2,1} - 132 s_{2,2} + 15957\right)^3 - 132\left(12 s_{2,1} + 133 s_{2,2} - 113\right)^3 + 2578,$$
$$12\left(-11 s_{2,1} - 132 s_{2,2} + 15957\right)^3 + 133\left(12 s_{2,1} + 133 s_{2,2} - 113\right)^3 + 8446 - y_1]$$

Remark

We can remove the first and the last equations: these equations are linear in x_1 and y_1. In general we can **remove** the first k equations and the last k equations.

CICO second example and optimizations

We take, $N = 2$, $m = 3$, $k = 1$ and $\mathscr{M} = \begin{bmatrix} -11 & -132 \\ 12 & 133 \end{bmatrix}$

We take for the input/output vectors:

$$X = [0, x_1, x_2] \text{ and } Y = [0, y_1, y_2]$$

We have the following equations (removing the linear equations):

$[{s_{1,1}}^3 - 1331\left(29260 - 19534 s_{0,2} - 24464 s_{0,3}\right)^3 + 19534\left(-31963 + 3309 s_{0,2} + 25225 s_{0,3}\right)^3 +$
$24464\left(-3157 + 16226 s_{0,2} - 760 s_{0,3}\right)^3 - 29116,$
${s_{1,2}}^3 + 1463\left(29260 - 19534 s_{0,2} - 24464 s_{0,3}\right)^3 - 3309\left(-31963 + 3309 s_{0,2} + 25225 s_{0,3}\right)^3 -$
$25225\left(-3157 + 16226 s_{0,2} - 760 s_{0,3}\right)^3 - 21978,$
${s_{1,3}}^3 - 133\left(29260 - 19534 s_{0,2} - 24464 s_{0,3}\right)^3 - 16226\left(-31963 + 3309 s_{0,2} + 25225 s_{0,3}\right)^3 +$
$760\left(-3157 + 16226 s_{0,2} - 760 s_{0,3}\right)^3 - 11587,$

$$1331\left(1331 s_{1,1} - 19534 s_{1,2} - 24464 s_{1,3} - 15445\right)^3 - 19534\left(-1463 s_{1,1} + 3309 s_{1,2} + 25225 s_{1,3} - 26929\right)^3 -$$
$$24464\left(133 s_{1,1} + 16226 s_{1,2} - 760 s_{1,3} - 22939\right)^3 - 31513]$$

Remark

Clearly we have 4 equations and 5 variables: we can fix arbitrarily $s_{0,3}$ (or equivalently x_2). In general we can **fix** the last $m - 2k$ variables of the input vector $X = [0, \ldots, 0, x_1, \ldots, x_k, \ , \ldots, \ ]$.

Structure of the new system of equations

CICO problem (k) for Rescue$_d(N,m,p)$
Assuming that $X = [0,\ldots,0,x_1,\ldots,x_k,\ ,\ldots,\ ]$
We have linear equations (that we can ignore):

$$\left\{ \ s^d_{0,i} - x_{i-k} - \alpha_i \text{ for i=(k+1),...,m} \right.$$

where $\alpha_i \in \mathbb{K}$ and we have non linear equations:

$$(\mathscr{S}) \left\{ \begin{array}{l} s^d_{1,i} - P_{1,i}(s_{0,k+1},\ldots,s_{0,2k}) \text{ for i=1,...,m} \\ \hline s^d_{2,i} - P_{2,i}(s_{1,1},\ldots,s_{1,m}) \text{ for i=1,...,m} \\ \quad \ldots \\ s^d_{N-1,i} - P_{N-1,i}(s_{N-2,1},\ldots,s_{N-2,m}) \text{ for i=1,...,m} \\ \hline P_{N,i}(s_{N-1,1},\ldots,s_{N-1,m}) \text{ for i=1,...,k} \end{array} \right.$$

where $P_{i,j}$ are polynomials of degree d.

Parameters

Name	Field	$n = \log_2 p$	N	m	k
Rescue128a	$\mathbb{F}_p$	125	16	4	2
Rescue128b	$\mathbb{F}_p$	253	22	12	1
Rescue128c	$\mathbb{F}_p$	125	10	12	2
Rescue128d	$\mathbb{F}_p$	61	10	12	4
Rescue128e	$\mathbb{F}_p$	253	10	11	1

We remark that $m \geq 2k$, so that we can apply the previous strategy and fix $m - 2k$ variables.

We report now some Gröbner bases experiments: as usual this is $TDRL + TFGLM$ where TDRL (resp. TFGLM) is the CPU time of the DRL Gröbner basis (resp. the time to change the monomial ordering).

When $k = 1$ or 2 is possible to speedup the computations by using the sparse structure of the equations.

Experiments: standard tools d=3

#	Nb	m	k	#Sol	TDRL	TFGLM	Dmax
#	2	2	1	3^3	0.0s	0.0s	6
#	2	3	1	3^4	0.0s	0.0s	7
#	2	4	1	3^5	0.0s	0.0s	9
#	2	4	2	3^6	0.09s	0.04s	12
#	3	2	1	3^5	0.0s	0.0s	8
#	3	3	1	3^7	0.42s	0.73s	13
#	3	4	1	3^9	124.9s	501.6s	16
#	4	2	1	3^7	0.22s	0.77s	12
#	5	2	1	3^9	44.4s	543.1s	16

Experiments: standard tools d=2

#	Nb	m	k	#Sol	TDRL	TFGLM	Dmax
#	2	4	2	2^6	0s	0s	7
#	2	5	2	2^7	0.01s	0s	8
#	3	2	1	2^5	0s	0s	5
#	3	4	2	2^10	0.47s	0.14s	11
#	3	5	2	2^12	19.8s	6.5 s	12
#	4	2	1	2^7	0s	0s	7
#	4	3	1	2^8	0.29s	0.13s	9
#	5	2	1	2^9	0.04s	0.02s	9

CICO Problem:complexity of the attack

From the experimental results, we see that all the algebraic systems behave like regular systems. So that we can derive sharp complexity bounds.
We observe similar results for $d = 2$ or $d = 3$ (but computations are much faster for $d = 2$ of course).
We count the number of solutions of the system $(\mathscr{S})$ by counting the number of equations: In $(\mathscr{S})$ we have,
$E = m + (N-2)m + k = (N-1)m + k$ equations of degree d.
Consequently,

Theorem (Number of Solutions)

The number of solutions of the CICO problem (k) for $\text{Rescue}_d(N, m, p)$:

$$D = 3^{(N-1)m+k}$$

CICO Problem:comparison with exhaustive search

The complexity of the attack will be polynomial in this number $D = 3^{(N-1)m+k}$. It is important to compare D with the cost of the exhaustive search which is $p^k \approx 2^{256}$.

Name	$n = \log_2 p$	N	m	k	p^k	D
Rescue128a	125	16	4	2	2^{250}	$2^{98.3}$
Rescue128b	253	22	12	1	2^{253}	$2^{401.0}$
Rescue128c	125	10	12	2	2^{250}	$2^{174.3}$
Rescue128d	61	10	12	4	2^{244}	$2^{177.5}$
Rescue128e	253	10	11	1	2^{253}	$2^{158.5}$

We see that the number D vary significantly. We have to investigate precisely to complexity of the attack.

CICO Problem:complexity of the attack

The complexity of the attack is polynomial in the degree D of the univariate polynomial: $O(D^{\omega})$
When $k = 1$ or 2, we take $\omega = 1$; when $k > 2$ then $\omega = 2$.

Proposition

The complexity of the attack is:

k	1 or 2	> 2
Complexity	$3^{(N-1)m+k}$	$3^{2((N-1)m+k)}$

Name	$n = \log_2 p$	N	m	k	p^k	CICO
Rescue128a	125	16	4	2	2^{250}	$2^{98.3}$
Rescue128b	253	22	12	1	2^{253}	$2^{401.0}$
Rescue128c	125	10	12	2	2^{250}	$2^{174.3}$
Rescue128d	61	10	12	4	2^{244}	2^{355}
Rescue128e	253	10	11	1	2^{253}	$2^{158.5}$

Hybrid method (1)

For Rescue128d the size of the ground field $\mathbb{K} = GF(2^{61} + 20\,2^{32} + 1)$ is “small”. An hybrid method is a combination between exhaustive search and Gröbner bases computation.

For the system $(\mathscr{S})$ the maximal degree occurring in the (DRL) Gröbner basis is given by

$$d_{max} = 1 + (d-1)E = 2((N-1)m + k) + 1$$

By exhaustive search, we can find the value of one variable of $(\mathscr{S})$: we keep the same the number of equations but we decrease by one the number of variables. Under semi-regularity assumption of the system, the new maximum degree is given by:

$$d_{max} = \left\lceil \frac{d_{max}}{2} \right\rceil = E + 1 = ((N-1)m + k) + 1$$

Experiments: standard tools d=3

We check experimentally the semi-regularity of the system when we fix one variable:

```
#   Nb    m     k    TDRL     Dmax
#   2     4     2    0.0s     7
#   2     5     2    0.08s    8
#   2     6     2    0.95s    9
#   3     4     2    259.4s   11
```

Conclusion: the system behaves exactly as expected.

Hybrid method (2)

$$d_{max} = \left\lceil \frac{d_{max}}{2} \right\rceil = E + 1 = ((N-1)m + k) + 1$$

However, the total complexity estimate of this computation is very pessimistic: it is equal to $\left(\begin{array}{c} E+E-1+1 \\ E-1 \end{array} \right)^{\omega} = \left(\begin{array}{c} 2E \\ E-1 \end{array} \right)^{\omega}$

In total, the complexity of the exhaustive search and the Gröbner basis is $C = 2^n \left(\begin{array}{c} 2E \\ E-1 \end{array} \right)^{\omega}$.

For Rescue128d, then N=10,m=12,k=4 we have $E = 114$, so that $C = 2^{61+223.7\omega}$. Since $k - 1 = 3 \gg 1$ we cannot expect to have $\omega = 2$.

☛ The Hybrid method is **not efficient** in this case.