Arion: Arithmetization-Oriented Permutation and Hashing from Generalized Triangular Dynamical Systems

Arnab Roy, Matthias Johann Steiner, and Stefano Trevisani

Alpen-Adria-Universität Klagenfurt, Universitätsstraße 65-67, 9020 Klagenfurt am Wörthersee, Austria
firstname.lastname@aau.at

Abstract. In this paper we propose the block cipher Arion and the hash function ArionHash over prime fields $\mathbb{F}_p$. The design of Arion is based on the newly introduced Generalized Triangular Dynamical System (GTDS), which generalizes and encapsulates the design strategies of Substitution Permutation Networks and Feistel Networks via a generic iterative polynomial dynamical system. At round level Arion is the first design which is instantiated using the new GTDS. We provide extensive security analysis of our construction including algebraic cryptanalysis (e.g. interpolation and Gröbner basis attacks) that are particularly decisive in assessing the security of permutations and hash functions over $\mathbb{F}_p$.
The aim of our constructions is to achieve low multiplicative complexity. From a application perspective, ArionHash is aimed for efficient implementation in zkSNARK protocols and zero-knowledge proof systems. From a theoretical point of view, we reveal that arbitrary bijections between graphs can lead to a more efficient implementation of arithmetization-oriented primitives. This also generalizes the recently revealed relation between arithmetization-orientation and CCZ-equivalence of graphs.
We compare the efficiency of ArionHash in a R1CS setting with other hash functions such as Poseidon, `Anemoi` and Griffin. For demonstrating the practical efficiency of ArionHash we implemented it with the zkSNARK library `libsnark`. Our result shows that ArionHash is significantly faster than Poseidon - a hash function designed for zero-knowledge proof systems. We also found that an aggressive version of ArionHash is considerably faster than Griffin in a practical zkSNARK setting.

1 Introduction

With the advancement of Zero-Knowledge (ZK), Multi-Party Computation (MPC) and Fully Homomorphic Encryption (FHE) in recent years new efficiency measures for symmetric-key primitives allowing efficient implementation in these schemes, namely low multiplicative complexity and low multiplicative depth, have been introduced. The block ciphers, permutations and hash functions with low multiplicative complexity are also referred to as Arithmetization-Oriented (AO) primitives. A significant number of these new types of AO primitives are

defined over large finite fields of prime characteristic $p \geq 2$. Our focus in this paper will be such a low multiplicative complexity construction over $\mathbb{F}_p$.

To put this paper into context with previous AO constructions we give a short overview of their developments. The AO primitives proposed in literature till now can be categorized into three generations.

Gen I: `LowMC` [4], `MiMC` [3]
Gen II: Hades [35], Neptune & Poseidon [34], `GMiMC` [2], Rescue-Prime [5]
Gen III: `Reinforced Concrete` [7], Griffin [33], `Anemoi` [18]

The first generation consists of constructions which demonstrated that one can construct secure and efficient ciphers and hash functions with low degree primitives at round level. In the second generations researcher tried to tweak the Feistel and the Substitution Permutation Network (SPN) to obtain new efficient primitives. This for example resulted in the widely adopted partial SPN strategy. The current third generation adopts new design principles which neither reduce to the Feistel or the SPN that culminated in the Generalized Triangular Dynamical System (GTDS) [1]. Moreover, this generation diverted from the consensus that one needs low degree polynomials to instantiate a secure and efficient AO primitive.

In this paper we propose new AO primitives - Arion (block cipher) and the hash function derived from it ArionHash. At round level Arion (and ArionHash) like Griffin, utilize(s) a polynomial of very high degree in one branch and low degree polynomials in the remaining branches to significantly cut the number of necessary rounds compared to the previous generations. `Anemoi` also utilizes a high degree permutation, the so-called open `Flystel`, at round level, but to limit the number of constraints in a prover circuit the authors proved that the open `Flystel` is CCZ-equivalent (cf. [22] and [18, §4.2]) to a low degree permutation, the so-called closed `Flystel`. Lastly, `Reinforce Concrete` is the first AOC that utilizes look-up tables which significantly reduces the number of necessary rounds of `Reinforced Concrete` and consequently also the number of constraints in a prover circuit.

1.1 Our Results

In this paper we propose the block cipher Arion and the hash function ArionHash, using the Generalized Triangular Dynamical System [1][1]. The block cipher and hash function are constructed over $\mathbb{F}_p$ with the target to achieve low multiplicative complexity. Utilizing the structure of GTDS enables us to provide a systematic security analysis of the newly proposed block cipher and hash function. The GTDS structure also allows us to choose the best suited parameters for the efficiency. We provide extensive security analysis of the proposed block cipher and hash function against state-of-the-art cryptanalysis techniques to justify their security. Our construction validates the soundness of the generic GTDS

[1] Paper is provided as supplementary material for the submission.

structure that uses polynomial dynamical system for constructing cryptographic permutations over finite fields.

Although Arion and ArionHash are defined on arbitrary finite fields $\mathbb{F}_p$, the parameters of the block cipher and hash function are chosen in such way to compatible with the primes chosen for the target ZK application namely, for BLS12 and BN254 curves. We propose aggressive versions of Arion and ArionHash namely, α-Arion and α-ArionHash. The difference between Arion (and ArionHash) and its aggressive version is that the former avoids a recently proposed probabilistic Gröbner basis [29] attack.

To demonstrate and compare the efficiencies of our constructions we implemented them using the zkSNARK library `libsnark` [45] - a `C++` library used in the privacy protecting digital currency Zcash. Our result shows that ArionHash is significantly (2x) faster than Poseidon - a efficient hash function designed for zkSNARK applications. The efficiency of ArionHash is comparable to the recently proposed (but not yet published at a peer-reviewed venue) hash function Griffin. We find that α-ArionHash for practical choices of parameters in a Merkle tree mode of hashing is faster than Griffin. We also reveal that a bijection between the graphs of the ArionHash GTDS and another closely related GTDS leads to a more efficient implementation of ArionHash in ZK applications compared to the naive circuit for ArionHash. In particular, this generalizes the recently revealed relation between arithmetization-orientation and CCZ-equivalence (cf. [18, §4.2]).

2 The (Keyed) Permutation and Hash Function

2.1 Overview of the Design Rationale

Before we define Arion and ArionHash we quickly summarize the design rationale behind our construction.

- By utilizing the GTDS to instantiate the permutation we aim to achieve fast degree growth in each component like in SPNs and non-linear mixing between the components as in Feistel network. Our GTDS, see Definition 1, incorporates the strength of both SPN and Feistel in a single primitive at round level.
- It follows from the generic security analysis in [1, §5] that the univariate permutations, the SPN part, of the GTDS determine worst case security bounds against differential cryptanalysis. Hence, we chose parameters that minimize this bound.
- To thwart interpolation attacks we opted for a design that can achieve a degree overflow in the input variables in the first round, see Lemma 2 and Remark 3. This is achieved by applying a low degree univariate permutation P_1 in all branches except the last one and by applying a high-degree inverse permutation P_2^{-1} in the last branch.
- We opted for a linear layer that mixes all branches in every round. This is achieved with a circulant matrix which has only non-zero entries.

– For the high degree inverse permutation, the naive circuit for $P_2^{-1}(x) = y$ introduces many multiplicative constraints, though one can always transform such a circuit into a circuit of P_2 at constant time, see Section 4. This trick plays a fundamental role in the efficiency of ArionHash circuits.

2.2 Keyed Permutation

We start with the definition of the generalized triangular dynamical system of Arion.

Definition 1 (GTDS of Arion). *Let $p \in \mathbb{P}$ be a prime, and let $\mathbb{F}_p$ be the field with p elements. Let $n, d_1, d_2, e \in \mathbb{Z}_{>1}$ be integers such that*

(i) d_1 *is the smallest positive integer such that* $\gcd(d_1, p-1) = 1$,
(ii) d_2 *is an arbitrary integer such that* $\gcd(d_2, p-1)$, *and*
(iii) $e \cdot d_2 \equiv 1 \mod p-1$.

For $1 \leq i \leq n-1$ let $\alpha_{i,1}, \alpha_{i,2}, \beta_i \in \mathbb{F}_p$ be such that $\alpha_{i,1}^2 - 4 \cdot \alpha_{i,2}$ is a quadratic non-residue modulo p. The generalized triangular dynamical system $\mathcal{F} = \{f_1, \ldots, f_n\}$ of Arion is defined as

$$\begin{aligned} f_i(x_1, \ldots, x_n) &= x_i^{d_1} \cdot g_i(\sigma_{i+1,n}) + h_i(\sigma_{i+1,n}), \qquad 1 \leq i \leq n-1, \\ f_n(x_1, \ldots, x_n) &= x_n^e, \end{aligned}$$

where

$$\begin{aligned} g_i(x) &= x^2 + \alpha_{i,1} \cdot x + \alpha_{i,2}, \\ h_i(x) &= x^2 + \beta_i \cdot x, \\ \sigma_{i+1,n} &= \sum_{j=i+1}^{n} x_j + f_j(x_1, \ldots, x_n). \end{aligned}$$

Since $\alpha_{i,1} - 4 \cdot \alpha_{i,2}$ is a non-residue modulo p for all $1 \leq i \leq n-1$ the polynomials g_i do not have any zeros over $\mathbb{F}_p$, therefore we can invert the GTDS with the procedure described in [1, Corollary 9]. For ArionHash and Arion we choose $d_2 \in \{121, 123, 125, 129, 161, 257\}$, because x^{d_2} can then be evaluated with 8 or 9 multiplications. To evaluate x^{121} one computes

$$y = \left(x^2\right)^2, \quad z = \left(y^2 \cdot y\right)^2, \quad x^{121} = \left(z^2\right)^2 \cdot z \cdot x. \tag{1}$$

To evaluate x^{123} one computes

$$y = x^2 \cdot x, \quad z = \left(\left(y^2\right)^2\right)^2, \quad x^{123} = \left(z^2\right)^2 \cdot z \cdot y. \tag{2}$$

To evaluate x^{125} one computes

$$y = \left(x^2\right)^2 \cdot x, \quad z = \left(\left(y^2\right)^2\right)^2, \quad x^{125} = z^2 \cdot z \cdot y. \tag{3}$$

To evaluate x^{161} one computes

$$y = \left(x^2\right)^2 \cdot x, \quad z = \left(\left(y^2\right)^2\right)^2, \quad x^{161} = \left(z^2\right)^2 \cdot x. \tag{4}$$

To evaluate x^{129} one computes $x^{128} \cdot x$ and x^{128} can be computed with 7 squaring operations[2]. To evaluate x^{257} one computes $x^{256} \cdot x$ and x^{256} can be computed with 8 squaring operations.

Let's compute the degrees of the polynomials in the GTDS.

Lemma 2. *Let $n, d_1, e \geq 1$ be integers. We consider the linear recurrence relation*

$$\begin{aligned} r_0 &= e, \\ r_i &= d_1 + 2 \cdot r_{i-1}, \ i \geq 1. \end{aligned}$$

Then

$$r_i = 2^i \cdot (d_1 + e) - d_1, \ i \geq 1,$$

and for the GTDS $\mathcal{F} = \{f_1, \ldots, f_n\}$ from Definition 1 we have that

$$\deg(f_i) = r_{n-i}.$$

Proof. After performing the index shift $i \mapsto n - i$ it is easy to see that the linear recurrence r_i indeed describes the degree of the f_i's. To compute the formula for r_i we use the method of generating functions. Let $R(x) = \sum_{i=0}^{\infty} r_i \cdot x^i$, we consider the equation

$$r_i \cdot x^i - 2 \cdot r_{i-1} \cdot x^i = d_1 \cdot x^i.$$

Summing the equation from 1 to ∞ yields

$$\sum_{i=1}^{\infty} r_i \cdot x^i - 2 \cdot \sum_{i=1}^{\infty} r_{i-1} \cdot x^i = d_1 \cdot \sum_{i=1}^{\infty} x^i = d_1 \cdot \left(\frac{1}{1-x} - 1\right).$$

If we compare the summands on the left-hand side with $R(x)$ and $x \cdot R(x)$ respectively, then we see that the first one is only missing the term r_0 and the second one is not missing any term. Therefore,

$$\begin{aligned} R(x) - e - 2 \cdot x \cdot R(x) &= d_1 \cdot \left(\frac{1}{1-x} - 1\right) \\ \Rightarrow R(x) &= \frac{e}{1 - 2 \cdot x} + \frac{d_1}{1 - 2 \cdot x} \cdot \left(\frac{1}{1-x} - 1\right) \\ &= \frac{e}{1 - 2 \cdot x} + \frac{d_1 \cdot x}{(1-x) \cdot (1 - 2 \cdot x)}. \end{aligned}$$

[2] Note that for the target primes BLS12 and BN254 x^{129} does not induce a permutation.

Applying the method of partial fractions to the second term yields

$$\begin{aligned} R(x) &= \frac{e}{1-2\cdot x} - \frac{d_1}{1-x} + \frac{d_1}{1-2\cdot x} \\ &= \sum_{i=0}^{\infty} \left((d_1+e)\cdot 2^i - d_1\right)\cdot x^i, \end{aligned}$$

and the claim follows. □

Remark 3. For

(1) $p = \text{BLS12}$ and $d_2 \in \{121, 123, 125, 161, 257\}$, and
(2) $p = \text{BN254}$ and $d_2 \in \{121, 123, 129, 161, 257\}$

we have that $4 \cdot e \geq p$. So for $n \geq 3$ and almost all specified choices for d_2 the Arion GTDS already achieves a degree overflow in the input variable x_n in its first component f_1.

To introduce mixing between the blocks we chose a circulant matrix whose product with a vector can be efficiently evaluated.

Definition 4 (Affine layer of Arion). *Let $p \in \mathbb{P}$ be a prime and let $\mathbb{F}_p$ be the field with p elements. The affine layer of Arion is defined as*

$$\mathcal{L}_{\mathbf{c}} : \mathbb{F}_p^n \to \mathbb{F}_p^n, \quad \mathbf{x} \mapsto \operatorname{circ}(1, \ldots, n)\,\mathbf{x} + \mathbf{c},$$

where $\operatorname{circ}(1, \ldots, n) \in \mathbb{F}_p^{n\times n}$ is the circular matrix with entries $1, \ldots, n$ in the first row and $\mathbf{c} \in \mathbb{F}_p^n$ is a constant vector.

Remark 5. For any prime number $p \in \mathbb{P}$ with at least $\log_2(p) \geq 39$ and all $2 \leq n \leq 12$ the matrix $\operatorname{circ}(1, \ldots, n)$ is a MDS matrix over $\mathbb{F}_p$.

The following algorithm provides an efficient way to evaluate the matrix vector product of $\operatorname{circ}(1, \ldots, n)$.

Algorithm 1 Efficient evaluation for circulant matrix vector product

Input
$\mathbf{v} = (v_1, \ldots, v_n)^\intercal \in \mathbb{F}_p^n$
Output
$\operatorname{circ}(1, \ldots, n)\,\mathbf{v} \in \mathbb{F}_p^n$

Initialize $\mathbf{w} = (0, \ldots, 0) \in \mathbb{F}_p^n$.
Compute $\sigma = \sum_{i=1}^{n} v_i$ and set $w_1 = \sigma + \sum_{i=1}^{n} (i-1)\cdot v_i$.
Set $i = 2$.
while $i \leq n$ **do**
    Set $w_i = w_{i-1} - \sigma + n \cdot v_{i-1}$.
    $i = i + 1$.
return w

To define a keyed permutation we need a key addition which we denote as

$$\mathcal{K}_{\mathbf{k}} : \mathbb{F}_p^n \times \mathbb{F}_p^n \to \mathbb{F}_p^n, \qquad (\mathbf{x}, \mathbf{k}) \mapsto \mathbf{x} + \mathbf{k}.$$

The keyed permutation Arion is now defined as follows.

Definition 6 (Arion). *Let $p \in \mathbb{P}$ be a prime and let $\mathbb{F}_p$ be the field with p elements, and let $n > 1$ and $r \geq 1$ be integers. For $1 \leq i \leq r$ let $\mathcal{F}^{(i)} : \mathbb{F}_p^n \to \mathbb{F}_p^n$ be generalized triangular systems from Definition 1 and for $1 \leq i \leq r$ let $\mathcal{L}_{\mathbf{c}_i} : \mathbb{F}_p^n \to \mathbb{F}_p^n$ be affine layers from Definition 4. The i^{th} round function of Arion is defined as*

$$\begin{aligned} \mathcal{R}_{\mathbf{k}}^{(i)} &: \mathbb{F}_p^n \times \mathbb{F}_p^n \to \mathbb{F}_p^n, \\ &(\mathbf{x}, \mathbf{k}) \mapsto \mathcal{K}_{\mathbf{k}} \circ \mathcal{L}_{\mathbf{c}_i} \circ \mathcal{F}^{(i)}(\mathbf{x}). \end{aligned}$$

Then Arion is defined as the following composition

$$\begin{aligned} \mathit{Arion} &: \mathbb{F}_p^n \times \mathbb{F}_p^{n \times (r+1)} \to \mathbb{F}_p^n, \\ &\left(\mathbf{x}, \mathbf{k}_0, \mathbf{k}_1, \ldots, \mathbf{k}_r, \right) \mapsto \mathcal{R}_{\mathbf{k}_r}^{(r)} \circ \cdots \circ \mathcal{R}_{\mathbf{k}_1}^{(1)} \circ \mathcal{L}_{\mathbf{0}} \circ \mathcal{K}_{\mathbf{k}_0}(\mathbf{x}). \end{aligned}$$

Further, we denote with Arion-π the unkeyed permutation.

2.3 Hash Function

For the hash function ArionHash over $\mathbb{F}_p^n$ we instantiate Arion-π in sponge mode [12,13]. The state size $n = r + c$ is split into the rate part r and the capacity part c. In [13, Theorem 2] it has been proven that for a random permutation the sponge construction is indistinguishable from a random oracle up to $\min\left\{p^r, p^{c/2}\right\}$ queries. Therefore, to provide κ bits of security, $p^r \geq 2^\kappa$ and $p^{c/2} \geq 2^\kappa$, we require that

$$r \geq \frac{\kappa}{\log_2(p)}, \qquad \text{and} \qquad c \geq \frac{2 \cdot \kappa}{\log_2(p)}. \tag{5}$$

Given an input message m we choose a similar padding rule as for Poseidon [34, §4.2], we add the smallest number of zeros $< r$ such that the size of $m \mid\mid 0^*$ is a multiple of r. If we have to pad the message, then we replace the initial value $\mathtt{IV} \in \mathbb{F}_p^c$ with $|m| \mid\mid \mathtt{IV}' \in \mathbb{F}_p^c$, where $|m| \in \mathbb{F}_p$ is the size of the input message m and $\mathtt{IV}' \in \mathbb{F}_p^{c-1}$ is an initial value.

2.4 Instantiations

In Table 1 we provide the parameters for Arion and ArionHash and their aggressive versions α-Arion and α-ArionHash over prime fields $p \geq 2^{60}$ and $d_1, d_2 \in \mathbb{Z}$ such that $\gcd(d_i, p-1) = 1$, $d_1 = 3, 5$ and $121 \leq d_2 \leq 257$. The number of rounds for Arion and ArionHash are chosen to provide 128-bit security against the most efficient probabilistic algorithm (available till date) for polynomial system solving in a Gröbner basis attack on ArionHash. The number of rounds for α-Arion and

α-Arion**Hash** are chosen to provide 128-bit security against the most efficient deterministic algorithm for polynomial system solving in a Gröbner basis attack on Arion**Hash**. For more details on the security with respect to Gröbner basis attacks we refer to Section 3.3 and Table 7. Our target primes for Arion and Arion**Hash** are BLS12 and BN254[3].

Table 1. Arion, Arion**Hash**, α-Arion and α-Arion**Hash** parameters for 128 bits of security and for primes $p \geq 2^{60}$ and $d_1, d_2 \in \mathbb{Z}$ such that $\gcd(d_i, p-1) = 1$ and $121 \leq d_2 \leq 257$.

		Rounds	
d_1	Blocks	Arion & Arion**Hash**	α-Arion & α-Arion**Hash**
3	3	6	5
5	3	6	4
3	4	6	4
5	4	5	4
3	5	5	4
5	5	5	4
3	6	5	4
5	6	5	4
3	8	4	4
5	8	4	4

3 Security Analysis of **Arion**

3.1 Statistical Attacks on **Arion**

Differential Cryptanalysis Differential cryptanalysis [16] and its variants are the most widely applied attack vectors against symmetric-key ciphers. It is based on the propagation of input differences through the rounds of a block cipher. In its base form an attacker requests the cipher texts for a large number of plain texts. Then he assumes that for $r-1$ rounds the input difference is $\boldsymbol{\Delta x} \in \mathbb{F}_q^n \setminus \{\mathbf{0}\}$ and the output difference is $\boldsymbol{\Delta y} \in \mathbb{F}_q^n$. Under the assumption that the differences in the last round are fixed the attacker can then deduce the possible keys. The key quantity to estimate the effectiveness of differential cryptanalysis is the so-called differential uniformity.

Definition 7. *Let $\mathbb{F}_q$ be a finite field, and let $f : \mathbb{F}_q^n \to \mathbb{F}_q^m$ be a function.*

(1) The differential distribution table of f at $\mathbf{a} \in \mathbb{F}_q^n$ and $\mathbf{b} \in \mathbb{F}_q^m$ is defined as

$$\delta_f(\mathbf{a}, \mathbf{b}) = \left|\{\mathbf{x} \in \mathbb{F}_q^n \mid f(\mathbf{x}+\mathbf{a}) - f(\mathbf{x}) = f(\mathbf{b})\}\right|.$$

[3] BLS12=0x73eda753299d7d483339d80809a1d80553bda402fffe5bfeffffffff00000001, BN254=0x30644e72e131a029b85045b68181585d2833e84879b9709143e1f593f0000001.

(2) The differential uniformity of f is defined as

$$\delta(f) = \max_{\substack{\mathbf{a}\in\mathbb{F}_q^n\setminus\{\mathbf{0}\},\\ \mathbf{b}\in\mathbb{F}_q^m}} \delta_f(\mathbf{a},\mathbf{b}).$$

Given the differential uniformity of a function one can upper bound the success probability of differential cryptanalysis with input differences $\boldsymbol{\Delta x} \in \mathbb{F}_q^n \setminus \{\mathbf{0}\}$ and $\boldsymbol{\Delta y} \in \mathbb{F}_q^m$ by

$$\mathbb{P}\left[f : \boldsymbol{\Delta x} \to \boldsymbol{\Delta y}\right] \leq \frac{\delta(f)}{q^n}. \tag{6}$$

Naturally, the lower the differential uniformity the stronger is the resistance of a block cipher against differential cryptanalysis.

By definition of the Arion GTDS, see Definition 1, we have that $d_2 \geq d_1$. With [1, Theorem 20] the differential distribution table of the Arion GTDS can be upper bounded by

$$\delta_{\mathcal{F}}(\boldsymbol{\Delta x}, \boldsymbol{\Delta y}) \leq p^{n-\mathrm{wt}(\boldsymbol{\Delta x})} \cdot d_2^{\mathrm{wt}(\boldsymbol{\Delta x})} \leq p^{n-1} \cdot d_2, \tag{7}$$

where $\boldsymbol{\Delta x} \in \mathbb{F}_p^n \setminus \{\mathbf{0}\}$, $\boldsymbol{\Delta y} \in \mathbb{F}_p^n$, and $1 \leq \mathrm{wt}\left(\boldsymbol{\Delta x}\right) \leq n$ denotes the Hamming weight, i.e. the number of non-zero entries, of $\boldsymbol{\Delta x}$. Henceforth, the maximal success probability of any differential is bounded by

$$\mathbb{P}\left[\mathcal{F} : \boldsymbol{\Delta x} \to \boldsymbol{\Delta y}\right] \leq \left(\frac{d_2}{p}\right)^{\mathrm{wt}(\boldsymbol{\Delta x})} \leq \frac{d_2}{p}. \tag{8}$$

If we assume that the differences are independent among the rounds of Arion, then the security level κ can be computed via

$$\mathbb{P}\left[\mathsf{Arion} : \boldsymbol{\Delta x} \to \boldsymbol{\Delta y}\right] \leq \left(\frac{d_2}{p}\right)^{r} \leq 2^{-\kappa}. \tag{9}$$

Thus, for $p \geq 2^N$ and $d_2 \leq 2^M$ we obtain

$$\kappa \leq r \cdot (N - M). \tag{10}$$

Now recall that Arion is supposed to be instantiated with a 64 bit prime number $p \gtrsim 2^{64}$ and that $d_2 \in \{121, 123, 125, 129, 161, 257\}$, then Equation (9) implies the following security levels for Arion.

Table 2. Security level of Arion against any differential characteristic for $p \geq 2^N$ and $d_2 \leq 2^9$.

r	N	κ (bits)
3	60	153
2	120	222
1	250	241

Truncated Differential, Impossible Differential and Rebound Attacks In a truncated differential attack [38] an attacker can only predict parts of the difference between pairs of text. Since the matrix $\mathrm{circ}(1, \ldots, n)$ provides full diffusion over a single round, a truncated differential with probability 1 holds for a single round only. To extend it an attacker requires that some differences are equal to zero, though this probability is p^{-1}. Moreover, since Equation (9) is a worst case bound that applies to all possible differentials we do not expect that this attack outperforms the classical differential cryptanalysis.

Impossible differential cryptanalysis [15] exploits differentials that occur with probability 0. For this attack one combines two (truncated) differentials with probability 1 that collide in the middle. Although this attack can be set up for two rounds with (truncated) differentials adding additional rounds protects against the attack.

In a rebound attack [40, 43] one has to find two input/output pairs such that the inputs satisfy a certain (truncated) input difference and the outputs satisfy a certain (truncated) output difference. Such an attack can be split into two phase: an *inbound* and an *outbound* phase. Let $P_{\mathsf{Arion}} : \mathbb{F}_q^n \to \mathbb{F}_q^n$ be the target permutation, then we split it into three sub-parts $P_{\mathsf{Arion}} = P_{fw} \circ P_{in} \circ P_{out}$. The inbound phase is placed in the middle followed by the two outbound phases. Then, in the outbound phase two high-probability (truncated) differential trails are constructed which are connected with the inbound phase. Since a truncated differential with probability 1 can only cover a single round an attacker can cover only $r - 2$ rounds with an inside-out approach. Thus, we have to add two additional rounds to the values from Table 2 to nullify this attack vector.

Other Statistical Attacks Since Arion and ArionHash are secure against classical and truncated differential attacks, we conjecture that they are also secure against other statistical attacks, including linear cryptanalysis [42], impossible differential and zero-correlation attacks [6, 17, 42], boomerang attack [47], integral [14, 25], multiple-of-n and mixture differential attacks [32, 36]. This conclusion is also supported by the fact full diffusion is achieved after a single round.

3.2 Algebraic Attacks

Higher-Order Differential & Interpolation Attacks Interpolation attacks [37] construct the polynomial vector representing a cipher without knowledge of the secret key. If such an attack is successful against a cipher, then an adversary can encrypt any plain text without knowledge of the secret key. For a hash function the interpolated polynomial vector can be exploited to set up collision or forgery attacks. The cost of interpolating a polynomial depends on the number of monomials present in the polynomial vector representing the cipher function. Recall that any function $F : \mathbb{F}_q^n \to \mathbb{F}_q$ can be represented by a unique polynomial $f \in \mathbb{F}_q[\mathbf{X}_n] = \mathbb{F}_q[x_1, \ldots, x_n]/\left(x_1^q - x_1, \ldots, x_n^q - x_n\right)$, thus at most q^n monomials can be present in f. Clearly, if f is dense, then an interpolation attack cannot be done faster than exhaustive search.

Let $\mathcal{R}^{(i)} \subset \mathbb{F}_q[\mathbf{X}_n]$ denote the polynomial vector of the Arion round function, we say that $\mathcal{R}^{(i+1)} \circ \mathcal{R}^{(i)}$ has a degree overflow if we have to reduce with at least one of the field equations to compute the unique representation in $\mathbb{F}_q[\mathbf{X}_n]$. It is immediate from the definition of the round function that after 2 rounds a degree overflow occurs in every component and by the binomial theorem some monomials with degree greater than or equal to $q-2$ will be present in the reduced polynomial representation of $\mathcal{R}^{(i+1)} \circ \mathcal{R}^{(i)}$. Moreover, after a degree overflow we expect that a big fraction of the monomials in $\mathbb{F}_q[\mathbf{X}_n]$ is present in the components of the polynomial vector of Arion. Further, to frustrate Meet-in-the-Middle (MitM) attacks we require that the number of rounds $r \geq 4$. We implemented Arion in SageMath [46] to compute the density of Arion-π for small primes. Our findings suggest that after two rounds Arion-π has already almost full density over $\mathbb{F}_p$.

Table 3. Observed minimum density for Arion-π for small primes, $n = 3, 4, 5$ and $d_1, d_2 = 3, 5$.

p	r	minimum density after 2 rounds
11	6	$\geq 82\%$
13	6	$\geq 91\%$
17	6	$\geq 91\%$
19	6	$\geq 93\%$
23	6	$\geq 95\%$

Higher-order differential attacks [14, 38, 39] exploit that higher differentials will vanish at some point. Since Arion achieves degrees greater than or equal to $q-2$ in the input variables we expect Arion to resist against higher-order differentials and distinguishers.

3.3 Gröbner Basis Attacks

In a Gröbner basis attack [19, 24] the adversary represents a cryptographic function as fully determined system of polynomial equations and then solves for the solutions of the system. Since the system is fully determined at least one solution of the polynomial system must contain the quantity of interest, e.g. the key of a block cipher or the preimage of a hash function. In general, a Gröbner basis attack proceeds in four steps:

(1) Model the cryptographic function with a (iterated) system of polynomials.
(2) Compute a Gröbner basis with respect to an efficient term order, e.g., the degree reverse lexicographic order.
(3) Perform a term order conversion to an elimination order, e.g., the lexicographic order.
(4) Solve the univariate equation.

Let's for the moment assume that an adversary has already found a Gröbner basis and discuss the complexity of the remaining steps. Let k be a field, let $I \subset k[x_1, \ldots, x_n]$ be an zero-dimensional ideal modeling a cryptographic function, and let $d = \dim_k \left(k[x_1, \ldots, x_n]/I\right)$ be the k-vector space dimension of the quotient space. With the original FGLM algorithm [30] the complexity of term order conversion is $\mathcal{O}\left(n \cdot d^3\right)$, but improved versions with probabilistic methods achieve $\mathcal{O}\left(n \cdot d^\omega\right)$ [29], where $2 \leq \omega < 2.3727$, and sparse linear algebra algorithms [31] achieve $\mathcal{O}\left(\sqrt{n} \cdot d^{2+\frac{n-1}{n}}\right)$. To solve the univariate equation[4] $f \in \mathbb{F}_q[x]$ with $d = \deg\left(d\right)$ most efficiently we perform a greatest common divisor method that has recently been described in [10, §3.1].

(1) Compute $g = x^q - x \mod f$.
The computation of $x^q \mod f$ requires $\mathcal{O}\Big(d \cdot \log\left(q\right) \cdot \log\left(d\right) \cdot \log\left(\log\left(d\right)\right)\Big)$ field operations with a double-and-add algorithm.
(2) Compute $h = \gcd\left(f, g\right)$.
By construction h has the same roots as f in $\mathbb{F}_q$ since $h = \gcd\left(f, x^q - x\right)$, but its degree is likely to be much lower.
This step requires $\mathcal{O}\left(d \cdot \log\left(d\right)^2 \cdot \log\left(\log\left(d\right)\right)\right)$ field operations.
(3) Factor h.
In general, the polynomial f coming from a 0-dimensional Gröbner basis has only a few roots in $\mathbb{F}_q$.
Thus, this step is negligible in complexity.

Note that this method is only performative if $\deg\left(f\right) < q$ else one has to exchange the roles of f and the field equation. Overall solving the polynomial system with probabilistic methods requires

$$\mathcal{O}\left(n \cdot d^\omega + d \cdot \log\left(q\right) \cdot \log\left(d\right) \cdot \log\left(\log\left(d\right)\right) + d \cdot \log\left(d\right)^2 \cdot \log\left(\log\left(d\right)\right)\right) \quad (11)$$

field operations, and with deterministic methods

$$\mathcal{O}\left(\sqrt{n} \cdot d^{2+\frac{n-1}{n}} + d \cdot \log\left(q\right) \cdot \log\left(d\right) \cdot \log\left(\log\left(d\right)\right) + d \cdot \log\left(d\right)^2 \cdot \log\left(\log\left(d\right)\right)\right) \quad (12)$$

field operations.

Let's now discuss the complexity of Gröbner basis computations. Today, the most efficient algorithms to compute Gröbner bases are Faugère's linear algebra-based algorithms F4 [27] and F5 [28]. The main idea of linear algebra-based Gröbner basis algorithms can be traced back to Lazard [41]. Let $\mathcal{F} = \{f_1, \ldots, f_m\} \subset P = k[x_1, \ldots, x_n]$ be a finite set of homogeneous polynomials over a field k. The *homogeneous Macaulay matrix* M_d of degree d has columns indexed by monomials in P_d and rows indexed by polynomials $m \cdot f_j$, where

[4] If I is an radical ideal, then the degree of the univariate polynomial in the elimination Gröbner is indeed d, though for non-radical ideals the degree can be larger than d. For a simple example in the non-radical case consider $(x^2) \subset k[x]$.

$m \in P$ is a monomial such that $\deg(m \cdot f_j) = d$. If $\mathcal{F}$ is an inhomogeneous system of polynomials, then one replaces M_d by $M_{\leq d}$ and the degree equality by an inequality. If one fixes a term order $>$ on P, then by performing Gaussian elimination on M_d respectively $M_{\leq d}$ for a large enough value of d one produces a $>$-Gröbner basis of $\mathcal{F}$. The least such d is called the solving degree $\mathrm{sd}_{>}(\mathcal{F})$ of $\mathcal{F}$. (The notion of solving degree was first introduced in [26], though we use the definition of [20].) The main improvement of F4/5 over Lazard's method is the choice of efficient selection criteria. Conceptually, the Macaulay matrix will contain many redundant rows, if one is able to avoid many of these rows with selection criteria, then the running time of an algorithm will improve. Nevertheless, with the notion of the solving degree it is possible to upper bound the maximal size of the computational universe of F4/5. It is well-known that the number of monomials in P of degree d is given by the multiset coefficient

$$N(n,d) = \left(\!\binom{n}{d}\!\right) = \binom{n+d}{d}. \tag{13}$$

We can now upper bound the size of the Macaulay matrix by $m \cdot d \cdot N(n,d) \times d \cdot N(n,d)$. Overall, we can bound the complexity of Gaussian elimination on the Macaulay matrix $M_{\leq d}$ by

$$\mathcal{O}\left(\binom{n+d}{d}^{\omega}\right), \tag{14}$$

where we absorb m and d in the implied constant since in general $N(n,d) \gg m, d$ and $\omega \geq 1$ is a linear algebra constant.

In the cryptographic literature a generic approach to bound the solving degree is the so-called *Macaulay bound*. Assume that $\deg(f_1) \geq \ldots \geq \deg(f_m)$ and let $l = \min\{m, n+1\}$, then the Macaulay bound of $\mathcal{F}$ is given by

$$\mathrm{MB}_{\mathcal{F}} = \sum_{i=1}^{l} \deg(f_1) + \ldots + \deg(f_l) - l + 1. \tag{15}$$

Up to date there are two known cases when one indeed has that $\mathrm{sd}_{DRL}(\mathcal{F}) \leq \mathrm{MB}_{\mathcal{F}}$:

(1) $\mathcal{F}$ is regular [8, 9], or
(2) $\mathcal{F}$ is in generic coordinates [20, Theorem 9, 10].

While the first case is well-known among cryptographers, the latter one has recently been introduced into mathematical cryptanalysis by Caminata and Gorla [20]. The notion of generic coordinates has first been introduced in [11], although it requires some knowledge of commutative and homological algebra its main idea is quite simple. Suppose we are given a set of homogeneous polynomials $\mathcal{F} = k[x_0, \ldots, x_n]$, where we consider x_0 as the homogenization variable. We would like to understand the projective variety $\mathcal{V}(\mathcal{F}) \subset \mathbb{P}_k^n$. Now recall that the points in the projective space $\mathbb{P}_k^n$ are given by equivalence classes of points in the affine space $\mathbb{A}_k^{n+1}$ such that $T \sim S \Leftrightarrow T = \lambda \cdot S$ where $T, S \in \mathbb{A}_k^{n+1} \setminus \{\mathbf{0}\}$ and

$\lambda \in k^\times$. On the other hand, if we would include the zero point $\mathbf{0}$, then this point would be equivalent to all other points in the projective space, clearly we want to avoid this case. Hence, the projective variety $\mathcal{V}(\mathcal{F})$ may not admit a solution at $x_0 = \ldots = x_n = 0$ and we say that its defining polynomials $\mathcal{F}$ are in generic coordinates if $\mathcal{V}(\mathcal{F})$ does not admit a solution with $x_0 = 0$. In technical terms this is expressed in [21, Definition 5]. Note that any homogeneous system can be transformed into generic coordinates with a suitable affine transformation [11, 2.10 Lemma]. Moreover, in [20, Theorem 11] it was proven that over a finite field $\mathbb{F}_q$ every polynomial system that contains all field equations is already in generic coordinates.

From a designers perspective, during all our small scale experiments the vector space dimension of the quotient space behaved more stable with respect to the chosen primes, branch sizes and round numbers than the observed complexity of Gröbner basis computation. **Thus, all our extrapolated security claims of** Arion and ArionHash **with respect to Gröbner basis attacks are expressed in the complexity of solving for the solutions of the polynomial system after a Gröbner basis has been found.**

Arion For the security of Arion against Gröbner basis attacks we consider the following model:

(i) We do not consider a key schedule, i.e., in every round we add the same key $\mathbf{k} = (k_1, \ldots, k_n) \in \mathbb{F}_p^n$.

(ii) We use a single plain/cipher pair $\mathbf{p}, \mathbf{c} \in \mathbb{F}_p^n$ given by Arion to set up a fully determined polynomial system $\mathcal{F}$.

(iii) For $1 \leq i \leq r-1$ we denote with $\mathbf{x}^{(i)} = \left(x_1^{(i)}, \ldots, x_n^{(i)}\right)$ the intermediate state variables, in addition we set $\mathbf{x}^{(0)} = \text{circ}\,(1, \ldots, n) \cdot (\mathbf{p} + \mathbf{k})$ and $\mathbf{x}^{(r)} = \mathbf{c}$. Further, for $1 \leq i \leq r$ we denote with $z^{(i)}$ an auxiliary variable. For our polynomial model we choose a slight modification $\tilde{\mathcal{F}} = \{\tilde{f}_1, \ldots, \tilde{f}_n\}$ of the Arion GTDS $\mathcal{F} = \{f_1, \ldots, f_n\}$, see Definition 1, where we set $\tilde{f}_n(x_n) = x_n$. For $1 \leq i \leq n-1$ the $\tilde{f}_i$ follow the same iterative definition as the original polynomials in the GTDS and we modify $\tilde{\sigma}_{i+1,n} = x_n + z_n + \sum_{j=i+1}^{n-1} x_j + \tilde{f}_j$. We consider the following polynomial model as the naive model $\mathcal{F}_{\text{naive}}$ for Arion

$$\text{circ}\,(1, \ldots, n) \cdot \begin{pmatrix} \tilde{f}_1^{(i)}\left(\hat{\mathbf{x}}^{(i-1)}, \mathbf{y}\right) \\ \vdots \\ \tilde{f}_n^{(i)}\left(\hat{\mathbf{x}}^{(i-1)}, \mathbf{y}\right) \end{pmatrix} + \mathbf{c}^{(i)} + \mathbf{k} - \begin{pmatrix} x_1^{(i)} \\ \vdots \\ x_n^{(i)} \end{pmatrix} = \mathbf{0},$$
$$\left(x_n^{(i)}\right)^e - z^{(i)} = 0,$$

where $\hat{\mathbf{x}}^{(i)} = \left(x_1^{(i)}, \ldots, x_{n-1}^{(i)}, z^{(i)}\right)$.

(iv) Obviously the naive polynomial system $\mathcal{F}_{\text{naive}}$ contains high degree equations given by the power permutation x^e. Though, if we replace the auxiliary

equations by

$$x_n^{(i)} - \left(z^{(i)}\right)^d = 0,$$

then we obtain a polynomial system $\mathcal{F}_{\mathsf{Arion}}$ whose polynomials are of small degree. Further, we expect the arithmetic of $\mathcal{F}_{\mathsf{Arion}}$ to be independent from the chosen prime, i.e., for primes $p, q \in \mathbb{P}$ such that $\gcd(d_i, p-1) = 1 = \gcd(d_i, q-1)$ we expect no notable difference for the complexity of a Gröbner basis attack.

Lemma 8. *Let $\mathbb{F}_p$ be a finite field, let $\mathcal{F}_{naive}$ and $\mathcal{F}_{\mathsf{Arion}}$ be the polynomial models from (iii) and (iv), and let F be the ideal of all field equations in the polynomial ring of $\mathcal{F}_{naive}$ and $\mathcal{F}_{\mathsf{Arion}}$. Then*

$$(\mathcal{F}_{naive}) + F = (\mathcal{F}_{\mathsf{Arion}}) + F.$$

Proof. By definition we have that $\left(x_n^{(i)}\right)^e \equiv z^{(i)} \mod (\mathcal{F}_{\text{naive}}) + F$, by raising this congruence to the d^{th} power we yield that

$$\left(\left(x_n^{(i)}\right)^e\right)^d \equiv x_n^{(i)} \equiv \left(z^{(i)}\right)^d \mod (\mathcal{F}_{\text{naive}}) + F$$

which proves the claim. □

I.e., on the solutions that solely come from the finite field $\mathbb{F}_p$, which are the solutions of cryptographic interest, the varieties corresponding to $\mathcal{F}_{\text{naive}}$ and $\mathcal{F}_{\mathsf{Arion}}$ coincide, i.e.,

$$\mathcal{V}(\mathcal{F}_{\mathsf{Arion}}) \cap \mathcal{V}(\mathcal{F}_{\text{naive}}) = \mathcal{V}(\mathcal{F}_{\mathsf{Arion}}) \cap \mathbb{F}_p^n.$$

Thus, $\mathcal{F}_{\mathsf{Arion}}$ is indeed a well-founded model for Arion.

To compute the Macaulay bound of $\mathcal{F}_{\mathsf{Arion}}$ we can use Lemma 2, we only have to set $e = 1$, further we have to account for the auxiliary equations, which yields

$$\begin{aligned} \text{MB}_{n,d_1,d_2}(r) &= r \cdot \left(n \cdot \left(2^{n-1} \cdot (d_1+1) - d_1\right) + d_2\right) + 1 - r \cdot (n+1) - 1 \\ &= r \cdot \left(n \cdot \left(2^{n-1} \cdot (d_1+1) - d_1 - 1\right) + d_2 - 1\right). \end{aligned} \tag{16}$$

We stress that we were unable to prove or disprove that $\mathcal{F}_{\mathsf{Arion}}$ is in generic coordinates, thus we cannot use the Macaulay bound as measure for the complexity of linear algebra-based Gröbner basis algorithms. Though, we will compare it to our experimentally observed solving degrees.

We implemented $\mathcal{F}_{\mathsf{Arion}}$ in the OSCAR computer algebra system [44] and computed the Gröbner basis with its F4 implementation. Unfortunately, the log function of F4 only prints the current working degree of the algorithm not its current solving degree. As remedy we estimate the empirical solving degree as follows: We sum up all positive differences of the working degrees between consecutive steps and add this quantity to the largest input polynomial

degree. After computing the Gröbner basis we also computed the $\mathbb{F}_p$-vector space dimension of the quotient ring. We conducted our experiments with the primes

$$\underbrace{\begin{aligned} p_1 &= 1013, \\ p_2 &= 10007, \end{aligned}}_{\gcd(3,p_i-1)=1} \quad \text{and} \quad \underbrace{\begin{aligned} p_3 &= 1033, \\ p_4 &= 15013. \end{aligned}}_{\gcd(5,p_i-1)=1}$$

All computations were performed on a AMD EPYC-Rome (48) CPU with 94GB RAM.

In Figure 1 we record our empirical results for $n = 2$ and in Figure 2 we record our empirical results for $n = 3$. From experiments we conclude that the solving degree is indeed bounded by the Macaulay bound and that the quotient space dimension grows exponential in r.

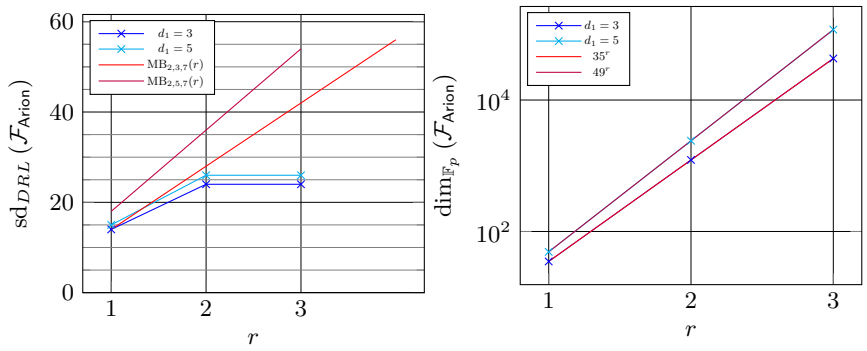

Fig. 1. Experimental solving degree and vector space dimension of the quotient ring for Arion with $n = 2$ and $d_2 = 7$.

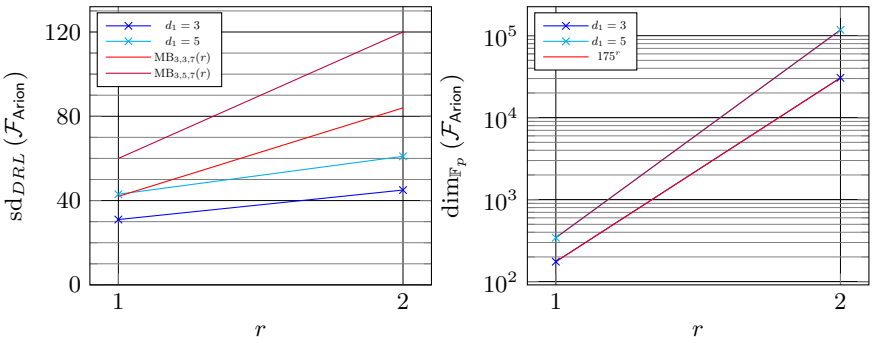

Fig. 2. Experimental solving degree and vector space dimension of the quotient ring for Arion with $n = 3$ and $d_2 = 7$.

To better understand the growth of the quotient space dimension we computed the quotient space dimension for $n \leq 4$ with $r = 1$, see Table 4.

Table 4. Empirical growth of the vector space dimension of the quotient space of Arion with $r = 1$.

n	d_1	d_2	$\dim_{\mathbb{F}_p,\text{emp}}$
2	3	7	35
2	3	257	1285
3	3	7	175
4	3	7	875
2	5	7	49
2	5	257	1799
3	5	7	343

From our experiments we deduce that the quotient space dimension grows or is bounded via

$$\dim_{\mathbb{F}_p}\left(\mathcal{F}_{\mathsf{Arion}}\right)(n,d,r) = \left(d_2 \cdot (d_1+2)^{n-1}\right)^r, \qquad n \geq 1. \tag{17}$$

To estimate the cost of a Gröbner basis computation we assume that the Macaulay bound always upper bounds the solving degree of Arion. Then we can combine Equations (14) and (15) to bound the complexity. For ease of computation we approximated the binomial coefficient with

$$\binom{n}{k} \approx \sqrt{\frac{n}{\pi \cdot k \cdot (n-k)}} \cdot 2^{n \cdot H_2(k/n)}, \tag{18}$$

where $H_2(p) = -p \cdot \log_2(p) - (1-p) \cdot \log_2(1-p)$ denotes the binary entropy (cf. [23, Lemma 17.5.1]). To estimate the cost of system solving we plug Equation (17) into Equations (11) and (12). For the probabilistic complexity estimate we assume that our adversary has an optimal algorithm with $\omega = 2$.

We base the security of Arion against Gröbner basis attacks solely on the complexity of solving the polynomial system. I.e., even if an adversary can compute a Gröbner basis in $\mathcal{O}(1)$ it must be computationally infeasible find a solution of the polynomial system. Thus, we estimate the security level κ of Arion against a Gröbner basis attack via

$$\kappa \leq \log_2\left(\mathcal{O}\left(\text{System solving}\right)\right). \tag{19}$$

Table 5. Empirical cost estimation of Gröbner basis attacks on Arion for primes $p \geq 2^{60}$ and $d_2 \geq 121$. The column GB contains complexity of Gröbner basis computations estimated via the Macaulay bound. For the probabilistic algorithm we assume that the adversary has an optimal algorithm with $\omega = 2$.

d_1	n	r	GB (bits)	Deterministic Solving (bits)	Probabilistic Solving (bits)
3	3	4	105	137	96
3	3	6	158	207	143
5	3	4	107	149	104
5	3	5	135	187	129
3	4	4	136	133	115
3	4	5	171	223	143
5	4	3	106	146	95
5	4	5	179	229	158
3	5	3	133	145	101
3	5	4	178	194	134
5	5	3	141	162	113
5	5	4	189	217	149
3	6	3	173	166	115
3	6	4	231	222	153
5	6	3	184	187	130
3	8	2	182	138	96
3	8	3	275	208	143
5	8	2	192	158	110
5	8	3	290	238	164

ArionHash In a preimage attack on ArionHash we are given a given hash value $\alpha \in \mathbb{F}_p$ and we have to find $\mathbf{x} \in \mathbb{F}_p^{r'}$ such that $\mathsf{ArionHash}(\mathbf{x}) = \alpha$. In a second-preimage attack we assume that we are given a message that consists of two input blocks, i.e. $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2) \in \left(\mathbb{F}_p^{r'}\right)^2$. Now we again have to find $\mathbf{x} \in \mathbb{F}_p^r$ such that $\mathsf{ArionHash}(\mathbf{y}) = \mathsf{ArionHash}(\mathbf{x})$. Consequently both preimage attacks on ArionHash reduce to the same equation system

$$\mathsf{Arion}\text{-}\pi \begin{pmatrix} \mathbf{x}_{\text{in}} \\ \mathtt{IV} \end{pmatrix} = \begin{pmatrix} \alpha \\ \mathbf{x}_{\text{out}} \end{pmatrix}, \tag{20}$$

where $\alpha \in \mathbb{F}_p$ is the output of ArionHash, $\mathtt{IV} \in \mathbb{F}_p^c$ is the initial value, and $\mathbf{x}_{\text{in}} = (x_{\text{in},1}, \ldots, x_{\text{in},r'})$ and $\mathbf{x}_{\text{out}} = (x_{\text{out},2}, \ldots, x_{\text{out},n})$ are indeterminates. Analog to Arion we construct an iterated polynomial system where each round is modeled with a polynomial vector. Note that Equation (20) is not fully determined if $n \geq 3$ and $r' > 1$, in such a case we have $n - 1 + r' > n$ many variables for $\mathbf{x}_{\text{in}}$ and $\mathbf{x}_{\text{out}}$ and $(r-1) \cdot n$ many intermediate state variables, but we only have $r \cdot n$ many equations. In order to obtain a fully determined system the adversary either has to guess some values for $\mathbf{x}_{\text{in}}$ or $\mathbf{x}_{\text{out}}$, but each guess has success probability $1/p$ so we can neglect this approach, or he has to add additional equations.

If an adversary is unable to exploit additional algebraic structures of Arion-π (which are unknown to us), then he has just one generic choice: he has to add field equations until the system is fully determined. This approach introduces polynomials of very high degree, thus we do not expect this attack to be feasible below our 128 bit Gröbner basis security claim. Therefore, in the analysis of this section we always choose $c = n - 1$ to obtain a fully determined system. Note that the Macaulay bound for ArionHash is identical to the one for Arion, see Equation (16).

We implemented ArionHash in the OSCAR [44] computer algebra system and computed the Gröbner basis of $\mathcal{F}_{\mathsf{ArionHash}}$ with F4. As initial value we chose $\mathtt{IV} = \mathbf{0}^c$. Further, we computed the $\mathbb{F}_p$-vector space dimension of the quotient space for ArionHash. We used the same primes as for Arion. In Figure 3 we record our empirical results for $n = 2$ and in Figure 4 we record our empirical results for $n = 3$. From experiments we conclude that the solving degree is indeed bounded by the Macaulay bound and that the quotient space dimension grows exponential in r.

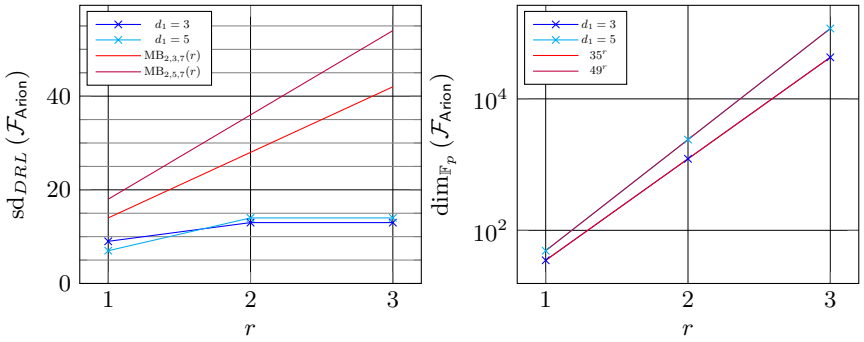

Fig. 3. Experimental solving degree and vector space dimension of the quotient ring for ArionHash with $n = 2$ and $d_2 = 7$.

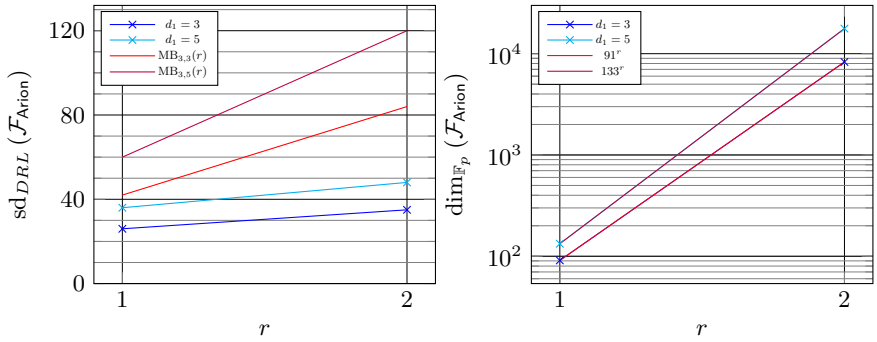

Fig. 4. Experimental solving degree and vector space dimension of the quotient ring for ArionHash with $n = 3$ and $d_2 = 7$.

To better understand the growth of the quotient space dimension we computed the quotient space dimension for $n \leq 4$ with $r = 1$, see Table 6.

Table 6. Empirical growth of the vector space dimension of the quotient space of ArionHash.

n	d_1	d_2	$\dim_{\mathbb{F}_p,\mathrm{emp}}$
2	3	7	35
2	3	257	1285
3	3	7	91
3	3	257	3341
4	3	7	203
5	3	7	427
2	5	7	49
2	5	257	1799
3	5	7	133
3	5	257	4833
4	5	7	301
5	5	7	637

From our experiments we deduce that the quotient space dimension grows or is bounded via

$$\dim_{\mathbb{F}_p}\left(\mathcal{F}_{\mathsf{ArionHash}}\right)(n,d,r) = \left(2^{n-1}\cdot d_2\cdot(d_1+1) - d_1\cdot d_2\right)^r, \qquad n \geq 1. \quad (21)$$

To estimate the cost of a Gröbner basis computation we again assume that the Macaulay bound always upper bounds the solving degree of ArionHash, and with Equation (21) we can estimate the cost of solving the polynomial system for ArionHash analog to Arion.

Table 7. Empirical cost estimation of Gröbner basis attacks on ArionHash for primes $p \geq 2^{60}$ and $d_2 \geq 121$. The column GB contains complexity of Gröbner basis computations estimated via the Macaulay bound. For the probabilistic algorithm we assume that the adversary has an optimal algorithm with $\omega = 2$.

d_1	n	r	GB (bits)	Deterministic Solving (bits)	Probabilistic Solving (bits)
3	3	5	131	158	110
3	3	6	159	190	132
5	3	4	107	133	93
5	3	6	162	200	138
3	4	4	136	141	98
3	4	6	206	212	146
5	4	4	143	147	103
5	4	5	179	185	128
3	5	4	178	154	107
3	5	5	224	193	133
5	5	3	189	161	111
5	5	5	237	201	139
3	6	4	231	167	115
3	6	5	290	208	143
5	6	3	184	130	91
5	6	5	308	217	149
3	8	3	275	143	100
3	8	4	367	191	132
5	8	3	290	148	103
5	8	4	388	198	137

An attacker can also set up a collision attack via the equation

$$\mathsf{ArionHash}(\mathbf{x}) = \mathsf{ArionHash}(\mathbf{y}), \tag{22}$$

where $\mathbf{x}, \mathbf{y} \in \mathbb{F}_p^*$ are variables. Though, any equation system for this problem is underdetermined. Even if an adversary can fix some variables to make the system fully determined he still has to deal with doubled number of equations compared to Equation (20). Thus, we do not expect that any Gröbner basis attack on the collision problem will be more performative than the preimage problem.

4 Performance Evaluation

In this section, we compare various instances of ArionHash, `Anemoi`, Griffin and Poseidon with respect to R1CS (Section 4.2). Though, first we discuss the theoretical foundation of an efficient implementation of a ArionHash circuit. In the `Anemoi` proposal it was revealed that CCZ-equivalence is a route to construct high degree permutations that can be verified with CCZ-equivalent low degree functions [18, §4.1]. In Section 4.1 we generalize CCZ-equivalence by

admitting arbitrary permutations. With the generalization we will then prove that a ArionHash circuit can be transformed into a low degree circuit. Note that in principal we perform the same trick as for Gröbner basis attacks (Section 3.3) on ArionHash to get rid of the high degree monomials x^e. We note that our transformation has also been observed and applied by the designers of Griffin, though they did not describe the theoretical foundations of their transform.

4.1 Reducing the Number of Constraints

By definition of the Arion GTDS a prover circuit will have to verify that

$$y = x^e, \tag{23}$$

though since e induces the inverse power permutation to $d_2 \in \{121, 123, 125, 129, 161, 257\}$ the naive circuit for Equation (23) will introduce many constraints. On the other hand from a prover's perspective Equation (23) is equivalent to

$$y^{d_2} = (x^e)^{d_2} = x, \tag{24}$$

for all $x \in \mathbb{F}_p$. Thus, in an implementation to reduce the number of multiplicative constraints we are well advised to implement the equivalent circuit instead of the naive circuit. We also would like to note that the same trick was applied in `Griffin` [33] to reduce the number of constraints.

In the design of `Anemoi` [18, §4] a new tool was introduced to reduce the number constraints in an `Anemoi` circuit: CCZ-equivalence [22]. The authors have found a high degree permutation, the open `Flystel`, which is CCZ-equivalent to a low degree function, the closed `Flystel`. Consequently, this can be exploited to significantly reduce the number of constraints in a prover circuit (cf. [18, Corollary 2]). From a high level perspective we can now ask ourselves whether our trick in Equation (24) can also be reformulated in terms of equivalence relations. The answer is yes and can easily be proven if we weaken CCZ-equivalence by admitting arbitrary permutations.

Definition 9. *Let $\mathbb{F}_q$ be a finite field, and let $F, G : \mathbb{F}_q^n \to \mathbb{F}_q^m$ be functions.*

(1) The graph of F is defined as

$$\Gamma_F = \Big\{ \big(\mathbf{x}, F(\mathbf{x})\big) \mid \mathbf{x} \in \mathbb{F}_q^n \Big\}.$$

(2) F and G are said to be CCZ-equivalent if there exists an affine permutation $\mathcal{A}$ of $\mathbb{F}_q^n \times \mathbb{F}_q^m$ such that

$$\Gamma_F = \mathcal{A}(\Gamma_G).$$

(3) F and G are said to be π-equivalent[5] if there exist a permutation π of $\mathbb{F}_q^n \times \mathbb{F}_q^m$ such that

$$\Gamma_F = \pi(\Gamma_G).$$

[5] Read as *pequivalent*.

Any function $\pi : \mathbb{F}_q^{2n} \to \mathbb{F}_q^{2n}$ can be decomposed into two functions $\pi_1, \pi_2 : \mathbb{F}_q^{2n} \to \mathbb{F}_q^n$ such that $\pi(\mathbf{x}, \mathbf{y}) = \big(\pi_1(\mathbf{x}, \mathbf{y}), \pi_2(\mathbf{x}, \mathbf{y})\big)$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n$. Thus, for any function $F : \mathbb{F}_q^n \to \mathbb{F}_q^n$ we have that $\pi\big(\mathbf{x}, F(\mathbf{x})\big) = \big(\pi_1(\mathbf{x}), \pi_2(\mathbf{x})\big)$, where

$$F_1(\mathbf{x}) = \pi_1\big(\mathbf{x}, F(\mathbf{x})\big), \tag{25}$$

$$F_2(\mathbf{x}) = \pi_2\big(\mathbf{x}, F(\mathbf{x})\big). \tag{26}$$

Therefore, the set $\pi(\Gamma_F) = \big\{\big(F_1(\mathbf{x}), F_2(\mathbf{x})\big) \mid \mathbf{x} \in \mathbb{F}_q^n\big\}$ is the graph of a function F' if and only if the function F_1 is a permutation. If π and π_1 are permutations, then $\pi(\Gamma_F) = \Gamma_{F'}$, where $F' = F_2 \circ F_1^{-1}$ and the functions F and F' are π-equivalent.

If one defines a new equivalence relation, then one also likes to identify equivalence classes of functions. At least for permutations this is a rather trivial task.

Lemma 10. *Let $\mathbb{F}_q$ be a finite field, and let $F, G : \mathbb{F}_q^n \to \mathbb{F}_q^n$ be permutations. Then F and G are π-equivalent.*

Proof. It suffices to prove the claim for $G = \mathrm{id}$. Obviously $\pi : (\mathbf{x}, \mathbf{y}) \mapsto \big(\mathbf{x}, F^{-1}(\mathbf{y})\big)$ defines a permutation of $\mathbb{F}_q^n \times \mathbb{F}_q^n$. Then $\pi(\Gamma_F) = \Gamma_{\mathrm{id}}$. □

Thus, when studying π-equivalence of permutations we will not learn anything new about the algebraic structure of (cryptographic) permutations. Nevertheless, we can still use it to reduce the number of constraints in a prover circuit for the Arion GTDS.

Proposition 11. *Let $\mathbb{F}_p$ be a prime field, and let $n, d_1, d_2, e \in \mathbb{Z}_{>1}$ be integers such that*

(i) d_1 is the smallest positive integer such that $\gcd(d_1, p-1) = 1$,
(ii) d_2 is an arbitrary integer such that $\gcd(d_2, p-1)$, and
(iii) $e \cdot d_2 \equiv 1 \mod p-1$.

Let $\mathcal{F} = \{f_1, \ldots, f_n\}$ be the Arion GTDS, let $g_i, h_i \in \mathbb{F}_q[x]$ be the polynomials that define $\mathcal{F}$, and let the GTDS $\mathcal{G} = \{\hat{f}_1, \ldots, \hat{f}_n\}$ be defined as

$$\hat{f}_i(x_1, \ldots, x_n) = x_i^{d_1} \cdot g_i(\tau_{i+1,n}) + h_i(\tau_{i+1,n}), \qquad 1 \le i \le n-1,$$
$$\hat{f}_n(x_1, \ldots, x_n) = x_n^{d_2},$$

where

$$\tau_{i+1,n} = x_n + x_n^e + \sum_{j=i+1}^{n-1} x_j + \hat{f}_j(x_j, \ldots, x_n).$$

Then $\mathcal{F}$ is π-equivalent to $\mathcal{G}$.

Proof. We consider the permutation

$$\pi : \mathbb{F}_q^{2n} \to \mathbb{F}_q^{2n}, \qquad \begin{pmatrix} \{x_i\}_{1 \le i \le n-1} \\ x_n \\ \{y_i\}_{1 \le i \le n-1} \\ y_n \end{pmatrix} \mapsto \begin{pmatrix} \{x_i\}_{1 \le i \le n-1} \\ y_n^{d_1} \\ \{y_i\}_{1 \le i \le n-1} \\ x_n^{d_1} \end{pmatrix}.$$

Then it is easy to verify that $\pi(\Gamma_{\mathcal{F}}) = \Gamma_{\mathcal{G}}$. □

Note that it follows from [1, Theorem 20] that the Arion GTDS $\mathcal{F}$ and its π-equivalent GTDS $\mathcal{G}$ from Proposition 11 are in the same security class with respect to differential and linear cryptanalysis.

Corollary 12. *Verifying that* $(y_1, \ldots, y_n) = \mathcal{F}(x_1, \ldots, x_n)$ *is equivalent to verifying that* $(y_1, \ldots, y_{n-1}, x_n) = \mathcal{G}(x_1, \ldots, x_{n-1}, y_n)$.

Remark 13. Unlike as for `Anemoi` the π-equivalent GTDS $\mathcal{G}$ is not a low degree function, though when implementing it as prover circuit we never have use multiplications to compute $\tilde{\sigma}_{i+1,n}$.

4.2 R1CS Performance of ArionHash

Estimating the number of multiplicative constraints in a R1CS circuit for ArionHash is straightforward.

Lemma 14. *Let* $\mathbb{F}_p$ *be a finite field, let* $r, n \geq 2$ *be integers, and let ArionHash with* r *rounds be defined over* $\mathbb{F}_p^n$. *For* $i = 1, 2$ *denote with* $d_{i,inc}$ *the minimal number of multiplications to compute the univariate power permutation* x^{d_i}. *Then a prover circuit for ArionHash needs*

$$N_{\mathsf{ArionHash}} = r \cdot \big((n-1) \cdot (d_{1,inc} + 2) + d_{2,inc} \big)$$

multiplicative constraints.

Proof. By Corollary 12 one needs $d_{2,\mathrm{inc}}$ constraints in the n^{th} branch. In each of the remaining $n-1$ branches one needs $d_{1,\mathrm{inc}}$ constraints for the power permutation, 1 constraints for the computation of g_i and h_i and 1 multiplication for the product of the power permutation and g_i. □

Analog the number of R1CS constraints for `Anemoi`, Griffin and Poseidon (cf. [18, §7], [33, §6] and [34]) are given by

$$N_{\textsc{Griffin}} = 2 \cdot r \cdot (d_{inc} + n - 2)\,, \tag{27}$$

$$N_{\texttt{Anemoi}} = \frac{r \cdot n}{2} \cdot (d_{inc} + 2)\,, \tag{28}$$

$$N_{\textsc{Poseidon}} = d_{inc} \cdot (n \cdot r_f + r_p)\,. \tag{29}$$

In Table 8 we compare the theoretical number of constraints for R1CS for various hash functions.

Table 8. R1CS constraint comparison 256 bit prime fields and 128 bits of security with $d_2 \in \{121, 123, 125, 161, 257\}$.

		Rounds				
d_1	n	ArionHash	α-ArionHash	Griffin	Anemoi	Poseidon
3	3	6	5	12		$r_f = 8,\ r_p = 84$
5	3	6	4	12		$r_f = 8,\ r_p = 56$
3	4	6	4	11	12	$r_f = 8,\ r_p = 84$
5	4	5	4	11	12	$r_f = 8,\ r_p = 56$
3	5	5	4			$r_f = 8,\ r_p = 84$
5	5	5	4			$r_f = 8,\ r_p = 56$
3	6	5	4		10	$r_f = 8,\ r_p = 84$
5	6	5	4		10	$r_f = 8,\ r_p = 84$
3	8	4	4	9	10	$r_f = 8,\ r_p = 84$
5	8	4	4	9	10	$r_f = 8,\ r_p = 56$
		R1CS Constraints				
d_1	n	ArionHash	α-ArionHash	Griffin	Anemoi	Poseidon
3	3	102	85	72		216
5	3	114	76	96		240
3	4	126	84	88	96	232
5	4	120	96	110	120	264
3	5	120	100			248
5	5	125	116			288
3	6	145	116		120	264
5	6	170	136		150	312
3	8	148	148	144	160	296
5	8	176	176	162	200	360

4.3 `libsnark` Implementation

We implemented ArionHash, Griffin and Poseidon using the `C++` library libsnark [45] that is used in the privacy-protecting digital currency Zcash.

Table 9. Performance of various hash functions for proving the membership of an Merkle tree accumulator over BN254. Proving times are in ms. For all hash functions verification requires less than 20 ms.

	Time (ms)											
Height	ArionHash			α-ArionHash			Griffin			Poseidon		
$d = 5$	$n = 3$	$n = 4$	$n = 8$	$n = 3$	$n = 4$	$n = 8$	$n = 3$	$n = 4$	$n = 8$	$n = 3$	$n = 4$	$n = 8$
4	101	103	142	73	87	143	88	99	133	186	212	274
8	211	216	294	145	177	294	181	209	270	386	417	566
16	392	401	554	278	334	553	338	387	505	745	805	1095
32	730	751	1046	509	646	1047	622	727	980	1422	1550	2111

The comparative result of our experiment is given in Table 9. All experiments were run on a system with an Intel Core i7-11800H CPU and 32GB RAM on a Clear Linux instance, using the `g++-12` compiler with `-O3 -march=native` flags. Our result shows that ArionHash significantly outperforms Poseidon showing 2x efficiency improvement, and the α-ArionHash is considerably faster than Griffin for practical parameter choices e.g. $n = 3$ or 4 in Merkle tree hashing mode.

References

1. Generalized triangular dynamical system: An algebraic system for constructing cryptographic permutations over finite fields, supplementary material
2. Albrecht, M.R., Grassi, L., Perrin, L., Ramacher, S., Rechberger, C., Rotaru, D., Roy, A., Schofnegger, M.: Feistel structures for MPC, and more. In: Sako, K., Schneider, S., Ryan, P.Y.A. (eds.) ESORICS 2019: 24th European Symposium on Research in Computer Security, Part II. Lecture Notes in Computer Science, vol. 11736, pp. 151–171. Springer, Heidelberg, Germany, Luxembourg (Sep 23–27, 2019). https://doi.org/10.1007/978-3-030-29962-0_8
3. Albrecht, M.R., Grassi, L., Rechberger, C., Roy, A., Tiessen, T.: MiMC: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology – ASIACRYPT 2016, Part I. Lecture Notes in Computer Science, vol. 10031, pp. 191–219. Springer, Heidelberg, Germany, Hanoi, Vietnam (Dec 4–8, 2016). https://doi.org/10.1007/978-3-662-53887-6_7
4. Albrecht, M.R., Rechberger, C., Schneider, T., Tiessen, T., Zohner, M.: Ciphers for MPC and FHE. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology – EUROCRYPT 2015, Part I. Lecture Notes in Computer Science, vol. 9056, pp. 430–454. Springer, Heidelberg, Germany, Sofia, Bulgaria (Apr 26–30, 2015). https://doi.org/10.1007/978-3-662-46800-5_17
5. Aly, A., Ashur, T., Ben-Sasson, E., Dhooghe, S., Szepieniec, A.: Design of symmetric-key primitives for advanced cryptographic protocols. IACR Transactions on Symmetric Cryptology **2020**(3), 1–45 (2020). https://doi.org/10.13154/tosc.v2020.i3.1-45
6. Baignères, T., Stern, J., Vaudenay, S.: Linear cryptanalysis of non binary ciphers. In: Adams, C.M., Miri, A., Wiener, M.J. (eds.) SAC 2007: 14th Annual International Workshop on Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 4876, pp. 184–211. Springer, Heidelberg, Germany, Ottawa, Canada (Aug 16–17, 2007). https://doi.org/10.1007/978-3-540-77360-3_13
7. Barbara, M., Grassi, L., Khovratovich, D., Lueftenegger, R., Rechberger, C., Schofnegger, M., Walch, R.: Reinforced concrete: Fast hash function for zero knowledge proofs and verifiable computation. Cryptology ePrint Archive, Report 2021/1038 (2021), https://eprint.iacr.org/2021/1038
8. Bardet, M., Faugère, J.C., Salvy, B.: Asymptotic behaviour of the index of regularity of semi-regular quadratic polynomial systems. In: MEGA 2005 - 8th International Symposium on Effective Methods in Algebraic Geometry. pp. 1–17. Porto Conte, Alghero, Sardinia, Italy (05 2005), https://hal.archives-ouvertes.fr/hal-01486845
9. Bardet, M., Faugère, J.C., Salvy, B., Yang, B.Y.: Asymptotic behaviour of the degree of regularity of semi-regular polynomial systems. In: MEGA 2005 - 8th

International Symposium on Effective Methods in Algebraic Geometry. Porto Conte, Alghero, Sardinia, Italy (05 2005)
10. Bariant, A., Bouvier, C., Leurent, G., Perrin, L.: Algebraic attacks against some arithmetization-oriented primitives. IACR Trans. Symmetric Cryptol. **2022**(3), 73–101 (9 2022). https://doi.org/10.46586/tosc.v2022.i3.73-101
11. Bayer, D., Stillman, M.: A criterion for detecting m-regularity. Invent. Math. **87**(1), 1–11 (2 1987). https://doi.org/10.1007/BF01389151
12. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Sponge functions. Ecrypt Hash Workshop (2007), https://keccak.team/files/SpongeFunctions.pdf
13. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: On the indifferentiability of the sponge construction. In: Smart, N.P. (ed.) Advances in Cryptology – EUROCRYPT 2008. Lecture Notes in Computer Science, vol. 4965, pp. 181–197. Springer, Heidelberg, Germany, Istanbul, Turkey (Apr 13–17, 2008). https://doi.org/10.1007/978-3-540-78967-3_11
14. Beyne, T., Canteaut, A., Dinur, I., Eichlseder, M., Leander, G., Leurent, G., Naya-Plasencia, M., Perrin, L., Sasaki, Y., Todo, Y., Wiemer, F.: Out of oddity - new cryptanalytic techniques against symmetric primitives optimized for integrity proof systems. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology – CRYPTO 2020, Part III. Lecture Notes in Computer Science, vol. 12172, pp. 299–328. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2020). https://doi.org/10.1007/978-3-030-56877-1_11
15. Biham, E., Biryukov, A., Shamir, A.: Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. In: Stern, J. (ed.) Advances in Cryptology – EUROCRYPT'99. Lecture Notes in Computer Science, vol. 1592, pp. 12–23. Springer, Heidelberg, Germany, Prague, Czech Republic (May 2–6, 1999). https://doi.org/10.1007/3-540-48910-X_2
16. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. Journal of Cryptology **4**(1), 3–72 (Jan 1991). https://doi.org/10.1007/BF00630563
17. Bogdanov, A., Rijmen, V.: Linear hulls with correlation zero and linear cryptanalysis of block ciphers. Des. Codes Cryptogr. **70**(3), 369–383 (03 2014). https://doi.org/10.1007/s10623-012-9697-z
18. Bouvier, C., Briaud, P., Chaidos, P., Perrin, L., Velichkov, V.: Anemoi: Exploiting the link between arithmetization-orientation and CCZ-equivalence. Cryptology ePrint Archive, Report 2022/840 (2022), https://eprint.iacr.org/2022/840
19. Buchberger, B.: Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal. Phd thesis, Universität Insbruck (1965)
20. Caminata, A., Gorla, E.: Solving multivariate polynomial systems and an invariant from commutative algebra. In: Bajard, J.C., Topuzoğlu, A. (eds.) Arithmetic of Finite Fields. pp. 3–36. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-68869-1_1
21. Caminata, A., Gorla, E.: Solving multivariate polynomial systems and an invariant from commutative algebra. arXiv: 1706.06319 (2021), Version: 6
22. Carlet, C., Charpin, P., Zinoviev, V.: Codes, bent functions and permutations suitable for DES-like cryptosystems. Des. Codes Cryptogr. **15**(2), 125–156 (11 1998). https://doi.org/10.1023/A:1008344232130
23. Cover, T.M., Joy, T.A.: Elements of Information Theory. John Wiley & Sons, Ltd, Hoboken, New Jersey, 2 edn. (2006). https://doi.org/10.1002/0471200611
24. Cox, D.A., Little, J., O'Shea, D.: Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra. Undergraduate

Texts in Mathematics, Springer International Publishing, 4 edn. (2015). https://doi.org/10.1007/978-3-319-16721-3
25. Daemen, J., Knudsen, L.R., Rijmen, V.: The block cipher Square. In: Biham, E. (ed.) Fast Software Encryption – FSE'97. Lecture Notes in Computer Science, vol. 1267, pp. 149–165. Springer, Heidelberg, Germany, Haifa, Israel (Jan 20–22, 1997). https://doi.org/10.1007/BFb0052343
26. Ding, J., Schmidt, D.: Solving degree and degree of regularity for polynomial systems over a finite fields. In: Fischlin, M., Katzenbeisser, S. (eds.) Number Theory and Cryptography: Papers in Honor of Johannes Buchmann on the Occasion of His 60th Birthday. pp. 34–49. Springer Berlin Heidelberg, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-42001-6_4
27. Faugère, J.C.: A new efficient algorithm for computing Gröbner bases (F4). J. Pure and Appl. Algebra **139**(1), 61–88 (1999). https://doi.org/10.1016/S0022-4049(99)00005-5
28. Faugère, J.C.: A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). In: Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation. p. 75–83. ISSAC '02, Association for Computing Machinery (2002). https://doi.org/10.1145/780506.780516
29. Faugère, J.C., Gaudry, P., Huot, L., Renault, G.: Sub-cubic change of ordering for Gröbner basis: A probabilistic approach. In: Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation. p. 170–177. ISSAC '14, Association for Computing Machinery, New York, NY, USA (2014). https://doi.org/10.1145/2608628.2608669
30. Faugère, J.C., Gianni, P., Lazard, D., Mora, T.: Efficient computation of zero-dimensional Gröbner bases by change of ordering. J. Symb. Comput. **16**(4), 329–344 (1993). https://doi.org/10.1006/jsco.1993.1051
31. Faugère, J.C., Mou, C.: Sparse FGLM algorithms. J. Symb. Comput. **80**, 538–569 (2017). https://doi.org/https://doi.org/10.1016/j.jsc.2016.07.025
32. Grassi, L.: Mixture differential cryptanalysis: a new approach to distinguishers and attacks on round-reduced AES. IACR Transactions on Symmetric Cryptology **2018**(2), 133–160 (2018). https://doi.org/10.13154/tosc.v2018.i2.133-160
33. Grassi, L., Hao, Y., Rechberger, C., Schofnegger, M., Walch, R., Wang, Q.: A new feistel approach meets fluid-SPN: Griffin for zero-knowledge applications. Cryptology ePrint Archive, Report 2022/403 (2022), https://eprint.iacr.org/2022/403
34. Grassi, L., Khovratovich, D., Rechberger, C., Roy, A., Schofnegger, M.: Poseidon: A new hash function for zero-knowledge proof systems. In: Bailey, M., Greenstadt, R. (eds.) USENIX Security 2021: 30th USENIX Security Symposium. pp. 519–535. USENIX Association (Aug 11–13, 2021)
35. Grassi, L., Lüftenegger, R., Rechberger, C., Rotaru, D., Schofnegger, M.: On a generalization of substitution-permutation networks: The HADES design strategy. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology – EUROCRYPT 2020, Part II. Lecture Notes in Computer Science, vol. 12106, pp. 674–704. Springer, Heidelberg, Germany, Zagreb, Croatia (May 10–14, 2020). https://doi.org/10.1007/978-3-030-45724-2_23
36. Grassi, L., Rechberger, C., Rønjom, S.: A new structural-differential property of 5-round AES. In: Coron, J.S., Nielsen, J.B. (eds.) Advances in Cryptology – EUROCRYPT 2017, Part II. Lecture Notes in Computer Science, vol. 10211, pp. 289–317. Springer, Heidelberg, Germany, Paris, France (Apr 30 – May 4, 2017). https://doi.org/10.1007/978-3-319-56614-6_10

37. Jakobsen, T., Knudsen, L.R.: The interpolation attack on block ciphers. In: Biham, E. (ed.) Fast Software Encryption – FSE'97. Lecture Notes in Computer Science, vol. 1267, pp. 28–40. Springer, Heidelberg, Germany, Haifa, Israel (Jan 20–22, 1997). https://doi.org/10.1007/BFb0052332
38. Knudsen, L.R.: Truncated and higher order differentials. In: Preneel, B. (ed.) Fast Software Encryption – FSE'94. Lecture Notes in Computer Science, vol. 1008, pp. 196–211. Springer, Heidelberg, Germany, Leuven, Belgium (Dec 14–16, 1995). https://doi.org/10.1007/3-540-60590-8_16
39. Lai, X.: Higher order derivatives and differential cryptanalysis. In: Blahut, R.E., Costello, D.J., Maurer, U., Mittelholzer, T. (eds.) Communications and Cryptography: Two Sides of One Tapestry. pp. 227–233. Springer US, Boston, MA (1994). https://doi.org/10.1007/978-1-4615-2694-0_23
40. Lamberger, M., Mendel, F., Rechberger, C., Rijmen, V., Schläffer, M.: Rebound distinguishers: Results on the full Whirlpool compression function. In: Matsui, M. (ed.) Advances in Cryptology – ASIACRYPT 2009. Lecture Notes in Computer Science, vol. 5912, pp. 126–143. Springer, Heidelberg, Germany, Tokyo, Japan (Dec 6–10, 2009). https://doi.org/10.1007/978-3-642-10366-7_8
41. Lazard, D.: Gröbner bases, Gaussian elimination and resolution of systems of algebraic equations. In: van Hulzen, J.A. (ed.) Computer Algebra. Lecture Notes in Computer Science, vol. 162, pp. 146–156. Springer Berlin Heidelberg (1983). https://doi.org/10.1007/3-540-12868-9_99
42. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth, T. (ed.) Advances in Cryptology – EUROCRYPT'93. Lecture Notes in Computer Science, vol. 765, pp. 386–397. Springer, Heidelberg, Germany, Lofthus, Norway (May 23–27, 1994). https://doi.org/10.1007/3-540-48285-7_33
43. Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: The rebound attack: Cryptanalysis of reduced Whirlpool and Grøstl. In: Dunkelman, O. (ed.) Fast Software Encryption – FSE 2009. Lecture Notes in Computer Science, vol. 5665, pp. 260–276. Springer, Heidelberg, Germany, Leuven, Belgium (Feb 22–25, 2009). https://doi.org/10.1007/978-3-642-03317-9_16
44. Oscar – open source computer algebra research system, version 0.10.2 (2022), https://oscar.computeralgebra.de
45. SCIPR Lab: libsnark: a c++ library for zkSNARK proofs (2017), https://github.com/scipr-lab/libsnark
46. The Sage Developers: SageMath, the Sage Mathematics Software System (Version 9.3) (2022), https://www.sagemath.org
47. Wagner, D.: The boomerang attack. In: Knudsen, L.R. (ed.) Fast Software Encryption – FSE'99. Lecture Notes in Computer Science, vol. 1636, pp. 156–170. Springer, Heidelberg, Germany, Rome, Italy (Mar 24–26, 1999). https://doi.org/10.1007/3-540-48519-8_12