A New Feistel Approach Meets *Fluid*-SPN: Griffin for Zero-Knowledge Applications

Lorenzo Grassi[1], Yonglin Hao[2], Christian Rechberger[3], Markus Schofnegger[3],
Roman Walch[3,4], and Qingju Wang[5]

[1] Radboud University, Nijmegen (The Netherlands)
[2] State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878 (China)
[3] Graz University of Technology (Austria)
[4] Know-Center GmbH (Austria)
[5] SnT, University of Luxembourg, Esch-sur-Alzette (Luxembourg)
`lgrassi@science.ru.nl`
`haoyonglin@yeah.net`
`firstname.lastname@iaik.tugraz.at`
`qingju.wang@uni.lu`

Abstract. Zero-knowledge (ZK) applications form a large group of use cases in modern cryptography, and recently gained in popularity due to novel proof systems. For many of these applications, cryptographic hash functions are used as the main building blocks, and they often dominate the overall performance and cost of these approaches. Therefore, in the last years several new hash functions were built in order to reduce the cost in these scenarios, including Poseidon and *Rescue* among others.

These hash functions often look very different from more classical designs such as AES or SHA-2. For example, they work natively with integer objects rather than bits. At the same time, for example Poseidon and *Rescue* share some common features, such as being SPN schemes and instantiating the nonlinear layer with invertible power maps. While this allows the designers to provide simple and strong arguments for establishing their security, it also introduces some crucial limitations in the design, which affects the performance in the target applications.

To overcome these limitations, we propose the `Horst` mode of operation, in which the addition in a Feistel scheme $(x, y) \mapsto (y+F(x), x)$ is replaced by a multiplication, i.e., $(x, y) \mapsto (y \times G(x), x)$.

By carefully analyzing the relevant performance metrics in SNARK and STARK protocols, we show how to combine an expanding `Horst` scheme and the strong points of existing schemes in order to provide security and better efficiency in the target applications. We provide an extensive security analysis for our new design Griffin and a comparison with all current competitors.

Keywords: Sponge Hash Function – Griffin – Zero-Knowledge Proof Systems – `Horst` – *Fluid*-SPN

1 Introduction

In the recent years, various cryptographic constructions were designed which are tailored towards specific use cases such as multi-party computation (MPC), homomorphic encryption (HE), signature schemes, and zero-knowledge (ZK) proof systems. All these applications favor cryptographic schemes with specific algebraic properties, for example a small number of multiplications. Considering $\mathbb{F}_p^t$ for a prime $p \geq 3$ and $t \geq 1$, existing designs include MiMC [3], GMiMC [2], HadesMiMC [38], Poseidon [36], *Rescue* [4, 67], Ciminion [29], Pasta [30], *Grendel* [66], and Neptune [39].

Naturally, the performance metrics (i.e., what determines the *cost* in a given setting) vary between the different use cases. While the cost in e.g. MPC is well-studied [42, 3, 38], ZK protocols often have more sophisticated optimization targets. In such a protocol, one party (the prover) proves to another party (the verifier) that they know a certain value x, without conveying any information apart from the fact that they know x. The two major classes of ZK proof systems are *zero-knowledge succinct non-interactive arguments of knowledge* (zk-SNARKs) and *zero-knowledge scalable transparent arguments of knowledge* (zk-STARKs), which are also the ones we focus on in this paper.

Recent hash functions proposed for these protocols include *Rescue*, Poseidon, *Grendel* and Neptune. Even though their algorithmic descriptions are quite different, their internal permutations are all SPN constructions. While this approach can have advantages for arguing security, it can also have various limitations affecting the performance in ZK protocols. The result of our analysis is the `Horst` construction, which allows to overcome these limits and to improve the performance significantly when combined with the *Rescue* approach.

1.1 Sponge Hash Functions for zk-SNARKs and zk-STARKs

Cost Metrics in ZK Protocols. In order to design a hash function for ZK settings, it is crucial to understand the cost metrics related to zk-SNARK and zk-STARK protocols. In these applications, the prover uses ZK proofs to convince a verifier that they know a preimage x of a given hash $y = \mathcal{H}(x)$, without revealing anything about x. The efficiency of these protocols depends on the details of the used hash function $\mathcal{H}$. Indeed,

- in zk-SNARKs, the cost of the proof is proportional to the number of non-linear operations one has to perform, and in some cases (e.g., Plonk [34]) the number of linear operations must also be considered, while
- in zk-STARKs, the cost is related to the degree and the depth of the circuit that must be verified.

In both cases, it is not required to re-evaluate $\mathcal{H}$ in order to determine if $y = \mathcal{H}(x)$. Indeed, one can verify any equivalent cheaper representation $\mathcal{F}(x, y) = 0$ which is satisfied if and only if $y = \mathcal{H}(x)$. As an example, consider the simple function $F(x) = x^{1/d}$ for a certain $d \geq 3$ such that $F : \mathbb{F}_p \to \mathbb{F}_p$ is invertible.

Instead of verifying $y = x^{1/d}$, the equivalent representation $y^d = x$ is cheaper to verify if $d \ll 1/d$. More details are given in Section 2 and App. B.1 and C.1.

Most designs in the past focused only on a subset of cost metrics. For example, the idea of MiMC, HadesMiMC, and Poseidon was to minimize the number of multiplications. As a result, they can be efficient in SNARKs, but have a high number of rounds due to the use of low-degree functions. Hence, the performance in STARKs, but also in Plonk due to having more linear layers, is not competitive. In contrast, *Rescue* has an inner structure tailored for STARKs, which results in comparably low round numbers and decent Plonk performance. However, the efficiency in other SNARKs, as well as the plain performance, suffers from this structure.

Strong-Arranged SPN Schemes and Power Maps. Currently, the most competitive hash functions for ZK protocols include *Rescue* and Poseidon. Both use an internal SPN permutation whose round function $\mathcal{R} : \mathbb{F}_p^t \to \mathbb{F}_p^t$ is

$$\mathcal{R}(\cdot) = c + M \times S(\cdot), \tag{1}$$

where c is a round constant, $M \in \mathbb{F}_p^{t\times t}$ is an MDS matrix, and $S : \mathbb{F}_p^t \to \mathbb{F}_p^t$ is an S-box layer whose details depend on the scheme. However, both are *strong-arranged* SPNs (based on [21, Definition 4]), i.e., the S-box layer is of the form

$$S(x_0, x_1, \ldots, x_{t-1}) = S_0(x_0) \mid\mid S_1(x_1) \mid\mid \cdots \mid\mid S_{t-1}(x_{t-1}), \tag{2}$$

where every $S_i : \mathbb{F}_p \to \mathbb{F}_p$ is bijective and instantiated with a power map, and the linear layer is defined via a multiplication with a matrix in $\mathbb{F}_p^{t\times t}$. For example, every round of *Rescue* consists of two subrounds, one in which all S_i correspond to $x \mapsto x^{1/d}$ and one in which all S_i correspond to $x \mapsto x^d$. Further, Poseidon uses two different rounds, a full one in which all S_i correspond to $x \mapsto x^d$ and a partial one in which $S_0(x) = x^d$ and $S_{i\neq 0}(x) = x$ (identity).

1.2 Our Contribution

Weaknesses of Strong-Arranged SPN Schemes. A strong-arranged SPN scheme allows for simple and strong security arguments regarding statistical attacks, including the (classical) differential [15] and linear [53] attacks. For example, the combination of a linear layer with a high branch number (e.g., an MDS matrix) and an S-box layer with a good maximum differential probability (e.g., certain classes of low-degree S-boxes) allows the designer to efficiently use the wide-trail design strategy [25] for this purpose. As a result, the main attack vectors against this kind of schemes (but more general, for arithmetization-oriented designs over larger fields) are algebraic ones [1, 14, 32, 47].

However, designing a strong-arranged SPN scheme over $\mathbb{F}_p^t$ has some limitations. First, for a permutation each S-box S_i in Eq. (2) must be invertible. Since no quadratic map is invertible over $\mathbb{F}_p$, one is forced to consider functions of degree at least 3. Secondly, no diffusion takes place among the several elements in the nonlinear layer, and achieving diffusion in both the linear and nonlinear layer can help against both statistical and algebraic attacks. Based on this, in Section 3.2 we conclude that a non-strong-arranged SPN scheme is a more efficient solution in our target applications.

`Horst` Schemes. One way of designing a non-strong-arranged SPN scheme is by considering a weak-arranged SPN scheme, as in the case of Neptune. The hash function Neptune is inspired by Poseidon, but the power maps in the external rounds are replaced by the concatenation of independent S-boxes over $\mathbb{F}_{p^2} \equiv \mathbb{F}_p^2$ based on the quadratic Lai–Massey function. As shown in [39, Section 7], such a scheme classifies as a weak-arranged SPN since the linear layer in the external rounds does not admit an equivalent representation in $\mathbb{F}_{p^2}^{t' \times t'}$, where $t = t' \cdot 2$. Other examples include Phoenix, Hermit, and Sneaky from the Marvellous family [6], whose nonlinear layers are defined as the concatenation of S-boxes over $\mathbb{F}_{p^2}$ constructed via Feistel functions.

Another possible approach is to consider non-SPN schemes as the Feistel ones, for example GMiMC. Given a function F over a generic field $\mathbb{F}$, a Feistel scheme is defined as the map $(x, y) \mapsto (y + F(x), x)$ over $\mathbb{F}_p^2$. Several generalizations over $\mathbb{F}_p^t$ are proposed in the literature [70, 60, 44]. No condition is imposed on F, thus it may be instantiated via the non-invertible quadratic power map $x \mapsto x^2$. The relation between $F(x)$ and y is linear, exactly as in the case of an aligned SPN scheme. Inspired by Amaryllises [40], in Section 3.2 we propose a modified Feistel scheme in which the linear relation between y and $F(x)$ is replaced by their product. To guarantee invertibility, we require that $F(x) \neq 0$ for each input x. We show how to construct such a low-degree (non-trivial) function and call this variant of the Feistel scheme the `Horst` approach.[6]

Griffin. In Section 4 we specify a new family of sponge hash functions called Griffin, instantiated with the internal permutation Griffin-π.[7] It differs from previous designs in both its nonlinear and linear layer. In particular, Griffin-π cannot be rewritten as in Eqs. (1) and (2) since its nonlinear layer is not defined as the concatenation of independent nonlinear S-boxes. Instead, it is the composition of two nonlinear sublayers defined via three different nonlinear functions. One is defined via the invertible power maps $x \mapsto x^d$ and $x \mapsto x^{1/d}$, which is inspired by *Rescue* and allows us to reach the maximum degree quickly while still being efficient in ZK applications. The other one is defined by our proposed `Horst` strategy, i.e., it uses the map $(x, y) \mapsto (x, y \cdot G(x))$, where G is a quadratic function so that $G(z) \neq 0$ for each input z. To better understand the

[6] The name `Horst` is due to the cryptographer Horst Feistel and has been chosen in order to emphasize the link between $(x, y) \mapsto (y + F(x), x)$ and $(x, y) \mapsto (y \cdot G(x), x)$.

[7] The griffin is a legendary creature with the body, tail, and back legs of a lion, and the head and wings of an eagle. For this reason, the name Griffin has been chosen since our design merges ideas of a *Fluid*-SPN and an unaligned construction as the `Horst` one into a single round function.

relation between Horst and the classical Feistel scheme, in Sections 3.5 and 5 we show that Griffin instantiated with Horst requires fewer rounds for security with respect to Griffin instantiated with the classical Feistel scheme, achieving better performance in terms of diffusion and multiplicative complexity.

Since the cost metrics in zk-SNARKs and zk-STARKs are mainly related to the number of nonlinear operations, instantiating the linear layer with an MDS matrix may be the simplest choice. However, the multiplication with such a matrix over $\mathbb{F}_p^t$ requires $\mathcal{O}(t^2)$ multiplications with constants, which could heavily affect the plain implementation efficiency for large t. Hence, we propose a matrix that can be implemented with a small number of additions. It is inspired by the linear layer of AES, i.e., it can be decomposed as the multiplication of two matrices. However, while in AES one of these two matrices (the one corresponding to the ShiftRows operation) only changes the position of the elements, both the matrices in the linear layer of Griffin-π provide full diffusion. This allows to achieve this property in each state word after a single round.

Security Analysis. A detailed security analysis of the proposed hash function is given in Section 5. From the algebraic perspective, Gröbner basis [19, 23] attacks at the round level are the most efficient attacks. We present several strategies that take into account the details of the proposed design. From the statistical perspective, well-known techniques like the wide-trail design strategy do not apply since our design is unaligned. For this reason, we propose an analysis which makes use of dedicated tools that help us to provide upper bounds on the success probability of statistical attacks.

Efficiency in Plain, zk-SNARKs, and zk-STARKs. Following the brief introduction to the cost metrics in Section 1.1, with Griffin we aim to find a beneficial tradeoff between all of them with a single design. We evaluate the resulting performance of Griffin when used in SNARKs using the R1CS arithmetization and compare it to various other constructions in Section 6. Our evaluation shows that Griffin is significantly better suited for these zk-SNARKs than any previously proposed design. In the case of zk-STARKs and Plonk (a SNARK with different arithmetization), Griffin provides similar performance to the currently best hash functions for STARKs and the best performance in Plonk. Due to page limit, we show the comparison between Griffin and its competitors in zk-STARKs and Plonk in App. B and App. C, respectively. As a result, and as was our goal, the unaligned round functions of Griffin provide an efficient tradeoff between the plain performance and the performance across different ZK proof systems.

2 Cost Metrics for Zero-Knowledge Proof Systems

In this section, we discuss different arithmetization techniques for various ZK proof systems and their respective cost metrics in order to later design an efficient hash function. For this, we directly focus on iterative functions to give the reader an intuition on how to translate hash functions for the different proof systems.

2.1 Zero-Knowledge Proofs

A zero-knowledge (ZK) proof system is a two-player protocol between a prover and a verifier, allowing the prover to convince the verifier that they know a witness w to a statement x without revealing anything about the witness beyond what can be implied by x. For example, the prover can use ZK proofs to convince a verifier that they know a preimage w of a given hash $y = \mathcal{H}(w)$ without revealing anything about w. The proof system needs to be complete and sound with a negligible soundness error ϵ, and must fulfil the zero-knowledge property, which informally states that the proof is independent of the witness w.

The two major classes of zero-knowledge proof systems are zk-SNARKs and zk-STARKs, with the main difference being that zk-SNARKs, contrary to zk-STARKs, require a trusted setup and are not post-quantum secure. In the recent years, many use cases involving ZK proofs have emerged, with two use cases mainly relying on hash functions: *set membership proofs* based on Merkle tree accumulators and *verifiable computation* based on recursive proofs. In both use cases one has to prove the knowledge of preimages of (chains of) hash functions, which is why the overall performance mainly depends on the efficiency of the used hash function in zk-SNARKs and zk-STARKs.

2.2 Arithmetization

To prove a solution of a computational problem in zero knowledge, one has to translate the problem into an algebraic representation. This step is known as arithmetization and the representation differs between the various proof systems. The performance (i.e., proof size, proving/verification time) then depends on this representation. Many proof systems and algebraic representations have been proposed in the literature, with rank-1 constraint satisfaction systems (R1CS) and Plonk gates being the most widely used representations in zk-SNARKs, and the algebraic intermediate representation (AIR) being used in zk-STARKs [10].

Concretely, in applications involving preimage proofs of hash functions, the algebraic representation describes the connection between the preimage and the final hash. The witness of the ZK proof then captures all intermediate values (including the preimage) required to satisfy this representation for a given instance of the problem (i.e., a specific hash value).

Let $\mathbb{F}$ be a field, let $t \geq 1$, and let $\mathcal{H} : \mathbb{F}^t \to \mathbb{F}^t$. Let us focus on the case where the statement $\mathcal{H}$ to prove is an iterative function, i.e.,

$$\mathcal{H}(a) = \mathcal{F}_{r-1} \circ \mathcal{F}_{r-2} \circ \cdots \circ \mathcal{F}_1 \circ \mathcal{F}_0(a),$$

where $\mathcal{F}_0, \mathcal{F}_1, \ldots, \mathcal{F}_{r-1} : \mathbb{F}^t \to \mathbb{F}^t$ are functions. Given $a, b \in \mathbb{F}^t$, the goal of a zero-knowledge application is to prove that $\mathcal{H}(a) = b$ without revealing a. In order to efficiently determine whether $\mathcal{H}(a) = b$, the prover can use the intermediate values $x_0 \equiv a, x_1, x_2, \ldots, x_{r-1} \equiv b$ s.t. $\mathcal{F}_i(x_i) - x_{i+1} = 0$ for $i \in \{0, 1, \ldots, r-1\}$. They can prove any *equivalent system of equations*, that is, they can introduce functions $\mathcal{G}_0, \mathcal{G}_1, \ldots, \mathcal{G}_{s-1} : (\mathbb{F}^t)^r \to \mathbb{F}^t$ s.t. the previous system of equations is satisfied if and only if $\mathcal{G}_i(x_0, x_1, \ldots, x_{r-1}) = 0$ for $i \in \{0, 1, \ldots, s-1\}$. Note that s does not need to be equal to r. The choice of the equivalent functions $\mathcal{G}_0, \mathcal{G}_1, \ldots, \mathcal{G}_{s-1}$ clearly depends on the cost metric of the given proof system. For the following, we use the term *Fluid-Scheme* for schemes that allow an equivalent representation which can be more efficiently proven or verified.[8] It is based on the notion of *non-procedural computation* first given in [4], which describes the idea of not only evaluating schemes in the plain direction, but using intermediate relations instead.

For example, given an invertible function $\mathcal{F} : \mathbb{F} \to \mathbb{F}$ defined as $\mathcal{F}(x) = x^{1/d}$, one can choose $\mathcal{G} : \mathbb{F}^2 \to \mathbb{F}$ as $\mathcal{G}(x, y) = x - y^d$. In this case, the equation $\mathcal{F}(x) - y = x^{1/d} - y = 0$ is proven via $\mathcal{G}(x, y) = x - y^d = 0$. Similarly, given an invertible function $\mathcal{F} : \mathbb{F} \setminus \{0\} \to \mathbb{F} \setminus \{0\}$ defined as $\mathcal{F}(x) = 1/x$, one can choose $\mathcal{G} : (\mathbb{F} \setminus \{0\})^2 \to \mathbb{F}$ as $\mathcal{G}(x, y) = x \cdot y - 1$.

Consider two functions $\mathcal{F}_0, \mathcal{F}_1 : \mathbb{F} \to \mathbb{F}$ defined as $\mathcal{F}_0(x) = \gamma + x^d$ and $\mathcal{F}_1(x) = x^{1/d}$, where $\gamma \neq 0$ and where the function $x \mapsto x^d$ is assumed to be invertible. The function $\mathcal{H} = \mathcal{F}_1 \circ \mathcal{F}_0$ can be efficiently proven via two functions $\mathcal{G}_0, \mathcal{G}_1 : \mathbb{F}^2 \to \mathbb{F}$ defined as $\mathcal{G}_0(x_0, x_1) = x_1 - \gamma - x_0^d$ and $\mathcal{G}(x_1, x_2) = x_2^d - x_1$. At the same time, it can also be proven via a single function $\mathcal{G} : \mathbb{F}^3 \to \mathbb{F}$ defined as $\mathcal{G}(x_0, x_1, x_2) = x_2^d - (\gamma + x_0^d)$. This corresponds to e.g. the arithmetization of *Rescue* in zk-STARKs. Both representations are valid and require the same number of multiplications, but they have different degrees when chained together. In this sense, *Rescue* is an example of a *Fluid*-SPN. Depending on the application, one of the two can minimize the cost in the different proof systems (recall that an infinite number of representations are in theory valid).

In the next section we analyze the cost metrics for R1CS-based SNARKs. For a similar analysis for AIR-based STARKs we refer to App. B.1 and for an analysis in the Plonk [34] proof system, we refer to App. C. We also discuss the relations between these three cost metrics in App. B.2.

2.3 Rank-1 Constraint Satisfaction Systems (R1CS)

Many different proof systems (e.g., Groth16 [43], Ligero [5], Aurora [11], Bulletproofs [20]) require to translate the computational problem into an R1CS system first, with Groth16 being the fastest proof system with the smallest proofs to date. A R1CS system is a set of q equations (i.e., q constraints) on the variables $a_0, \ldots, a_m \in \mathbb{F}$ (with $a_0 = 1$) of the form

$$\left(\sum_i u_{i,q} \cdot a_i\right) \cdot \left(\sum_i v_{i,q} \cdot a_i\right) = \left(\sum_i w_{i,q} \cdot a_i\right),$$

[8] A fluid is a material that continuously deforms (flows) under an applied external force. In our case, the scheme adapts its algebraic representation to the target ZK protocol.

where $u_{i,q}, v_{i,q}, w_{i,q} \in \mathbb{F}$ are constants describing the q-th constraint. These constants are derived from the hash functions when proving the knowledge of preimages of hashes and are independent of the given hash value. An assignment to the variables $a_0, \ldots, a_m$ is then the witness of the ZK proof and captures all intermediate values (including the preimage) when computing a given hash value. The role of the zk-SNARK is to prove that the witness satisfies the R1CS system without revealing the witness itself. The efficiency then depends on the number of constraints q in the constraint systems, i.e., the prover complexity is in $\mathcal{O}(q)$.

R1CS constraints require that every statement needs to be translated into multiplications of linear combinations of the witness variables. Consequently, linear operations can be embedded into subsequent constraints and do not require constraints on their own. For nonlinear operations, the designer has to find a representation which fully captures the relation between the input and the output of the operation, while minimizing the number of degree-2 equations.

Cost Metric. We measure the number of R1CS constraints, i.e., the minimum number of multiplications of linear combinations of witness variables required to fully represent any (equivalent) relation between the preimage and the hash.

3 The Birth of Griffin

3.1 SPN Schemes: Strong and Weak Points

In order to minimize our cost metrics discussed in the previous section, ideally the iterated round function should have a low-degree representation with a small number of multiplications (and additions), be efficiently computable in plain, and should lead to a hash function with a small number of rounds.

Weaknesses of Strong-Arranged and/or Aligned SPN Schemes. Let $q = 2$ or $q = p \geq 3$ be a prime. Let $t = n \cdot t'$ where $1 < n < t$ and $1 < t' < t$. As we have already mentioned in the introduction, many of the ZK-friendly schemes are SPN schemes, and hence their nonlinear layer is defined as

$$S(x_0, \ldots, x_{t-1}) = S_0(x_0, \ldots, x_{n-1}) \,||\, S_1(x_n, \ldots, x_{2n-1}) \,||\, \cdots \,||\, S_{t'-1}(x_{t-n}, \ldots, x_{t-1}),$$

where $||$ denotes concatenation and $S_0, S_1, \ldots, S_{t'-1} : \mathbb{F}_{q^n} \to \mathbb{F}_{q^n}$ are invertible functions. Following [21, 17], SPNs can be divided into sub-categories including

(1) strong-arranged SPNs if the linear layer $M \in \mathbb{F}_q^{t \times t}$ has degree one when written over $\mathbb{F}_{q^n}^{t' \times t'}$ and weak-arranged SPNs otherwise[9], and

(2) aligned SPNs if two (or more) consecutive rounds admit a Super-Sbox structure [27] and unaligned SPNs otherwise.[10]

[9] Every matrix in $\mathbb{F}_{q^n}^{t' \times t'}$ admits an equivalent representation over $\mathbb{F}_q^{t \times t}$, while the reverse is not true.

[10] We point out that a scheme must be an SPN in order to be aligned. See [17, Section 3]: "The nonlinear layer N consists of the parallel application of n S-boxes of size m to disjoint parts of the state, indexed by B_i".

We refer to [21, 17] for the formal definitions, pointing out that these two concepts are not equivalent.

Since many designs such as SHARK [65], AES [26], and more generally AES-like schemes are aligned and/or strong-arranged SPN schemes, several techniques such as the *wide-trail design strategy* [25] have been developed in order to study their security. For example, using the branch number of the linear layer and the maximum differential probability of the S-boxes, we can provide a simple and strong security argument against classical differential (and linear) attacks, whereas in the case of unaligned schemes dedicated tools are required.

However, since both the nonlinear and the linear layer are defined over the same field $\mathbb{F}_{q^n}^{t'}$, statistical attacks that exploit this "arrangement" may be powerful. For example, a truncated differential [48] can be set up by exploiting the fact that every element of $\mathbb{F}_{q^n}$ is either active or passive. The same property is crucial for attacks and distinguishers like the multiple-of-n [41] and the mixture differential one [35]. In contrast, these strategies do not work in the case of an unaligned scheme over $\mathbb{F}_{q^n}$, as also pointed out recently in e.g. [17]. Similar advantages also hold when considering algebraic attacks. Indeed, since in an unaligned scheme the diffusion takes place both in the linear and the nonlinear layer, the algebraic equations that describe the scheme can be more dense.

Limits of SPN: Choice of the Nonlinear Layer. The invertibility of an SPN scheme follows from the invertibility of the linear layer M and from the invertibility of the nonlinear layer $S : \mathbb{F}_{p^n} \to \mathbb{F}_{p^n}$. For $n = 1$, a common way to instantiate it is to use power maps $x \mapsto x^d$, which are invertible if and only if $\gcd(d, p-1) = 1$. Since the square function is not a permutation over $\mathbb{F}_p$, one has to use a function of degree $d \geq 3$ to ensure invertibility.

These functions can also be used for $n \geq 2$. Unfortunately, the algebraic representation over $\mathbb{F}_p^n$ becomes more complicated, and the number of multiplications in $\mathbb{F}_p$ increases exponentially. [11] Other invertible degree-2 functions over $\mathbb{F}_p^n$ constructed via a local map have recently been presented in [39], and they include, for example, the Lai–Massey one [49] and its generalization. However, many of these functions either are invertible only for some particular values of p or they admit invariant subspaces [68, 51], i.e., a subspace $\mathfrak{X}$ that is invariant through the function. These facts may put some restrictions on the values of p, or potentially affect the performance due to an increased number of rounds.

3.2 Non-SPN Schemes: From Feistel to Horst

To overcome these limitations, we can consider non-SPN schemes. Well-known examples are the Feistel ones, such as GMiMC (broken in [14]), or Ciminion.

[11] Given $d = \sum_{i=0}^{\lfloor \log_2(d) \rfloor} d_i \cdot 2^i$ for $d_i \in \{0, 1\}$, evaluating $x \mapsto x^d$ can require computing x^{2^j} for each $j \in \{0, 1, \ldots, \lfloor \log_2(d) \rfloor\}$ for a cost of $\lfloor \log_2(d) \rfloor$ multiplications, plus other $\text{hw}(d) - 1$ multiplications to get $x \mapsto x^d$ (where $\text{hw}(\cdot)$ is the Hamming weight, defined in Definition 4).

Another example is `Amaryllises` [40], defined over $\mathbb{F}_q^t$ (where $q = p^s$) as

$$\forall j \in \{0, 1, \ldots, t-1\}: \quad y_j = x_j \cdot G\left(\sum_{i=0}^{t-1} \alpha_i \cdot x_i\right) + F\left(\sum_{i=0}^{t-1} \alpha_i \cdot x_i\right),$$

where $F(0) \neq 0$, $x \mapsto x \cdot G(x)$ is invertible over $\mathbb{F}_q$, and $\sum_{i=0}^{t-1} \alpha_i = 0$ if $F \neq 0$.

Our idea is to apply the `Amaryllises` approach for Lai–Massey schemes to the case of Feistel schemes. Given a function $F : \mathbb{F}_q \to \mathbb{F}_q$, the nonlinear layer of a Feistel scheme over $\mathbb{F}_q^2$ is defined as $(x, y) \mapsto (x, y + F(x))$, which is invertible independently of F. Instead of considering a linear relation between y and $F(x)$, here we combine y and $F(x)$ in a nonlinear way without losing the advantageous properties of Feistel schemes. The simplest way is to replace the sum with a multiplication, but then the invertibility cannot be guaranteed anymore. We solve this with a stronger assumption on F.

Definition 1 (`Horst` Scheme). *Let $G : \mathbb{F}_q \to \mathbb{F}_q \setminus \{0\}$, and let $F : \mathbb{F}_q \to \mathbb{F}_q$. We define the `Horst` scheme over $\mathbb{F}_q^2$ as in Fig. 1a, that is, as*

$$(y, x) \mapsto (x, y \times G(x) + F(x)).$$

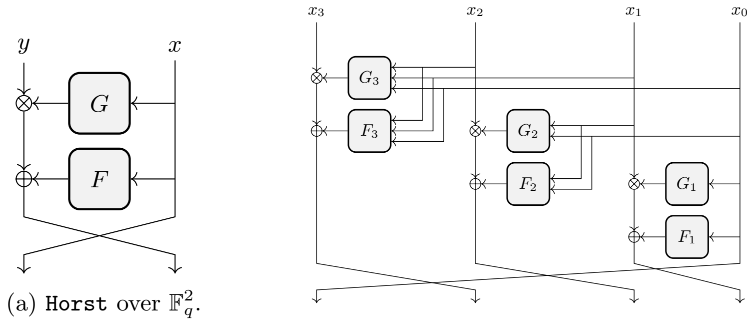
We point out that if $G(x) = 1$ for each $x \in \mathbb{F}_q$, then `Horst` reduces to a (classical) Feistel scheme. Moreover, we sometimes use the notation $\texttt{Horst}^{\times}$ to denote a `Horst` scheme in which $F = 0$ (i.e., $(y, x) \mapsto (x, y \times G(x))$).[12] Since $\mathbb{F}_q$ is a field and $G(x) \neq 0$ for each $x \in \mathbb{F}$, it follows that `Horst` defined as in Definition 1 is invertible.

S-Box in Streebog and Kuznyechik. In the case $F(x) = 0$ for each $x \in \mathbb{F}_q$, we note that a Feistel scheme based on a nonlinear relation between the branches was allegedly also used in order to set up the 8-bit S-boxes of Streebog [31] and Kuznyechik [33], two Russian standards of a hash function and a block cipher, respectively. This was discovered in [16], where the authors reconstructed the design of the S-box from its lookup table definition. The nonlinear diffusion in this case consists of multiplications in $\mathbb{F}_{2^4}$ between the two branches.

However, to the best of our knowledge, no generalization from $\mathbb{F}_{2^4}$ to larger binary extension fields or larger prime fields is publicly available. Indeed, given $(x, y) \mapsto x \cdot G(y)$, while a brute-force approach may be sufficient in order to guarantee invertibility (i.e., $G(y) \neq 0$ for each y) and efficiency in terms of linear or nonlinear operations for small fields, this does not seem feasible when considering larger fields. We solve this problem in the following, by showing how to construct G in an efficient way in the `Horst` approach given above.

Initial Security Considerations. The security of Feistel schemes [52] has been heavily analyzed both from the indistinguishable point of view [61, 62, 55, 63, 64]

[12] More generally, this is the case where F and G are related by a relation of the form $F(x) = \alpha \cdot G(x) + \beta$ for certain $\alpha, \beta \in \mathbb{F}_q$ (i.e., $(y, x) \mapsto (x, (y + \alpha) \times G(x) + \beta)$).



(a) Horst over $\mathbb{F}_q^2$.

(b) Generalized Horst over $\mathbb{F}_q^4$.

Fig. 1: The generalized Horst scheme over $\mathbb{F}_q^t$.

and from the indifferentiable one [22, 45, 24, 28].[13] We leave the problem of finding the minimum number of rounds for which the Horst construction is indistinguishable and/or indifferentiable from a pseudo-random permutation/function (PRP/PRF) open for future work.

Here we limit ourselves to make some initial considerations about the security of r rounds of Horst based on the attacks on Feistel schemes presented e.g. in [62]. In the following, $F^{(i)}, G^{(i)} : \mathbb{F}_q \to \mathbb{F}_q$ denotes the i-th round functions for $i \in \{0, \dots, r-1\}$. In App. A we show that

- if $r \in \{1, 2\}$, there exists a distinguisher for (i) Feistel and (ii) Horst, and
- if $r \in \{3, 4, 5\}$, there exists a distinguisher for (i) Feistel and (ii) Horst$^\times$ (under the condition that $F^{(i)}$ is equal to zero).

We prove this second result by working over the commutative group $(\mathfrak{G}, *)$ and by considering a generic scheme of the form $(y, x) \mapsto (x, y * H(x))$, where $H^{(i)} : \mathfrak{G} \to \mathfrak{G}$ denotes the i-th round function for $i \in \{0, \dots, r-1\}$. We also define $\zeta \in \mathfrak{G}$ s.t. $\zeta * x = x * \zeta = x$ for $x \in \mathfrak{G}$. The corresponding result on the Feistel scheme is obtained by replacing $*$ with the addition operation, ζ with 0, and H with F. The corresponding result on Horst$^\times$ is obtained by replacing $*$ with the multiplication operation, ζ with 1, and H with G.

The problems of setting up distinguishers for more than 2 rounds of Horst and for more than 6 rounds of Feistel or Horst$^\times$ are open for future research.

Generalized Horst. Next, we generalize the Horst scheme over $\mathbb{F}_q^t$ for $t \geq 2$.

Definition 2 (Generalized Horst). *Let $t \geq 2$. For each $i \in \{0, 1, \dots, t-2\}$, let $G_i : \mathbb{F}_q^{i+1} \to \mathbb{F}_q \setminus \{0\}$, and let $F_i : \mathbb{F}_q^{i+1} \to \mathbb{F}_q$. We define the Generalized*

[13] Roughly speaking, in the first case, the attacker does not have any information about the functions F_i that define the round. In the second case, they can e.g. query such functions, which are public available.

Horst scheme over $\mathbb{F}_q^t$ *as*

$$\begin{aligned}(x_0, x_1, \ldots, x_{t-1}) \mapsto (&x_1 \times G_0(x_0) + F_0(x_0), x_2 \times G_1(x_0, x_1) + F_1(x_0, x_1),\\ &\ldots, x_{t-1} \times G_{t-2}(x_0, x_1, \ldots, x_{t-2}) + F_{t-2}(x_0, x_1, \ldots, x_{t-2}), x_0)\,.\end{aligned} \tag{3}$$

We refer to Fig. 1b for $t = 4$. Based on [70, 60, 44], we see the following.

- If $G_i = 1$ for $i \in \{0, \ldots, t-2\}$, $F_j = 0$ for $j \in \{1, \ldots, t-2\}$, and without a condition on F_0, we have Type-I Feistel. If $G_i = 1, F_i = 0$ for $i \in \{1, \ldots, t-2\}$, and without a condition on G_0 and F_0, we have Type-I Horst.
- If $G_i = 1$ for $i \in \{0, \ldots, t-2\}$ and $F_j(x_0, \ldots, x_{j-1}) = F'_j(x_{j-1})$ for $j \in \{0, \ldots, t-2\}$, we have Type-III Feistel. If $G_j(x_0, \ldots, x_{j-1}) = G'_j(x_j)$ and $F_j(x_0, \ldots, x_{j-1}) = F'_j(x_{j-1})$ for $j \in \{0, \ldots, t-2\}$, we have Type-III Horst.

The results are similar for Type-II, expanding, and contracting constructions.

3.3 An Efficient Nonzero Function G

One way of instantiating G is to exploit the following result proposed in [40].

Lemma 1. *Let* $G : \mathbb{F}_q \to \mathbb{F}_q$ *such that* $G'(x) := G(x) \cdot x$ *is a permutation over* $\mathbb{F}_q$ *and* $G(0) \neq 0$*. Then,* $G(x) \neq 0$ *for* $x \in \mathbb{F}_q$*.*

The proof exploits the fact that G' is a permutation and $G'(0) = G(0) \cdot 0 = 0$ by definition, hence $G'(x) \neq 0$ for each $x \neq 0$. It follows that $G(x) = G'(x)/x \neq 0$ for $x \in \mathbb{F}_q \setminus \{0\}$ and $G(0) \neq 0$ by assumption. As shown in [40], let $d \geq 3$ be the smallest integer such that $x \mapsto x^d$ is invertible over $\mathbb{F}_q$, hence $\gcd(d, q-1) = 1$. The function G over $\mathbb{F}_q$ introduced in there and defined as $G(z) = \frac{(z \pm \alpha)^d \mp \alpha^d}{z} = \sum_{i=1}^{d} \binom{d}{i} z^{i-1} \cdot (\pm\alpha)^{d-i}$ satisfies the requirements of Lemma 1.

In the case of binary fields $\mathbb{F}_{2^n}$, Lemma 1 can be exploited by noting that $x \mapsto x^{2^i}$ are linear operations over $\mathbb{F}_2^n$. Indeed, by defining $G(x) = \sum_{i=0}^{d} \alpha_i \cdot x^{2^i - 1}$ for $\alpha_0 \in \mathbb{F}_{2^n} \setminus \{0\}$ and $\alpha_1, \alpha_2, \ldots, \alpha_d \in \mathbb{F}_{2^n}$, due to Lemma 1, G satisfies the required property if and only if the matrix corresponding to $G'(x) = x \cdot G(x) = \sum_{i=0}^{d} \alpha_i \cdot x^{2^i}$ rewritten over $\mathbb{F}_2^n$ is invertible.

Generic Result for Prime Fields. In the case of a prime field $\mathbb{F}_p$ for $p \geq 3$, we can also exploit the fact that the quadratic map $x \mapsto x^2$ is not invertible over $\mathbb{F}_p$ in order to construct G. Let $\alpha, \beta \in \mathbb{F}_p$ such that $\alpha^2 - 4\beta$ is a quadratic nonresidue modulo p, that is, $\alpha^2 - 4\beta \neq w^2$ for each $w \in \mathbb{F}_p$. In this case,

$$G(x) = x^2 + \alpha x + \beta$$

satisfies the required property. Indeed, the solutions of $x^2 + \alpha x + \beta = 0$ are given by $x_\pm = -(\alpha \pm \sqrt{\alpha^2 - 4\beta})/2$. Since $\alpha^2 - 4\beta$ is a quadratic nonresidue, no solution $x_\pm$ exists. Note that the function G just given does in general not satisfy the requirement of Lemma 1.

Lemma 2 **([58, Corollary 2.9]).** *The function $H(x) = \eta \cdot x^3 + \psi \cdot x^2 + \varphi \cdot x$ over $\mathbb{F}_p$ is invertible if and only if $p = 2 \mod 3$ and $\psi^2 = 3 \cdot \eta \cdot \varphi \mod p$.*

Since $G'(x) = G(x) \cdot x = x^3 + \alpha \cdot x^2 + \beta \cdot x$, $G(x)$ is never a permutation if (i) $p = 1 \mod 3$ or if (ii) $p = 2 \mod 3$ and $\alpha^2 = 3 \cdot \beta$ does not satisfy the condition that $\alpha^2 - 4 \cdot \beta$ is a quadratic nonresidue modulo p.

3.4 Comparing SPN, P-SPN, Feistel, and Horst

Here we compare SPN, partial SPN (P-SPN), Feistel, and Horst schemes keeping in mind our target applications. For simplicity, we here assume $n = 1$ and $t = t'$.

P-SPN and Type-I Feistel/Horst. P-SPN schemes and Type-I Feistel designs are similar and share many properties. [14] Comparing one round of a P-SPN scheme (whose nonlinear layer includes one nonlinear function S_0 and $t - 1$ identity functions) and one round of a Type-I Feistel scheme [70], we get

$$M \times (S_0(x_0) \,||\, x_1 \,||\, \cdots \,||\, x_{t-1}) \quad \text{and} \quad M' \times (x_0 \,||\, x_1 \,||\, \cdots \,||\, x_{t-1} + F(x_0)),$$

where $M, M' \in \mathbb{F}_q^{t \times t}$ are invertible matrices. In more detail, the j-th components of the two schemes are respectively

$$M_{j,0} \cdot S_0(x_0) + \sum_{l=1}^{t-1} M_{j,l} \cdot x_l \quad \text{and} \quad M'_{j,t-1} \cdot F(x_0) + \sum_{l=0}^{t-1} M'_{j,l} \cdot x_l.$$

The sum in the Feistel case contains x_0 and F does not have to be invertible, but in both schemes the nonlinear part depends only on x_0. Hence, they have similar advantages and disadvantages. For example, they only need a small number of nonlinear operations per round, but at least $t-1$ rounds can be skipped via e.g. an invariant subspace trail for which the nonlinear function is not active (i.e., x_0 is constant). Since the (nonlinear) diffusion is slow, more rounds than in an SPN or a Horst scheme are in general needed for security. This can be a disadvantage for e.g. STARK protocols in which one aims to minimize the depth.

In the case of a Type-I Horst, we get

$$M'' \times (x_0 \,||\, x_1 \,||\, \cdots \,||\, G(x_0) \times x_{t-1} + F(x_0)),$$

where $M'' \in \mathbb{F}_q^{t \times t}$ is an invertible matrix and $F, G : \mathbb{F}_q \to \mathbb{F}_q$ are defined as before. With respect to P-SPN schemes and Type-I Feistel designs, a nonlinear mixing takes place between x_0 and x_{t-1}. However, this round function is linear if x_0 is fixed, which means that a Type-I Horst scheme suffers from the same subspace problem as P-SPN and Type-I Feistel schemes.

[14] The results discussed in this paragraph holds also in the case in which the Type-I Feistel scheme is replaced by an expanding or a contracting Feistel scheme.

SPN and Type-III Feistel/`Horst`*.* For an SPN and a Type-III Feistel [70], we get

$$M \times (S_0(x_0) \mid\mid \cdots \mid\mid S_{t-1}(x_{t-1})) \quad \text{and}$$
$$M' \times (x_0 \mid\mid x_1 + F_1(x_0) \mid\mid \cdots \mid\mid x_{t-1} + F_{t-1}(x_{t-2}))\,,$$

where $M, M' \in \mathbb{F}_p^{t \times t}$ are again invertible matrices and where $F_i : \mathbb{F}_p \to \mathbb{F}_p$ are nonlinear functions. Similar to before, the main differences are that the function describing a Feistel scheme is linear in x_{t-1} and that F_i are not required to be invertible (as before). However, in both cases there is no nonlinear mixing between different components x_l. This is partially solved in a Type-III `Horst` as

$$M^{''} \times (x_0 \mid\mid G_1(x_0) \times x_1 + F_1(x_0) \mid\mid \cdots \mid\mid G_{t-1}(x_{t-2}) \times x_{t-1} + F_{t-1}(x_{t-2}))\,,$$

where a nonlinear mixing between x_i and x_{i+1} take place.

Generalized Feistel and Generalized `Horst`*.* A generalized Feistel scheme uses

$$M \times (x_0 \mid\mid x_1 + F_1(x_0) \mid\mid x_2 + F_2(x_0, x_1) \mid\mid \cdots \mid\mid x_{t-1} + F_{t-1}(x_0, x_1, \ldots, x_{t-2}))$$

in its round, where $M \in \mathbb{F}_p^{t \times t}$ is again an invertible matrix and where $F_i : \mathbb{F}_p^i \to \mathbb{F}_p$ are nonlinear functions. Compared to the previous cases, nonlinear diffusion can take place among $x_0, x_1, \ldots, x_{i-1}$ via the function F_i. However, the combination between x_i and $F_i(x_0, x_1, \ldots, x_{i-1})$ is again linear. This problem does not arise in a generalized `Horst` scheme as defined in Eq. (3), since nonlinear diffusion takes place between x_i and $G_i(x_0, x_1, \ldots, x_{i-1})$.

3.5 The Road to Griffin

A *Fluid*-SPN scheme whose nonlinear layer is composed both of $x \mapsto x^d$ and $x \mapsto x^{1/d}$ (where $d \geq 3$ is the smallest integer ensuring invertibility) can be efficiently proven/verified in ZK protocols. Further, the overall degree of the function increases quickly due to the degree-$(1/d)$ S-boxes, while the round-level constraints remain of degree d. This prevents algebraic attacks exploiting the degree of the entire function and forces e.g. Gröbner basis attacks to work at round level. However, while this representation is efficient in STARKs, such a nonlinear layer may be too expensive for SNARKs and for the plain performance.

An unarranged scheme based on generalized `Horst` seems beneficial since it provides diffusion in the nonlinear layer. To minimize the multiplicative complexity, we work with quadratic functions G_j in Eq. (3), while we fix all F_i functions to zero for efficiency reasons. Further, we work with $G_j(x_0, x_1, \ldots, x_{j-1}) = G'_j(\sum_{l=0}^{j-1} \lambda_l \cdot x_l)$, where $G'_j : \mathbb{F}_p \to \mathbb{F}_p$ for each $j \in \{2, \ldots, t-1\}$.

Nonlinear Layer. By combining a *Fluid*-SPN scheme and `Horst` in a single nonlinear layer, we get $S : \mathbb{F}_p^t \to \mathbb{F}_p^t$ defined as $S(\cdot) = S'' \circ S'(\cdot)$, where

$$(S'(x_0, \ldots, x_{t-1}))_i = \begin{cases} {x_1}^{1/d} & \text{if } i = 0, \\ {x_0}^d & \text{if } i = 1, \\ x_i & \text{otherwise,} \end{cases}$$

$$(S''(x_0, \ldots, x_{t-1}))_i = \begin{cases} x_i & \text{if } i \in \{0, 1\}, \\ x_i \left(z_{i-1}^2 + \alpha_i z_{i-1} + \beta_i\right) & \text{otherwise,} \end{cases}$$

where $\alpha_i^2 - 4\beta_i$ is a quadratic nonresidue for each i and $z_i = \lambda' \cdot y_0 + \lambda^{''} \cdot y_1 + \sum_{l=0}^{i-1} \lambda_l \cdot x_l$ is a linear combination of the inputs $\{x_0, \ldots, x_{i-1}\}$ and the outputs $\{y_0, y_1\}$. Clearly, S' is inspired by the nonlinear layer of *Rescue*, while S'' is based on the `Horst` function previously defined.

Note that this nonlinear layer is invertible. Indeed, the power maps $x \mapsto x^d$ and $x \mapsto x^{1/d}$ are invertible if $\gcd(d, p-1) = 1$ due to Hermite's criterion. Moreover, $x_i = y_i / (z_{i-1}^2 + \alpha_i \cdot z_{i-1} + \beta_i)$ for $i \geq 3$, where $w^2 + \alpha_i \cdot w + \beta_i \neq 0$ for each $w \in \mathbb{F}_p$ by choosing (α_i, β_i) such that $\alpha_i^2 - 4\beta_i$ is a quadratic nonresidue.

Hence, the number of multiplications per round for the verification process is $2 \cdot (\mathrm{hw}(d) + \lfloor \log_2(d) \rfloor - 1)$ for S' and $2 \cdot (t-2)$ for S'', for a total of

$$2 \cdot t + 2 \cdot (\mathrm{hw}(d) + \lfloor \log_2(d) \rfloor - 3),$$

that is, $2t$ (plus a constant) $\mathbb{F}_p$ multiplications per round. Hence, for large t, the cost of our design is almost independent of the value of d. For example, $(\mathrm{hw}(d) + \lfloor \log_2(d) \rfloor - 1) \cdot t$ $\mathbb{F}_p$ multiplications per round are required for *Rescue* or for a full round of Poseidon. Our design requires $2t$ and $2t + 2$ multiplications per round for $d = 3$ and $d = 5$ respectively, while *Rescue* requires $4t$ and $6t$ respectively. In the case of Neptune (a recent variant of Poseidon), each external full round needs only t $\mathbb{F}_p$ multiplications, but the high number of internal partial rounds instantiated with $x \mapsto x^d$ makes it less competitive with respect to Griffin, as we will show in the following.

Griffin *with Feistel.* In order to better understand the advantages of `Horst` with respect to the classical Feistel, we consider a variant of Griffin instantiated with a classical Feistel instead of `Horst`, i.e., with S'' replaced by $\widehat{S''}$ defined as

$$\widehat{S''}(x_0, \ldots, x_{t-1}))_i = \begin{cases} x_i & \text{if } i \in \{0, 1\}, \\ x_i + \left(z_{i-1}^2 + \alpha_i \cdot z_{i-1} + \beta_i\right) & \text{otherwise,} \end{cases} \qquad (4)$$

where as before $\alpha_i^2 - 4\beta_i$ is a quadratic nonresidue for each i and z_i is a linear combination of the inputs $\{x_0, x_1, \ldots, x_{i-1}\}$ and the outputs $\{y_0, y_1\}$. As we discuss in Section 5.4, the security of this variant against algebraic attacks as the Gröbner bases one is smaller and more difficult to argue than with Griffin, and a higher number of rounds is needed to provide security. Besides that, a slower diffusion occurs, and more rounds are potentially needed for guaranteeing security. This has a crucial impact on the performance in the target ZK applications, as e.g. in zk-STARK in which we aim to minimize the depth of the evaluated hash function. This concrete comparison allows us to better understand the importance and the impact of the nonlinear combination in the `Horst` scheme.

Linear Layer. In many recent SNARK/STARK-friendly designs, an MDS matrix is used for every state size t. Even with optimized representations e.g. for Poseidon, the number of linear operations is an element in $O(t^2)$ in all cases. Moreover, since our target applications mostly use large primes (e.g., $p \geq 2^{64}$) for a security level of 128 or 256 bits, an MDS matrix for large t is not required from a statistical point of view. For example, security against (classical) differential and linear attacks can also be provided with smaller branch numbers.

Therefore, in Griffin we only use a cheap MDS matrix for $t \in \{3, 4\}$, and we use an efficient linear layer for $t > 4$. Still, we want to achieve full diffusion over a single round to resist e.g. truncated differential, impossible differential, and rebound attacks. For this goal and for the case $t = 4 \cdot t' \geq 8$, we consider the linear layer of AES, which can be rewritten as the multiplication of two matrices, namely $M = M_{\text{MC}} \times M_{\text{SR}}$ where

$$M_{\text{SR}} = \text{diag}(I, I_2, I_3, I_4), \qquad M_{\text{MC}} = \text{circ}(2 \cdot I, 3 \cdot I, I, I),$$

where I is the 4×4 identity matrix, $I_2 = \text{circ}(0, 1, 0, 0)$, $I_3 = \text{circ}(0, 0, 1, 0)$, and $I_4 = \text{circ}(0, 0, 0, 1)$. As is well-known, $M = M_{\text{MC}} \times M_{\text{SR}}$ does not provide full diffusion over a single round, due to the fact that each I_i is sparse. In particular, M_{SR} only changes the position of the input words, without mixing them. Therefore, we decided to replace every I_i with the MDS matrix circ$(3, 2, 1, 1)$, and we generalize the matrix M_{MC} via the circulant matrix circ$(2 \cdot I, I, \ldots, I)$ (where $I \in \mathbb{F}_p^{4 \times 4}$ is the identity matrix).

This change has a considerable impact on the design. First, we can achieve full diffusion over a single round (M contains only nonzero entries). This advantage comes at the price that e.g. the wide-trail design strategy is not applicable both due to the choice of M' and of the nonlinear layer. Secondly, regarding the plain performance, the multiplication with our matrix M is very efficient. Indeed, for $t \geq 8$ (and similarly for $t \in \{3, 4\}$), the multiplication with circ$(3, 2, 1, 1)$ only needs 15 additions (e.g., $3x_0 + 2x_1 + x_2 + x_3 = y + (x_0 + x_0) + x_1$ where $y = x_0 + x_1 + x_2 + x_3$ is computed only once), resulting in a total of $15 \cdot t' = 15(t/4) \approx 4t$ additions. Further, $M^{''}$ can be efficiently computed with $4(t/4) + t = 2t$ additions (using the same approach proposed for M'). Hence, our linear layer M only requires around $6t \in \mathcal{O}(t)$ additions.

4 Griffin and Griffin-π

Griffin is a sponge hash function over $\mathbb{F}_p^t$ instantiated with the permutation Griffin-π, where $p > 2^{32}$ (or $\lceil \log_2(p) \rceil > 32$) is a prime and $t \in \{3, 4t'\}$ for a positive integer $t' \in \{1, 2, \ldots, 6\}$, i.e., t is either 3 or a multiple of 4. We limit ourselves to $t \leq 24$, since this is sufficient for the applications we have in mind. The security level is κ bits, where $80 \leq \kappa \leq \min\{256, \lfloor \log_2(p) \cdot t/3 \rfloor\}$. We assume there exists $d \in \{3, 5, 7, 11\}$ such that $\gcd(d, p-1) = 1$.[15]

4.1 Sponge Hash Functions

The sponge construction introduced in [12, 13] builds upon an internal permutation and can be used to achieve various goals such as encryption, authentication, or hashing. Both the input and the output may be of arbitrary size. We use the classical sponge construction to build a hash function using the Griffin-π permutation. The state size is split into $t = r + c$, where r and c denote the number of elements in the rate (outer) and capacity (inner) part, respectively. The security of this construction then depends on r and c. Indeed, it has been proven in [13] that, if the inner permutation is a random one, the sponge construction is indifferentiable from a random oracle up to around $\min\{p^r, p^{c/2}\}$ queries. Equivalently, in order to provide κ bits of security, $p^{c/2} \geq 2^\kappa$ (i.e., $c \geq \lceil 2\kappa \cdot \log_p(2) \rceil$) and $p^r \geq 2^\kappa$ (i.e., $r \geq \lceil \kappa \cdot \log_p(2) \rceil$). From now on, we impose $c \geq \lceil 2\kappa \cdot \log_p(2) \rceil$ and $r \geq \lceil \kappa \cdot \log_p(2) \rceil$, which implies $p^t \geq 2^{3 \cdot \kappa}$. Given an input message m, the padding rule for Griffin resembles the one proposed for Poseidon in [36, Section 4.2], and consists of adding the smallest number $< r$ of zeros such that the size of $m \,||\, 0^*$ is a multiple of r and of replacing the initial value $\texttt{IV} \in \mathbb{F}_p^c$ instantiating the inner part with $|m| \,||\, \texttt{IV}' \in \mathbb{F}_p^c$, where $|m| \in \mathbb{F}_p$ is the size of the input message m and $\texttt{IV}' \in \mathbb{F}_p^{c-1}$ is an initial value.

4.2 Specification of Griffin-π

The Griffin-π permutation $\mathcal{G}^\pi : \mathbb{F}_p^t \to \mathbb{F}_p^t$ is defined by

$$\mathcal{G}^\pi(\cdot) := \mathcal{F}_{R-1} \circ \cdots \circ \mathcal{F}_1 \circ \mathcal{F}_0(M \times \cdot),$$

where $M \in \mathbb{F}_p^{t \times t}$ is an invertible matrix and $\mathcal{F}_i : \mathbb{F}_p^t \to \mathbb{F}_p^t$ is a round function of the form $\mathcal{F}_i(\cdot) = c^{(i)} + M \times S(\cdot)$ for a round constant $c^{(i)} \in \mathbb{F}_p^t$, a nonlinear layer $S : \mathbb{F}_p^t \to \mathbb{F}_p^t$, and $i \in \{0, 1, \ldots, R-1\}$. The same matrix M is applied to the input and in every round. We assume $c^{(R-1)} = 0$. The overall design is shown in Fig. 2, and the details of the components are provided in the following.

The Nonlinear Layer *S*. Let $d \in \{3, 5, 7, 11\}$ be the smallest integer such that $\gcd(d, p-1) = 1$. Let $(\alpha_i, \beta_i) \in \mathbb{F}_p^2 \setminus \{(0, 0)\}$ be pairwise distinct such that $\alpha_i^2 - 4\beta_i$ is a quadratic nonresidue modulo p for $2 \leq i \leq t-1$. The nonlinear

[15] Griffin and Griffin-π may be defined also if there exists no $d \in \{3, 5, 7, 11\}$ such that $\gcd(d, p-1) = 1$. However, the following security analysis and the number of rounds must be adapted for this case.

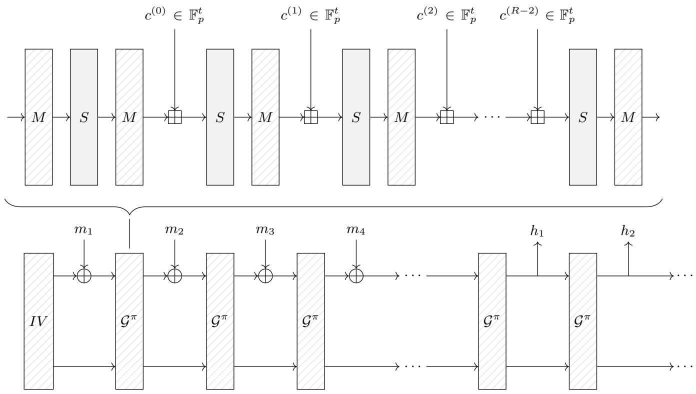


Fig. 2: Griffin-π (top) and the Griffin sponge (bottom), where $\boxplus$ and $\oplus$ denote the element-wise addition of two vectors in $\mathbb{F}_p^t$ and $\mathbb{F}_p^r$, respectively.

layer $S(x_0, \dots, x_{t-1}) = y_0 \,||\, \cdots \,||\, y_{t-1}$ is then defined by

$$y_i = \begin{cases} {x_0}^{1/d} & \text{if } i = 0, \\ {x_1}^{d} & \text{if } i = 1, \\ x_2 \cdot \big((L_i(y_0, y_1, 0))^2 + \alpha_2 \cdot L_i(y_0, y_1, 0) + \beta_2\big) & \text{if } i = 2, \\ x_i \cdot \big((L_i(y_0, y_1, x_{i-1}))^2 + \alpha_i \cdot L_i(y_0, y_1, x_{i-1}) + \beta_i\big) & \text{otherwise,} \end{cases} \tag{5}$$

where $L_i : \mathbb{F}_p^3 \to \mathbb{F}_p$ are linear functions of the form $L_i(z_0, z_1, z_2) = \gamma_i \cdot z_0 + z_1 + z_2$ for arbitrary pairwise distinct $\gamma_i \in \mathbb{F}_p \setminus \{0\}$ (e.g., $\gamma_i = i - 1$).

The Linear Layer *M*. For $t \in \{3, 4\}$, the matrices are $M_{t=3} = \mathrm{circ}(2, 1, 1)$ and $M_{t=4} = \mathrm{circ}(3, 2, 1, 1)$, and they are invertible and MDS for $p \geq 2^{32}$. If $t = 4t' \geq 8$, M is defined as

$$M = M'' \times M' \equiv M' \times M'' = \begin{pmatrix} 2 \cdot M_{t=4} & M_{t=4} & \dots & M_{t=4} \\ M_{t=4} & 2 \cdot M_{t=4} & \dots & M_{t=4} \\ \vdots & \vdots & \ddots & \vdots \\ M_{t=4} & M_{t=4} & \dots & 2 \cdot M_{t=4} \end{pmatrix}, \tag{6}$$

where $M', M'' \in \mathbb{F}_p^{t \times t}$ are defined as

$$M' = \mathrm{diag}(M_{t=4}, M_{t=4}, \dots, M_{t=4}), \quad M'' = \mathrm{circ}(2 \cdot I, I, \dots, I),$$

where $M_{t=4} = \mathrm{circ}(3, 2, 1, 1)$ is an MDS matrix and where I is the 4×4 identity matrix. For $t = 4t' \geq 8$, the matrix is invertible since[16]

$$\det(M) = \det(M') \cdot \det(M'') = (\det(\mathrm{circ}(3, 2, 1, 1)))^{t'} \cdot (t'+1) = 35^{t'} \cdot (t'+1) \neq 0 \mod p.$$

[16] Note that $(1, 1, \dots, 1)^T$ is an eigenvector of M'' with the eigenvalue $t' + 1$. All other $t - 1$ eigenvectors (of the form $(1, 0, \dots, 0, -1, 0, \dots, 0)^T$) of M'' have eigenvalue equal to 1. Hence, $\det(M'') = t' + 1$.

Table 1: Instances of Griffin-π with security margin. We focus on the most common cases, namely $d \in \{3, 5\}$, $\kappa = 128$, $p \approx 2^{256}$, and $c = \lceil 2\kappa / \log_2(p) \rceil$.

t	R $(d = 3)$	R $(d = 5)$
3	16	12
4	14	11
8	11	9
$12, 16, 20, 24$	10	9

Choosing the Constants. We use a pseudo-random number generator based on SHAKE [59] in order to choose our round constants $\{c^{(i)}\}_{i=0}^{R-2}$ and the constants $\{(\alpha_2, \beta_2)\}$ and λ_0, λ_1 that define the nonlinear layer. The other constants $\{(\alpha_i, \beta_i)\}_{i=3}^{t-1}$ are defined as $\alpha_i = (i-1) \cdot \alpha_2$ and $\beta_i = (i-1)^2 \cdot \beta_2$. Note that $L_p(\alpha_i^2 - 4 \cdot \beta_i) = L_p((i-1)^2 \cdot (\alpha_2^2 - 4 \cdot \beta_2)) = L_p(\alpha_2^2 - 4 \cdot \beta_2) = -1$.

4.3 Security of Griffin-π

For κ-bit security, where $80 \leq \kappa \leq \min\{256, \lfloor \log_2(p) \cdot t/3 \rfloor\}$, the number of rounds R including a security margin of 20% must satisfy

$$R \geq \lceil 1.2 \cdot \max\{6, 1 + R_{\mathrm{GB}}\} \rceil \,,$$

where $R_{\mathrm{GB}} \geq 1$ is the smallest integer such that

$$\min\left\{ \binom{R_{\mathrm{GB}} \cdot (d+t) + 1}{1 + t \cdot R_{\mathrm{GB}}}, \binom{d^{R_{\mathrm{GB}}} + 1 + R_{\mathrm{GB}}}{1 + R_{\mathrm{GB}}} \right\} \geq 2^{\kappa/2} \,.$$

These numbers are supported by our security analysis given in Section 5. Some instances for Griffin-π are given in Table 1.

5 Security of Griffin and Griffin-π

5.1 Sponge Hash Function Security

A hash function $\mathcal{H} : \mathbb{F}_p^\star \to \mathbb{F}_p^t$ needs to fulfill certain security properties. Informally, it should be computationally hard to find

(collision resistance) $x, x' \neq x$ such that $\mathcal{H}(x) = H(x')$,
(preimage resistance) x given y such that $\mathcal{H}(x) = y$, or
(second-preimage resistance) x' given $x \neq x'$ such that $\mathcal{H}(x') = \mathcal{H}(x)$.

We refer to [57] for a formal definition of these conditions. In our context, we aim to guarantee that the complexity of enforcing any of them is 2^κ for a security level of κ bits. For this purpose, we introduce the CICO problem.

Definition 3. *The invertible function $\mathcal{P} : \mathbb{F}_p^t \to \mathbb{F}_p^t$ is κ-secure against the CICO (t_1, t_2)-problem (where $0 < t_1, t_2 < t$ and $t_1 + t_2 = t$) if no algorithm with expected complexity smaller than 2^κ finds $I_2 \in \mathbb{F}_p^{t_2}$ and $O_2 \in \mathbb{F}_p^{t_1}$ for given $I_1 \in \mathbb{F}_p^{t_1}$ and $O_1 \in \mathbb{F}_p^{t_2}$ such that $\mathcal{P}(I_1 \mid\mid I_2) = O_1 \mid\mid O_2$.*

A sponge hash function instantiated with a pseudo-random permutation can provide $\min\{r \cdot \log_2(p), c/2 \cdot \log_2(p)\}$ bits of collision and (second-)preimage resistance. In practice, however, the inner permutation $\mathcal{P}$ is replaced by a concrete iterated permutation. Two possible approaches are usually taken to choose the number of rounds of this permutation. First, as done by the designers of Keccak, it is possible to choose the number of rounds of $\mathcal{P}$ in order to provide security against any (known-/chosen-) distinguisher. Equivalently, this means that this number guarantees that $\mathcal{P}$ does not present any non-random/structural property (among the ones known in the literature). However, more recently the designers tend to choose the number of rounds with the goal of preventing only attacks on the hash function itself. In other words, designers do not consider distinguishers on the internal permutation that cannot be exploited in order to set up an attack on the hash function (as the zero-sum partition one). In this paper, we make use of the same approach.

5.2 Statistical Attacks on Griffin-π

In this section, we present an analysis of the best statistical attacks against Griffin-π, which include the differential attack [15] and the rebound attack [56, 54]. Our theoretical security analysis is supported by dedicated automatic MILP tools which we designed in order to search for bounds on the minimal number of rounds against (truncated) differential, linear, and rebound attacks. Due to page limitation, other attacks as linear cryptanalysis, impossible differential, zero-correlation, integral/square, multiple-of-n, and mixture differential attacks are analyzed in App. G.1 instead.

Differential Cryptanalysis. Differential cryptanalysis [15] and its variations are the most widely used techniques to analyze symmetric-key primitives. Given pairs of inputs with fixed input differences, differential cryptanalysis considers the probability distribution of the corresponding output differences. Let $\Delta_I, \Delta_O \in \mathbb{F}_p^t$ be respectively the input and the output differences through a permutation $\mathcal{P}$ over $\mathbb{F}_p^t$. The differential probability (DP) for the output difference Δ_O given the input difference Δ_I is

$$\text{Prob}(\Delta_I \to \Delta_O) = (|\{x \in \mathbb{F}_p^t \mid \mathcal{P}(x + \Delta_I) - \mathcal{P}(x) = \Delta_O\}|)/p^t.$$

Its maximum DP is $\text{DP}_{\max} = \max_{\Delta_I, \Delta_O \in \mathbb{F}_p^t \setminus \{0\}} \text{Prob}(\Delta_I \to \Delta_O)$. As Griffin-$\pi$ is an iterated scheme, we search for ordered sequences of differences over any number of rounds, i.e., differential characteristics/trails. Assuming independent rounds, the DP of a differential trail is the product of the DPs of its one-round differences. Our goal is to find the minimum number of rounds such that each characteristic's probability is smaller than $2^{-2.5\kappa}$. We chose this value since more characteristics can be used simultaneously to set up a differential attack, and hence each probability must be smaller than $2^{-\kappa}$ for security. For this purpose, we first compute $\mathrm{DP}_{\max}$ of the components of the nonlinear layer S and the branch number of the matrix M.

Lemma 3. *Let $d \geq 3$ be an integer s.t. $\gcd(d, p-1) = 1$. Then, $\mathrm{DP}_{\max}(x \mapsto x^d) = \mathrm{DP}_{\max}(x \mapsto x^{1/d}) = (\min\{d, 1/d\} - 1)/p$, where $\min\{d, 1/d\} = d$.*

Lemma 4. *Let $\alpha, \beta \in \mathbb{F}_p \setminus \{0\}$ such that $\alpha^2 - 4\beta$ is a quadratic nonresidue modulo p. Let $F : \mathbb{F}_p^2 \to \mathbb{F}_p$ be defined as $F(x, \ell) = x \cdot (\ell^2 + \alpha \cdot \ell + \beta)$. Given an input difference $\Delta_I = (\delta_x, \delta_\ell) \neq (0, 0)$ and an output difference Δ_O,*

$$Prob(\Delta_I \to \Delta_O) \leq \begin{cases} \frac{2}{p} & \text{if } \delta_\ell = 0 \text{ or } \delta_x = \Delta_O = 0, \\ \frac{p-1}{p^2} \leq \frac{1}{p} & \text{otherwise.} \end{cases}$$

In particular, if $\delta_\ell = 0$, then $\Delta_O \neq 0$.

Note that the previous probability is always smaller than $(d-1)/p$ for each $d \geq 3$. The proofs for Lemmas 3 and 4 are given in App. D.1.

Proposition 1. *Let $t = 4t' \in \{8, 12, \ldots, 24\}$. Given $M \in \mathbb{F}_p^{t \times t}$ defined as in Eq. (6), its branch number is $t' + 4$.*

We practically verified and proved the branch numbers for our instantiations, and give details in App. D.2. We recall that $M \in \mathbb{F}_p^{t \times t}$ is an MDS matrix for $t \in \{3, 4\}$ and its branch number is equal to $t+1$ in these cases. Even if we cannot directly use the wide-trail design strategy [25] since our design is weak-arranged, each active word at the input of S activates at least one word at its output. In particular, x_{t-1} affects only y_{t-1}, x_i affects y_i and y_{i+1} for $i \in \{2, \ldots, t-2\}$, x_0 affects all the output words except y_1, and x_1 affects all the output words except y_0. Further, due to the branch number of the matrix, at least $\#b$ words are active every two rounds. Due to these facts, if $t = 4 \cdot t' \geq 8$, the probability of any differential characteristic over 2 rounds is at most

$$\frac{(d-1)^4 \cdot 2^{\#b-4}}{p^{\#b}} = \frac{(d-1)^4 \cdot 2^{t'}}{p^{4+t'}} \overset{(i)}{\leq} \frac{(d-1)^4 \cdot 2^{t'}}{p^4 \cdot 2^{(3\cdot\kappa)/4}}$$
$$\overset{(ii)}{<} \frac{2^6 \cdot (d-1)^4}{2^{\kappa/2 + (3\cdot\kappa)/4}} = 2^{-\kappa} \cdot \frac{2^6 \cdot (d-1)^4}{2^{\kappa/4}} \overset{(iii)}{\leq} 2^{-\kappa} \cdot \frac{(d-1)^4}{2^{14}},$$

where (i) is motivated by $2^{3\cdot\kappa} \leq p^t$, (ii) is motivated by $t' \leq 6$ and $p > 2^{32} \geq 2^{\kappa/8}$ since $\kappa \leq 256$, and (iii) is motivated by $\kappa \geq 80$. Similarly, for the case $t \in \{3, 4\}$, the probability of any differential characteristic over 2 rounds is at most

$$\frac{(d-1)^4 \cdot 2^{t-3}}{p^5} \leq \left(\frac{d-1}{p}\right)^5 \overset{(iv)}{\leq} \left(\frac{d-1}{2^\kappa}\right)^5 = 2^{-3\cdot\kappa} \cdot \left(\frac{(d-1)^5}{2^\kappa}\right)^2 \leq 2^{-3\cdot\kappa},$$

which is always satisfied for $d \leq 1 + 2^{16}$ (about (iv), note that $2^{3\cdot\kappa} \leq p^t$, hence $2^\kappa \leq p$ for $t = 3$ and $2^\kappa \leq p \leq p^{4/3}$ for $t = 4$). Hence, if $d \in \{3, 5, 7, 11\}$, the probability of any differential characteristic over 2 consecutive rounds is strictly smaller than $2^{-\kappa}$. Further, the probability of any differential characteristic over 6 consecutive rounds is smaller than $2^{-2.5\cdot\kappa}$ if $d \le 113$ (equivalently, $(d-1)^6 \le 2^{41} \le 2^{21} \cdot 2^{\kappa/4}$), which we assume to be satisfied. As a result, 6 consecutive rounds are largely sufficient for preventing (classical) differential attacks.

Truncated Differential and Rebound Attacks. Truncated differential cryptanalysis [48] is a variant of classical differential cryptanalysis where the attacker specifies only part of the difference between pairs of texts or specifies conditions between the differences. This method is particularly efficient in the case of aligned schemes (e.g., AES-like schemes). Considering the case of active/inactive words, since diffusion takes place in the nonlinear layer and since the matrix M provides full diffusion over a single round, a truncated differential with probability 1 holds for one single round only. In order to extend it, it would be necessary that some differences are equal to zero. However, the probability that one word is inactive is p^{-1}, exactly as for a pseudo-random permutation. Hence, we do not expect that this attack outperforms the classical differential one just proposed.

In a rebound attack [50, 56], the goal of the attacker is to find two (input, output) pairs such that the two inputs satisfy a certain (truncated) input difference and the corresponding outputs satisfy a certain (truncated) output difference. The approach consists of the *inbound* and the *outbound* phase. According to these phases, the internal permutation $\mathcal{P}$ of the hash function is split into three subparts, that is, $\mathcal{P} = \mathcal{P}_{\text{fw}} \circ \mathcal{P}_{\text{in}} \circ \mathcal{P}_{\text{bw}}$. The inbound phase is placed in the middle of the permutation and the two outbound phases are placed next to the inbound part. In the outbound phase, two high-probability (truncated) differential trails are constructed, which are then connected in the inbound phase. We claim that 6 rounds are sufficient against this attack. From our analysis, we know that there exist truncated differentials with probability 1 over a single round, but they cannot be extended over more rounds, and any classical differential characteristic over 2 rounds has a probability smaller than $2^{-\kappa}$ (for common d). Hence, by using an inside-out approach, the attacker can cover less than 4 rounds in the inbound phase. Since one round can be covered with a truncated differential characteristic of probability 1, the attacker can cover two rounds (one in each direction) in the outbound phase. Thus, no rebound attack on 6 rounds of Griffin-π can be set up.

Verification with Dedicated Tools. Our results have been verified via a dedicated mixed integer linear programming (MILP) tool. The results obtained with the tools for (classical/truncated) differential attacks and rebound attacks are presented in App. E. They support the conclusion that 6 rounds are sufficient in order to provide security against these attacks.

5.3 Algebraic Attacks

Algebraic attacks exploit weak algebraic properties of the design (e.g., low degrees or low density). Our analysis suggests that interpolation attacks and Gröbner basis attacks are the most efficient ones against Griffin. For this purpose, we analyze the algebraic properties of the obtained equation systems and also practically implement Griffin-π to obtain better estimates.

We also claim security against higher-order differentials, which is implied by the security against interpolation attacks. We do not claim security against zero-sum partitions [18]. We refer to App. G.2 for more details.

Interpolation Attacks. The goal of an interpolation attack [46] is to construct an interpolation polynomial describing the function. In the case of a hash function, an interpolation polynomial can potentially be exploited to set up collisions or forgery attacks. The cost of the attack growths with the number of different monomials in the interpolation polynomial, where (an upper/lower bound of) the number of different monomials can be estimated given the degree of the function. If the number of unknown monomials is sufficiently large, this cannot be done faster than by exhaustive search. Roughly speaking, if the interpolation polynomial is dense and if its degree is maximum, this attack does not work.

In our case, 3 rounds are sufficient to reach the maximum degree. Indeed, due to Fermat's little theorem, $1/d \equiv d'$ where $(d \cdot d' - 1) \mod (p-1) = 0$. Since $d \geq 3$ is the smallest integer such that $\gcd(d, p-1) = 1$, this implies that d' is of the same order of p. In order to frustrate variants of the interpolation attack like MitM approaches or inside-out approaches starting from the middle of the constructions, we double the number of rounds, conjecturing that $2 \cdot 3 = 6$ rounds are sufficient to prevent interpolation attacks and their variants. We further refer to App. F.1 for a more detailed analysis about the density of Griffin-π.

Gröbner Basis Attacks. In Gröbner basis [19, 23] attacks, the goal is to represent the cryptographic construction as a system of equations, and then solve for a set of variables depending on the attack goals. In general, a Gröbner basis attack consists of three steps. First, the attacker needs to set up the equation system and compute a Gröbner basis for it. Secondly, they perform a change of term ordering for the basis, usually going to a term order which makes it easier to eliminate variables and find the solutions. Finally, the attacker uses the system obtained in the second step in order to start solving for the variables. As is usually done in the literature, here we focus on the complexity of the first step (i.e., computing a Gröbner basis), which can be estimated by

$$\mathcal{O}\left(\binom{D_{\text{reg}} + n_v}{n_v}^{\omega}\right),$$

where D_{reg} is the degree of regularity, n_v is the number of variables, and $2 \leq \omega < 3$ is a constant representing the complexity of a matrix multiplication. Theoretical estimations of the degree of regularity are known only for regular and semi-regular system of equations [8, 9]. For example, in the case of a regular system of equations with $n_e = n_v$, where n_e denotes the number of polynomials in the system, the degree of regularity is estimated by $D_{\text{reg}} = 1 + \sum_{i=1}^{n_e} (d_i - 1)$, where d_i is the degree of the i-th equation. Since most of our equation systems will not exhibit the properties of regular sequences, we will compute the actual degrees reached during the computations (i.e., the practical degree of regularity) for reduced versions of Griffin-π, and then use the new estimates to compute the final round numbers.

Here we focus on a preimage attack on the hash function, using the algebraic properties of the permutation Griffin-π. This approach has also been adopted in the literature before, e.g. for Poseidon/Neptune, *Rescue*, and *Grendel*. Moreover, it naturally extends to the CICO problem (see Definition 3) by simply reordering the elements. The collision attack is analogous (see App. F.4).

Intermediate Variables. Using the inputs and outputs of Griffin-π directly is infeasible since the degree is maximum and the polynomials are dense. A possible strategy to overcome this problem consists in introducing intermediate variables. This is a method to decrease the degrees in the equation system (and thus in general also the number of appearing monomials) at the cost of more variables. For Griffin-π, we can introduce new variables in each round in order to avoid reaching a degree of $1/d$. Let us denote the state before a nonlinear layer by $x = x_0 \mid\mid \cdots \mid\mid x_{t-1}$ and the state after the same nonlinear layer by $y = y_0 \mid\mid \cdots \mid\mid y_{t-1}$. Then, the relation between x and y can be described by 2 equations of degree d and $t-2$ equations of degree 3, using the fact that $y_1 = {x_1}^{1/d}$ can be rewritten as $y_1^d = x_1$ and the definition of our nonlinear layer given in Eq. (5). The variables in this equation system are $x_0, \ldots, x_{t-1}$ and $y_0, \ldots, y_{t-1}$. In order to connect two rounds with this approach, we denote the input of the next nonlinear layer by affine functions in $y_0, \ldots, y_{t-1}$, depending on the linear layer matrix M and on the round constants. In this way, we introduce t variables for each round, except for the last one, where we simply use the desired output values. Using this representation, we have $n_v = r + Rt$ variables (where r is the rate) and the same number of equations $n_e = n_v$. Of these equations, $2R$ equations are of degree d and $(t-2)R$ equations are of degree 3. The degree of the remaining equations depends on r. We focus on $r = 1$, since by experiments this case is the easiest from the attacker's point of view.

When implementing this system in `Sage` and `Magma`, the observed degrees of regularity are $\geq D_{\text{est}}^{(1)} = dR$ for a degree-d nonlinear layer after R rounds (see App. F.2 for details). Using $D_{\text{est}}^{(1)}$, we obtain an estimated complexity of $\binom{D_{\text{est}}^{(1)}+n_v}{n_v}^{\omega} = \binom{dR+n_v}{n_v}^{\omega}$ operations. By setting $\omega = 2$ (optimistic from the attacker's point of view) and for a security level of κ bits, R must satisfy

$$\log_2\left(\binom{D_{\text{est}}^{(1)} + n_v}{n_v}\right) = \log_2\left(\binom{dR+1+tR}{1+tR}\right) \geq \frac{\kappa}{2}. \tag{7}$$

Partial Intermediate Variables. Another strategy consists in introducing only a single intermediate variable for each round in order to avoid the high degree growth in the second word. The other state words go through the nonlinear layer without adding any more variables. In more detail, we introduce a single new equation $0 = {y_1}^d - x_1$ in each round, where y_1 is the new variable. Hence, we have $r + R$ variables in total, and we again focus on $r = 1$. The degree of the equations increases in each round, however not as fast as it would without adding a variable for the second word. By practical experiments, we found that the degree of regularity can be estimated conservatively by $D_{\text{est}}^{(2)} = d^R$ for this strategy (see App. F.2 for details). Even if the equations here have a higher degree than in the first strategy, the number of variables and equations is smaller, since only one relation is added in each round (instead of t). Still, there is one crucial difference. Adding intermediate variables for all state words leads to a complexity which scales significantly with t. In this case, we add only one variable in each round, regardless of t. This means that we require

$$\log_2\left(\binom{D_{\text{est}}^{(2)} + n_v}{n_v}\right) = \log_2\left(\binom{d^R + 1 + R}{1 + R}\right) \geq \frac{\kappa}{2}. \tag{8}$$

Gröbner Basis Summary. Given the results just presented, we require that Eq. (7) and Eq. (8) are fulfilled for a κ-bit security level. However, due to the particular structure of our nonlinear layer, it is possible to choose the input such that the degrees in the first round are lower than expected. In particular, an attacker may choose the input such that $y_0 = x_0^{1/d} = u_1$ and $y_1 = x_1^d = u_2$, where u_1, u_2 are two fixed constants chosen by the attacker. This can be done by simply solving a linear equation system with these constraints. Consequently, the first two words are constant, the third word is linear, and only then the degree starts to grow. In order to in order to protect from this attack, we add 1 round to the final round number needed for preventing Gröbner basis attacks.

For completeness, we also describe two additional attack strategies in App. F.3. However, they are less competitive than the ones just presented, and hence do not determine the number of rounds.

5.4 Security of Griffin Instantiated with Feistel

We consider the security of Griffin instantiated with a Feistel scheme as in Eq. (4) with respect to the two Gröbner basis approaches discussed in Section 5.3.

In the first Gröbner basis strategy we introduce intermediate variables for the whole state, i.e., we add t new variables and equations per round. In our experiments with `Sage` and `Magma` we could observe that the practical degree of regularity was constant regardless of the number of rounds in our tests for $R \geq 2$. Indeed, we were able to compute Gröbner bases in practice for the round numbers proposed for Griffin (with `Horst`). We emphasize that this does not necessarily mean that the complexity of an attack changes only slightly with increased round numbers, but rather that it is harder for the designer to argue security. A similar behaviour was reported in [1, Section 6.1] for MiMC, where computing the Gröbner basis is efficient with intermediate variables, but the other steps in the full attack (monomial reordering, factorization) are not.

For the second strategy, where we only introduce an intermediate variables to avoid the degree-$(1/d)$ growth in each round, it is easier to argue security. Still, the maximum degree in each round is reduced due to the missing multiplication instead of the addition. In particular, the difference is $\deg(R_{i-1})$ in each round, where $\deg(R_i)$ is the degree in the i-th round. Additionally, we could observe faster Gröbner basis computations for the Feistel version compared to the `Horst` version. Concretely, the difference is about a factor of 8 between the two versions.

Hence, even with a detailed analysis of the first strategy, the number of rounds would have to be increased due to the second strategy. This would severely impact the plain performance and the efficiency in the STARK use case (which we recall is related to the depth of the hash function), which suggests that using the multiplication instead of the addition is better when aiming for security and efficiency in the applications discussed in this paper.

6 Performance Evaluation

In this section, we evaluate the performance of Griffin and compare it to Poseidon, *Rescue*-Prime [67] (a newer variant of *Rescue* with less security margin), $\text{GMiMC}_{\text{erf}}$, *Grendel*, and Neptune. Since $\text{GMiMC}_{\text{erf}}$ was broken in [14], we use the updated round numbers proposed in [29, App. G]. We further point out that *Grendel* has recently been broken [37], leading to an adaptation of the round numbers by the designers. Our evaluation includes the updated numbers.

First we evaluate the plain performance, then we compare the efficiency when used in R1CS-based SNARKs. For an evaluation in STARKs and Plonk we refer to App. B.3 and C.2. We instantiate all hash functions to provide 128 bits of security. All benchmarks were obtained on Linux using an Intel Xeon E5-2699 v4 CPU (2.2 GHz, turboboost up to 3.6 GHz) using stable Rust version 1.59 and the `target-cpu=native` flag. Each of the individual benchmarks has only access to one thread.

Remark 1. The Pedersen hash function [69, Sec. 5.4.1.7] is also relevant for ZK proof systems. However, since it is not preimage-resistant, uses hardness assumptions vulnerable to quantum computers, and requires more R1CS constraints than Poseidon and *Rescue* [7], we do not consider it in our benchmarks.

Remark 2. As is often the case in symmetric cryptography, it is difficult to consider versions with equal security margins in the comparisons. For example, adding the same number of rounds or nonlinear functions to two designs with different structures may affect both the security and the performance of the two designs differently. Therefore, we focus on the original specifications given by the designers, noting that the security margins may vary between the different constructions.

6.1 Plain Performance

In Table 2, we compare the plain performance of the permutations when instantiated with the scalar fields of the commonly used BLS12 and BN254 elliptic

Table 2: Plain performance of different permutations in Rust (measured in μs).

Permutation	State size t								
	3	4	5	8	9	12	16	20	24
				BLS12 ($d = 5$)					
Griffin	113.97	105.45	–	89.32	–	93.76	98.19	**103.78**	**107.96**
GMiMC$_{\text{erf}}$	20.14	20.70	**21.65**	**26.07**	**26.44**	**37.72**	**65.94**	107.45	167.75
Neptune	–	**19.54**	–	30.87	–	60.20	93.14	128.95	171.97
Poseidon	**18.61**	24.36	30.60	55.52	63.10	95.84	149.61	212.85	286.75
Rescue-Prime	412.91	434.13	451.49	645.79	739.24	1005.20	1363.40	1759.10	2147.80
Grendel	822.54	959.92	1001.30	1154.60	1215.60	1283.30	1425.30	1411.90	1459.20
				BN254 ($d = 5$)					
Griffin	106.90	99.33	–	84.97	–	88.21	92.08	**96.85**	**100.10**
GMiMC$_{\text{erf}}$	18.67	19.34	**20.08**	**23.44**	**24.63**	**34.05**	**69.49**	107.82	156.35
Neptune	–	**17.38**	–	29.83	–	58.41	89.89	125.87	166.11
Poseidon	**17.56**	23.23	29.37	51.06	58.96	89.20	139.68	196.64	267.80
Rescue-Prime	379.78	400.87	411.16	598.86	683.81	929.89	1275.50	1639.30	2006.10
Grendel	703.36	808.78	849.89	994.20	1034.30	1094.20	1213.30	1196.00	1253.50

curves.[17] As the table shows, the fastest permutation for $t \leq 16$ is GMiMC$_{\text{erf}}$. However, as we show later, it has the worst performance when used with SNARKs and STARKs. *Rescue*-Prime and *Grendel* have the worst plain performance due to having either t high-degree $x^{1/d}$ or Legendre symbol evaluations per round. Griffin also uses $x^{1/d}$, but only once per round. Thus, Griffin scales significantly better with larger t than the other designs. Indeed, for small t Griffin is slower than Poseidon and Neptune, but the differences get smaller for larger t, until Griffin is faster than Poseidon and Neptune if $t \geq 16$.

6.2 R1CS-Based SNARKs with Griffin

Here we evaluate the efficiency of Griffin when used in R1CS-based zk-SNARKs and compare it to its competitors by giving the number of R1CS constraints, as well as concrete runtimes for proving knowledge of preimages and membership witnesses for Merkle tree accumulators. Our implementation is written in Rust using the bellman_ce library[18] for creating Groth16 [43] proofs.[19]

Describing Griffin as a R1CS system is straightforward. The first two words of the nonlinear layer (i.e., y_0, y_1 in Eq. (5)) each require $\lfloor \log_2(d) \rfloor + \text{hw}(d) - 1$ constraints (2 constraints if $d = 3$, 3 constraints if $d = 5$). The squaring of each $L(\cdot)$ and each word of the remaining state require an additional constraint each. Since the linear layers can be incorporated into the constraints of the subsequent nonlinear layers (see Section 2.3), the total number of R1CS constraints for

[17] p_{BLS381} = 0x73eda753299d7d483339d80809a1d80553bda402fffe5bfeffffffff00000001,
p_{BN254} = 0x30644e72e131a029b85045b68181585d2833e84879b9709143e1f593f0000001.

[18] https://docs.rs/bellman_ce/0.3.5/bellman_ce/

[19] bellman_ce is used in the Zcash protocol: https://z.cash/technology/zksnarks/

Table 3: Bellman_ce performance of various hash functions (one permutation per call) for different state sizes t. Performance numbers are for proving the knowledge of preimages of hashes (`Perm`) and for proving the membership of a Merkle tree accumulator with 2^{24} elements (`MT`). Proving times are given in ms.

Hash		State size t (`MT` arity)											
		3 (2 : 1)		4 (2 : 1)		5 (4 : 1)		8 (4 : 1)		9 (8 : 1)		12 (8 : 1)	
		Prove	R1CS	Prove	R1CS	Prove	R1CS	Prove	R1CS	Prove	R1CS	Prove	R1CS
BLS12 ($d = 5$)													
GRIFFIN	`Perm`	**39.08**	**96**	**42.46**	**110**	–	–	**60.54**	**162**	–	–	**82.29**	**234**
	`MT`	**451.88**	**2637**	**495.74**	**2712**	–	–	**422.50**	**2136**	–	–	**424.07**	**2192**
NEPTUNE	`Perm`	–	–	71.53	228	–	–	95.54	264	–	–	120.55	306
	`MT`	–	–	969.71	5544	–	–	728.11	3360	–	–	747.22	2768
POSEIDON	`Perm`	75.31	240	88.29	264	93.43	288	108.40	363	114.35	387	132.54	459
	`MT`	1013.70	5832	1093.00	6408	654.85	3648	877.17	4548	630.17	3416	719.52	3992
Rescue-Prime	`Perm`	75.12	252	77.55	264	78.01	270	96.71	384	106.61	432	138.93	576
	`MT`	851.56	6120	872.26	6408	512.97	3432	726.84	4800	541.93	3776	737.59	4928
GMiMC$_{\text{erf}}$	`Perm`	173.71	678	176.91	684	180.20	690	190.01	708	193.76	714	253.53	942
	`MT`	3060.80	16344	2842.40	16488	1537.40	8472	1640.80	8688	1118.20	6032	1535.60	7856
Grendel	`Perm`	148.76	870	160.50	1000	191.33	1050	216.12	1200	223.85	1260	231.53	1320
	`MT`	2297.70	20952	2535.40	24072	1403.20	12792	1505.40	14592	1249.70	10400	1268.00	10880
BN254 ($d = 5$)													
GRIFFIN	`Perm`	**22.48**	**96**	**24.24**	**110**	–	–	**35.08**	**162**	–	–	**48.05**	**234**
	`MT`	**266.77**	**2637**	**294.07**	**2712**	–	–	**251.90**	**2136**	–	–	**257.31**	**2192**
NEPTUNE	`Perm`	–	–	42.75	228	–	–	61.30	264	–	–	86.31	306
	`MT`	–	–	621.76	5544	–	–	512.69	3360	–	–	569.48	2768
POSEIDON	`Perm`	43.47	240	51.58	264	54.35	288	64.46	363	70.82	387	79.86	459
	`MT`	604.91	5832	656.77	6408	391.55	3648	542.02	4548	385.03	3416	446.87	3992
Rescue-Prime	`Perm`	43.54	252	44.36	264	44.87	270	54.52	384	61.51	432	80.97	576
	`MT`	510.03	6120	520.01	6408	306.12	3432	436.83	4800	323.67	3776	445.66	4928
GMiMC$_{\text{erf}}$	`Perm`	101.81	678	104.95	684	107.36	690	115.99	708	119.02	714	164.38	942
	`MT`	2148.60	16344	1791.20	16488	952.34	8472	1049.80	8688	717.61	6032	1046.70	7856
Grendel	`Perm`	86.85	870	94.12	1000	113.33	1050	127.31	1200	131.54	1260	135.80	1320
	`MT`	1401.20	20952	1523.60	24072	854.51	12792	920.43	14592	759.53	10400	776.86	10880

describing the whole GRIFFIN-π permutation is

$$(2 \cdot \lfloor \log_2(d) \rfloor + 2 \cdot \mathrm{hw}(d) + 2 \cdot t - 6) \cdot R\,,$$

i.e., $2 \cdot R \cdot t$ R1CS constraints if $d = 3$ and $R \cdot (2 \cdot t + 2)$ ones if $d = 5$. In Table 3 we compare the number of R1CS constraints, as well as the concrete runtime to create a ZK proof using the bellman_ce library when instantiated with two different elliptic curves (BLS12-381, BN254). We compare the performance of the hash functions when *(1)* proving knowledge of a preimage of a specific hash, and *(2)* when proving membership of a Merkle tree accumulator with 2^{24} elements. For the latter case, we construct Merkle trees with arity $(x : 1)$ such that x is the largest power of two smaller than t. In all cases, verifying the created ZK proof took < 4 ms which is why we do not explicitly list this runtime in Table 3.

Table 3 shows that Griffin requires the smallest number of R1CS constraints to prove knowledge of a preimage of a hash for several state sizes t. However, since Griffin is defined for state sizes $t = 3$ or $t = 4t'$ it cannot be instantiated with $t = 5$ or $t = 9$ (the smallest state sizes for Merkle trees with arities 4 and 8, respectively). Thus, to create trees of this arity, Griffin requires a larger state size (e.g., more words in the inner part of the sponge) compared to its competitors. As shown in Table 3, this still results in significantly fewer R1CS constraints and smaller proving times compared to the other hash functions. Concretely, using Griffin results in nearly half of the required constraints compared to Poseidon and *Rescue* and two third of the constraints compared to the recently proposed Neptune. Consequently, Griffin has the fastest proving times which also lead to the fastest membership proving times when used as a hash function in Merkle tree accumulators. Hence, we recommend to use Griffin and Merkle tree arities 8 : 1 for the membership witness use cases, since it provides the fastest proofs in combination with a decent plain performance (8 times faster than *Rescue*-Prime, and nearly as fast as Poseidon), which heavily influences the initial Merkle tree accumulation runtime.

Acknowledgments. The authors thank the Eurocrypt'22 reviewers for pointing out an attack on the previous (unpublished) version of Griffin. We took it into account in order to design the version of Griffin presented in this paper. We also thank them for the suggestion of the name `Horst`, and for pointing out the similarity between `Horst` and the S-box used in Streebog. Lorenzo Grassi is supported by the European Research Council under the ERC advanced grant agreement under grant ERC-2017-ADG Nr. 788980 ESCADA. Roman Walch is supported by the "DDAI" COMET Module within the COMET – Competence Centers for Excellent Technologies Programme, funded by the Austrian Federal Ministry for Transport, Innovation and Technology (bmvit), the Austrian Federal Ministry for Digital and Economic Affairs (bmdw), the Austrian Research Promotion Agency (FFG), the province of Styria (SFG) and partners from industry and academia. The COMET Programme is managed by FFG. Yonglin Hao is supported by National Natural Science Foundation of China (Grant No. 62002024), National Key Research and Development Program of China (No. 2018YFA0306404). Qingju Wang is funded by Huawei Technologies Co., Ltd (Agreement No.: YBN2020035184).

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: *ASIACRYPT 2019*. Vol. 11923. LNCS. 2019, pp. 371–397.

[2] M. R. Albrecht, L. Grassi, L. Perrin, S. Ramacher, C. Rechberger, D. Rotaru, A. Roy, and M. Schofnegger. "Feistel Structures for MPC, and More". In: *ESORICS 2019*. Vol. 11736. LNCS. 2019, pp. 151–171.

[3] M. R. Albrecht, L. Grassi, C. Rechberger, A. Roy, and T. Tiessen. "MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity". In: *ASIACRYPT 2016*. Vol. 10031. LNCS. 2016, pp. 191–219.

[4] A. Aly, T. Ashur, Eli Ben-Sasson, S. Dhooghe, and A. Szepieniec. "Design of Symmetric-Key Primitives for Advanced Cryptographic Protocols". In: *IACR Trans. Symmetric Cryptol.* 2020.3 (2020), pp. 1–45.

[5] S. Ames, C. Hazay, Y. Ishai, and M. Venkitasubramaniam. "Ligero: Lightweight Sublinear Arguments Without a Trusted Setup". In: *CCS*. ACM, 2017, pp. 2087–2104.

[6] T. Ashur and S. Dhooghe. "Prelude to Marvellous (With the Designers' Commentary, Two Bonus Tracks, and a Foretold Prophecy)". In: *IACR Cryptol. ePrint Arch.* (2020), p. 568.

[7] M. Barbara, L. Grassi, D. Khovratovich, R. Lüftenegger, C. Rechberger, M. Schofnegger, and R. Walch. "Reinforced Concrete: Fast Hash Function for Zero Knowledge Proofs and Verifiable Computation". In: *IACR Cryptol. ePrint Arch.* 2021 (2021), p. 1038.

[8] M. Bardet, J. Faugere, B. Salvy, and B. Yang. "Asymptotic behaviour of the index of regularity of quadratic semi-regular polynomial systems". In: *The Effective Methods in Algebraic Geometry Conference (MEGA)*. 2005, pp. 1–14.

[9] M. Bardet, J.-C. Faugére, B. Salvy, and B.-Y. Yang. "Asymptotic behaviour of the degree of regularity of semi-regular polynomial systems". In: *Proc. of MEGA*. Vol. 5. 2005.

[10] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. *Scalable, transparent, and post-quantum secure computational integrity.* Cryptology ePrint Archive, Report 2018/46. 2018.

[11] E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. P. Ward. "Aurora: Transparent Succinct Arguments for R1CS". In: *EUROCRYPT 2019*. Vol. 11476. LNCS. 2019, pp. 103–128.

[12] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. *Sponge functions.* In: Ecrypt Hash Workshop 2007, http://www.csrc.nist.gov/pki/HashWorkshop/PublicComments/2007_May.html. 2007.

[13] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. "On the Indifferentiability of the Sponge Construction". In: *EUROCRYPT 2008*. Vol. 4965. LNCS. 2008, pp. 181–197.

[14] T. Beyne, A. Canteaut, I. Dinur, M. Eichlseder, G. Leander, G. Leurent, M. Naya-Plasencia, L. Perrin, Y. Sasaki, Y. Todo, and F. Wiemer. "Out of Oddity - New Cryptanalytic Techniques Against Symmetric Primitives Optimized for Integrity Proof Systems". In: *CRYPTO 2020*. Vol. 12172. LNCS. 2020, pp. 299–328.

[15] E. Biham and A. Shamir. "Differential Cryptanalysis of DES-like Cryptosystems". In: *CRYPTO 1990*. Vol. 537. LNCS. 1990, pp. 2–21.

[16] A. Biryukov, L. Perrin, and A. Udovenko. "Reverse-Engineering the S-Box of Streebog, Kuznyechik and STRIBOBr1". In: *EUROCRYPT 2016*. Vol. 9665. LNCS. 2016, pp. 372–402.

[17] N. Bordes, J. Daemen, D. Kuijsters, and G. V. Assche. "Thinking Outside the Superbox". In: *CRYPTO (3)*. Vol. 12827. LNCS. 2021, pp. 337–367.

[18] C. Boura, A. Canteaut, and C. D. Cannière. "Higher-Order Differential Properties of Keccak and *Luffa*". In: *FSE 2011*. Vol. 6733. LNCS. 2011, pp. 252–269.

[19] B. Buchberger. “Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal”. PhD thesis. University of Innsbruck, 1965.

[20] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. “Bulletproofs: Short Proofs for Confidential Transactions and More”. In: *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2018, pp. 315–334.

[21] C. Cid, L. Grassi, A. Gunsing, R. Lüftenegger, C. Rechberger, and M. Schofnegger. *Influence of the Linear Layer on the Growth of the Algebraic Degree in SP-Networks*. Cryptology ePrint Archive, Report 2020/536. https://ia.cr/2020/536 – accepted at FSE/ToSC 2022. 2020.

[22] J.-S. Coron, J. Patarin, and Y. Seurin. “The Random Oracle Model and the Ideal Cipher Model Are Equivalent”. In: *CRYPTO 2008*. Vol. 5157. LNCS. 2008, pp. 1–20.

[23] D. A. Cox, J. Little, and D. O’Shea. *Ideals, Varieties, and Algorithms – An Introduction to Computational Algebraic Geometry and Commutative Algebra*. 2nd ed. Undergraduate Texts in Mathematics. Springer, 1997.

[24] D. Dachman-Soled, J. Katz, and A. Thiruvengadam. “10-Round Feistel is Indifferentiable from an Ideal Cipher”. In: *EUROCRYPT 2016*. Vol. 9666. LNCS. 2016, pp. 649–678.

[25] J. Daemen and V. Rijmen. “The Wide Trail Design Strategy”. In: *Cryptography and Coding - IMA International Conference 2001*. Vol. 2260. LNCS. 2001, pp. 222–238.

[26] J. Daemen and V. Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.

[27] J. Daemen and V. Rijmen. “Understanding Two-Round Differentials in AES”. In: *SCN 2006*. Vol. 4116. LNCS. 2006, pp. 78–94.

[28] Y. Dai and J. P. Steinberger. “Indifferentiability of 8-Round Feistel Networks”. In: *Advances in Cryptology - CRYPTO 2016*. Vol. 9814. LNCS. 2016, pp. 95–120.

[29] C. Dobraunig, L. Grassi, A. Guinet, and D. Kuijsters. “Ciminion: Symmetric Encryption Based on Toffoli-Gates over Large Finite Fields”. In: *EUROCRYPT 2021*. Vol. 12697. LNCS. 2021, pp. 3–34.

[30] C. Dobraunig, L. Grassi, L. Helminger, C. Rechberger, M. Schofnegger, and R. Walch. *Pasta: A Case for Hybrid Homomorphic Encryption*. Cryptology ePrint Archive, Report 2021/731. 2021.

[31] V. Dolmatov and A. Degtyarev. “GOST R 34.11-2012: Hash Function”. In: *RFC* 6986 (2013), pp. 1–40.

[32] M. Eichlseder, L. Grassi, R. Lüftenegger, M. Øygarden, C. Rechberger, M. Schofnegger, and Q. Wang. “An Algebraic Attack on Ciphers with Low-Degree Round Functions: Application to Full MiMC”. In: *ASIACRYPT 2020*. Vol. 12491. LNCS. 2020, pp. 477–506.

[33] Federal Agency on Technical Regulation and Metrology. *GOST R 34.12-2015: Block Cipher*. 2015.

[34] A. Gabizon, Z. J. Williamson, and O. Ciobotaru. *PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge*. Cryptology ePrint Archive, Report 2019/953. 2019.

[35] L. Grassi. “Mixture Differential Cryptanalysis: a New Approach to Distinguishers and Attacks on round-reduced AES”. In: *IACR Trans. Symmetric Cryptol.* 2018.2 (2018), pp. 133–160.

[36] L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofnegger. “Poseidon: A New Hash Function for Zero-Knowledge Proof Systems”. In: *USENIX Security Symposium*. USENIX Association, 2021, pp. 519–535.

[37] L. Grassi, D. Khovratovich, S. Rønjom, and M. Schofnegger. “The Legendre Symbol and the Modulo-2 Operator in Symmetric Schemes over $\mathbb{F}_p^n$”. In: *IACR Trans. Symmetric Cryptol.* 2022.1 (2022). To appear.

[38] L. Grassi, R. Lüftenegger, C. Rechberger, D. Rotaru, and M. Schofnegger. “On a Generalization of Substitution-Permutation Networks: The HADES Design Strategy”. In: *EUROCRYPT 2020*. Vol. 12106. LNCS. 2020, pp. 674–704.

[39] L. Grassi, S. Onofri, M. Pedicini, and L. Sozzi. *Invertible Quadratic Non-Linear Layers for MPC-/FHE-/ZK-Friendly Schemes over $\mathbb{F}_p^n$*. Cryptology ePrint Archive, Report 2021/1695. https://ia.cr/2021/1695. 2021.

[40] L. Grassi, M. Øygarden, M. Schofnegger, and R. Walch. *From Farfalle to Megafono via Ciminion: The PRF Hydra for MPC Applications*. Cryptology ePrint Archive, Report 2022/342. https://ia.cr/2022/342. 2022.

[41] L. Grassi, C. Rechberger, and S. Rønjom. “A New Structural-Differential Property of 5-Round AES”. In: *EUROCRYPT 2017*. Vol. 10211. LNCS. 2017, pp. 289–317.

[42] L. Grassi, C. Rechberger, D. Rotaru, P. Scholl, and N. P. Smart. “MPC-Friendly Symmetric Key Primitives”. In: *CCS*. ACM, 2016, pp. 430–443.

[43] J. Groth. “On the Size of Pairing-Based Non-interactive Arguments”. In: *EUROCRYPT 2016*. Vol. 9666. LNCS. 2016, pp. 305–326.

[44] V. T. Hoang and P. Rogaway. “On Generalized Feistel Networks”. In: *CRYPTO*. Vol. 6223. LNCS. 2010, pp. 613–630.

[45] T. Holenstein, R. Künzler, and S. Tessaro. “The equivalence of the random oracle model and the ideal cipher model, revisited”. In: *STOC 2011*. ACM, 2011, pp. 89–98.

[46] T. Jakobsen and L. R. Knudsen. “The Interpolation Attack on Block Ciphers”. In: *FSE 1997*. Vol. 1267. LNCS. 1997, pp. 28–40.

[47] N. Keller and A. Rosemarin. “Mind the Middle Layer: The HADES Design Strategy Revisited”. In: *EUROCRYPT 2021*. Vol. 12697. LNCS. 2021, pp. 35–63.

[48] L. R. Knudsen. “Truncated and Higher Order Differentials”. In: *FSE 1994*. Vol. 1008. LNCS. 1994, pp. 196–211.

[49] X. Lai and J. L. Massey. “A Proposal for a New Block Encryption Standard”. In: *EUROCRYPT 1990*. Vol. 473. LNCS. 1990, pp. 389–404.

[50] M. Lamberger, F. Mendel, C. Rechberger, V. Rijmen, and M. Schläffer. “Rebound Distinguishers: Results on the Full Whirlpool Compression Function”. In: *ASIACRYPT 2009*. Vol. 5912. LNCS. 2009, pp. 126–143.

[51] G. Leander, M. A. Abdelraheem, H. AlKhzaimi, and E. Zenner. “A Cryptanalysis of PRINTcipher: The Invariant Subspace Attack”. In: *CRYPTO 2011*. Vol. 6841. LNCS. 2011, pp. 206–221.

[52] M. Luby and C. Rackoff. “How to Construct Pseudorandom Permutations from Pseudorandom Functions”. In: *SIAM J. Comput.* 17.2 (1988), pp. 373–386.

[53] M. Matsui. “Linear Cryptanalysis Method for DES Cipher”. In: *EUROCRYPT 1993*. Vol. 765. LNCS. 1993, pp. 386–397.

[54] K. Matusiewicz, M. Naya-Plasencia, I. Nikolic, Y. Sasaki, and M. Schläffer. “Rebound Attack on the Full Lane Compression Function”. In: *ASIACRYPT 2009*. Vol. 5912. LNCS. 2009, pp. 106–125.

[55] U. M. Maurer and K. Pietrzak. “The Security of Many-Round Luby-Rackoff Pseudo-Random Permutations”. In: *EUROCRYPT 2003*. Vol. 2656. LNCS. 2003, pp. 544–561.

[56] F. Mendel, C. Rechberger, M. Schläffer, and S. S. Thomsen. “The Rebound Attack: Cryptanalysis of Reduced Whirlpool and Grøstl”. In: *FSE 2009*. Vol. 5665. LNCS. 2009, pp. 260–276.

[57] A. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[58] R. Mollin and S. C. “On Permutation Polynomials Over Finite Fields”. In: *International Journal of Mathematics and Mathematical Sciences* 10 (Jan. 1987).

[59] National Institute of Standards and Technology. “SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions”. In: *Federal Information Processing Standards Publication (FIPS)* (202 2015). http://ws680.nist.gov/publication/get_pdf.cfm?pub_id=919061.

[60] K. Nyberg. “Generalized Feistel Networks”. In: *ASIACRYPT*. Vol. 1163. LNCS. 1996, pp. 91–104.

[61] J. Patarin. “About Feistel Schemes with Six (or More) Rounds”. In: *FSE 1998*. Vol. 1372. LNCS. 1998, pp. 103–121.

[62] J. Patarin. “Generic Attacks on Feistel Schemes”. In: *ASIACRYPT 2001*. Vol. 2248. LNCS. 2001, pp. 222–238.

[63] J. Patarin. “Luby-Rackoff: 7 Rounds Are Enough for $2^{n(1-\epsilon)}$ Security”. In: *CRYPTO 2003*. Vol. 2729. LNCS. 2003, pp. 513–529.

[64] J. Patarin. “Security of Random Feistel Schemes with 5 or More Rounds”. In: *CRYPTO 2004*. Vol. 3152. LNCS. 2004, pp. 106–122.

[65] V. Rijmen, J. Daemen, B. Preneel, A. Bosselaers, and E. D. Win. “The Cipher SHARK”. In: *FSE 1996*. Vol. 1039. LNCS. 1996, pp. 99–111.

[66] A. Szepieniec. “On the Use of the Legendre Symbol in Symmetric Cipher Design”. In: *IACR Cryptol. ePrint Arch.* (2021), p. 984.

[67] A. Szepieniec, T. Ashur, and S. Dhooghe. *Rescue-Prime: a Standard Specification (SoK)*. Cryptology ePrint Archive, Report 2020/1143. 2020.

[68] S. Vaudenay. “On the Lai-Massey Scheme”. In: *ASIACRYPT 1999*. Vol. 1716. LNCS. 1999, pp. 8–19.

[69] *ZCash protocol specification*. https://github.com/zcash/zips/blob/master/protocol/protocol.pdf. 2021.

[70] Y. Zheng, T. Matsumoto, and H. Imai. “On the Construction of Block Ciphers Provably Secure and Not Relying on Any Unproved Hypotheses”. In: *CRYPTO*. Vol. 435. LNCS. 1989, pp. 461–480.

Source Code

The source code used in our evaluations is publicly available at https://extgit.iaik.tugraz.at/krypto/zkfriendlyhashzoo/-/tree/master/bellman [19].

A Security of Horst Schemes

As we mention in Section 3.2, here we show that

- if $r \in \{1, 2\}$, there exists a distinguisher for (i) Feistel and (ii) Horst, and
- if $r \in \{3, 4, 5\}$, there exists a distinguisher for (i) Feistel and (ii) $\texttt{Horst}^{\times}$ (under the condition that $F^{(i)}$ are equal to zero).

We prove the second result by working over the commutative group $(\mathfrak{G}, *)$ and by considering a generic scheme of the form

$$(y, x) \mapsto (x, y * H(x)),$$

where $H^{(i)} : \mathfrak{G} \to \mathfrak{G}$ denotes the i-th round function for $i \in \{0, \ldots, r-1\}$. We also define $\zeta \in \mathfrak{G}$ s.t. $\zeta * x = x * \zeta = x$ for $x \in \mathfrak{G}$.

1 to 2 Rounds. First of all, 1 or 2 rounds of Horst and/or Feistel can always be distinguished from a PRP. Starting with a difference $(\delta, 0) \in \mathbb{F}_q^2$ we get an output difference of the form $(\delta, \delta') \in \mathbb{F}_q^2$ in the case of 2-round Feistel (for an unknown $\delta' \in \mathbb{F}_q$). In the case of Horst, consider three inputs of the form $(y_i, x) \in \mathbb{F}_q^2$ for $i \in \{0, 1, 2\}$ and the corresponding outputs $(z_i, w_i) \in \mathbb{F}_q^2$ for $i \in \{0, 1, 2\}$, where $z_i := y_i \cdot G^{(0)}(x) + F^{(0)}(x)$ and for unknown $w_0, w_1, w_2 \in \mathbb{F}_q$. Hence, in the case of 2-round Horst, we have

$$(y_2 - y_0) \cdot (z_1 - z_0) = (z_2 - z_0) \cdot (y_1 - y_0)$$

with probability 1, while this occurs with probability $1/q$ in the case of a PRP.

3 Rounds. In the case of Feistel/$\texttt{Horst}^{\times}$, we can consider inputs of the form $(\zeta, x_i) \in \mathfrak{G}^2$ for several $i \in \mathbb{N}$ and the corresponding outputs $(z_i, w_i) \in \mathfrak{G}^2$. We then have that $z_i = x_i * H^{(1)}(H^{(0)}(x_i))$. Since $H^{(0)}$ and $H^{(1)}$ are PRFs, the probability to have a collision of the form $z_i * x_j = z_j * x_i$ for $j \neq i$ is around $2/q$ in the case of Feistel/$\texttt{Horst}^{\times}$ and around $1/q$ in the case of a PRP. Indeed, note that $z_i * x_j = z_j * x_i$ can occur if either $H^{(0)}(x_i) = H^{(0)}(x_j)$ (which happens with a probability of $1/q$) or $H^{(0)}(x_i) \neq H^{(0)}(x_j)$ and $H^{(1)}(H^{(0)}(x_i)) = H^{(1)}(H^{(0)}(x_j))$ (which happens with a probability of $(1 - 1/q) \cdot 1/q \approx 1/q$). In the generic case of a Horst scheme in which $G^{(i)}$ and $F^{(i)}$ are unrelated, the problem to set up a distinguisher for 3 rounds is open for future research.

4 Rounds. The previous 3-round distinguisher can easily be extended to 4 rounds. Indeed, it is sufficient to start with inputs of the form $(y_i, \zeta) \in \mathfrak{G}^2$, and to reuse the previous distinguisher, by noting that the outputs of the first rounds are of the form $(\zeta, x_i) \in \mathfrak{G}^2$.

5 Rounds. Finally, it is possible to set up a distinguisher on 5-round Feistel/$\texttt{Horst}^{\times}$. Let $|\mathfrak{G}|$ be the cardinality of the group $\mathfrak{G}$. The idea is to consider $|\mathfrak{G}|^{7/4}$ inputs of the form $(x_i, y_i) \in \mathfrak{G}^2$, and the corresponding outputs $(z_i, w_i) \in \mathfrak{G}^2$ after 5 rounds. If there exist $\{(x_i, y_i), (z_i, w_i)\}_{i \in \{1,2,3,4\}}$ such that $x_1 \neq x_3$, $y_1 \neq y_2$ and

$$\begin{aligned} &y_1 = y_3, \qquad y_2 = y_4, \qquad x_1 * x_4 = x_2 * x_3, \qquad z_1 = z_3, \qquad z_2 = z_4, \\ &z_1 * y_2 = y_1 * z_2, \qquad w_1 * x_3 = x_1 * w_3, \qquad w_1 * w_4 = w_2 * w_3, \end{aligned}$$

we can conclude that the analyzed scheme is a 5-round Feistel/$\texttt{Horst}^{\times}$. Otherwise, the outputs have been generated by a PRF. This result can be proven by (easily) adapting the proof proposed in [25, Section 4].

B STARKs with Griffin

We first analyze the cost metric in STARKs, and then the performance of GRIFFIN compared to various competitors.

B.1 Algebraic Intermediate Representation (AIR)

zk-STARKs [4] require to translate the computational problem into an algebraic intermediate representation (AIR). The AIR consists of a sequence of machine states (the algebraic execution trace (AET)) and multivariate polynomials describing the transition between those states. The machine states consist of w registers each, and the sequence has a length of T machine states. Thus, the AET is a $T \times w$ matrix, where the i-th row describes the machine state at timestep i. With $d_{\max}$ being the maximum degree of all multivariate transition polynomials, the efficiency of the proof system (i.e., proof size, prover/verification time) depends on w, T, and $d_{\max}$, where smaller values lead to more efficient proofs. In the literature, several different performance metrics have been proposed to compare the efficiency of different AIRs, such as $w \cdot T \cdot d_{\max}$ in [2], $8 \cdot w \cdot T \cdot d_{\max} \cdot \log_2(w \cdot T)$ in [16], or $\left\lceil \frac{\log_2(|\mathbb{F}|)}{64} \right\rceil^2 \cdot (d_{\max} + w) \cdot T \cdot \log_2(T)$ in [5].

There exist many different ways to design an AIR from a given circuit. For arithmetic hash functions purely built with additions and multiplications, an AIR can be built as follows. *(1)* The AET consists of a machine with a state size of w, which is equal to the state size t of the hash function, and represents the state after each round. *(2)* The length T of the AET sequence equals the number of rounds r of the hash function. *(3)* The multivariate update polynomials correspond to the applied round function, which is why $d_{\max}$ corresponds to the maximum degree of the representation of the nonlinear layers.

One can also consider various tradeoffs, e.g., increasing the state size w of the AIR by adding intermediate variables to reduce the maximum degree $d_{\max}$ of the update polynomials. The designer must find an optimal AIR representation which minimizes the cost. In many cases, though, starting with the straightforward AIR described in this section will give the best result. Indeed, the designers of POSEIDON and *Rescue* propose such AIRs with slight modifications.[20]

Cost Metric. We approximate the cost metric to be in $\mathcal{O}(d_{\max} \cdot T)$, i.e., the number of rounds times the degree of the round function representation.

[20] POSEIDON uses heavy preprocessing to combine t partial rounds into one to reduce the effective number of partial rounds.

B.2 Relations Between SNARK and STARK Cost Metrics

Similar to e.g. HE or MPC use cases, the performance of the proof systems scales with the number of nonlinear operations. However, the metrics can differ significantly. While for HE it is important to minimize the multiplicative depth, for MPC it is crucial to minimize the total number of multiplications. In ZK proof systems, on the other hand, it is important to find an efficient *equivalent representation* which minimizes the degree and/or the number of multiplications. Thus, while having to compute $y = x^{1/d}$ for small d is inefficient in MPC due to the large number of multiplications, it is efficient in SNARKs/STARKs by switching the representation to $y^d = x$. Further, observe that the degree of a nonlinear relation differs from the number of multiplications. The relations $y_3 = x^3$ and $y_4 = x^4$ have a different degree (i.e., y_3 is more beneficial in STARKs), however, they require the same number of multiplications to compute, which make them equivalent in MPC and SNARKs. Similarly, the relations $y_5 = x^5$ and $y'_5 = \sum_{i=0}^{5} x^i$ have the same degree and are equally beneficial in STARKs, but y'_5 requires more multiplications, making it worse for MPC and SNARKs.

B.3 STARK Performance of Griffin

Here we analyze the efficiency of Griffin and its competitors in zk-STARKs by comparing the AIR representations of the different hash functions. For this purpose, we transform Griffin into an efficient AIR representation: We construct a straightforward AIR, where the machine size w equals the state size t and represents the state of Griffin after each round. Thus, the length of the AET sequence T corresponds to the number of rounds R. Describing the nonlinear layer as $y_0^d = x_0$, $y_1 = x_1^d$, $y_2 = x_2\left((\gamma_2 \cdot y_0 + y_1)^2 + \alpha_2(\gamma_2 \cdot y_0 + y_1) + \beta_2\right)$, and $y_i = x_i\left((\gamma_i \cdot y_0 + y_1 + x_{i-1})^2 + \alpha_i(\gamma_i \cdot y_0 + y_1 + x_{i-1}) + \beta_i\right)$ for $i \in \{3, \ldots, t-1\}$, one can observe that the maximum degree of the multivariate update polynomials is given by d, since per definition $d \geq 3$. As before, the linear layers can be included into the description of the nonlinear layers.

Compared to the AIR representation of Poseidon, *Rescue*/*Rescue*-Prime, and *Grendel*, the structure of the Griffin AIR constraints is simpler. Indeed, Poseidon requires heavy precomputations to combine its partial rounds for a more compact AIR representaiton, the AIR of *Rescue* needs to combine the two nonlinear layers per round with a meet-in-the-middle approach, and *Grendel* requires to prove the Legendre symbol via introducing additional witness variables. However, the AIR of Griffin is just a straightforward translation, and is therefore easier to use in practice.

In Table 4 we compare the AIRs of the six different permutations when instantiated with 128-bit security, a 256-bit prime field, four different state sizes t, and for the most common cases where $d = 3$ and $d = 5$. For some prime fields, *Grendel* can be instantiated with $d = 2$ which is why we also give numbers for this case. We give the AIR numbers by first listing the values for w, T, and $d_{\max}$, and then use their product as metric as in [2].

Table 4: Comparison of AIR cost for different permutations instantaited with a 256-bit prime field. The total AIR cost is given as the product of $w \cdot T \cdot d_{\max}$.

	$d = 2$					$d = 3$							$d = 5$						
	R	w	T	$d_{\max}$	AIR	R_F	R_P	R	w	T	$d_{\max}$	AIR	R_F	R_P	R	w	T	$d_{\max}$	AIR
$t = 3$:																			
Griffin	–	–	–	–	–	–	–	16	**3**	**16**	**3**	**144**	–	–	12	3	12	5	180
Rescue-Prime	–	–	–	–	–	–	–	18	3	18	3	162	–	–	14	3	14	5	210
Poseidon	–	–	–	–	–	8	83	–	3	36	3	324	8	56	–	3	27	5	405
GMiMC$_{\text{erf}}$	–	–	–	–	–	–	–	328	1	328	3	984	–	–	256	1	256	5	1280
Grendel	38	9	38	4	1368	–	–	33	9	33	4	1188	–	–	29	9	29	6	1566
$t = 4$:																			
Griffin	–	–	–	–	–	–	–	14	**4**	**14**	**3**	**168**	–	–	11	4	11	5	220
Rescue-Prime	–	–	–	–	–	–	–	14	**4**	**14**	**3**	**168**	–	–	11	4	11	5	220
Poseidon	–	–	–	–	–	8	84	–	4	29	3	348	8	56	–	4	22	5	440
Neptune	–	–	–	–	–	6	68	–	4	23	4	368	6	68	–	4	23	5	460
GMiMC$_{\text{erf}}$	–	–	–	–	–	–	–	330	1	330	3	990	–	–	228	1	288	5	1440
Grendel	31	12	31	4	1488	–	–	28	12	28	4	1344	–	–	25	12	25	6	1800
$t = 8$:																			
Griffin	–	–	–	–	–	–	–	11	8	11	3	264	–	–	9	8	9	5	360
Rescue-Prime	–	–	–	–	–	–	–	8	**8**	**8**	**3**	**192**	–	–	8	8	8	5	320
Neptune	–	–	–	–	–	6	72	–	8	15	4	480	6	72	–	8	15	5	600
Poseidon	–	–	–	–	–	8	84	–	8	19	3	456	8	57	–	8	16	5	640
GMiMC$_{\text{erf}}$	–	–	–	–	–	–	–	338	1	338	3	1014	–	–	236	1	236	5	1180
Grendel	17	24	17	4	1632	–	–	16	24	16	4	1536	–	–	15	24	15	6	2160
$t = 12$:																			
Griffin	–	–	–	–	–	–	–	10	12	10	3	360	–	–	9	12	9	5	540
Rescue-Prime	–	–	–	–	–	–	–	8	**12**	**8**	**3**	**288**	–	–	8	12	8	5	480
Poseidon	–	–	–	–	–	8	85	–	12	16	3	576	8	57	–	12	13	5	780
Neptune	–	–	–	–	–	6	78	–	12	13	4	624	6	78	–	12	13	5	780
GMiMC$_{\text{erf}}$	–	–	–	–	–	–	–	346	1	346	3	1038	–	–	314	1	314	5	1570
Grendel	12	36	12	4	1728	–	–	12	36	12	4	1728	–	–	11	36	11	6	2376

The optimized AIR representations of all hash functions, with the exception of *Grendel* and GMiMC$_{\text{erf}}$, use a machine state with a size w equal to the state size t. Moreover, the maximum degree $d_{\max}$ equals the degree of the nonlinear layers d. Consequently, the length of the AET sequence T, which is equal to the number of rounds in *Rescue* and Griffin and depends on the number of rounds and the state size t in Poseidon and Neptune, constitutes the performance difference between these four designs, with smaller T yielding a more efficient AIR. In any case, Poseidon and Neptune have the largest T, i.e., a worse AIR compared to Griffin and *Rescue*. The difference between the number of rounds of Griffin and *Rescue* depends on both t and d. Thus, there are cases, where Griffin has a better AIR (e.g., $t = 3$), and case where *Rescue* has fewer rounds (e.g., $t \in \{8, 12\}$). In all cases, though, Griffin has significantly better plain performance than *Rescue*, reducing the time to build the AET.

Even though *Grendel* can be instantiated with $d = 2$ in some prime fields, the resulting round numbers, in addition to $d_{\max} \geq 4$, lead to a significantly worse AIR compared to Griffin and *Rescue*. Further, even though $\text{GMiMC}_{\text{erf}}$ can be represented with $w = 1$ [5] in an AIR, the large number of rounds make it less efficient than the other designs.

C Plonk with Griffin

In this section we first analyze the cost metric in the Plonk proof system, and then the performance of Griffin compared to different competitors.

C.1 Plonk Arithmetization

The Plonk [14] proof system is a zk-SNARK proof system which does not use R1CS constraints. Its arithmetization is based on Plonk gates, more concretely, the constraints are of the form

$$q_{L_i} \cdot a_{L_i} + q_{R_i} \cdot a_{R_i} + q_{O_i} \cdot a_{O_i} + q_{M_i} \cdot (a_{L_i} a_{R_i}) + q_{C_i} = 0, \tag{9}$$

where the a values are again the witness variables and the q values describe a given constraint. Using this equation, one can either describe a 2-fan-in addition (setting $q_{M,i} = 0$) or a 2-fan-in multiplication (setting $q_{L,i} = q_{R,i} = 0$). Thus, to use the Plonk proof system one needs to describe the given circuit using 2-fan-in addition and multiplication gates. As a result, contrary to R1CS constraints, additions cannot be embedded into subsequent multiplication constraints anymore and require Plonk gates on their own.

Cost Metric. We measure the number of Plonk gates, i.e., the minimum number of 2-fan-in additions and multiplications of witness variables required to fully represent any (equivalent) relation between the preimage and the hash.

Remark 3. The plonk proof system is in general very flexible and can easily be modified to use constraints different to Eq. (9). Thus, some implementations of the Plonk system extend Eq. (9) to allow 3-fan-in addition gates which are beneficial in some use cases. For this implementations the cost metric changes accordingly, i.e., the cost is then the minimum number of 2-fan-in multiplications and 3-fan-in additions of witness variables required to fully represent any (equivalent) relation between the preimage and the hash.

C.2 Plonk Performance of Griffin

Describing Griffin as Plonk gates can be done as follows. Each affine layer requires $t \cdot (t-1)$ addition gates. The first two words in the nonlinear layer require $\lfloor \log_2(d) \rfloor + \text{hw}(d) - 1$ multiplication gates. Computing $L(\cdot)$ requires one addition gate for $i = 2$ and two gates for $i > 2$. Computing $z_i = L_i(\cdot)^2 + \alpha_i L_i(\cdot) + \beta_i$

Table 5: Number of Plonk gates to describe various hash functions when instantiated with a 256-bit prime field. Numbers are given for Plonk implementations using either 2-fan-in addition gates or 3-fan-in addition gates.

	State size t											
Hash	2-fan-in addition gates						3-fan-in addition gates					
	3	4	5	8	9	12	3	4	5	8	9	12
$d = 3$												
Griffin	**214**	**334**	–	**969**	–	**1882**	**163**	**260**	–	**626**	–	**1132**
Reinforced Concrete	372	–	–	–	–	–	270	–	–	–	–	–
Rescue-Prime	432	560	**720**	1152	**1440**	2496	324	448	**480**	768	**864**	1536
Poseidon	760	1336	2088	5448	6936	12638	487	968	1168	3240	3624	7058
Neptune	–	1069	–	4625	–	11457	–	766	–	2726	–	6354
Grendel	1485	1792	2040	2560	2835	3456	1386	1680	1800	2176	2295	2736
GMiMC$_{\text{erf}}$	1312	1650	1992	3042	3400	4498	984	1320	1328	2028	2040	2768
$d = 5$												
Griffin	**186**	**287**	–	**821**	–	**1725**	**147**	**228**	–	**536**	–	**1044**
Reinforced Concrete	378	–	–	–	–	–	276	–	–	–	–	–
Rescue-Prime	420	528	**630**	1280	**1584**	2688	336	440	**450**	896	1008	1728
Poseidon	624	1032	1568	4003	5067	9039	432	776	928	2443	2727	5139
Neptune	–	1137	–	4697	–	11535	–	834	–	2798	–	6432
Grendel	1392	1700	1890	2520	2772	3300	1305	1600	1680	2160	2268	2640
GMiMC$_{\text{erf}}$	1130	1368	1610	2360	2618	4396	904	1140	1150	1652	1666	2826

requires one gate, plus an additional multiplication gate for $y_i = x_i \cdot z_i$. Summing up, Griffin requires

$$(R+1) \cdot t \cdot (t-1) + R \cdot (2 \cdot \lfloor \log_2(d) \rfloor + 2 \cdot \mathrm{hw}(d) + 4t - 11)$$

Plonk gates, i.e., $(R+1) \cdot t \cdot (t-1) + R \cdot (4t-5)$ gates if $d = 3$ and $(R+1) \cdot t \cdot (t-1) + R \cdot (4t-3)$ gates if $d = 5$.

Remark 4. When 3-fan-in addition gates are available in the Plonk implementation (see Remark 3), then Griffin requires $(R+1) \cdot t \cdot \lceil (t-1)/2 \rceil + R \cdot (2 \cdot \lfloor \log_2(d) \rfloor + 2 \cdot \mathrm{hw}(d) + 3t - 8)$ Plonk gates.

In Table 5, we compare the efficiency of the different hash function when used in the Plonk proof system by comparing the number of Plonk gates required to represent one permutation. One can observe that Griffin always requires the smallest number of Plonk gates due to having a small number of multiplications (Section 6.2) and a small number of rounds implying a small number of linear layers. We also compare Griffin to Reinforced Concrete [3], a hash function with a fixed state size $t = 3$ introducing novel techniques to use lookup tables in $\mathbb{F}_p$ designs. These lookup tables lead to fast plain performances, but potentially also introduce the risk of side-channel attacks. Further, they prevent Reinforced Concrete from (efficiently) being used in R1CS-based SNARKs or AIR-based STARKs. It is, however, usable and *specifically designed* for Plookup [13], an extension to Plonk allowing lookup tables. Interestingly though, Griffin requires fewer Plonk gates than Reinforced Concrete when using Plonk with the Plookup extension.

D Proofs – Differential Cryptanalysis

D.1 Maximum Differential Probability of S

Proof of Lemma 3 – Maximum Differential Probability of $\boldsymbol{x \mapsto x^{1/d}}$. First we prove that $\text{DP}_{\max}(x \mapsto x^d) = \text{DP}_{\max}(x \mapsto x^{1/d})$. Given the input and output differences δ_I and δ_O, we want to analyze the maximal number of solutions x of $(x+\delta_I)^{1/d} - x^{1/d} = \delta_O$, or equivalently of

$$(x+\delta_I)^{1/d} = x^{1/d} + \delta_O.$$

Computing the power of d of both sides, we have

$$x + \delta_I = (x^{1/d} + \delta_O)^d \implies (x^{1/d} + \delta_O)^d - x = \delta_I.$$

By making use of the change of variable $y = x^{1/d}$ or $x = y^d$ in the above equation, we get

$$(y + \delta_O)^d - y^d = \delta_I.$$

Since every step is invertible, we have that the number of solutions of $(x+\delta_I)^{1/d} - x^{1/d} = \delta_O$ for $y = x^d$ corresponds to the number of solutions of $(y+\delta_O)^d - y^d = \delta_I$ for $y = x^{1/d}$. That is, $\text{DP}_{\max}(x \mapsto x^d) = \text{DP}_{\max}(x \mapsto x^{1/d})$ for each d such that $\gcd(d, p-1) = 1$.

We then prove that $\text{DP}_{\max}(x \mapsto x^{d'}) = \frac{d'-1}{p}$ for a generic d' such that $\gcd(d', p-1) = 1$. The equation $(x+\delta_I)^{d'} - x^{d'} = \delta_O$ equals to $\sum_{i=0}^{d'-1} \binom{d'}{i} \cdot x^i \cdot \delta_I^{d'-i} = \delta_O$. The maximal degree of the left-hand side with respect to x is $d'-1$, hence the maximal number of solutions x is $d'-1$, therefore $DP_{\max}(x \mapsto x^{d'}) = (d'-1)/p$. This completes the proof.

Proof of Lemma 4 – Maximum Differential Probability of $\boldsymbol{(x,\ell) \mapsto y = x \cdot (\ell^2 + \alpha\ell + \beta)}$. Given $\Delta_I = (\delta_x, \delta_\ell)$ and $\Delta_O = \delta_y$, we look for the number of solutions of

$$\begin{aligned}\delta_y &= (\delta_x + x) \cdot \left((\delta_\ell + \ell)^2 + \alpha(\delta_\ell + \ell) + \beta\right) - x \cdot (\ell^2 + \alpha\ell + \beta)\\ &= \delta_x \cdot \left(\delta_\ell^2 + \delta_\ell \cdot (2\ell + \alpha) + (\ell^2 + \alpha \cdot \ell + \beta)\right) + x \cdot \delta_\ell \cdot (\delta_\ell + (2\ell + \alpha)).\end{aligned}$$

We distinguish the cases $\delta = 0$ and $\delta \neq 0$.

- If $\delta_\ell = 0$ and $\delta_x \neq 0$, then

  $$\delta_x \cdot \left(\ell^2 + \alpha \cdot \ell + \beta\right) = \delta_y \implies \ell^2 + \alpha \cdot \ell + (\beta - \delta_y/\delta_x) = 0,$$

  which admits at most two different solutions in ℓ (independently of x) if $\delta_y \neq 0$, i.e., $\text{DP}_{\max}((\delta_\ell = 0, \delta_x) \mapsto \delta_y) \leq 2/p$. If $\delta_y = 0$, then no solution is possible (since $\ell^2 + \alpha \cdot \ell + \beta \neq 0$ for each $\ell \in \mathbb{F}_p$), thus $\text{DP}_{\max}((\delta_\ell = 0, \delta_x) \mapsto \delta_y = 0) = 0$.

- If $\delta_\ell \neq 0$ and $\delta_x = 0$, we have

$$x \cdot \delta_\ell \cdot (\delta_\ell + (2\ell + \alpha)) = \delta_y.$$

If $\delta_y = 0$, this equation admits $x = 0$ or $\ell = -(\alpha+\delta_\ell)/2$ as possible solutions, which means that $\mathrm{DP}_{\max}((\delta_\ell, \delta_x = 0) \mapsto \delta_y = 0) \leq (2p-1)/p^2 \leq 2/p$. Otherwise, if $\delta_y \neq 0$ and $\ell \neq -(\alpha + \delta_\ell)/2$,

$$x = \frac{\delta_y}{\delta_\ell \cdot (\delta_\ell + (2\ell + \alpha))}.$$

As a result, $\mathrm{DP}_{\max}((\delta_\ell, \delta_x = 0) \mapsto \delta_y) \leq (p-1)/p^2 \leq 1/p$.

- If $\delta_\ell \neq 0$ and $\delta_x \neq 0$, the solutions are given by

$$x = \frac{\delta_y - \delta_x \cdot \left(\delta_\ell^2 + \delta_\ell \cdot (2\ell + \alpha) + (\ell^2 + \alpha \cdot \ell + \beta)\right)}{\delta_\ell \cdot (\delta_\ell + (2\ell + \alpha))}$$

if $\ell \neq -(\alpha + \delta_\ell)/2$. Hence, again $\mathrm{DP}_{\max}((\delta_x, \delta_\ell) \mapsto \delta_y) \leq (p-1)/p^2 \leq 1/p$. This result also holds for $\delta_y = 0$.

D.2 Branch Number of M

Practical Tests for the Branch Number of M. We practically test the branch number of $M = M^{''} \times M^{'}$ with $M^{'} = \mathrm{diag}(M_0, M_1, \ldots, M_{t'-1})$, where $M_i = \mathrm{circ}(3, 2, 1, 1)$ and $M^{''} = \mathrm{circ}(2I, I, \ldots, I)$ over $\mathbb{F}_p$, for $p = 11, 13, \ldots, 31$ and $p \approx 2^{80}$. The results are given in Table 6. We observe that for our applications with large p (e.g., $p \geq 2^{32}$), and $8 \leq t \leq 24$, the branch number of M is $t' + 4$, and for $t > 24$ the branch number can be smaller. Next, we prove that the branch number of M is $t' + 4$ for $2 \leq t' \leq 6$.

Table 6: Branch numbers of M with $t = 4t' \geq 8$ over $\mathbb{F}_p$ found in our tests.

t	t'	$\log_2 p$							
		11	13	17	19	23	29	31	$\approx 2^{80}$
8	2	5	6	5	6	6	6	6	6
12	3	6	7	6	7	7	7	7	7
16	4	7	8	7	8	8	8	8	8
20	5	8	9	8	9	9	9	9	9
24	6	8	10	8	10	10	10	10	10

Proof of the Branch Number of M. The quality of a linear diffusion layer can be reflected by its branch number. For a vector $\vec{a}$ over the field $\mathbb{F}_p$, its hamming weight $\mathrm{hw}(\vec{a})$ is defined in the following.

Definition 4. *The Hamming weight of a vector, denoted as* $\mathrm{hw}(\cdot)$*, is defined as the number of nonzero elements.*

For an arbitrary matrix $M \in \mathbb{F}_p^{n\times m}$, the branch number $\#b$ of M is

$$\#b := \min_{\vec{a}\in\mathbb{F}_p^m\setminus\{\vec{0}\}} \{\mathrm{hw}(\vec{a}) + \mathrm{hw}(M\vec{a})\}\,. \tag{10}$$

According to Section 4.2, for $t = 8, 12, \ldots, 24$, we have $M \in \mathbb{F}_p^{t\times t}$ and $t' = t/4 \in \{2, \ldots, 6\}$. Denote the column vectors of I as $\vec{e}_0, \ldots, \vec{e}_3$, where $I \in \mathbb{F}_p^{4\times 4}$ is the 4×4 identity matrix, we have $I = (\vec{e}_0, \ldots, \vec{e}_3)$. We also define a 4×4 circulant matrix $A = \mathrm{circ}(3, 2, 1, 1)$ which is an MDS matrix having branch number 5. Then, the $t \times t$ matrix $M \in \mathbb{F}_p^{t\times t}$ can be represented as

$$M = \begin{pmatrix} 2A & A & \ldots & A \\ A & 2A & \ldots & A \\ \vdots & & \ddots & \vdots \\ A & A & \ldots & 2A \end{pmatrix}.$$

The column vectors of A are denoted as $\vec{h}_0, \ldots, \vec{h}_3$ such that $A = (\vec{h}_0, \ldots, \vec{h}_3)$. With the column vectors

$$\vec{a} = (a_{0,0}, \ldots, a_{0,3}, a_{1,0}, \ldots, a_{1,3}, \ldots, a_{t'-1,0}, \ldots, a_{t'-1,3})^T, \tag{11}$$

$$\vec{b} = (b_{0,0}, \ldots, b_{0,3}, b_{1,0}, \ldots, b_{1,3}, \ldots, b_{t'-1,0}, \ldots, b_{t'-1,3})^T \tag{12}$$

$\in \mathbb{F}_p^{4t'}$, according to the definition of the branch number in Eq. (10), there must be vectors $\vec{a}, \vec{b} \in \mathbb{F}_p^t$ satisfying

$$M\vec{a} = \vec{b}, \tag{13}$$

$$\mathrm{hw}(\vec{a}) + \mathrm{hw}(\vec{b}) = \#b. \tag{14}$$

In the remainder of this part, without specific instructions, we constantly use $\vec{a}, \vec{b}$ to represent the vectors satisfying Eq. (13) and Eq. (14) simultaneously so as to prove $\#b = \mathrm{hw}(\vec{a}) + \mathrm{hw}(\vec{b}) = t' + 4$ when $t' = 2, \ldots, 6$.

Inequality: $\#b \le t' + 4$. For arbitrary t', we can easily prove that $\#b \le t' + 4$. Indeed, it is sufficient to set $a_{0,0} = 1 + \frac{t'-1}{t'}$, $a_{1,0} = \cdots = a_{t'-1,0} = \frac{-1}{t'}$, $(b_{0,0}, \ldots, b_{0,3}) = (-3, -1, -1, -2)$ and other entries of $\vec{a}, \vec{b}$ equal to 0 so as to satisfy Eq. (13). Therefore, we have $\#b \le t' + 4$.

Equality: $\#b = t' + 4$. For $j = 0, \ldots, 3$, let us define the summation s_j and column vectors $\vec{a}_j, \vec{a}'_j$ as

$$s_j = \sum_{i=0}^{t'-1} a_{i,j}, \tag{15}$$

$$\vec{a}_j = (a_{0,j}, \ldots, a_{t'-1,j})^T, \tag{16}$$

$$\vec{a}'_j = (a_{0,j} + s_j, \ldots, a_{t'-1,j} + s_j)^T. \tag{17}$$

We prove the following lemma.

Lemma 5. *For $\vec{a}_j, \vec{a}'_j$ ($j = 0, \ldots, 3$) in Eq. (16) and Eq. (17), when* $\mathrm{hw}(\vec{a}'_j) \leq \frac{t'}{2}$*, there is* $\mathrm{hw}(\vec{a}_j) \geq \mathrm{hw}(\vec{a}'_j)$*.*

Proof. Without loss of generality, we let $a_{i,j} + s_j = 0$ for $i = \lfloor \frac{t'}{2} \rfloor, \ldots, t'-1$, and $(a_{0,j} + s_j, \ldots, a_{\lfloor \frac{t'}{2} \rfloor -1,j} + s_j) \neq \vec{0}$.

- If $s_j = 0$, clearly $\vec{a}_j = \vec{a}'_j$ and $\mathrm{hw}(\vec{a}_j) = \mathrm{hw}(\vec{a}'_j)$.
- If $s_j \neq 0$, we have $a_{i,j} = -s_j \neq 0$ for $i = \lfloor \frac{t'}{2} \rfloor, \ldots, t'-1$. Therefore, $\mathrm{hw}(\vec{a}_j) \geq \lfloor \frac{t'}{2} \rfloor + 1 \geq \frac{t'}{2} \geq \mathrm{hw}(\vec{a}'_j)$. □

Note that Eq. (13) can be rewritten as

$$\begin{pmatrix} a_{0,0}+s_0 & a_{0,1}+s_1 & a_{0,2}+s_2 & a_{0,3}+s_3 \\ a_{1,0}+s_0 & a_{1,1}+s_1 & a_{1,2}+s_2 & a_{1,3}+s_3 \\ \vdots & \vdots & \vdots & \vdots \\ a_{t'-1,0}+s_0 & a_{t'-1,1}+s_1 & a_{t'-1,2}+s_2 & a_{t'-1,3}+s_3 \end{pmatrix} \times \begin{pmatrix} \vec{h}_0 \\ \vec{h}_1 \\ \vec{h}_2 \\ \vec{h}_3 \end{pmatrix}$$

$$= \begin{pmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ b_{0,0} & b_{0,1} & b_{1,2} & b_{1,3} \\ \vdots & \vdots & \vdots & \vdots \\ b_{t'-1,0} & b_{t'-1,1} & b_{t'-1,2} & b_{t'-1,3} \end{pmatrix} \times \begin{pmatrix} \vec{e}_0 \\ \vec{e}_1 \\ \vec{e}_2 \\ \vec{e}_3 \end{pmatrix}.$$

For $j = 0, \ldots, t'-1$, we define $\vec{\alpha}_j = (a_{j,0}+s_0, \ldots, a_{j,3}+s_3)$ and $\vec{\beta}_j = (b_{j,0}, \ldots, b_{j,3})$. Since $A = (\vec{h}_0, \ldots, \vec{h}_3)$, $A\vec{\alpha}_j = \vec{\beta}_j$. Since the branch number of A is 5,

$$\mathrm{hw}(\vec{\alpha}_j) + \mathrm{hw}(\vec{\beta}_j) = \begin{cases} 0 \text{ if } \mathrm{hw}(\vec{\alpha}_j) = 0, \\ 5 \text{ otherwise,} \end{cases}$$

and combining the definition of $\vec{\beta}_j$ and $\vec{b}$ in App. D.2, we can deduce

$$\mathrm{hw}(\vec{b}) = \sum_{j=0}^{t'-1} \mathrm{hw}(\vec{\beta}_j). \tag{18}$$

Moreover, from Eq. (16) and App. D.2, $\vec{a}_0, \ldots, \vec{a}_3$ form a partition of $\vec{a}$ s.t.

$$\mathrm{hw}(\vec{a}) = \sum_{j=0}^{3} \mathrm{hw}(\vec{a}_j). \tag{19}$$

Lemma 6. *If there is only one $j = 0, \ldots, 3$ such that $\vec{\beta}_j \neq \vec{0}$, then* $\mathrm{hw}(\vec{\beta}_j) = 4$*.*

Proof. Without loss of generality, we let $\vec{\beta}_0 \neq \vec{0}$. In this situation, for arbitrary $j = 0, \ldots, 3$ and $a_{0,j} + s_j \neq 0$, we can prove that $a_{i,j} \neq 0$ for all $i = 0, \ldots, t'-1$.

- If $s_j = 0$, there must be $a_{1,j} = \cdots = a_{t'-1,j} = 0$. But $a_{0,j} \neq 0$ contradicts the definition of s_j in Eq. (15).

– If $s_j \neq 0$, there must be $a_{1,j} = \cdots = a_{t'-1,j} = -s_j \neq 0$ and $a_{0,j} = t's_j \neq 0$ according to Eq. (15). Therefore, $\text{hw}(\vec{a})$ can be computed as $\text{hw}(\vec{a}) = t' \cdot \text{hw}(\vec{\alpha}_0)$. There is also $\text{hw}(\vec{\beta}_0) + \text{hw}(\vec{\alpha}_0) \geq 5$. Therefore, we have

$$\#b = \text{hw}(\vec{a}) + \text{hw}(\vec{b}) = t' \cdot \text{hw}(\vec{\alpha}_0) + \text{hw}(\vec{\beta}_0) \geq (t'-1) \cdot \text{hw}(\vec{\alpha}_0) + 5.$$

Since $\#b \leq t' + 4$, $\text{hw}(\vec{\alpha}_0) = 1$ is the only solution when $t' \geq 2$.

□

Theorem 1. *For $2 \leq t' \leq 6$, the branch number of M is $\#b = t' + 4$.*

Proof. Let η be the number of nonzero $\vec{\beta}_j$'s.

– If $1 \leq \frac{t'}{2} < \eta \leq t'$, there must be some $0 \leq j \leq 3$ such that $\text{hw}(\vec{a}_j) > 0$. Without loss of generality, let $\vec{\beta}_0, \ldots, \vec{\beta}_{\eta-1}$ be the η nonzero vectors.
  1. There is just one j such that $\vec{a}_j \neq 0$. There must be $\vec{a}'_j \neq 0$ and $\text{hw}(\vec{b}) = 4\eta$. If $s_j = 0$, we have $\text{hw}(\vec{a}_j) = \text{hw}(\vec{a}'_j) = \eta \geq t' - \eta + 1$ and that $\text{hw}(\vec{a}) + \text{hw}(\vec{b}) = \#b = 5\eta \geq 10$, contradicting $\#b \leq t' + 4$. If $s_j \neq 0$, we have $a_{\eta,j} = \cdots = a_{t'-1,j} = -s_j \neq 0$, $a_{0,j} \neq 0$. We have $\text{hw}(\vec{a}) + \text{hw}(\vec{b}) \geq t' + 3\eta + 1 > t' + 4$, contradicting $\#b \leq t' + 4$.
  2. There exist two j's such that $\vec{a}_j \neq 0$. Then there must be $\vec{a}'_j \neq 0$ and $\text{hw}(\vec{b}) \geq 3\eta$. Moreover, there is constantly $\text{hw}(\vec{a}_j) \geq t' - \eta + 1$. We have $\text{hw}(\vec{a}) + \text{hw}(\vec{b}) \geq 2t' + \eta + 2 \geq \frac{5t'}{2} + 2$, contradicting $\#b \leq t' + 4$.
  3. There exist three j's such that $\vec{a}_j \neq 0$. Then there must be $\vec{a}'_j \neq 0$ and $\text{hw}(\vec{b}) \geq 2\eta$. Moreover, there is constantly $\text{hw}(\vec{a}_j) \geq t' - \eta + 1$. We have $\text{hw}(\vec{a}) + \text{hw}(\vec{b}) \geq 3t' - \eta + 3 \geq 2t' + 3$, contradicting $\#b \leq t' + 4$.
  4. For $j = 0, 1, 2, 3$, $\vec{a}_j \neq 0$. There must be $\vec{a}'_j \neq 0$ and $\text{hw}(\vec{b}) \geq \eta$. Moreover, there is constantly $\text{hw}(\vec{a}_j) \geq t' - \eta + 1$. We have $\text{hw}(\vec{a}) + \text{hw}(\vec{b}) \geq 4t' - 3\eta + 4 \geq t' + 4$. Adding the constraint that $\#b = \text{hw}(\vec{a}) + \text{hw}(\vec{b}) \leq t' + 4$, the only solution is $\#b = t' + 4$ and it can only be acquired when $\eta = t'$, $\text{hw}(\vec{b}) = t'$, and $\text{hw}(\vec{a}_j) = 1$ are all satisfied.

  Therefore, for $\eta > \frac{t'}{2}$, the branch number can only be $\#b = t' + 4$.
– If $\eta \leq \frac{t'}{2}$, we have $\text{hw}(\vec{a}'_j) \leq \eta \leq \frac{t'}{2}$ for all $j = 0, \ldots, 3$. It can be deduced from Lemma 5 that $\text{hw}(\vec{a}_j) \geq \text{hw}(\vec{a}'_j)$ and according to Eq. (18) and Eq. (19), there is $\#b \geq 5\eta$. Adding the constraint that $\#b \leq t' + 4$, all possible solutions are the following.
  1. For $t' = 6$, we have $\eta = 1$ and $\eta = 2$. For $\eta = 1$ and $t' = 6$, with Lemma 6, we have $\text{hw}(\vec{a}) + \text{hw}(\vec{b}) = t' + 4 = 10$. For $\eta = 2$ and $t' = 6$, we have $\text{hw}(\vec{a}) + \text{hw}(\vec{b}) \geq 5\eta = 10$. Therefore, there is $\#b = t' + 4$ for $t' = 6$.
  2. For $2 \leq t' < 6$, there is only one satisfying value $\eta = 1$. So $\#b = t' + 4$ is the direct application of Lemma 6.

To sum up, $\#b = t' + 4$ for all $2 \leq t' \leq 6$. □

E Dedicated Automatic Tools for Differential Propagations of Griffin-π – Differential Attack and Rebound Attack

To verify the differential bound in Section 5.2, we set up a dedicated mixed integer linear programming (MILP) tool to look for upper bounds of the probabilities of the (classical) differential characteristics given in Section 5.2. Let us denote this upper bound for the probability by $\overline{\mathrm{DP}}$. With the tool, a MILP model $\mathcal{M}$ is constructed to find the truncated differential patterns with the highest probability. First, truncated word differences are represented in $\mathcal{M}$ as binary variables denoted as $\mathcal{M}.\texttt{var}$: 0 for zero differences (also known as *inactive*) and 1 for nonzero ones (also known as *active*). Then, the truncated differential propagation rules are described as the linear constraints denoted as $\mathcal{M}.\texttt{con}$. Finally, the objective function $\mathcal{M}.\texttt{obj}$ is set so as to upper bound the propagation probability. In the case of an aligned scheme [24, 27], $\mathcal{M}.\texttt{obj}$ aims at minimizing the summation of the binary variables corresponding to all S-box inputs, since the highest probability corresponds to the fewest active (having nonzero input and output difference) S-boxes. Hence, in the case of a strong-arranged scheme, $\overline{\mathrm{DP}}$ can be computed as $\overline{\mathrm{DP}} = (\mathrm{DP}_{\max}(S))^{\#a}$, where $\#a$ is the solution of $\mathcal{M}.\texttt{obj}$ and the $\mathrm{DP}_{\max}(S)$ corresponding to each S-box can be acquired with its differential distribution table (DDT).

In our weak-arranged design `Horst`, structural nonlinear operations in S take 2 input words and produce 1 output word. For each nonlinear operation in Griffin-π, we add an additional binary variable $\tau \in \mathcal{M}.\texttt{var}$ so as to describe the corresponding differential propagation probabilities. Hence, the objective function is simply minimizing the summation of all τ's as $\mathcal{M}.\texttt{obj} = \min \sum \tau$. With the solution $\mathcal{M}.\texttt{obj} = \#a$, $\overline{\mathrm{DP}}$ can be computed as $(d/p)^{\#a}$ where the $\mathrm{DP}_{\max}$ of the 3 kinds of nonlinear functions composing S are bounded as $\mathrm{DP}_{\max}(S) \leq \frac{d}{p}$ (see Lemma 3 and Lemma 4).

Constructing the MILP Models. The core of our dedicated tools is the construction of MILP model $\mathcal{M}$'s capturing the truncated differential propagation and the corresponding probabilities of Griffin-π. In the round function $\mathcal{F}_i(\cdot) = c^{(i)} + M \times S(\cdot)$, the round constant addition operation with $c^{(i)}$ does not change the difference propagation, so we omit it in the MILP model to simplify the differential propagation of the i-th round as follows:

$$\Delta X^i \xrightarrow{S} \Delta Y^i \xrightarrow{M} \Delta X^{i+1}.$$

The truncated differences ΔX^i and ΔY^i are represented in the model $\mathcal{M}$ as follows:

$$\Delta X^i = (\delta_{x_0^i}, \ldots, \delta_{x_{t-1}^i}) \xrightarrow{S} \Delta Y^i = (\delta_{y_0^i}, \ldots, \delta_{y_{t-1}^i}),$$

where $\delta \in \mathcal{M}.\texttt{var}$ are binary variables: $\delta = 1$ for nonzero difference and $\delta = 0$ for zero ones. When the context allows, we omit the index. In strong-aligned S-box based primitives, the summation of δ_x has strong correlation with the propagation probability [24, 27, 1, 26, 11]. However, for Griffin-π, the diffusion also takes place in the nonlinear layer where the first two words affect the other $(t-2)$ words. Therefore, in our dedicated tools, additional binary variables $\tau \in \mathcal{M}.\texttt{var}$ are introduced for tracking the probability.[21] Such τ's correspond to the nonlinear operation in S in differential attacks and the word condition in rebound attacks.

E.1 MILP Models for Differential Propagation of Griffin-π

R-round Griffin-π can resist differential attacks only if the corresponding probability upper bound $\overline{\mathrm{DP}}$ satisfies $\overline{\mathrm{DP}} \le 2^{-2\kappa}$. The computation of $\overline{\mathrm{DP}}$ correlates to the maximum differential propagation probability of the 3 kinds of nonlinear operations in the S layer. Such a probability is denoted as $\mathrm{DP}_{\max}$ and is analyzed in detail as follows.

The $\mathbf{DP_{max}}$ for $\boldsymbol{S}$. The nonlinear layer S of Griffin-π is composed of four nonlinear operations, namely $x_0 \mapsto {x_0}^{1/d} = y_0$, $x_1 \mapsto {x_1}^d = y_1$, $(\ell, x_i) \mapsto x_i \cdot (\ell^2 + \alpha\ell + \beta)$ for $2 \le i \le t-1$, where ℓ is a linear function of the first two words y_0 and y_1 for $i = 2$, and another function of three words y_0, y_1 and x_{i-1} for $3 \le i \le t-1$. Since the outputs of the first two words are independent of the others, the corresponding differential propagation probability is bounded by $\frac{d-1}{p}$ (see Lemma 3 and the proofs in App. D.1).

MILP Model for $\boldsymbol{S}$. We now detail the truncated differential propagation of the three nonlinear operations of S as well as the assignment of the corresponding τ variables. The nonlinear operations corresponding to the first two words, namely $y_0 = {x_0}^{1/d}$ and $y_1 = {x_1}^d$, are strong-aligned so there is constantly $\delta_{x_i} = \delta_{y_i} = \tau_i \in \mathcal{M}.\texttt{con}$ for $i = 0, 1$.

The nonlinear operation for computing the remaining y_i's $(i = 2, \ldots, t-1)$ can be divided into three kinds of steps: the two linear steps $\ell = L(y_0, y_1) = \gamma \cdot y_0 + y_1$ and $\ell' = L(y_0, y_1, x) = \ell + x$, where $\gamma \neq 0$, the nonlinear steps $y_i = S(\ell, x_i) = x_i \cdot (\ell^2 + \alpha\ell + \beta)$ for $i = 2$, and the nonlinear step $y_i = S(\ell', x_i) = x_i \cdot ({\ell'}^2 + \alpha\ell' + \beta)$ for $3 \le i \le t-1$, where $\alpha^2 - 4\beta \neq 0 \mod p$. The three steps are modeled separately.

The linear step $\ell'_i = \ell_i + x_{i-1}$ for $i = 3, \ldots, t-1$ follows exactly the truncated differential propagation rule of ordinary linear combinations, so the available values for $(\delta_{\ell_i}, \delta_{x_{i-1}}, \delta_{\ell'_i})$'s can be easily deduced in Table 7a and captured by

[21] Additional probability correlated binary variables are also used in [12] for describing the differential and linear propagation probabilities of the unaligned modular add operation.

Table 7: The truncated differential propagation rules for computing y_i where $i = 2, \ldots, t-1$. We list all possible truncated input and output difference values for the linear step $\ell'_i = L(y_0, y_1, x_{i-1})$, $i = 3, \ldots, t-1$ on the left side, and those of the nonlinear step $y_i = S(\ell, x_i)$ for $i = 2$ and $y_i = S(\ell', \delta_{x_i})$ for $i = 3, \ldots, t-1$ on the right side.

(a) The linear step $\ell'_i = \ell_i + x_{i-1}$

δ_{ℓ_i}	$\delta_{x_{i-1}}$	$\delta_{\ell'}$
0	0	0
1	0	1
0	1	1
1	1	0
1	1	1

(b) The nonlinear step $y_i = S(\ell, x_i)$ or $S(\ell', x_i)$

$\delta_\ell/\delta_{\ell'}$	δ_{x_i}	δ_{y_i}	τ_i
0	0	0	0
1	0	1	1
1	0	0	1
0	1	1	1
1	1	0	1
1	1	1	1

the MILP model as Eq. (20) originated in [27].

$$\begin{aligned}\mathcal{M}.\texttt{var} \leftarrow & \delta_{\ell_i}, \delta_{x_{i-1}}, \delta_{\ell'_i} \text{ are binary,}\\ \mathcal{M}.\texttt{con} \leftarrow & \begin{cases} \delta_{\ell_i} + \delta_{x_{i-1}} - \delta_{\ell'_i} \geq 0, \\ \delta_{\ell_i} - \delta_{x_{i-1}} + \delta_{\ell'_i} \geq 0, \\ -\delta_{\ell_i} + \delta_{x_{i-1}} + \delta_{\ell'_i} \geq 0. \end{cases}\end{aligned} \tag{20}$$

Note that Eq. (20) is the classical MILP description of the truncated differential propagation rule of the ordinary linear combinations. For simplicity, we use Eq. (21) to represent the constraints in Eq. (20):

$$\mathcal{M}.\texttt{con} \leftarrow (\delta_{x_{i-1}}, \delta_{\ell'_i}, \delta_{\ell_i})_+. \tag{21}$$

The linear step $\ell_i = \gamma_i \cdot y_0 + y_1$ for $i = 2, \ldots, t-1$ also follows the truncated differential propagation rule of ordinary linear combinations, so we can use inequalities as in Eq. (20) to express the constraints, i.e., $\mathcal{M}.\texttt{con} \leftarrow (\delta_{y_0}, \delta_{y_1}, \delta_{\ell_i})_+$ for $i = 2, \ldots, t-1$. Besides, for $(\delta_{y_0}, \delta_{y_1}) = (1, 1)$, since γ_i's are pairwise distinct, the number of non-zero δ_{ℓ_i}'s $(i = 2, \ldots, t-1)$ is at least $t-3$: there can be at most one $i \in \{2, \ldots, t-2\}$ satisfying $\delta_{\ell_i} = 0$. Taking this condition in consideration, the available values for $(\delta_{y_0}, \delta_{y_1}, \delta_{\ell_2}, \ldots, \delta_{\ell_{t-1}})$'s can be captured with the MILP model in Eq. (22) after introducing a dummy variable d_ℓ:

$$\begin{aligned}\mathcal{M}.\texttt{var} \leftarrow & \delta_{y_0}, \delta_{y_1}, \delta_{\ell_2}, \ldots, \delta_{\ell_{t-1}}, d_\ell \text{ are binary,}\\ \mathcal{M}.\texttt{con} \leftarrow & (\delta_{y_0}, \delta_{y_1}, \delta_{\ell_i})_+, \quad i = 2, \ldots, t-1,\\ \mathcal{M}.\texttt{con} \leftarrow & \begin{cases} d_\ell \geq \delta_{y_0}, \delta_{y_1}, \\ \delta_{y_0} + \delta_{y_1} \leq 2 \cdot d_\ell, \\ (t-3) \cdot d_\ell \leq \sum_{i=2}^{t-1} \delta_{\ell_i} \leq (t-2) \cdot d_\ell. \end{cases}\end{aligned} \tag{22}$$

Next, we consider the nonlinear step $y_i = S(\ell, x_i)$ for $i = 2$. The differential probabilities $\mathrm{DP}_{\max}((\delta_\ell, \delta_{x_i}) \mapsto \delta_{y_i})$ can be deduced by Lemma 4 as follows.

- If $(\delta_\ell, \delta_{x_i}) = (0, 0)$, we must have $\delta_{y_i} = 0$, so $\mathrm{DP}_{\max}[(0, 0) \mapsto 0] = 1 = p^0$.
- If $\delta_\ell \neq 0 \bigwedge \delta_{x_i} \neq 0$, the probability for a static δ_{y_i} is p^{-1}.
- If $\delta_\ell \neq 0 \bigwedge \delta_{x_i} = 0$, the probability for a static δ_{y_i} is p^{-1} (there is also $\delta_{y_i} = 0$ when $x_i = 0$).
- If $\delta_\ell = 0 \bigwedge \delta_{x_i} \neq 0$, there must be $\delta_{y_i} \neq 0$ and the probability for a static δ_{y_i} is $2p^{-1} = \frac{2}{p}$.

Note that for the linear steps $y_i = S(\ell', x_i)$ $(3 \leq i \leq t-1)$, the differential probabilities $\mathrm{DP}((\delta_{\ell'}, \delta_{x_i}) \mapsto \delta_{y_i})$ can be deduced in exactly the same way. Therefore, the binary variable τ_i for $2 \leq i \leq t-1$ tracks the probability s.t.

$$\mathrm{DP}_{\max}((\delta_\ell, \delta_{x_i}) \mapsto \delta_{y_i}) \leq (2/p)^{\tau_i}.$$

The possible values for $(\delta_\ell, \delta_{x_i}, \delta_{y_i}, \tau_i)$ or $(\delta_{\ell'}, \delta_{x_i}, \delta_{y_i}, \tau_i)$, are given in Table 7b.

In the following, we express $(\delta_\ell, \delta_{x_i}, \delta_{y_i}, \tau_i)$ (for the case $(\delta_{\ell'}, \delta_{x_i}, \delta_{y_i}, \tau_i)$, replace δ_ℓ with $\delta_{\ell'}$) in the MILP language as follows [27]:

$$\mathcal{M}.\mathtt{var} \leftarrow \delta_\ell, \delta_{x_i}, \delta_{y_i}, \tau_i \text{ are binary,}$$
$$\mathcal{M}.\mathtt{con} \leftarrow \begin{cases} -\delta_\ell + \tau_i \geq 0, \\ -\delta_{x_i} + \tau_i \geq 0, \\ -\delta_{y_i} + \tau_i \geq 0, \\ \delta_\ell + \delta_{x_i} - \tau_i \geq 0, \\ \delta_\ell + \delta_{y_i} - \tau_i \geq 0. \end{cases}$$

Note that $d > 2$, so the differential probability for $\Delta X \mapsto \Delta Y$ can be uniformly upper-bounded by

$$\mathrm{DP}_{\max}(\Delta X \mapsto \Delta Y) \leq (d/p)^{\tau_0 + \tau_1} \times (2/p)^{\sum_{i=2}^{t-1} \tau_i} \leq (d/p)^{\sum_{i=0}^{t-1} \tau_i}. \tag{23}$$

MILP Model for M. Assume the truncated input and output differences of M are $\Delta Y^r = (\delta_{y_0}, \ldots, \delta_{y_{t-1}})$ and $\Delta X^{r+1} = (\delta_{x_0}, \ldots, \delta_{x_{t-1}})$. For $t = 3, 4$, the corresponding M are MDS matrices, so the relationship between ΔY^r and ΔX^{r+1} can be described simply with the knowledge of the branch number $\#b = t + 1$. The description of the MILP model in this situation is as Eq. (24), where d_M is a dummy variable.

$$\mathcal{M}.\mathtt{var} \leftarrow \delta_{y_i}, \delta_{x_i}, d_M \text{ as binaries,}$$
$$\mathcal{M}.\mathtt{con} \leftarrow \begin{cases} d_M \leq \sum_{i=0}^{t-1} \delta_{y_i}, \sum_{i=0}^{t-1} \delta_{x_i} \leq t \cdot d_M, \\ \mathcal{M}.\mathtt{con} \leftarrow \sum_{i=0}^{t-1} \delta_{y_i} + \sum_{i=0}^{t-1} \delta_{x_i} \geq \#b \cdot d_M. \end{cases} \tag{24}$$

For $t = 4t' \geq 8$, according to Eq. (6), the linear layer M has branch number $\#b = t' + 4$ and can be decomposed as $M'' \times M'$. The branch number property of M can easily be captured by Eq. (24). However, we introduce a new method of describing the decomposition property of M, aiming at deriving more accurate bounds. First, We use an additional variable ΔW^r to represent the output truncated difference of M':

$$\Delta Y^r \xrightarrow{M'} \Delta W^r \xrightarrow{M''} \Delta X^{r+1}.$$

According to the definition of M', the transformation from ΔY^r to ΔW^r can be regarded as t' parallel 4×4 linear layers, each of which can be described as Eq. (24) with branch number $\#b = 5$. Next, we introduce and combine two methods to describe the truncated differential propagation of M''.

- Assume $\Delta W^r = (\delta_{w_0}, \dots, \delta_{w_{t-1}})$. The transformation $\Delta W^r \to \Delta X^{r+1}$ is additions of words over $\mathbb{F}_p$, so the truncated differential propagation of each word addition can be described by equations in Eq. (20) directly:

  $$\mathcal{M}.\mathtt{con} \leftarrow (\delta_{w_i}, \delta_{w_{i+4}}, \dots, \delta_{w_{i+t-4}}, \delta_{x_j})_+, i = 0, 1, 2, 3; j = i+4\cdot k, k = 0, \dots, t/4-1.$$

- In order to get the tighter bounds, we further decompose M'' as $M'' = \mathrm{circ}(2 \cdot I, I, \dots, I) = \mathrm{circ}(I, I, \cdots, I) + \mathrm{circ}(I, 0, \cdots, 0) = M_1 + M_2$. We express the truncated differential propagation of M_1 and M_2 in two steps. Note they follow the propagation rules of ordinary linear combinations, after defining four dummy binary variables δ_{tmp_i} for $i = 0, 1, 2, 3$, we can use the two MILP descriptions in Eq. (20) to respectively describe them:

  $$\mathcal{M}.\mathtt{con} \leftarrow \begin{cases} (\delta_{w_i}, \delta_{w_{i+4}}, \dots, \delta_{w_{t-4}}, \delta_{\mathrm{tmp}_i})_+, & i = 0, 1, 2, 3 \\ (\delta_{w_j}, \delta_{\mathrm{tmp}_{(j \bmod 4)}}, \delta_{x_j})_+, & j = 0, 1, \dots, t-1 \end{cases}$$

Finally, we take the maximal value of the objective function of the models using two descriptions. In this way, the differential bound of Griffin-π can be derived in a more precise manner than simply considering the branch number $\#b = t' + 4$ of M.

Objective Function of the MILP Model. As has been analyzed in Section 5.2, in order to acquire $\overline{\mathrm{DP}}$ of R-round Griffin-π, the objective function of the MILP model should be set to minimize the summation of all τ's as

$$\mathcal{M}.\mathtt{obj} = \min \sum_{j=0}^{R-1} \sum_{i=0}^{t-1} \tau_i^j,$$

where τ_i^j corresponds to the τ_i in Eq. (23) at the j-th round. The solution to such an objective function is denoted as $\#a = \mathcal{M}.\mathtt{obj}$.

Table 8: The MILP model solution $\#a$'s corresponding to different R and t settings. The maximal differential propagation probability $\overline{\mathrm{DP}}$'s can be bounded as $\overline{\mathrm{DP}} \leq (d/p)^{\#a}$.

$R\backslash t$	3	4	8	12	16	20	24
3	5	6	8	9	11	13	13
4	8	10	12	14	16	19	20
5	9	11	15	17	21	24	24

$\overline{\mathrm{DP}}$ for $R = 5$. For $t = 3, 4, 8, \dots, 24$, our MILP-based dedicated tool can construct models for R-round Griffin-π, where $R = 4, 5, 6$. The solutions $\#a$'s are given in Table 8.

Since the maximal DP of the S layer is bounded as $\mathrm{DP}_{\max} \leq \frac{d}{p}$, and $3 \leq t \leq 24$ for Griffin-π, the $\overline{\mathrm{DP}}$'s for $R = 5$ rounds and for the most relevant p are

$$\begin{aligned}
\overline{\mathrm{DP}} &\leq (d/p)^{17} \approx d^{17} \cdot 2^{-544} && \text{for } p \approx 2^{32},\ t \geq 12,\\
\overline{\mathrm{DP}} &\leq (d/p)^{15} \approx d^{15} \cdot 2^{-960} && \text{for } p \approx 2^{64},\ t \geq 8,\\
\overline{\mathrm{DP}} &\leq (d/p)^{9} \leq d^{9} \cdot 2^{-1152} && \text{for } p \geq 2^{128},\ t \geq 3.
\end{aligned}$$

For all $d \in \{3, 5, 7, 11\}$, there is constantly $\overline{\mathrm{DP}} < 2^{-544+17\cdot 3.5} \leq (2^{-128})^2$, which is smaller than our security claim of $\kappa = \min\{256, \log_2(p) \times t/3\} = 128$ bits. Therefore, we conclude that 5 rounds are sufficient for providing security against differential attacks. The relevant truncated differential characteristics are demonstrated in Table 9.

E.2 MILP Models for Griffin-π in Rebound Attacks

In differential attacks, since only the nonlinear operations can contribute to $\overline{\mathrm{DP}}$, τ's only appear in the description of the S layer. In rebound attacks, the complexities are decided by the number of word conditions which can be imposed by both nonlinear and linear operations. Word conditions are actually equations of words or word differences because particular differential propagation can only happen when the corresponding word conditions are satisfied. The word conditions lying in the inbound phase can be satisfied manually with the message modification technique for free while those in the outbound phase can only be satisfied randomly with probability p^{-1}.[22] In our model, for an operation $f : \mathbb{F}_p^m \to \mathbb{F}_p$, a word condition is imposed when there is a differential propagation $\vec{0} \neq \Delta X \xrightarrow{f} \Delta Y = 0$. Such a word condition is tracked in $\mathcal{M}$ with a binary variable. If the word condition lies in the inbound phase, the binary variable has the value 0. Otherwise, it has the value 1. Furthermore, the inbound-outbound manner of rebound attacks considers both forward and backward directions, so the corresponding word condition deduction should involve not only S and M,

[22] Such a technique is widely used in collision attacks of hash functions [29].

Table 9: 5-round differential paths deduced from the MILP model.

t	Interm.	$R=0$	$R=1$	$R=2$	$R=3$	$R=4$	$R=5$	$\sum_{R=0}^{5}\tau^R$
3	$\varDelta X^R$	2	0∼2	2	0∼2	2	0∼2	9
	$\varDelta Y^R$	2	0∼2	2	0∼2	2	–	
4	$\varDelta X^R$	3	0∼3	3	0∼3	3	0∼3	11
	$\varDelta Y^R$	3	0∼3	3	0∼3	3	–	
8	$\varDelta X^R$	3	0,2,4,6	2,7	0,1,3,5,6	7	0∼7	15
	$\varDelta Y^R$	3,4	0,2,4,6	2,7	0,1,3,5,6	7	–	
	$\varDelta W^R$	0∼7	0∼7	0∼7	1∼3,5∼7	4∼7	–	
12	$\varDelta X^R$	7	0,2,4,6,8,10	3	0,2,4,6,8,10	11	0∼11	17
	$\varDelta Y^R$	7,8	0,2,4,6,8,10	3,4	0,2,4,6,8,10	11	–	
	$\varDelta Z^R$	4∼11	0,2∼4,6∼8,10,11	0∼7	0∼11	8∼11	–	
16	$\varDelta X^R$	3,11	3,9,14,15	3,15	3,8∼11	15	0∼15	21
	$\varDelta Y^R$	3,4,11,12	3,4,9,10,14,15	3,4,15	3,4,8∼12	15	–	
	$\varDelta Z^R$	0∼15	0∼15	0∼15	0∼15	12∼15	–	
20	$\varDelta X^R$	15	12∼19	15,19	12∼19	15,19	0,3,4,7,8,11∼19	24
	$\varDelta Y^R$	15,16	12∼19	15,16,19	12∼19	15,16,19	–	
	$\varDelta Z^R$	12∼19	15,19	12∼19	15,19	12∼18	–	
24	$\varDelta X^R$	19	16∼23	19,23	16∼23	19,23	2,6,10,14,16∼23	24
	$\varDelta Y^R$	19,20	16∼23	19,20,23	16∼23	19,23	–	
	$\varDelta W^R$	16∼23	19,23	16∼23	19,23	16∼23	–	

but their inverses S^{-1} and M^{-1} as well. For clear interpretation, we still use binary variables τ's to track the word conditions in the nonlinear layer S, S^{-1} and μ's for those in the linear layer M, M^{-1}. Details are provided in the following.

MILP Models for Conditions in $\boldsymbol{S}$. For $i = 0, 1$, there is constantly $\delta_{y_i} = \delta_{x_i}$ and no word condition is introduced, so we have $\tau_i = 0 \in \mathcal{M}.\texttt{con}$. For $i = 2, \ldots, t-1$, since $y_i = x_i(\ell^2 + \alpha_i \ell + \beta_i)$ where $\ell = \lambda_0 \cdot y_0 + \lambda_1 \cdot y_1$, we only discuss cases depending on values of $(\delta_\ell, \delta_{x_i}, \delta_{y_i})$:

- $(\delta_\ell, \delta_{x_i}, \delta_{y_i}) \in \{(1,1,1), (0,0,0), (0,1,1)\}$: no word condition is introduced, so we set $\tau_i = 0$.
- $(\delta_\ell, \delta_{x_i}, \delta_{y_i}) = (1,0,0)$: one word condition is introduced as $x_i = 0$, so we set $\tau_i = 1$.
- $(\delta_\ell, \delta_{x_i}, \delta_{y_i}) = (1,1,0)$: one word condition is introduced from $\delta_{y_i} = 0$ as

$$(x_i + \delta_{x_i})[(\ell + \delta_\ell)^2 + \alpha_i(\ell + \delta_\ell) + \beta_i] - x_i(\ell^2 + \alpha_i \ell + \beta_i) = 0, \qquad (25)$$

  so we set $\tau_i = 1$.

We summarize all possible values for $(\delta_\ell, \delta_{x_i}, \delta_{y_i}, \tau_i)$ in Table 10. The relations

Table 10: All possible values of $(\delta_\ell, \delta_{x_i}, \delta_{y_i}, \tau_i)$ for S and $(\delta_\ell, \delta_{y_i}, \delta_{x_i}, \tau_i)$ for S^{-1}.

$(\delta_\ell,$	$\delta_{x_i},$	$\delta_{y_i},$	$\tau_i)$
(0,	0,	0,	0)
(1,	0,	1,	0)
(1,	0,	0,	1)
(0,	1,	1,	0)
(1,	1,	0,	1)
(1,	1,	1,	0)

of $\delta_\ell, \delta_{x_i}, \delta_{y_i}, \tau_i$ can be represented in a MILP model as

$$\begin{aligned}
&\mathcal{M}.\texttt{var} \leftarrow \delta_\ell, \delta_{x_i}, \delta_{y_i}, \tau_i \text{ as binaries,}\\
&\mathcal{M}.\texttt{con} \leftarrow -\delta_{y_i} - \tau_i + 1 \geq 0,\\
&\mathcal{M}.\texttt{con} \leftarrow -\delta_\ell + \delta_{y_i} + \tau_i \geq 0,\\
&\mathcal{M}.\texttt{con} \leftarrow \delta_\ell + \delta_{x_i} - \delta_{y_i} - \tau_i \geq 0,\\
&\mathcal{M}.\texttt{con} \leftarrow \delta_\ell - \tau_i \geq 0,\\
&\mathcal{M}.\texttt{con} \leftarrow -\delta_{x_i} + \delta_{y_i} + \tau_i \geq 0.
\end{aligned} \tag{26}$$

MILP Models for Conditions in S^{-1}. Since $x_0 = {y_0}^{1/d}$ and $x_1 = {y_1}^d$, there is constantly $\delta_{y_i} = \delta_{x_i}$, no condition is introduced for $i = 0, 1$. For $i = 2, \ldots, t-1$, similarly, several cases of word conditions can be identified depending on $(\delta_\ell, \delta_{y_i}, \delta_{x_i})$. An additional binary variable τ_i is also used to represent the number of word conditions for the tuple $(\delta_\ell, \delta_{y_i}, \delta_{x_i})$. We find that values of $(\delta_\ell, \delta_{y_i}, \delta_{x_i}, \tau_i)$ for S^{-1} are exactly the same as those of $(\delta_\ell, \delta_{x_i}, \delta_{y_i}, \tau_i)$ for S, and they are given in Table 10.

MILP Models for Conditions in M. We retain the MILP description of the differential propagation $\Delta Y \xrightarrow{M} \Delta Z$ in App. E.1. The extra word conditions for M are introduced when cancellation happens. Let the matrix of M be $(a_{i,j})_{t\times t}$. If $\delta_{z_i} = 0$, one word condition is introduced as

$$\sum_{j=0}^{t-1} (a_{i,j} \cdot \delta_{y_j}) = 0.$$

We use a binary variable μ_i to represent the number of word conditions introduced by δ_{z_i} as follows.

1. We add two variables $\mathcal{M}.\texttt{var} \leftarrow \mu_i, d_M$ as binaryies.
2. For $j = 0, \ldots, t-1$, if $a_{i,j} \neq 0$, we add a constraint $\mathcal{M}.\texttt{con} \leftarrow d_M \geq \delta_{y_j}$.
3. We add a constraint $\mathcal{M}.\texttt{con} \leftarrow \mu_i = d_M - \delta_{z_i}$.

MILP Models for Conditions in M^{-1}. For M^{-1}, we denote the corresponding matrix as $(b_{i,j})_{t\times t}$. If $\delta_{y_i} = 0$, one word condition is introduced as

$$\sum_{j=0}^{t-1}(b_{i,j} \cdot \delta_{z_j}) = 0.$$

The number of word condition introduced by δ_{y_i} can be modeled as follows.

1. We add two variables $\mathcal{M}.\mathtt{var} \leftarrow \mu_i, d_M$ as binaries.
2. For $j = 0, \ldots, t-1$, if $b_{i,j} \neq 0$ we add a constraint $\mathcal{M}.\mathtt{con} \leftarrow d_M \geq \delta_{z_j}$.
3. We add a constraint $\mathcal{M}.\mathtt{con} \leftarrow \mu_i = d_M - \delta_{y_i}$.

Secure Bounds against Rebound Attacks. In rebound attacks, the adversary aims at constructing a pair of intermediate states at round r_m ($1 \leq r_m \leq R-1$), denoted as $(Y^{r_m}, \hat{Y}^{r_m})$, having nonzero truncated difference ΔY^{r_m} whose truncated differential propagation follows some predefined characteristic $\Delta X^0 \xrightarrow{\mathcal{F}^R} \Delta X^R$.

According to the parameters in Table 1, we require c words in ΔX^0 and $\left\lfloor \frac{\kappa}{\log_2 p} \right\rfloor$ words in ΔX^R are 0. Reflected to the MILP model $\mathcal{M}$, we add the following constraints:

$$\mathcal{M}.\mathtt{con} \leftarrow \sum_{i=0}^{t-1} \delta_{x_i^0} \leq t - c,$$

$$\mathcal{M}.\mathtt{con} \leftarrow \sum_{i=0}^{t-1} \delta_{x_i^R} \leq t - \left\lfloor \frac{\kappa}{\log_2 p} \right\rfloor.$$

We further assume that all the word conditions between ΔY^{r_m-1} and ΔY^{r_m+1} can manually be satisfied by modifying words in Y^{r_m}. This has upper-bounded the power of the message modification technique [29] because each word can only be used once and can hardly modify words after S layers. Therefore, the objective function of the MILP model can be defined as follows:

$$\mathcal{M}.\mathtt{obj} \leftarrow \min\left(\sum_{i\leq r_m-1 \bigvee i>r_m+1} \sum_{j=1}^{t-1} \tau_j^i + \sum_{i<r_m-1 \bigvee i\geq r_m+1} \sum_{j=0}^{t-1} \mu_j^i\right).$$

For $r_m = 1, \ldots, R-1$, we construct the model for the r_m-th round, and the solution is the minimum number of unfixed word conditions, denoted as $\theta_{R,r_m} = \mathcal{M}.\mathtt{obj}$. If the condition

$$\left\lfloor \frac{\kappa}{\log_2 p} \right\rfloor \leq \min_{1\leq r_m\leq R-1} \theta_{R,r_m}$$

holds for all r_m settings, we know that R rounds are sufficient to resist rebound attacks. Otherwise, we may update $R \leftarrow R+1$ to the model and repeat the process. In this way, we are able to acquire the secure bound.

Table 11: Secure instances of Griffin considering only rebound attacks. The best setting corresponding to R is always $r_m = 1$: such $r_m = 1$ indicates that any start-from-the-middle strategy ($r_m > 1$) cannot provide a result better than a pure random search.

$\lceil \log_2 p \rceil$	κ	c	t	R
32	128	8	≥ 12	3
32	256	16	≥ 20	3
64	128	4	≥ 8	3
64	256	8	≥ 12	3
128	128	2	$3, 4, \geq 8$	3
128	256	4	≥ 8	3
256	128	1	$3, 4, \geq 8$	3
256	256	2	$3, 4, \geq 8$	3

The number of rounds required to resist rebound attacks for values of p we focus on are listed in Table 11. We claim that it is not possible to mount a rebound attack on more than 3 rounds of Griffin-π. Equivalently, 4 rounds are sufficient for providing security against this attack.

F Algebraic Attacks – Details

F.1 Density of Griffin-π

Since the only high-degree nonlinear function of Griffin-π is $x \mapsto x^{1/d}$, it is important to analyze the density of the construction, and in particular the density of the polynomials in each word. First, note that we apply a linear layer in the beginning, in order to ensure that the variables are mixed before the first nonlinear operation in the sponge setting.[23] Hence, the input x_1 to the nonlinear function in the second word is a linear combination of all input variables. In practice, we evaluated $x \mapsto x^{1/d}$ and found that it provides (almost) full density over $\mathbb{F}_p$. This behaviour was also observed after a small number of rounds in multiple words for different t due to the mixing. Moreover, we compared Griffin-π to the *Rescue* permutation and could not find any significant differences regarding the polynomial density. Thus, we claim that the polynomial representation of our construction is dense after 3 rounds, and in particular with the round numbers we propose (e.g., $R \geq 6$ for statistical attacks).

F.2 Practical Results for Gröbner Bases

Intermediate Variables. Concrete data points for practical Gröbner basis computations when introducing t intermediate variables in each round are shown

[23] Skipping this linear layer would only delay the variable mixing for one single round.

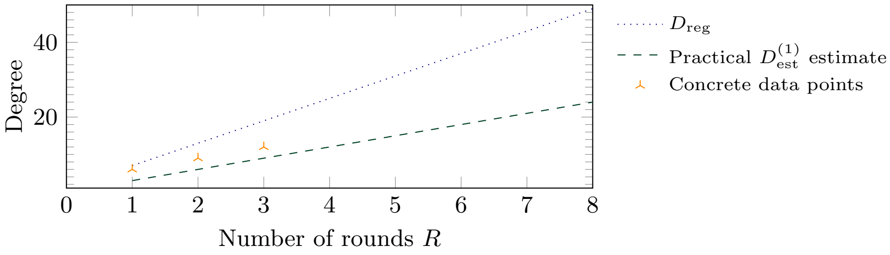


Fig. 3: Comparison of the theoretical estimation D_{reg} for $d = 3, t = 3$, the adapted estimation $D_{\mathrm{est}}^{(1)}$ using the practical results, and concrete data points from our practical tests with `Sage` (degree growth is the same in `Magma`).

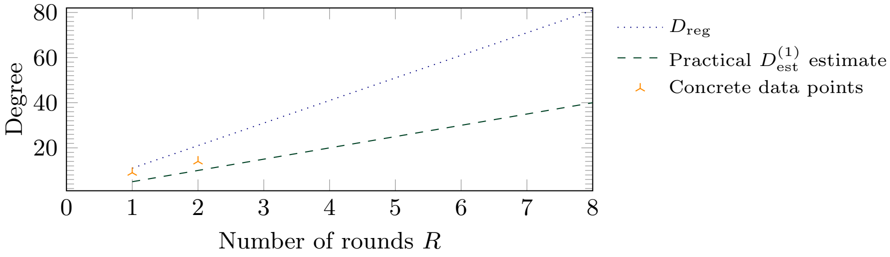


Fig. 4: Comparison of the theoretical estimation D_{reg} for $d = 5, t = 3$, the adapted estimation $D_{\mathrm{est}}^{(1)}$ using the practical results, and concrete data points from our practical tests with `Sage` (degree growth is the same in `Magma`).

in Fig. 3 for $d = 3$ and in Fig. 4 for $d = 5$. The theoretical estimation for the degree of regularity is given by

$$D_{\mathrm{reg}}^{(1)} = 1 + \sum_{i=1}^{n_e} (\deg(f_i) - 1) = 1 + R(2(d-1)) + 2(t-2)) \,.$$

Partial Intermediate Variables. Concrete data points for practical Gröbner basis computations when introducing 1 intermediate variable in each round in order to avoid the high-degree growth are shown in Fig. 5 for $d = 3$ and in Fig. 6 for $d = 5$. Since the degree of the equation in the $(i-1)$-th round is $\deg(R_{i-1})$ for $R \geq 2$ and the degree of the equation in the next round is $2d \cdot \deg(R_{i-1}) +$

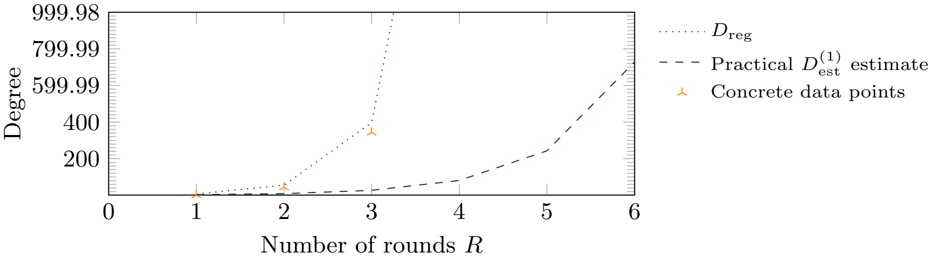


Fig. 5: Comparison of the theoretical estimation D_{reg} for $d = 3, t = 3$, the adapted estimation $D_{\text{est}}^{(2)}$ using the practical results, and concrete data points from our practical tests with `Sage`.

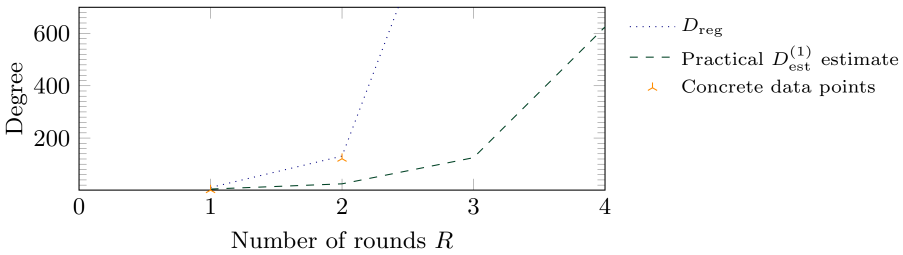


Fig. 6: Comparison of the theoretical estimation D_{reg} for $d = 5, t = 3$, the adapted estimation $D_{\text{est}}^{(2)}$ using the practical results, and concrete data points from our practical tests with `Sage`.

$\deg(R_{i-1})$, we have that

$$\begin{aligned}
D_{\text{reg}}^{(2)} &= 1 + \sum_{i=1}^{n_e} (\deg(f_i) - 1) \\
&= 1 + ((2d+1) - 1) + (2d(2d+1) + 2d + 1 - 1) + \cdots \\
&= 1 + (2d+1) - 1 + (2d+1)^2 - 1 + (2d+1)^3 - 1 + \cdots \\
&= 1 + \left(\sum_{i=1}^{R} (2d+1)^i \right) - R.
\end{aligned}$$

F.3 Additional Strategies for Gröbner Bases

Apart from the main strategies given in Section 5.3, there also exist other approaches which we briefly describe here. However, they are in general weaker, and hence we do not use them to determine the final number of rounds.

Full-Round Equation System. When considering the nonlinear layer described in Section 4.2 and given in Eq. (5), we see that $y_1 = {x_1}^{1/d}$, and hence the round function exhibits a high degree for low d. However, the starting variables may be chosen such that x_1 is constant. Hence, we need full diffusion in order to inrease the degree of a polynomial containing at least one input variable significantly. Since full diffusion is achieved after only one round and our representation is dense, we conclude that $\lceil \kappa / \log_2(p) \rceil + 1$ rounds of Griffin-π are sufficient against this attack. Note that using more than a single input variable increases the complexity in this case.

Adding an Intermediate Variable for L. A technique that can be combined with both main strategies described in Section 5.3 is to add an additional variable for the degree-2 function L. This increases the number of equations and variables by R, while only slightly reducing the overall degrees of the equations. Indeed, in our practical evaluations, we found that combining this technique with any of the above strategies leads to higher solving complexities.

F.4 Collisions via Gröbner Bases

The attacker can also directly write down the equation system for a collision. That is, they can use two different inputs to the permutation (with at least two different variables) and set the output difference to zero. The resulting system of equations has more variables than equations, also when considering the different attack strategies discussed before. Consequently, it may be possible to exploit the additional degree of freedom and fix specific variables in order to reduce the number of appearing monomials and make the system easier to solve. However, we conjecture that *(1)* no such degenerate cases exist which can be observed after the entire permutation and that *(2)* the collision attack is not cheaper compared to the preimage attack for the algebraic approaches described before, since the cost is essentially doubled when considering two absorption descriptions for the collision attack. This is also confirmed by practical experiments.

G Security Analysis – Other Attacks

G.1 Other Statistical Attacks

Linear Attack. Linear cryptanalysis [23] is often used to deduce distinguishers against the underlying permutation of sponge-based hash functions. Similar to the differential attacks, we are also able to construct MILP-model-based dedicated tools so as to search for optimal linear characteristics upper-bounding the probability. According to the analysis with the tools, we are able to conclude that, for both the hash function Griffin and the permutation Griffin-π, 5 rounds are well enough for resisting linear attacks.

Impossible Differential and Zero-correlation Attacks. According to the definition of M, the difference of one word can affect the whole t-word state only by one round function call. Therefore, the impossible differential [7, 21] and zero-correlation [8] attacks can hardly be mounted on 3 or more rounds.

Boomerang Attack. The boomerang attack [28] is a variant of differential attacks. Instead of constructing pairs satisfying differential paths, boomerang attacks look for quartets satisfying two differential paths simultaneously. For hash functions using compression functions, where message blocks are involved in the computation of intermediate internal states, the boomerang attack can be effective since the difference in the whole state can be eliminated with specifically designed message block differences. As studied in Section 5.2, differential trails with high probability are rather unlikely to occur for more than 6 rounds. Hence, the number of rounds necessary to prevent the differential attacks and/or the rebound attacks are sufficient to prevent the boomerang attack as well.

Integral/Square Attack, Multiple-of-n & Mixture Differential Cryptanalysis. As a scheme working natively on larger field elements, Griffin could be potentially attacked by all attacks vectors that exploit the strong alignment on symmetric schemes, as the integral and square attacks [10], or more recently the multiple-of-n differential cryptanalysis [17] and the mixture differential one [15]. However, we claim that such attacks become quickly infeasible, since the nonlinear layer S is not aligned and the matrix M provides full diffusion after a single round.

G.2 Higher-Order Differentials and Zero-Sum Partitions

Given a vectorial Boolean function $\mathcal{F}$ over $\mathbb{F}_2^n$ of degree d, the higher-order differential attack [22, 20] exploits the fact that

$$\sum_{x \in \mathfrak{V}+v} x = \sum_{x \in \mathfrak{V}+v} \mathcal{F}(x) = 0$$

for each affine subspace $\mathfrak{V} + v \subseteq \mathbb{F}_2^n$ of dimension strictly larger than d (i.e., $\dim(\mathfrak{V}) \geq d + 1$). The corresponding attack in the case of a prime field $\mathbb{F}_p$ has been recently proposed in [6]. Since this result is related to the degree of the polynomial that describes the permutation, we claim that the number of rounds necessary to guarantee security against the interpolation attack provides security against this attack as well.

A possible variant of higher-order sums in the case of permutations is the zero-sum partition attack/distinguisher.

Definition 5 (Zero-Sum Partition [9]). *Let P be a permutation over $\mathbb{F}_q^t$ for a prime $q \geq 2$. A zero-sum partition for $\mathcal{P}$ of size $l < t$ is a collection of l disjoint sets $\{X_1, \ldots, X_l\}$ with the following properties:*

- $X_i \subset \mathbb{F}^t$ *for each* $i \in \{1, \ldots, l\}$ *&* $\bigcup_{i=1}^{l} X_i = \mathbb{F}^t$,
- $\forall i \in \{1, \ldots, l\} : \sum_{x \in X_i} x = \sum_{x \in X_i} \mathcal{P}(x) = 0$.

This direction has been investigated e.g. in [9] for two SHA-3 candidates, *Luffa* and Keccak. Since it is expected that a randomly chosen function does not have many zero sums, the existence of several such sets can be seen as a distinguishing property of the internal function. Here we explicitly state that we do not make claims about the security of Griffin against zero-sum partitions. This choice is motivated by the gap present in the literature between the number of rounds of the internal permutation that can be covered by a zero-sum partition and by the number of rounds in the corresponding sponge hash function that can be broken e.g. via a preimage or a collision attack. As a concrete example, consider the case of Keccak. While 24 rounds of Keccak-f can be distinguished from a random permutation using a zero-sum partition [9] (that is, *full* Keccak-f), preimage/collision attacks on Keccak can only be set up for up to 6 rounds of Keccak-f [18]. This suggests that zero-sum partitions should be largely ignored for practical applications.

References for Supplementary Material

[1] A. Abdelkhalek, Y. Sasaki, Y. Todo, M. Tolba, and A. M. Youssef. “MILP Modeling for (Large) S-boxes to Optimize Probability of Differential Characteristics”. In: *IACR Trans. Symmetric Cryptol.* 2017.4 (2017), pp. 99–129.

[2] A. Aly, T. Ashur, Eli Ben-Sasson, S. Dhooghe, and A. Szepieniec. “Design of Symmetric-Key Primitives for Advanced Cryptographic Protocols”. In: *IACR Trans. Symmetric Cryptol.* 2020.3 (2020), pp. 1–45.

[3] M. Barbara, L. Grassi, D. Khovratovich, R. Lüftenegger, C. Rechberger, M. Schofnegger, and R. Walch. “Reinforced Concrete: Fast Hash Function for Zero Knowledge Proofs and Verifiable Computation”. In: *IACR Cryptol. ePrint Arch.* 2021 (2021), p. 1038.

[4] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. *Scalable, transparent, and post-quantum secure computational integrity.* Cryptology ePrint Archive, Report 2018/46. 2018.

[5] E. Ben-Sasson, L. Goldberg, and D. Levit. *STARK Friendly Hash - Survey and Recommendation.* Cryptology ePrint Archive, Report 2020/948. 2020.

[6] T. Beyne, A. Canteaut, I. Dinur, M. Eichlseder, G. Leander, G. Leurent, M. Naya-Plasencia, L. Perrin, Y. Sasaki, Y. Todo, and F. Wiemer. “Out of Oddity - New Cryptanalytic Techniques Against Symmetric Primitives Optimized for Integrity Proof Systems”. In: *CRYPTO 2020.* Vol. 12172. LNCS. 2020, pp. 299–328.

[7] E. Biham, A. Biryukov, and A. Shamir. “Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials”. In: *EUROCRYPT 1999.* Vol. 1592. LNCS. 1999, pp. 12–23.

[8] A. Bogdanov and M. Wang. “Zero Correlation Linear Cryptanalysis with Reduced Data Complexity”. In: *FSE 2012.* Vol. 7549. LNCS. 2012, pp. 29–48.

[9] C. Boura, A. Canteaut, and C. D. Cannière. “Higher-Order Differential Properties of Keccak and *Luffa*”. In: *FSE 2011.* Vol. 6733. LNCS. 2011, pp. 252–269.

[10] J. Daemen, L. R. Knudsen, and V. Rijmen. “The Block Cipher Square”. In: *FSE.* Vol. 1267. LNCS. 1997, pp. 149–165.

[11] M. Eichlseder, M. Nageler, and R. Primas. “Analyzing the Linear Keystream Biases in AEGIS”. In: *IACR Trans. Symmetric Cryptol.* 2019.4 (2019), pp. 348–368.

[12] K. Fu, M. Wang, Y. Guo, S. Sun, and L. Hu. “MILP-Based Automatic Search Algorithms for Differential and Linear Trails for Speck”. In: *FSE 2016*. Vol. 9783. LNCS. 2016, pp. 268–288.

[13] A. Gabizon and Z. J. Williamson. “plookup: A simplified polynomial protocol for lookup tables”. In: *IACR Cryptol. ePrint Arch.* (2020), p. 315.

[14] A. Gabizon, Z. J. Williamson, and O. Ciobotaru. *PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge*. Cryptology ePrint Archive, Report 2019/953. 2019.

[15] L. Grassi. “Mixture Differential Cryptanalysis: a New Approach to Distinguishers and Attacks on round-reduced AES”. In: *IACR Trans. Symmetric Cryptol.* 2018.2 (2018), pp. 133–160.

[16] L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofnegger. “Poseidon: A New Hash Function for Zero-Knowledge Proof Systems”. In: *USENIX Security Symposium*. USENIX Association, 2021, pp. 519–535.

[17] L. Grassi, C. Rechberger, and S. Rønjom. “A New Structural-Differential Property of 5-Round AES”. In: *EUROCRYPT 2017*. Vol. 10211. LNCS. 2017, pp. 289–317.

[18] J. Guo, G. Liao, G. Liu, M. Liu, K. Qiao, and L. Song. “Practical Collision Attacks against Round-Reduced SHA-3”. In: *J. Cryptol.* 33.1 (2020), pp. 228–270.

[19] *Hash functions for Zero-Knowledge applications Zoo*. https://extgit.iaik.tugraz.at/krypto/zkfriendlyhashzoo. IAIK, Graz University of Technology. Aug. 2021.

[20] L. R. Knudsen. “Truncated and Higher Order Differentials”. In: *FSE 1994*. Vol. 1008. LNCS. 1994, pp. 196–211.

[21] L. R. Knudsen. *DEAL – A 128-bit Block Cipher*. 1998.

[22] X. Lai. “Higher Order Derivatives and Differential Cryptanalysis”. In: *Communications and Cryptography: Two Sides of One Tapestry*. Springer US, 1994, pp. 227–233.

[23] M. Matsui. “Linear Cryptanalysis Method for DES Cipher”. In: *EUROCRYPT 1993*. Vol. 765. LNCS. 1993, pp. 386–397.

[24] N. Mouha, Q. Wang, D. Gu, and B. Preneel. “Differential and Linear Cryptanalysis Using Mixed-Integer Linear Programming”. In: *Inscrypt 2011*. Vol. 7537. LNCS. 2011, pp. 57–76.

[25] J. Patarin. “Generic Attacks on Feistel Schemes”. In: *ASIACRYPT 2001*. Vol. 2248. LNCS. 2001, pp. 222–238.

[26] S. Sun, D. Gérault, P. Lafourcade, Q. Yang, Y. Todo, K. Qiao, and L. Hu. “Analysis of AES, SKINNY, and Others with Constraint Programming”. In: *IACR Trans. Symmetric Cryptol.* 2017.1 (2017), pp. 281–306.

[27] S. Sun, L. Hu, P. Wang, K. Qiao, X. Ma, and L. Song. “Automatic Security Evaluation and (Related-key) Differential Characteristic Search: Application to SIMON, PRESENT, LBlock, DES(L) and Other Bit-Oriented Block Ciphers”. In: *ASIACRYPT 2014*. Vol. 8873. LNCS. 2014, pp. 158–178.

[28] D. A. Wagner. “The Boomerang Attack”. In: *FSE*. Vol. 1636. LNCS. 1999, pp. 156–170.

[29] X. Wang, Y. L. Yin, and H. Yu. “Finding Collisions in the Full SHA-1”. In: *CRYPTO 2005*. Vol. 3621. LNCS. 2005, pp. 17–36.