Starkad and Poseidon: New Hash Functions for Zero-Knowledge Proof Systems

Lorenzo Grassi[1], Dmitry Khovratovich[2,3], Christian Rechberger[1], Arnab Roy[4], and Markus Schofnegger[1]

[1] IAIK, Graz University of Technology
[2] Evernym Inc.
[3] ABDK Consulting
[4] University of Bristol, Bristol, UK
firstname.lastname@iaik.tugraz.at, khovratovich@gmail.com, arnab.roy@bristol.ac.uk

Abstract. The area of practical computational integrity proof systems, like SNARKs, STARKs, Bulletproofs, is seeing a very dynamic development with several constructions appeared in 2019 with improved properties and relaxed setup requirements. Many use cases of such systems involve, often as their most expensive apart, proving the knowledge of a preimage under a certain cryptographic hash function, which is expressed as a circuit over a large prime (sometimes binary) field. A zero-knowledge proof of coin ownership in the Zcash cryptocurrency is a notable example, where the inadequacy of SHA-256 hash function for such a circuit caused a huge computational penalty.

In this paper, we present a modular framework and concrete instances of cryptographic hash functions which either work natively with $GF(p)$ objects or on binary field elements. Our $GF(p)$ hash-function Poseidon uses up to 8x fewer constraints per message bit than Pedersen Hash, whereas our binary hash-function Starkad wins by a substantial margin over the other recent designs. Our construction is not only expressed compactly as a circuit, but also can be tailored for various proof systems using specially crafted polynomials, thus bringing another boost in performance. We demonstrate this by implementing a 1-out-of-a-billion membership proof using Merkle-trees in less than a second.

Keywords: Zero-Knowledge Proof Systems, SNARK, STARK, Hash Function

Table of Contents

1 Introduction

The recent advances in computational integrity proof systems made a number of computational tasks verifiable in short time and/or in zero knowledge. Several protocols appeared that require one party to prove the knowledge of a seed-derived secret, of an element being part of a large set, or their combination. Whereas accumulator-based solutions [16,15] and algebraic Schnorr proofs exist in the area, they are quite involving and thus error-prone, require a trusted setup, are limited in statement language, and are often slow. An alternative is to express secret derivation via application of cryptographic hash functions, and to prove set membership via presenting an opening in a properly chosen Merkle tree, also built on a cryptographic hash function. Such hash-based protocols require a computational integrity proof system, which can be applied to an arbitrary arithmetic circuit. However, for the protocol to be efficient, proofs must be generated and verified in reasonable time, which in turn requires the hash function to be cheap in a certain metric depending on the proof system.

At the beginning of 2020, the most popular proof systems are ZK-SNARKs (Pinocchio [43], Groth16 [26], Plonk [22], Marlin [18] to name a few), Bulletproofs [14], ZK-STARKs [8]. The former two groups have been already applied to a number of real-world protocols, whereas the latter is the most promising from the perspective of performance and post-quantum security. These three systems use two quite different circuit descriptions so that the proof size and generation time are computed differently:

- R1CS (Rank-1 quadratic constraints) describes the circuit as a set of special quadratic polynomials of the form $L_1(X) \cdot L_2(X) = L_3(X)$, where X is the tuple of internal and input variables, L_i are linear forms and $\cdot$ is the field multiplication, and (possibly in an affine-equivalent form) is used in almost all SNARKs and Bulletproofs. The circuit multiplication and addition gates over a prime field $GF(p)$. The proof generation complexity is directly proportional to the number T of constraints, which often corresponds to the number of multiplicative gates. The prime field $GF(p)$ is the scalar field of an elliptic curve, where for ZK-SNARKs the curve should be pairing-friendly and for Bulletproofs it should be just a secure curve.
- The AET metric is used in ZK-STARKs. The computation is expressed as a set of internal program states related to each other by polynomial equations of degree d. The state consists of w binary $GF(2^n)$ field elements and undergoes T transformations. The proof generation is roughly proportional to the product $w \cdot d \cdot T$, where n should be 32 or higher. The number and sparsity of polynomial constraints do not play a major role.

Our goal was to design a family of hash functions that are optimal in either the R1CS or the AET metric, and for different finite field sizes. Even though the metrics are different we tried to make the hash functions to share as many components as possible to reuse the analysis. It turned out that the substitution-

Table 1: Our primary proposals and their competitors. 'Tree' means Merkle tree arity and is equal to the rate/capacity ratio. Curve denotes the curve whose (subgroup) scalar field determines the prime size with BLS being BLS12-381, BN being BN254, Ed being Ristretto group. The R1CS/bit and AET/bit costs are obtained by dividing the R1CS (resp. AET) prover costs by message rate. Note that AET costs are measured in field operations, whose costs in software/hardware grow quadratically with the field size.

Name	**S-Box**	**Security** M	**Rate** $n \cdot t - 2M$	**SB size** $(\log_2 p)$	**Tree** $(nt/(2M-1))$	$\mathbf{R_F}$	$\mathbf{R_P}$	**Curve** Scalar field	**R1CS** /perm.	**R1CS** /bit
Poseidon-256	x^5	128	510	255	2:1	8	55	BLS/BN/Ed	237	0.46
	x^5	128	1020	255	4:1	8	56		288	0.28
	x^5	128	2040	255	8:1	8	57		387	0.19
Pedersen Hash	-	128	516	-	2:1	-		BLS12-381	869	1.68
Rescue	x^3 & $x^{1/3}$	128	508	254	2:1	22	-	-	264	0.52
Name	**S-Box**	**Security** M	**Rate** $n \cdot t - 2M$	**SB size**	**Tree** $(nt/(2M-1))$	$\mathbf{R_F}$	$\mathbf{R_P}$	**Field**	**AET** /perm.	**AET** /bit
Starkad-256	x^3	127	504	63	2:1	8	48	$GF(2^{63})$	24779	49
	x^3	127	1016	63	4:1	8	48		41086	41
	x^3	127	2032	63	8:1	8	48		73369	36
Friday-256	x^{-1}	128	256	-	-	-		$GF(2^{256})$	10371	41
Vision	x^{-1}	128	504	63	2:1	10	-	$GF(2^{63})$	34202	67

permutation network (SPN) design, well known in symmetric cryptography, allows a generic hash function framework where the only security-critical parameter that has to be changed for each instance is the number of rounds, and we provide an efficient and transparent strategy for its choice. The S-Box is chosen as power maps $x \mapsto x^d$, where $d \geq 3$ is usually chosen as the smallest integer that guarantees invertibility and provides non-linearity. In particular, the cube function x^3 is almost universally chosen, apart from cases of fields where this function is not a bijection. Instead, we suggest other S-Boxes such as x^5 or $1/x$ for these cases. Thanks to a succinct representation of the functions and low S-box degree, we are able to optimize the circuit significantly for Plonk and RedShift proof systems, with up to 40x performance increase.

Our Contributions. We design and analyze two families of hash functions: Starkad and Poseidon, which are both based on the HadesMiMC strategy [24]. The latter is a permutation design with t field elements forming the internal state, and each round is a composition of the S-Box layer, a linear transformation, and a round constant addition. We aim to support 128- and 256-bit security, where the security level is the same for collision and preimage resistance. For each pair (basic field, security level) we suggest a concrete instance of either the Starkad (for binary fields) or the Poseidon (for prime fields) permutation. Some middle rounds (called *partial*) carry only 1 rather than t S-Boxes to save up R1CS or AET cost. Each hash function is a certain permutation in the sponge mode of operation, where a few S-Box elements are reserved

for the capacity (roughly double the security level in bits), and the rest for the rate. The permutation width is determined by an application: it is set close to 1500 bits for long-message hashing, whereas for Merkle trees we support various width to enable 2:1, 4:1, and 8:1 arities and thus higher ZK performance.

We provide an extensive cryptanalysis of both families with an accent on algebraic methods as these prove to be the most effective. We explore different variants of interpolation, Gröbner basis, and higher-order differential attacks. As our permutations are quite wide, we do not aim for them behaving like randomly chosen permutations. Instead, for security level of M bits we require that no attack could exhibit a non-random property of a permutation faster than in 2^M queries. We then calculate the maximum number of rounds for each field, security level, and fixed permutation width that can be attacked. Then we select the number of rounds for concrete instances together with a security margin.

We have evaluated the number of constraints in Poseidon instances for the R1CS metric and the STARK complexity in Starkad instances for the AET metric. Our primary proposals Poseidon-252, Poseidon-256, and Starkad-252 are listed in Table 1 and are compared to similar-purpose designs. As supplementary material we provide reference implementations of various instances of designs, scripts to choose and create those instances, as well as code for benchmarks which we describe later in the paper[5].

Related Work. The Zcash designers introduced a new 256-bit hash function called Pedersen hash [28, p.134], which is effectively a vectorized Pedersen commitment in elliptic curve groups with short vector elements. For the claimed 128-bit security level, it utilizes 869 constraints per 516-bit message chunks, thus having 1.7 constraints per bit, whereas our Poseidon instances use from 0.2 to 0.45 constraints per bit, depending on the underlying prime field.

For the binary field case, Ashur and Dhooghe [5] have recently introduced the STARK-friendly block cipher Jarvis and its derivative hash function Friday with several instances and security levels. They use a key-alternating structure with a single inverse S-Box, followed by an affine transformation (with low degree in the extension field). However, both Jarvis and Friday were successfully attacked shortly after their publication [3]. In the response, the authors created a new family of SNARK/STARK-friendly family of hash functions with Vision (binary fields) and Rescue (prime fields) being main instances [4]. The latter share some similarity with our design with two important differences: all S-box layers are full; and every second layer has S-boxes of type $x^{1/d}$ for small d. This approach prevents some algebraic attacks but is also more expensive in software as the resulting power functions have high Hamming weight and thus require many squarings.

Structure of the Paper. We provide an overview of our design strategy in Section 2. We summarize the cryptanalysis results in Section 3 with the details

[5] `https://github.com/anonymous-ccs/material-ccs`

in the appendix. We explain the rationale for the choice of the number of rounds in Section 4. Then we suggest concrete parameters (permutation size, number of rounds, round constant generation) for our designs Starkad and Poseidon in Section 5. We estimate R1CS costs of Starkad instances in Section 6.1 and AET (STARK) costs in Section 6.2.

2 The Starkad and Poseidon Hash Functions

2.1 Overview

In the following we propose two hash functions:

- the hash function[6] Starkad-Hash for the *binary case* is constructed by instantiating a sponge construction [10] with Starkad-Permutation – denoted by Starkad$^\pi$;
- the hash function[7] Poseidon-Hash for the *prime case* is constructed by instantiating a sponge construction [10] with Poseidon-Permutation – denoted by Poseidon$^\pi$.

Both permutations are variants of HadesMiMC– the block cipher proposed in [24] – instantiated by a fixed key, e.g. 0^κ.

2.2 Sponge Construction for Starkad-Hash and Poseidon-Hash

We recall that when the internal permutation $\mathcal{P}$ of an N-bit sponge function (composed of a c-bit capacity and an r-bit rate: $N = c + r$) is modeled as a randomly chosen permutation, it has been proven by Bertoni *et al.* [10] to be indifferentiable from a random oracle up to $2^{c/2}$ calls to $\mathcal{P}$. In other words, a sponge with a capacity of c provides $2^{c/2}$ collision and $2^{c/2}$ (second) preimage resistance. Given a permutation of size N and a desired security level s, we can hash $r = N - 2s$ bits per call to the permutation. Following this design strategy, we choose the number of rounds of the inner permutations Poseidon$^\pi$ and Starkad$^\pi$ in order to ensure that such a permutation does not exhibit non-generic properties up to 2^M queries[8], where M is the desired security level.

As usual, the message is first padded according to the sponge specification so that the number of message blocks is a multiple of r, where r is the rate in the sponge mode. In our case, we use the Poseidon$^\pi$ or the Starkad$^\pi$ permutation,

[6] *About the name:* Starkad was a legendary hero in Norse mythology, who used to hash his enemies with 2^2 swords in 2^3 arms.

[7] *About the name:* Poseidon – brother of Zeus and Hades – was god of the Sea and other waters, of earthquakes and of horses.

[8] In other words, such a permutation cannot be distinguished from a randomly drawn permutation.

where $N \geq 4 \cdot M$. For Poseidon-256 (analogous for Starkad-256), we thus use the Poseidon permutation with $N = n \cdot t \geq 1024$. The capacity is chosen to be 256. This choice allows e.g. for processing more input bits then SHA-256 (512 bits) while at the same time offering collision security and (second) preimage security of 128 bits. Similar considerations hold as well for Poseidon-128 and/or Starkad-128.

2.3 Security Claims

In terms of concrete security, we expect it to be infeasible for all our hash functions to find collision attacks or preimage attacks with a cost substantially lower than 2^M evaluations of the permutation.

To help increase confidence in our design and simplify external cryptanalysis, we also explicitly state another claim about our internal permutation about the difficulty of the so-called constrained-input constrained-output problem of the permutation: *We expect it to be infeasible for all our permutations, to solve the CICO problem by fixing m_1 bits of the input and m_2 bits of the output of the permutation with a cost substantially lower than $2^{\min(M, m_1, m_2)}$ evaluations of the permutation.*

Even though an attack below this threshold may not affect any concrete application of our hash functions, we would still consider it an important cryptanalytic result.

2.4 The Hades Strategy

Cryptographic permutations are typically designed by iterating an efficiently implementable round function many times in the hope that the resulting composition behaves like a randomly drawn permutation. In general, *the same round function is iterated enough times to make sure that any symmetries and structural properties that might exist in the round function vanish.*

Instead of considering the same round function in order to construct the cipher (to be more precise, the same non-linear layer for all rounds), in [24] authors propose to consider *a variable number of S-Boxes per round*, that is, to use different S-Box layers in the round functions.

Due to our goals, in the following we shortly describe the cipher presented in [24] as a keyless permutation, by replacing the *AddRoundKey* operation with an *AddRoundConstant* operation. Similar to other SPN designs, each round of a keyless permutation based on Hades is composed of three steps:

1. *AddRoundConstant* – denoted by $ARC(\cdot)$;
2. *SubWords* – denoted by S-Box$(\cdot)$ or $SB(\cdot)$;

3. *MixLayer* – denoted by $M(\cdot)$.

A final round constant addition is usually added after the last round, but we do not use this in the STARKAD/POSEIDON hash functions for uniformity:

$$\underbrace{ARC \rightarrow SB \rightarrow M}_{1st \text{ round}} \rightarrow ... \rightarrow \underbrace{ARC \rightarrow SB \rightarrow M}_{(R-1)\text{-}th \text{ round}} \rightarrow \underbrace{ARC \rightarrow SB \rightarrow M}_{R\text{-}th \text{ round}}$$

The crucial property of HADES is that *the number of S-Boxes per round is not the same for every round*:

- a certain number of rounds – denoted by R_F – has a *full* S-Box layer, i.e., t S-Box functions;
- a certain number of rounds – denoted by R_P – has a *partial* S-Box layer, i.e., $1 \leq s < t$ S-Boxes and $(t - s)$ identity functions.

In the following, we limit ourselves to consider only the case $s = 1$, that is, R_P rounds have a single S-Box per round and $t - 1$ identity functions.

In more details, assume $R_F = 2 \cdot R_f$ is an even number[9]. Then

- the first R_f rounds have a full S-Box layer,
- the middle R_P rounds have a partial S-Box layer (i.e., 1 S-Box layer),
- the last R_f rounds have a full S-Box layer.

Figure 1 shows the HADES strategy. Note that the rounds with a partial S-Box layer are "masked" by the rounds with a full S-Box layer, which means that an attacker should not (directly) take advantage of the rounds with a partial S-Box layer.

Behind the HADES Strategy. The crucial point of our design is that it contains *both rounds with full S-Box layers and rounds with partial S-Box layers*. This allows to provide *simpler argumentation about the security against statistical attacks* than the one proposed for P-SPN permutations.

In more details, a certain number of rounds $R_F^{stat} = 2 \cdot R_f^{stat}$ with full S-Box layers situated at the beginning and the end provides security against statistical attacks. Indeed, even without the middle part, they are sufficient in order to apply the "wide-trail" strategy, in a way that we are going to show in the following. Security against all algebraic attacks is achieved working both with rounds $R_F = R_F^{stat} + R'_F \geq R_F^{stat}$ with a full S-Box layer and rounds $R_P \geq 0$ with a partial S-Box layer. Even if few (even one) S-Boxes per round are potentially sufficient to increase the degree of the encryption/decryption function (which mainly influences the cost of an algebraic attack), other factors can play

[9] $R_F = 2 \cdot R_f$ is even in order to have a "symmetric" permutation. Note that some attacks – like the statistical ones – have the same performance both in the forward and in the backward direction. Thus a "symmetric" permutation with $R_F = 2 \cdot R_f$ guarantees the same security against these attacks both in the chosen-/known-input scenario and in the chosen-/known-output one.

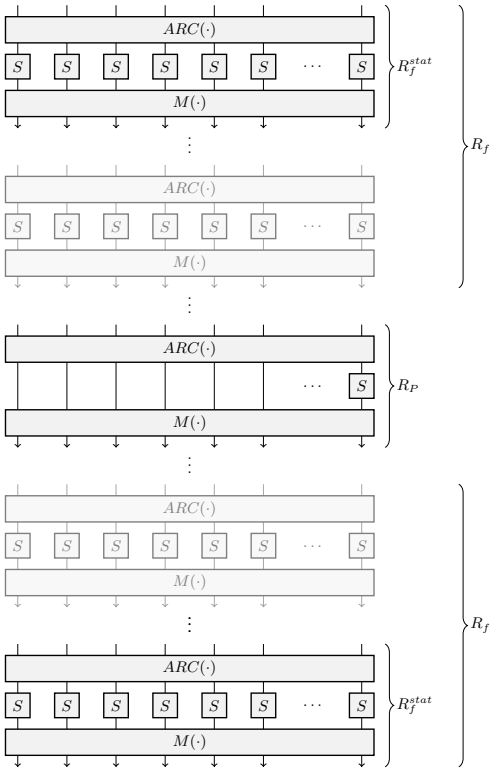

Fig. 1: Construction of the HADES-like permutation.

a crucial role in the cost of such attacks (e.g. a Gröbner basis attack depends also on the number of non-linear equations to solve).

With this in mind, the idea is to construct "something in the middle" between an SPN and a P-SPN permutation. Moreover, since we aim to have the same security w.r.t. chosen-input and chosen-output attacks, we consider a permutation which is "symmetric": in other words, the same number of rounds with full non-linear layer is applied at the beginning and at the end, where the rounds with partial non-linear layers are in the middle and they are "masked" by the rounds with full non-linear layers. As a result, depending on the cost metric that one aims to minimize (e.g. the total number of non-linear operations) and on the size of the S-Box, in the following we provide the *best ratio* between the number of rounds with full S-Box layer and with partial ones in order to both achieve security and minimize the cost metric.

For more details about the HADES strategy, we refer to [24].

What about the choice of the linear and of the non-linear layer? This strategy does not pose any restriction/constriction on the choice of the linear

layer and/or on the choice of the S-Box. The idea is to consider a "traditional" SPN permutation based on the wide-trail strategy, and then to replace a certain number of rounds with full S-Box layers with the same number of rounds with partial S-Box layers in order to minimize the number of non-linear operations, but without affecting the security. The HADES strategy has a huge impact especially for permutations with low-degree S-Box, since in this case a large number of rounds is required to provide security against algebraic attacks.

2.5 The Permutations STARKAD$^\pi$ and POSEIDON$^\pi$

Roughly speaking, the permutations POSEIDON$^\pi$ and STARKAD$^\pi$ are obtained by applying the HADES strategy to the cipher SHARK [44], proposed by Rijmen *et al.* in 1996, and omitting the keys. Both permutations work with texts of $t \geq 2$ words in $\mathbb{F}_p$ or $\mathbb{F}_{2^n}$, where p is a prime of size $p \approx 2^n$.

The linear layer of POSEIDON$^\pi$ and STARKAD$^\pi$ consists of the multiplication with a fixed $t \times t$ MDS matrix[10]. The number of rounds $R = 2 \cdot R_f + R_P$ depends on the choice of the S-Box and of the parameters n and t. For the applications we have in mind, we focus on

- the cubic S-Box S-Box$(x) = x^3$ – remember that the cube S-Box is a bijection in $GF(2^n)$ iff n is odd and it is a bijection in $GF(p)$ iff $p = 2 \mod 3$; in the following, these permutations are called "x^3 – POSEIDON$^\pi$" for the prime case and "x^3 – STARKAD$^\pi$" for the binary case;
- the S-Box S-Box$(x) = x^5$ – remember that x^5 is a bijection in $GF(2^n)$ iff $2^n \neq 1 \mod 5$, and it is a bijection in $GF(p)$ iff $p \neq 1 \mod 5$; in the following, these permutations are called "x^5 – POSEIDON$^\pi$" for the prime case and "x^5 – STARKAD$^\pi$" for the binary case;
- the inverse one S-Box$(x) = x^{-1}$; in the following, these permutations are called "x^{-1} – POSEIDON$^\pi$" for the prime case and "x^{-1} – STARKAD$^\pi$" for the binary case.

About the MDS Matrix. A $t \times t$ MDS matrix[11] with elements in $GF(2^n)$ (or $GF(p)$ where $p \approx 2^n$) exists if the condition (see [37] for details)

$$2t + 1 \leq 2^n \qquad \text{or/and} \qquad 2t + 1 \leq p$$

(or equivalently $t \cdot \log_2(2t + 1) \leq N$) is satisfied.

Given n and t, there are several ways to construct an MDS matrix (similar for prime case). One of them is using a Cauchy matrix [45], which we recall here briefly. Let $x_i, y_i \in \mathbb{F}_{2^n}$ for $i = 1, ..., t$ s.t.

[10] The schemes can be easily modified in order to also allow near-MDS matrices.

[11] A matrix $M \in \mathbb{F}^{t \times t}$ is called *Maximum Distance Separable* (MDS) matrix iff it has branch number $\mathcal{B}(M)$ equal to $\mathcal{B}(M) = t + 1$. The branch number of M is defined as $\mathcal{B}(M) = \min_{x \in \mathbb{F}^t} \{wt(x) + wt(M(x))\}$, where wt is the hamming weight. Equivalently, a matrix M is MDS iff every submatrix of M is non-singular.

– $\forall i \neq j: \qquad x_i \neq x_j, \quad y_i \neq y_j,$

– for $1 \leq i \leq t$ and $1 \leq j \leq t$: $\qquad x_i \oplus y_j \neq 0.$

To fulfill these conditions, one can simply consider x_i s.t. the $t - \log_2(t)$ most significant bits are zero. Then, choosing $r \in \mathbb{F}_{2^n}$ s.t. the $t - \log_2(t)$ most significant bits are non zero, let $y_i = x_i \oplus r$. Let A be the Cauchy matrix defined by

$$a_{i,j} = \frac{1}{x_i \oplus y_j}.$$

It follows that A is MDS. A similar construction works for $\mathbb{F}_p$.

Efficient Implementation. We refer to App. A for a complete description about possible strategies for efficient Poseidon and Starkad implementations. The main advantage of these strategies consists of reducing the number of constant multiplications in each round with a partial S-Box layer from t^2 to $2t$, which is particularly useful for large t and R_P. For example, we implemented Poseidon $^\pi$ with $(t, R_F, R_P) = (24, 8, 42)$ in `Sage`, and we could observe that the performance improves by a factor of about 5, with the average computation time being 4 ms for the optimized version.

3 Cryptanalysis Summary of the Starkad and Poseidon Hashes

As for any new design, it is paramount to present a concrete security analysis. In the following, we provide an in-depth analysis of the security of our construction. Due to a lack of any method to ensure that an hash function based on a sponge construction is secure against all possible attacks, we base our argumentation on the following consideration. As we just recalled in the previous section, when the internal permutation $\mathcal{P}$ of an $N = c + r$ bit sponge function is modeled as a randomly chosen permutation, the sponge hash function is indifferentiable from a random oracle up to $2^{c/2}$ calls to $\mathcal{P}$. Thus, we choose *the numbers of rounds of the inner permutation case in order to guarantee security against any (secret-/known-/chosen-) distinguisher. Equivalently, this means that such number of rounds guarantee that $\mathcal{P}$ does not present any non-random/structural property (among the ones known in the literature[12]).*

Now we list the main points of our cryptanalysis results. The number of rounds we can break depends on the security level M and the number of S-Boxes t, which we specify for each concrete hash function instance in the next section. Before going on, we highlight that *all details about cryptanalysis are provided as Supplementary Material.*

$\mathbb{F}_p^t$ ***versus*** $\mathbb{F}_{2^n}^t$**.** From the designer's point of view, the prime field version $\mathbb{F}_p^t$ appears stronger than the binary field version $\mathbb{F}_{2^n}^t$, since fewer known attacks

[12] We do not exclude that a non-random property can be discovered in the future.

apply. In particular, the designer must take into account the higher-order differential attack when they determine the number of rounds to provide security in $\mathbb{F}_{2^n}^t$. Vice versa, this attack does not apply (or is much less powerful) in $\mathbb{F}_p^t$ (due to the fact that the only subspaces of $\mathbb{F}_p$ are $\{0\}$ and $\mathbb{F}_p$ itself).

Statistical Attacks. As we show in the following, for any S-Box $(x^3, x^5, 1/x)$, at least 6 rounds with full S-Box layer are necessary to provide security against statistical attacks (differential, linear, truncated/impossible differential attacks, rebound attack) we consider. In more details:

$$R_F \geq \begin{cases} 6 & \text{if } \mathfrak{C} \times (t+1) \leq N + n - M \\ 10 & \text{otherwise} \end{cases}$$

(where $\log_2 p = n$) are sufficient to prevent statistical attacks, where $\mathfrak{C} = 1$ for the cubic S-Box and $\mathfrak{C} = 2$ for the other two considered S-Boxes.

Algebraic Attacks. In order to estimate the security against algebraic attacks, we evaluate the degree of the reduced-round permutations and their inverses. Roughly speaking, our results can be summarized as following (where $n \simeq \log_2(p)$ for the prime field):

Interpolation Attack. The interpolation attack depends on the number of different monomials of the interpolation polynomial, where (an upper/lower bound of) the number of different monomials can be estimated given the degree of the function. The idea of such an attack is to construct an interpolation polynomial that describes the function. If the number of monomials is too big, then such a polynomial cannot be constructed faster than via a brute force attack. For a security level of M bits, the numbers of rounds that can be attacked are

$$\begin{aligned} S(x) = x^3: &\quad R_F + R_P \leq \log_3(2) \cdot \min\{n, M\} + \log_2 t \\ S(x) = x^5: &\quad R_F + R_P \leq \log_5(2) \cdot \min\{n, M\} + \log_2 t \\ S(x) = 1/x: &\quad R_F \log_2(t) + R_P \leq \log_2(t) + 0.5 \cdot \min(M, n) \end{aligned} \tag{1}$$

Gröbner Basis. In a Gröbner basis attack, one tries to solve a system of non-linear equations that describe the function. The cost of such an attack depends obviously on the degree of the equations, but also on the number of equations and on the number of variables. Working on the cubic S-Box case (analogous for the others), we show that the attack complexity is about $O(D^{2t})$, therefore for a security level of M bits the attack works at most on $\log_3 2^{\min\{n/2, M/2\}}$ rounds, which is smaller than for the interpolation attack. If a partial S-Box layer is used, it could become more efficient to consider degree-3 equations for single S-Boxes. In this case, more rounds can be necessary to guarantee security against this attack. With optimistic (for the adversary) complexity of the Gaussian elimination, we obtain for each S-Box two attacks which are faster than 2^M if either

condition is satisfied:

$$
\begin{aligned}
S(x) = x^3: &\quad \begin{cases} R_F + R_P \leq 0.32 \cdot \min(M, n) \\ (t-1) \cdot R_F + R_P \leq 0.18 \cdot \min(M, n) - 1 \end{cases} \\
S(x) = x^5: &\quad \begin{cases} R_F + R_P \leq 0.21 \cdot \min(M, n) \\ (t-1) \cdot R_F + R_P \leq 0.14 \cdot \min(M, n) - 1 \end{cases} \\
S(x) = 1/x: &\quad \begin{cases} R_F \log_2(t) + R_P \leq \log_2(t) + 0.5 \cdot \min(M, n)) \\ (t-1) \cdot R_F + R_P \leq 0.25 \cdot \min(M, n) - 1 \end{cases}
\end{aligned} \tag{2}
$$

Higher-Order Differential Attack. The higher-order differential attack depends on the *boolean degree*, where the boolean degree δ of a function $f(x) = x^d$ is given by $\delta = hw(d)$ where $hw(\cdot)$ is the hamming weight. The idea of such an attack is based on the property that given a function $f(\cdot)$ of boolean degree δ, then $\bigoplus_{x \in V \oplus \phi} f(x) = 0$ if the dimension of the subspace V satisfies $dim(V) \geq \delta + 1$. If the boolean degree is sufficiently high, then the attack does not work. The attack applies to the binary field case, where we use the x^3 S-Box only. We obtained that the boolean degree grows accordingly to the algebraic degree, as the polynomial becomes dense and any monomial of degree d implies the existence of almost all monomials of smaller degree, which contain, among others, a monomial with degree of weight $\log_2 d$. Eventually we obtain the following condition for the attack to work in the binary field case:

$$R_F + R_P \leq \log_3(2) \cdot \min(M, n) + 2 + \log_2 t. \tag{3}$$

Zero-Sum Partition. The zero-sum partition distinguisher can be applied for $q = q_1 + q_2$ rounds as long as the boolean degree in the forward direction for q_1 and in the backward direction for q_2 does not exceed M. In the case in which the S-Box is defined by a power map with low degree, this allows attacking the same number of rounds as for the higher-order differential attack (as the inverse function has high algebraic degree). In all other cases, one has to approximately double the number of rounds considered in a higher-order differential attack in order to guarantee security (as the inverse function has almost the same algebraic degree).

Security Margin. Given the *minimum* number of rounds necessary to provide security against all attacks known in the literature, we *arbitrarily* decided to add:

- two more rounds with full S-Box layers (+2 R_F);
- 7.5% more rounds with partial S-Box layers (+7.5% R_P).

4 Number of Rounds Needed for Security

The design goal is to offer a family of hash functions which minimize the R1CS costs (Starkad instances, Section 6.1) or AET (STARK) costs in (Poseidon instances, Section 6.2). It turns out that for the fixed S-Box function the minimum costs are delivered by a primitive with the smallest number of S-Boxes, though the field size also plays a role. For each combination (security level M, prime/binary field type, S-Box size, S-Box function) we minimize the number of S-Boxes taking into account Equations (1),(2),(3).

Minimize "Number of S-Boxes" In our design strategy, we always exploit the "wide-trail" strategy in order to guarantee security against statistical attacks. In other words, for this class of attacks, we work only with rounds with full S-Box layers in order to guarantee security. All our instances are secure against statistical attacks if

$$R_F^{stat} \geq \begin{cases} 6 & \text{if } \mathfrak{C} \times (t+1) \leq N + n - M \\ 10 & \text{otherwise} \end{cases}$$

where $\mathfrak{C} = 1$ for the cubic S-Box and $\mathfrak{C} = 2$ for the other two considered S-Boxes. In order to minimize the number of S-Boxes for given n and t, the goal is to find the best ratio between R_P and R_F that minimizes

$$\textit{number of S-Boxes } = t \cdot R_F + R_P, \tag{4}$$

where $t \geq 2$ and where the number of non-linear operations is proportional to the number of S-Boxes.

Overall, the S-Box type and the number of rounds should be chosen as follows:

- If you plan to use a binary field $\mathbb{F}_{2^n}$ with n odd[13]:
  - Use S-Box x^3;
  - Select R_F to be 6 or higher.
  - Select R_P that minimizes $tR_F + R_P$ such that no inequation (1),(2) is satisfied.
- If you plan to use a prime field $\mathbb{F}_p$ and $GCD(q, p-1) = 1$ for $q = 3$ or $q = 5$:
  - Use S-Box x^q;
  - Select R_F to be 6 or higher.
  - Select R_P that minimizes $tR_F + R_P$ such that no inequation (1),(2) is satisfied.

[13] Note that $x \mapsto x^3$ is a permutation over $\mathbb{F}_{2^n}$ if and only if $GCD(3, 2^n - 1) = 1$, that is if and only if n is odd.

– for all other cases, we suggest to:[14]

- Use S-Box x^{-1};
- Select R_F to be 6 or higher.
- Select R_P that minimizes $tR_F + R_P$ such that no inequation (1),(2) is satisfied.

Before going on, we mention that other S-Boxes can be used as well (e.g., $x \mapsto x^7$). We have set up a script that calculates the number of rounds accordingly, using the security margin further described at the end of Section 3. Our resulting instances are given in Tables 5-6.

Results via Script. A complete analysis on how to set up the script – in order to guarantee security and to find the best ratio between R_P and R_F – for this case has been proposed in [24]. For this reason, we refer to [24], and we limit ourselves here to report the minimum number of rounds necessary to provide security.

For completeness, we mention that the simplest way to set up the script is to test (e.g., by brute force) all possible values R_P and R_F that provide security (equivalently, for which the previous inequalities are satisfied), and find the ones that minimize the cost metric.

5 Concrete Instantiations – Poseidon$^{\pi}$ and Starkad$^{\pi}$

For our applications, we are interested in the cases:

– text size: $N = 1\,536 = 3 \cdot 2^9$ (where $N = n \cdot t \simeq t \cdot \log_2 p$);

– security level in bits: $M = 128$ and $M = 256$.

All our MDS matrices are Cauchy matrices, and the method to construct them is further described in Section 2.5. We use ascending sequences of integers (or elements in $\mathbb{F}_{2^n}$) for the construction.

The round constants are generated using the Grain LFSR [27] in a self-shrinking mode:

1. Initialize the state with 80 bits $b_0, b_1, \ldots, b_{79}$, where
   (a) b_0, b_1 describe the field,
   (b) b_i for $2 \leq i \leq 5$ describe the S-Box,

[14] Since we do not present any concrete application that makes use of this S-Box (and due to page limitation), we present a complete cryptanalysis of this case only on an extended version of the paper.

(c) b_i for $6 \leq i \leq 17$ are the binary representation of n,

(d) b_i for $18 \leq i \leq 29$ are the binary representation of t,

(e) b_i for $30 \leq i \leq 39$ are the binary representation of R_F,

(f) b_i for $40 \leq i \leq 49$ are the binary representation of R_P,

(g) b_i for $50 \leq i \leq 79$ are set to 1.

2. Update the bits using $b_{i+80} = b_{i+62} \oplus b_{i+51} \oplus b_{i+38} \oplus b_{i+23} \oplus b_{i+13} \oplus b_i$.
3. Discard the first 160 bits.
4. Evaluate bits in pairs: If the first bit is a 1, output the second bit. If it is a 0, discard the second bit.

Using this method, the generation of round constants depends on the specific instance, and thus different round constants are used even if some of the chosen parameters (e.g., n and t) are the same.

If a randomly sampled integer is bigger than (or equal to) p, we discard this value and take the next one. Note that cryptographically strong randomness is not needed for the round constants, and other methods can also be used. We give both the matrices and the round constants in auxiliary files for three example instantiations:

- "x^3 – Poseidon"-Permutation in $\mathbb{F}_p$ with $p = 2^{64} - 2^8 - 1$, $n = 64$, $t = 24$, $N = 1536$,
- "x^{-1} – Poseidon"-Permutation in $\mathbb{F}_p$ with $p = 2^{252} + 27742317777372353535851937790883648493$, $n = 253$, $t = 6$, $N = 1518$,
- "x^3 – Starkad"-Permutation in $\mathbb{F}_{2^n}$ with $p(x) = x^{63} + x + 1$, $n = 63$, $t = 24$, $N = 1512$.

We also make reference implementations for various instantiations and scripts to calculate the round numbers, the round constants, and the MDS matrices available online[15].

5.1 Domain Separation for Starkad-Hash and Poseidon-Hash

For some of our use cases, we require independent hash functions, i.e., different instances of our initial hash function. This can be done using domain separation in the following way. We define the original hash function to be Starkad-Hash$_{0000}$, which has an initial state of $(0^r \mid\mid 0000 \mid\mid 0^{c-4})$. Using this technique, we specify Starkad-Hash$_i$ to be a hash function with $(0^r \mid\mid i \mid\mid 0^{c-4})$ as its

[15] `https://github.com/anonymous-ccs/material-ccs`

initial sponge state, where $0 \leq i \leq 15$. The same approach is applied to Poseidon-Hash, resulting in 16 different hash functions for Starkad-Hash and Poseidon-Hash each. Note that we essentially increase the rate r by 4 bits, while at the same time reducing the capacity c by 4 bits, which means that we lose a small amount of security.

5.2 Merkle Tree Instances of Poseidon and Starkad

As a hash function used in a Merkle tree of a fixed arity always gets a message input of bounded length, it makes sense to have a compact padding of input elements. Concretely, we suggest the following:

- Poseidon and Starkad instances with width t are used for Merkle trees with arity $t - c$ where c is the capacity (at word-level, namely c elements in $\mathbb{F}$). As the hash output is equal in size to the capacity, we should have $c|(t-c)$. In the prime field setting with 128-bit security and 256-bit field we will have $c = 1$ and arity $t - 1$.
- A tree node may have from 0 to $t - c$ child elements. The missing child element is denoted by $\emptyset$ and we denote $\widehat{\mathbb{F}} = \mathbb{F} \cup \{\emptyset\}$.
- A node hash function $\widehat{H}$ maps $\widehat{\mathbb{F}}^{t-c}$ to $\mathbb{F}^c$. Therefore, a missing subtree of depth 1 (a single node) is represented as $\emptyset$, a missing subtree of depth 2 has the hash $\widehat{H}^2_\emptyset = \widehat{H}(\emptyset, \emptyset, \ldots, \emptyset)$, and a missing subtree of depth d has the hash

$$\widehat{H}^d_\emptyset = \widehat{H}(\widehat{H}^{d-1}_\emptyset, \widehat{H}^{d-1}_\emptyset, \ldots, \widehat{H}^{d-1}_\emptyset).$$

- The node hash function $\widehat{H}$, based on the permutation Π of width t, is defined as follows:

$$\widehat{H}(X_{c+1}, X_{c+2}, \ldots, X_t) = \Pi_{c+1\ldots 2c}(\widetilde{X}_1, \widetilde{X}_2, \widetilde{X}_3, \ldots, \widetilde{X}_t),$$

  where $\Pi_{c+1\ldots 2c}$ are the first c non-capacity elements of the output of Π and

$$\widetilde{X}_1 = \ldots = \widetilde{X}_{c-1} = 0, \quad \widetilde{X}_c = \sum_i 2^i [X_i \neq \emptyset]; \quad \widetilde{X}_{i>c} = \begin{cases} X_i, & X_i \neq \emptyset; \\ 0, & X_i = \emptyset \end{cases}$$

  where [] is the Iverson bracket (1 if the input is true, 0 otherwise).

Sponge Padding. For a variable-length sponge instance of Poseidon and Starkad, we pad all message strings from $\mathbb{F}^*$ with a single element $1 \in \mathbb{F}$ and then, if necessary, with as many zero elements as needed to have the message length be a multiple of $t - c$.

6 Poseidon and Starkad in Zero-Knowledge Proof Systems

Our hash functions have been designed to be friendly to zero-knowledge applications. Concretely, we aim to minimize proof generation time, proof size, and verification time (when it varies). Before presenting concrete results, we make a small overview of ZK proof systems to date.

State of the Art. Let $\mathcal{P}$ be a circuit over some finite field $\mathbb{F}$ where gates are some (low-degree) polynomials over $\mathbb{F}$ with I and O being input and output variables, respectively: $\mathcal{P}(I) = O$. The *computational integrity problem* consists of proving that some given O_0 is the result of the execution of $\mathcal{P}$ over some I_0: $\mathcal{P}(I_0) = O_0$. It is not difficult to show that any limited-time program on a modern CPU can be converted to such a circuit [9], and making the proof zero-knowledge is often possible with little overhead. The seminal PCP series of papers states that for any program $\mathcal{P}$ it is possible to construct a proof of computational integrity, which can be verified in time sublinear in the size of $\mathcal{P}$. However, for long time the prover algorithms were so inefficient that this result remained merely theoretical. Only recently, proof systems where the prover costs are polynomial in $|\mathcal{P}|$ were constructed, but they required a trusted setup: a verifier or someone else (not the prover) must process the circuit with some secret s and output a reference string S, used both by the prover and the verifier. In this setting, the prover' work can even be made linear in $|\mathcal{P}|$, and the verifier's costs are constant. Such systems were called SNARKs for proof succinctness. The first generation of SNARKs, known as Pinocchio and Groth16 [43,26], require a separate trusted setup for each circuit. The next generation, which includes Sonic [38], Plonk [22], and Marlin [18], can use one reference string of size d for all circuits with at most d gates, thus simplifying the setup and its reuse. Later on, proof systems without trusted setups appeared, of which we consider Bulletproofs [14], STARKs [8], and RedShift [32] the most interesting, though all of them come with deficiencies: Bulletproofs have linear verifier times (but rather short proofs), STARKs work with iterative programs, and RedShift has big proofs (up to 1 MB for millions of gates).

Current benchmarks demonstrate that programs with millions of gates can be processed within a few seconds with the fastest proof systems, which solves the computational integrity problem for some practical programs. Among them, privacy-preserving cryptocurrencies, mixers, and private voting are prominent examples. Shortly, such applications work as follows:

1. Various users add publicly hashes of some secret and public values to some set V, which is implemented as a Merkle tree. Hashes can be currency transaction digests, public keys, or other credentials.
2. Only those who know a secret behind some hash are declared eligible for an action (e.g., to vote or to spend money);

3. A user who wants to perform the action proves that they know a tree leaf L and a secret K such that L is both the hash of K and a leaf in V. If the proof passes, the user is allowed to perform and action (e.g., to vote). If an action must be done only once, a deterministic hash of the secret and leaf position can be computed and published.

This paradigm is behind the cryptocurrency Zcash and the Ethereum mixer Tornado[16].

The bottleneck of such a system is usually the proof creation time, which took 42 seconds in the early version of Zcash, and sometimes the verifier's time. Both are determined by the size of the circuit that describes a Merkle proof and are thus dependent on the hash function that constitutes the tree.

Unfortunately, a single hash function cannot be optimal for all ZK proof systems, because they use different arithmetizations: STARKs can use prime and binary fields, Bulletproofs uses any prime field, whereas most SNARKs use a prime field based on a scalar field of a pairing-friendly elliptic curve. Therefore, for each proof system a new instance of Starkad or Poseidon may be needed. In the future text we describe how this is done and how to optimize a circuit for some proof systems.

6.1 SNARKs with Poseidon$^{\pi}$

In SNARKs, the prime field is typically the scalar field of some point on a pairing-friendly elliptic curve. The primitive Poseidon$^{\pi}$ can be represented as such a circuit with reasonably few gates, but the parameters of Poseidon$^{\pi}$ must have been determined first by p. Concretely, after p is fixed, we first check if x^3 or x^5 are bijections in $GF(p)$, which is true if $p \bmod 3 \neq 1$ (resp., $p \bmod 5 \neq 1$). If both inequalities are not satisfied, we have to use the inverse S-Box or consider another prime power for the S-box.

Groth16

Groth16 [26] is an optimization of the Pinocchio proof system and to date the fastest SNARK with the smallest proofs. The Groth16 prover complexity is $O(s)$ where s is the number of rank-1 constraints – quadratic equations of the form $(\sum_i u_i X_i)(\sum_i v_i X_i) = \sum_i w_i X_i$, where u_i, v_i, w_i are field elements and X_i are program variables. It is easy to see that the S-Box x^3 is represented by 2 constraints, the S-Box x^5 by 3 constraints, and the S-Box $1/x$ by 3 constraints (1

[16] `https://medium.com/@tornado.cash/introducing-private-transactions-on-ethereum-now-42ee915babe0`

for non-zero case, and two more for the zero case). Thus, in total we have

$$2tR_F + 2R_P \text{ constraints for } x^3\text{-based } \textsc{Poseidon}^\pi; \tag{5}$$

$$3tR_F + 3R_P \text{ constraints for } x^5\text{-based } \textsc{Poseidon}^\pi; \tag{6}$$

$$3tR_F + 3R_P \text{ constraints for } x^{-1}\text{-based } \textsc{Poseidon}^\pi. \tag{7}$$

It requires a bit more effort to see that we do not need more constraints as the linear layers and round constants can be incorporated into these ones. However, it is necessary to do some preprocessing. For example, in the $\textsc{Poseidon}^\pi$ setting, the full S-Box layers are followed by linear transformation $M = (M_{i,j})$. Each round with a full S-Box layer can be represented by the following constraints in the SNARK setting:

$$\left(\sum_j M_{i,j}x_{i,j}\right)\cdot\left(\sum_j M_{i,j}x_{i,j}\right) = y_i \;\; 1 \le i \le t, \tag{8}$$

$$y_i \cdot \left(\sum_j M_{i,j}z_{i,j}\right) = z_i, \tag{9}$$

where $M = I_{t\times t}$ for the first round. However, in a round with a partial S-Box layer, we will have only one such constraint for $j = 1$. For the rest of the $t - 1$ variables we will have linear constraints of the form

$$\sum_j M_{i,j}x_{i,j} = u_i \;\; \text{, where } 2 \le i \le t.$$

Since the linear constraints have little complexity effect in the Groth16, in the partial S-Box rounds linear constraints can be composed with the ones from the previous round(s) using the following equation:

$$\sum_k M_{i,k}\left(\sum_j M_{i,j}x_{i,j}\right) = v_k \;\; 2 \le k \le t.$$

We can now calculate the number of constraints for the sponge mode of operation and for Merkle trees. In sponges, the $2M$ bits are reserved for the capacity, so $N - 2M$ bits are fed with the message. Therefore, we get

- $\frac{2tR_F+2R_P}{N-2M}$ constraints per bit for x^3-based $\textsc{Poseidon}^\pi$;
- $\frac{3tR_F+3R_P}{N-2M}$ constraints per bit for x^5-based $\textsc{Poseidon}^\pi$;
- $\frac{3tR_F+3R_P}{N-2M}$ constraints per bit for x^{-1}-based $\textsc{Poseidon}^\pi$.

For the Merkle tree, we suggest a 1-call sponge where all branches must fit into the rate. Then a Merkle tree has arity $\frac{N}{2M} - 1$. Based on that we can calculate how many constraints we need to prove the opening in a Merkle tree of, for example, 2^{32} elements (the recent ZCash setting). The tree will have $\frac{32}{\log_2[N/(2M)-1]}$ levels

Table 2: `libsnark`[1] performance of Poseidon-Hash with S-box x^5 over prime field and 127-bit collision resistance.

Arity	Width	libsnark ZK proof time for one hash		R1CS constraints
		Prove	Verify	
2:1	3	57.5ms	1.2ms	344
3:1	4	84.0ms	1.2ms	507
8:1	9	120.2ms	1.3ms	834

Table 3: Number of constraints for a circuit proving a Merkle tree entry in the set of 2^{30} elements, with the S-box x^5 over a suitable prime field with 128-bit collision resistance.

Poseidon				
Arity	Width	R_F	R_P	Total constraints
2:1	3	8	55	7110
4:1	5	8	56	4320
8:1	9	8	57	3870
Pedersen hash				
510	171	-	-	43936
Rescue				
2:1	3	22	-	11880
4:1	5	14	-	6300
8:1	9	10	-	5400

with the number of constraints in each according to the above. The libsnark performance of Poseidon-Hash is given in table 2.

As an example, we calculate the concrete number of constraints for a Merkle tree proof, where the tree has 2^{30} elements, assuming a security level of 128 bits and a prime field of size close to 2^{256}. We take the S-Box equal to x^5 as it fits many prime fields: Ristretto[17], BN254, and BLS12-381 scalar fields. The results are in Table 3.

Bulletproofs

Bulletproofs [14] is a proof system that does not require a trusted setup. It is notable for short proofs which are logarithmic in the program size, and also for the shortest range proofs that do not require a trusted setup. However, its verifier is linear in the program size. For the use cases where the trusted setup is not an option, the Bulletproofs library 'dalek' is among the most popular ZK primitives.

[17] `https://ristretto.group`

Table 4: Bulletproofs performance to prove 1 out of 2^{30}-Merkle tree.

Field	Arity	Merkle 2^{30}-tree ZK proof Bulletproofs time		R1CS Constraints
		Prove	Verify	
Poseidon hash				
BLS12-381	2:1	16.8s	1.5s	7110
	4:1	13.8s	1.65s	4320
	8:1	11s	1.4s	3870
BN254	2:1	11.2s	1.1s	7110
	4:1	9.6s	1.15s	4320
	8:1	7.4s	1s	3870
Ristretto	2:1	8.4s	0.78s	7110
	4:1	6.45s	0.72s	4320
	8:1	5.25s	0.76s	3870

We have implemented[18] a Merkle tree prover for Poseidon in Bulletproofs using the same constraint system as for Groth16 with results outlined in Table 4. The performance varies since the underlying curves are based on prime fields of different size and weight: BN254 uses a 255-bit prime whereas BLS12-381 uses a 381-bit one (the reason for that is the recent reevaluation of discrete logarithm algorithms specific to pairing-friendly curves).

Plonk

Plonk [22] is a new SNARK using universal trusted setup, where a structured reference string of size d can be used for any circuit of d gates or less. The setup is pretty simple as for the secret k the values $k^d \cdot B$ are stored, where B is an elliptic curve point and $\cdot$ denotes scalar multiplication. The setup is used for short polynomial commitments and proofs by Kate et al. [31].

The standard version of Plonk works with the same constraint system as we have described, plus it uses special machinery to lay out wires in the circuit. A prover first crafts three polynomials of degree equal to the number of gates, which are responsible for left input, right input, and output, respectively. Then he allocates several supplementary polynomials to describe the wire layout. The prover complexity for a Poseidon permutation with the S-Box x^5 of width w and R rounds is $11(w(w+6)+3)R$ point multiplications, and the proof has 7 group elements and 7 field elements.

As we have almost identical rounds, the Plonk compiler can be heavily optimized. Concretely, we suggest:

[18] `https://github.com/anonymous-ccs/material-ccs/blob/master/bulletproofs/gadget_poseidon.rs`

– Define a separate polynomial for each S-box line;

– Get rid of wire layout polynomials;

– Express round transitions as a system of affine equations over polynomial values at adjacent points.

As a result, our optimized Plonk compiler makes only $(w+11)R$ point multiplications for a single permutation call, whereas the proof consists of $((w+3)$ group elements and $2w$ field elements. This brings a 25-40x increase in performance depending on w.

RedShift

RedShift [32] is a STARK-inspired proof system which works with arbitrary set of constraints. It can be viewed as Plonk with pairing-based polynomial commitments with the trusted setup being replaced by Reed-Solomon trustless commitments. The RedShift proof is $c_\lambda \log d^2$ KB large, where d is the degree of circuit polynomials and $c_\lambda \approx 2.5$ for 120-bit security. Due to similarity, we can make the same optimizations as in Plonk, so that the entire Merkle tree proof requires polynomials of degree 4800 for width 5, resulting in the entire proof being around 12 KB in size. Unfortunately, no RedShift library is publicly available so far, and hence we could not measure the actual performance.

6.2 STARKs with Starkad$^\pi$

ZK-STARKs [8] is a proof system for the computational integrity, which is not vulnerable to quantum computers and does not use a trusted setup. STARKs operate with programs whose internal state can be represented as a set of w registers, each belonging to a binary field $GF(2^n)$ or to a 2^n-subgroup of a prime-order group.

Here $n = 32$ and higher are preferred. The program execution is then represented as a set of T internal states. The computational integrity is defined as the set of all wT registers satisfying certain s polynomial equations (constraints) of degree d.

Before going into the details, we mention the ongoing public "StarkWare hash challenge" competition[19], which aims to evaluate the security of proposed STARK-Friendly Hash (SFH) candidates including Starkad and Poseidon. The challenge is proposed at four security levels: low security, medium security, target security, and high security in multiple scenarios.

[19] `https://starkware.co/hash-challenge/`

STARK Costs. According to [36], the number of constraints does not play a major role in the prover, verifier, or communication complexity, which are estimated as follows:

$$\text{Prover Operations in } GF(2^n) = 8w \cdot T \cdot d \cdot \log(wT), \tag{10}$$

$$\text{Prover Memory} = \Omega(w \cdot T \cdot n), \tag{11}$$

$$\text{Communication} = \text{Verifier Time} = n \cdot (m + \log^2(8Td)), \tag{12}$$

where m is the maximum number of variables in a constraint polynomial.

The primitive Starkad^π can be represented as such a program with few registers and number of steps and low degree. For x^3 to be invertible, n has to be odd, so we select $n = 63$ for our primary instance of Starkad^π to be close to 64 bits in order to efficiently utilize the carry-less multiplication (CLMUL) instruction set available in recent CPUs to speed up finite field operations. Following the same approach as for SNARKs in Section 6.1, we keep in registers only S-Box inputs and the permutation outputs. Setting $w = t$, we get $T = R_F + \lceil R_P/t \rceil$ and $wT = tR_F + R_P$. Thus, the complexity is calculated as follows:

$$\text{Prover Operations in } GF(2^n) = 24(tR_F + R_P) \cdot \log_2(tR_F + R_P), \tag{13}$$

$$\text{Prover Memory} = \Omega(63 \cdot (tR_F + R_P)), \tag{14}$$

$$\text{Communication} = \text{Verifier Time} = 63 \cdot (t + \log_2^2(24(tR_F + R_P))). \tag{15}$$

We are flexible in choosing the number of S-Boxes t. This number can be chosen by the application, for instance a Merkle tree with arity a and 128-bit security would require $t = 4a + 4$. For example, if we choose the binary tree then $t = 12$, which requires to set $R_F = 8, R_P = 48$ (security level of $M = 128$ bits) to both protect from known attacks and have a reasonable security margin. In the sponge setting, we reserve 4 S-Boxes for the capacity in the 128-bit security level, and 8 S-Boxes for the capacity in the 256-bit security level. Thus, for our primary instance $\text{Starkad} - 256$, we get an AET cost of 24779 in $GF(2^{63})$ for each permutation call. As we process 504 bits per call, we obtain a prover complexity of 49 operations per bit.

Vision. For fair comparison, we include the AET costs of Vision in the same methodology. The authors of [4] describe a set of AIR constraints of width $2t$ (with t being the number of S-boxes) and degree 2 for half a round. For the S-box of size 63 and 2:1 compression rate and 128-bit security we need 12 S-boxes and 10 rounds so the prover costs are $8 \cdot 24 \cdot 20 \cdot \log(24 \cdot 20) = 34202$.

Acknowledgements. This work is partially supported by the Ethereum foundation, Starkware Ltd, and IOV42 Ltd.

References

1. C++ library for zkSNARK, https://github.com/scipr-lab/libsnark
2. Abdelraheem, M.A., Ågren, M., Beelen, P., Leander, G.: On the Distribution of Linear Biases: Three Instructive Examples. In: CRYPTO 2012. LNCS, vol. 7417, pp. 50–67 (2012)
3. Albrecht, M.R., Cid, C., Grassi, L., Khovratovich, D., Lüftenegger, R., Rechberger, C., Schofnegger, M.: Algebraic Cryptanalysis of STARK-Friendly Designs: Application to MARVELlous and MiMC. In: Advances in Cryptology – ASIACRYPT 2019. LNCS, vol. 11923, pp. 371–397 (2019)
4. Aly, A., Ashur, T., Ben-Sasson, E., Dhooghe, S., Szepieniec, A.: Design of symmetric-key primitives for advanced cryptographic protocols. Cryptology ePrint Archive, Report 2019/426 (2019), https://eprint.iacr.org/2019/426
5. Ashur, T., Dhooghe, S.: Marvellous: a stark-friendly family of cryptographic primitives. Cryptology ePrint Archive, Report 2018/1098 (2018), https://eprint.iacr.org/2018/1098
6. Bardet, M., Faugere, J., Salvy, B., Yang, B.: Asymptotic behaviour of the index of regularity of quadratic semi-regular polynomial systems. In: The Effective Methods in Algebraic Geometry Conference (MEGA). pp. 1–14 (2005)
7. Beierle, C., Canteaut, A., Leander, G., Rotella, Y.: Proving Resistance Against Invariant Attacks: How to Choose the Round Constants. In: CRYPTO 2017. LNCS, vol. 10402, pp. 647–678 (2017)
8. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Scalable, transparent, and post-quantum secure computational integrity. IACR Cryptology ePrint Archive **2018**, 46 (2018)
9. Ben-Sasson, E., Chiesa, A., Tromer, E., Virza, M.: Succinct non-interactive zero knowledge for a von neumann architecture. In: USENIX Security Symposium. pp. 781–796. USENIX Association (2014)
10. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: On the Indifferentiability of the Sponge Construction. In: EUROCRYPT 2008. LNCS, vol. 4965, pp. 181–197 (2008)
11. Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystems. Journal of Cryptology **4**(1), 3–72 (1991)
12. Biham, E., Shamir, A.: Differential Cryptanalysis of the Data Encryption Standard. Springer (1993)
13. Boura, C., Canteaut, A., De Cannière, C.: Higher-Order Differential Properties of Keccak and *Luffa*. In: FSE 2011. LNCS, vol. 6733, pp. 252–269 (2011)
14. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: IEEE Symposium on Security and Privacy. pp. 315–334. IEEE Computer Society (2018)
15. Camenisch, J., Kohlweiss, M., Soriente, C.: An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In: Public Key Cryptography. LNCS, vol. 5443, pp. 481–500. Springer (2009)
16. Camenisch, J., Lysyanskaya, A.: Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. In: CRYPTO 2002. LNCS, vol. 2442, pp. 61–76. Springer (2002)
17. Carlet, C., Charpin, P., Zinoviev, V.A.: Codes, Bent Functions and Permutations Suitable For DES-like Cryptosystems. Designs, Codes Cryptography **15**(2), 125–156 (1998)

18. Chiesa, A., Hu, Y., Maller, M., Mishra, P., Vesely, N., Ward, N.: Marlin: Preprocessing zksnarks with universal and updatable srs. Cryptology ePrint Archive, Report 2019/1047 (2019), https://eprint.iacr.org/2019/1047
19. Cox, D.A., Little, J., O'Shea, D.: Ideals, varieties, and algorithms - an introduction to computational algebraic geometry and commutative algebra (2. ed.). Undergraduate texts in mathematics, Springer (1997)
20. Daemen, J., Knudsen, L.R., Rijmen, V.: The Block Cipher Square. In: FSE 1997. LNCS, vol. 1267, pp. 149–165 (1997)
21. Dinur, I., Kales, D., Promitzer, A., Ramacher, S., Rechberger, C.: Linear Equivalence of Block Ciphers with Partial Non-Linear Layers: Application to LowMC. In: Advances in Cryptology – EUROCRYPT 2019. LNCS, vol. 11476, pp. 343–372 (2019)
22. Gabizon, A., Williamson, Z.J., Ciobotaru, O.: PLONK: permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. IACR Cryptology ePrint Archive **2019**, 953 (2019)
23. Grassi, L.: Mixture Differential Cryptanalysis: a New Approach to Distinguishers and Attacks on round-reduced AES. IACR Trans. Symmetric Cryptol. **2018**(2), 133–160 (2018)
24. Grassi, L., Lüftenegger, R., Rechberger, C., Rotaru, D., Schofnegger, M.: On a Generalization of Substitution-Permutation Networks: The HADES Design Strategy. Cryptology ePrint Archive (2019), accepted at EUROCRYPT'20
25. Grassi, L., Rechberger, C., Rønjom, S.: A New Structural-Differential Property of 5-Round AES. In: EUROCRYPT 2017. LNCS, vol. 10211, pp. 289–317 (2017)
26. Groth, J.: On the size of pairing-based non-interactive arguments. In: EUROCRYPT 2016. LNCS, vol. 9666, pp. 305–326. Springer (2016)
27. Hell, M., Johansson, T., Maximov, A., Meier, W.: The Grain Family of Stream Ciphers. In: The eSTREAM Finalists, LNCS, vol. 4986, pp. 179–190. Springer (2008)
28. Hopwood, D., Bowe, S., Hornby, T., Wilcox, N.: Zcash protocol specification: Version 2019.0-beta-37 [overwinter+sapling]. Tech. rep., Zerocoin Electric Coin Company (2019), available at https://github.com/zcash/zips/blob/master/protocol/protocol.pdf
29. Jakobsen, T., Knudsen, L.R.: The Interpolation Attack on Block Ciphers. In: FSE 1997. LNCS, vol. 1267, pp. 28–40 (1997)
30. Jean, J., Naya-Plasencia, M., Peyrin, T.: Multiple Limited-Birthday Distinguishers and Applications. In: SAC 2013. LNCS, vol. 8282, pp. 533–550 (2013)
31. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: ASIACRYPT. Lecture Notes in Computer Science, vol. 6477, pp. 177–194. Springer (2010)
32. Kattis, A., Panarin, K., Vlasov, A.: Redshift: Transparent snarks from list polynomial commitment iops. Cryptology ePrint Archive, Report 2019/1400 (2019), https://eprint.iacr.org/2019/1400
33. Knudsen, L.R.: Truncated and Higher Order Differentials. In: FSE 1994. LNCS, vol. 1008, pp. 196–211 (1994)
34. Lamberger, M., Mendel, F., Rechberger, C., Rijmen, V., Schläffer, M.: Rebound Distinguishers: Results on the Full Whirlpool Compression Function. In: ASIACRYPT 2009. LNCS, vol. 5912, pp. 126–143 (2009)
35. Leander, G., Abdelraheem, M.A., AlKhzaimi, H., Zenner, E.: A Cryptanalysis of PRINTcipher: The Invariant Subspace Attack. In: CRYPTO 2011. LNCS, vol. 6841, pp. 206–221 (2011)

36. Ltd, S.I.: The complexity of STARK-friendly cryptographic primitives. Private communication (2018)
37. MacWilliams, F.J., Sloane, N.J.A.: The Theory of Error-Correcting Codes. North-holland Publishing Company (1978)
38. Maller, M., Bowe, S., Kohlweiss, M., Meiklejohn, S.: Sonic: Zero-knowledge snarks from linear-size universal and updateable structured reference strings. Cryptology ePrint Archive, Report 2019/099 (2019), https://eprint.iacr.org/2019/099
39. Matsui, M.: Linear Cryptanalysis Method for DES Cipher. In: EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397 (1993)
40. Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: The Rebound Attack: Cryptanalysis of Reduced Whirlpool and Grøstl. In: FSE 2009. LNCS, vol. 5665, pp. 260–276 (2009)
41. Nyberg, K.: Differentially uniform mappings for cryptography. In: EUROCRYPT 1993. LNCS, vol. 765, pp. 55–64 (1994)
42. Nyberg, K., Knudsen, L.R.: Provable Security Against Differential Cryptanalysis. In: CRYPTO 1992. LNCS, vol. 740, pp. 566–574 (1992)
43. Parno, B., Howell, J., Gentry, C., Raykova, M.: Pinocchio: Nearly practical verifiable computation. In: IEEE Symposium on Security and Privacy. pp. 238–252. IEEE Computer Society (2013)
44. Rijmen, V., Daemen, J., Preneel, B., Bosselaers, A., Win, E.D.: The cipher SHARK. In: Fast Software Encryption – FSE 1996. LNCS, vol. 1039, pp. 99–111. Springer (1996)
45. Youssef, A.M., Mister, S., Tavares, S.E.: On the Design of Linear Transformations for Substitution Permutation Encryption Networks. In: School of Computer Science, Carleton University. pp. 40–48 (1997)

SUPPLEMENTARY MATERIAL

A Efficient Implementation

Like for LowMC, the fact that the non-linear layer is partial in R_P rounds can be used to reduce the size of the round constants required in each round R_P. Referring to [21], we recall here an equivalent representation of an SPN with partial non-linear layer for an efficient implementation.

Round Constants. In the description of an SPN, it is possible to swap the order of the linear layer and the round constant addition as both operations are linear. The round constant then needs to be exchanged with an equivalent one. For round constant $c^{(i)}$, the equivalent one can be written as $\hat{c}^{(i)} = MC^{-1}(c^{(i)})$, where MC is the linear layer in the i-th round. If one works with partial non-linear layers, it is possible to use this property to move parts of the original round constants from the last round all the way through the permutation to the beginning. To arrive at such a reduced variant, we work as following:

- First, we find an equivalent round constant that is applied before the affine layer.
- Then we split the round constants in two parts, one that applies to the S-Box part of the non-linear layer and one that applies to the identity part of the non-linear layer. The constant part that only applies to the non-linear layer part can now move further up.
- Working in this way for all round constants, we finally end up with an equivalent representation in which round constants are only added to the output of the S-Boxes apart from one constant which is applied to the entire state after the first R_f rounds.

This simplified representation can in certain cases also reduce the implementation cost of an SPN permutation with a partial non-linear layer. For instance, the standard representation of HadesCubic requires constants matrices of total size $t \cdot n \cdot (R + 1)$, where $R = R_P + R_F$ is the number of rounds. The optimized representation only requires $t \cdot n \cdot (R_F + 1) + n \cdot R_P$, thus potentially greatly reducing the amount of needed memory and calculation to produce the round constants.

Linear Layer. For our design the situation is simpler than for LowMC, since we can guarantee the existence of invertible sub matrices. Hence, a similar trick can be used also for the matrix multiplication.

Focusing on the rounds with a single S-Box, let M be the $t \times t$ MDS matrix of the linear layer:

$$M = \left[\begin{array}{c|ccc} M_{0,0} & M_{0,1} & \cdots & M_{0,t} \\ \hline M_{1,0} & & & \\ M_{2,0} & & & \\ \vdots & & \hat{M} & \\ M_{t-1,0} & & & \\ M_{t,0} & & & \end{array}\right] \equiv \left[\begin{array}{c|c} M_{0,0} & v \\ \hline w & \hat{M} \end{array}\right]$$

where $\hat{M}$ is a $(t-1) \times (t-1)$ MDS matrix (note that since M is MDS, every submatrix of M is also MDS), v is a $1 \times (t-1)$ matrix and w is a $(t-1) \times 1$ vector. By simple computation, the following equivalence holds:

$$M = \underbrace{\left[\begin{array}{c|c} 1 & 0 \\ \hline 0 & \hat{M} \end{array}\right]}_{M'} \times \underbrace{\left[\begin{array}{c|c} M_{0,0} & v \\ \hline \hat{w} & I \end{array}\right]}_{M''}, \tag{16}$$

where

$$\hat{w} = \hat{M}^{-1} \times w$$

and I is the $(t-1) \times (t-1)$ identity matrix. Note that both M' and M'' are two invertible matrices[20].

As for the round constants discussed previously, it is possible to use the equivalence (16) in order *to swap the S-Box layer (formed by a single S-Box and $t-1$ identity functions) and the matrix multiplication with the matrix M'*. As a result, each linear part in the R_P rounds is defined only by a multiplication with a matrix of the form M'', which is a *sparse matrix*, since $(t-1)^2-(t-1) = t^2-3t+2$ coefficients of M'' are equal to zero (moreover, $t-1$ coefficients of M'' are equal to one). It follows that this optimized representation potentially greatly reduces the number of operations needed to compute the linear layer multiplication.

B Security Analysis – STARKAD and POSEIDON with S-Box(x) = x^3

B.1 Security Analysis – Statistical Attacks

Differential Cryptanalysis. Differential cryptanalysis [11,12] and its variations are the most widely used techniques to analyze symmetric-key primitives. The differential probability of any function over the finite field $\mathbb{F}_{2^n}$ is defined as

$$Prob[\alpha \rightarrow \beta] := |\{x : f(x) \oplus f(x \oplus \alpha) = \beta\}|/(2^n).$$

[20] First of all, $\det(M') = \det(\hat{M}) \neq 0$ since $\hat{M}$ is an MDS matrix, and so it is invertible. Secondly, $\det(M) = \det(M') \cdot \det(M'')$. Since $\det(M) \neq 0$ and $\det(M') \neq 0$, it follows that $\det(M'') \neq 0$.

Since the cubic function $f(x) = x^3$ is an almost perfect non-linear permutation (APN) [42,41], it has an optimal differential probability over a prime field or $\mathbb{F}_{2^n}$ (where n is odd). In other words, for this function the probability is bounded above by $2/2^n$ or $2/|\mathbb{F}_p|$.

As largely done in the literature, we claim that STARKAD and POSEIDON are secure against differential cryptanalysis if each differential has probability at most 2^{-N}. Since it is in general hard to compute the probability of a differential, we assume that this fact is satisfied if each characterestic has probability at most $2^{-2 \cdot N}$. In order to compute the minimum number of rounds to guarantee this, we work only with the rounds with full S-Box layers. In other words, we limit ourselves to work with a "weaker" version of the permutation defined as

$$R^{R_f} \circ L \circ R^{R_f}(\cdot), \tag{17}$$

where

- L is an *invertible linear layer* (which is the "*weakest*" possible assumption),
- $R(\cdot) = M \circ \text{S-Box} \circ ARK(\cdot)$ where S-Box$(\cdot)$ is a full S-Box layer (remember that M is an MDS matrix).

We are going to show that this "weaker" permutation is secure against differential cryptanalysis for $R_F = 2R_f = 10$. As a result, it follows that also STARKAD/POSEIDON (instantiated with R_F rounds with full S-Box layers) is secure against such an attack. Indeed, if the linear layer L (which we only assume to be invertible) is replaced by R_P rounds of STARKAD/POSEIDON, its security cannot decrease. *The same strategy is exploited in the following in order to prove security against all attacks in this subsection.*

In order to prove the result just given, we need a lower bound on the number of minimum number of active S-Boxes. Observe that the minimum number of "active" S-Boxes in the permutation

$$R^s \circ L \circ R^r(\cdot) \equiv SB \circ \underbrace{M \circ SB}_{s-1 \text{ times}} \circ \underbrace{L'}_{\equiv L \circ M(\cdot)} \circ SB \circ \underbrace{M \circ SB}_{r-1 \text{ times}}(\cdot)$$

(where $s, r \geq 1$, $R(\cdot)$ is a round with full S-Box layer and where L' is an invertible linear layer) are at least[21]

$$\text{number } \textit{active} \text{ S-Boxes} \geq \underbrace{\big(\lfloor s/2 \rfloor + \lfloor r/2 \rfloor\big) \cdot (t+1)}_{\text{due to final/initial rounds}} + \big(s \bmod 2\big) + \big(r \bmod 2\big).$$

We emphasize that the (middle) linear $L'(\cdot) \equiv L \circ M(\cdot)$ plays *no* role in the computation of the previous number. Since at least $2 \cdot (t+1)$ S-Boxes are active

[21] If $s = 2 \cdot s'$ is even, then the minimum number of active S-Boxes over $R^s(\cdot)$ rounds with full S-Box layer is $\lfloor s/2 \rfloor \cdot (t+1)$. Instead, if $s = 2 \cdot s' + 1$ is odd, then the minimum number of active S-Boxes over $R^s(\cdot)$ rounds with full S-Box layer is $\lfloor s/2 \rfloor \cdot (t+1) + 1$.

in the 4 middle rounds of $R^2 \circ L \circ R^2(\cdot)$, and since the maximum differential probability of the cubic S-Box is $DP_{max} = 2^{-n+1}$, each characteristic has probability at most

$$(2^{-n+1})^{2\cdot(t+1)} = \begin{cases} 2^{-2N} \cdot 2^{-2n+2t+2} \leq 2^{-2N} & \text{if } n \geq t+1, \\ 2^{-1.25\cdot N} \cdot 2^{-0.75\cdot N - 2n + 2t + 2} < 2^{-1.25\cdot N} \end{cases}$$

(where the second inequality holds since $0.75 \cdot N + 2n \geq 2t + 2$) for each $t \geq 2$ and $n \geq 3$ (note that $0.75 \cdot N + 2t = n \cdot (0.75t + 2)$). By doubling this number of rounds, we get that each charaterestic has probability at most $2^{-2.5\cdot N}$. Finally, 1 more round guarantees that no differential attack can be set up.

Security up to $2^M \leq 2^N$. For completeness, we present the number of rounds necessary to provide security up to 2^M (that is, data and computational cost of the attacker upper bounded by 2^M). Using the same analysis as before, it turns out that

$$R_F = \begin{cases} 6 & \text{if } t+1 \leq N+n-M \\ 10 & \text{otherwise} \end{cases}$$

guarantees that no differential attack can be set up.

Linear Cryptanalysis. Similar to differential attacks, linear attacks [39] pose no threat to the Starkad/Poseidon families of permutations instantiated with the same number of rounds previously defined for classical differential cryptanalysis. This follows from the fact that the cubic function is almost bent (AB), which means that its maximum square correlation is limited to 2^{-n+1} (see [2] for details). As a result, it offers the best possible resistance against linear cryptanalysis much like an APN function provides optimal resistance against differential cryptanalysis.

For completeness, we remember a function $f(\cdot)$ is AB and/or APN if and only if its inverse $f^{-1}(\cdot)$ is AB and/or APN [17]. As a result, both the forward and the inverse permutation are secure against linear and differential cryptanalysis[22].

Truncated Differential. A variant of classical differential cryptanalysis is the truncated differential one [33], in which the attacker can specify only part of the difference between pairs of texts.

We consider the "weaker" permutation described in (17) again. Focusing only on active/passive bytes (and not on the actual differences), there exist several differentials with probability 1 for a maximum of 1 round of Starkad/Poseidon, e.g.

$$[\alpha, 0, ..., 0]^T \xrightarrow{R(\cdot)} M \times [\beta, 0, ..., 0]^T$$

[22] Remember that if a matrix M is MDS, then also M^{-1} is MDS.

where α, β denote non-zero differences. Due to the next S-Box layer, the linear relations given by $M \times (\beta, 0, ..., 0)^T$ are destroyed in the next round. As a result, no probability-one truncated differential covers more than a single round.

Since no linear relation survives the S-Box layer, it seems hard to set up a truncated differential for more than 2 rounds. As a result, it turns out that 4 rounds with full S-Box layer makes HadesCubic$^\pi$ secure against this attack.

Rebound Attacks. The rebound attacks [34,40] have much improved the best known attacks on many hash functions, especially for AES-based schemes. The goal of this attack is to find two (input, output) pairs (p^1, c^1) and (p^2, c^2) such that the two inputs satisfy a certain (truncated) input difference and the corresponding outputs satisfy a certain (truncated) output difference.

The rebound attack consists of two phases, called *inbound* and *outbound* phase. According to these phases, the internal permutation of the hash function is split into three sub-parts. Let f be the permutation, then we get $f = f_{fw} \circ f_{in} \circ f_{bw}$. The part of the inbound phase is placed in the middle of the permutation and the two parts of the outbound phase 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. Since the rebound attack is a differential attack, as first thing an attacker needs to construct a *"good" (truncated) differential trail*. A good trail used for a rebound attack should have a high probability in the outbound phases and can have a rather low probability in the inbound phase. In the first phase, the attacker uses the knowledge of the key to find pairs of texts that satisfy the middle rounds of the truncated differential trail. In the second one, they propagate the solutions found in the first phase in the forward and in the backward directions, and check if at least one of them satisfies the entire differential trail.

The best rebound attack on AES proposed in [30] covers 8 rounds. Here we claim that 6 rounds with full S-Box layers are sufficient to protect Starkad/Poseidon from this attack. To support it, note that *(1st)* 1 round of Starkad/Poseidon provides full diffusion while 2 rounds of AES are necessary to provide it and *(2nd)* the best truncated differential covers 1 round of Starkad/Poseidon *vs* 3 rounds of AES[23]. Since the best results on AES in the literature cover at most 8 rounds, due to the similarity between AES and Starkad/Poseidon and due to the previous observations, we argue that it is not possible to mount a rebound attack on more than 5 rounds with full S-Box layers of Starkad/Poseidon. Hence, 6 rounds of Starkad/Poseidon with full S-Box layers are sufficient to guarantee security against this attack.

Multiple-of-n and Mixed Differential Cryptanalysis. The "Multiple-of-8" distinguisher [25] was proposed at Eurocrypt 2017 by Grassi *et al.* as the

[23] The best truncated differential distinguisher with prob. 1 covers 2 rounds of AES.

first 5-round secret-key distinguisher for AES that exploits a property which is independent of the secret key and of the details of the S-Box. It is based on a new structural property for up to 5 rounds of AES: by appropriate choices of a number of input pairs it is possible to make sure that the number of times that the difference of the resulting output pairs lie in a particular subspace is always a multiple of 8. The input pairs of texts that satisfy a certain output difference are related by linear/differential relations. Such relations are exploited by a variant of such a distinguisher, called the "mixture differential" distinguisher [23] proposed at FSE/ToSC 2019.

Regarding STARKAD/POSEIDON, it is possible to set up such distinguishers on 2 rounds only. In particular, consider a set of texts with $2 \leq s \leq t$ active words (and $t - s$ constants words). The number of pairs of texts that satisfy an (arbitrary) output truncated differential is always a multiple of 2^{s-1}. Moreover, the relations of the input pairs of texts exploited by mixture differential cryptanalysis are known. The proofs of these two properties are analogous to the ones proposed in [25] and in [23]. E.g., consider two texts T^1 and T^2 of the form

$$T^1 = C \oplus \left[x_0, x_1, 0, ..., 0\right]^T, \qquad T^2 = C \oplus \left[y_0, y_1, 0, ..., 0\right]^T$$

for some constant C and where $x_i \neq y_i$ for $i = 0, 1$. After one round, the difference in each word is of the form

$$M_0 \cdot [\mathrm{SB}(x_0 \oplus c_0) \oplus \mathrm{SB}(x_1 \oplus c_1)] \oplus M_1 \cdot [\mathrm{SB}(y_0 \oplus c_0) \oplus \mathrm{SB}(y_1 \oplus c_1)],$$

where M_0, M_1 depend on the MixLayer and c_0, c_1 depend on the secret key. By simple observation, the same output difference is given by the pair of texts

$$\hat{T}^1 = C \oplus \left[y_0, x_1, 0, ..., 0\right]^T, \qquad \hat{T}^2 = C \oplus \left[x_0, y_1, 0, ..., 0\right]^T.$$

Combining this result with a 1-round truncated differential with prob. 1, it is possible to set up a multiple-of-n distinguisher (where $n = 2^{s-1}$) and a mixture differential one on 2 rounds of STARKAD/ POSEIDON. Using the inside-out approach, it is possible to set up such attack on 4-round of STARKAD/POSEIDON. As a result, it turns out that 6 rounds with full S-Box layers make it secure against these attacks.

Invariant Subspace Attack. The invariant subspace attack [35] makes use of affine subspaces that are invariant under the round function. As the round constant addition translates this invariant subspace [7], random round constants provides a good protection against such attacks.

Integral/Square Attack. Integral cryptanalysis is a technique first applied on SQUARE [20] and is particularly efficient against designs based on substitution-permutation networks, like AES or STARKAD/POSEIDON.

The idea is to study the propagation of sums of values. For the case of Starkad/Poseidon, it is possible to set up an integral distinguisher over two rounds, e.g.

$$\begin{bmatrix} A \\ C \\ \dots \\ C \end{bmatrix} \xrightarrow{\text{S-Box}(\cdot)} \begin{bmatrix} A \\ C \\ \dots \\ C \end{bmatrix} \xrightarrow{M(\cdot)} \begin{bmatrix} A \\ A \\ \dots \\ A \end{bmatrix} \xrightarrow{\text{S-Box}(\cdot)} \begin{bmatrix} A \\ A \\ \dots \\ A \end{bmatrix} \xrightarrow{M(\cdot)} \begin{bmatrix} B \\ B \\ \dots \\ B \end{bmatrix}$$

where A denotes an active word, C a constant one and B a balanced one[24]. Using the inside-out approach, it is possible to set up such attack on 4-round of HadesCubic$^\pi$. As a result, it turns out that 6 rounds with full S-Box layers make HadesCubic$^\pi$ secure against this attack.

B.2 Security Analysis – Algebraic Attacks

First we introduce a simple lemma, which follows from the iterative structure of the HadesCubic permutation.

Lemma 1. *The algebraic degree $D_3(r)$ of r-round* Starkad/Poseidon *with S-Box x^3 as a function of input and, optionally, key variables is at most 3^r, no matter if partial or full rounds are used.*

Interpolation Attack. One of the most powerful attacks is the interpolation attack, introduced by Jakobsen and Knudsen [29] in 1997. In the case of a keyed function $E_k : \mathbb{F}_{2^N} \to \mathbb{F}_{2^N}$, the strategy of the attack is to construct a polynomial representation of the function without knowledge of the secret key. If an adversary can construct such a polynomial then it can compute any output without knowing the key, thus enabling forgeries (for MAC settings) and other attacks. The interpolation polynomial $P(x)$ representing $E_k(x)$ can be constructed using e.g. the Vandermonde matrix – *cost* approximately of $\mathcal{O}(t^2)$ – or the Lagrange's theorem – *cost* approximately of $\mathcal{O}(t \cdot \log t)$, where x is the indeterminate corresponding to the input.

Such attack can be opportunely modified for the case of an unkeyed permutation $E(\cdot)$. In such a case, assume it is possible to construct the interpolation polynomial without using the full code-book. In this case, such a polynomial can be exploited to set up a forgery attack on the permutation E, which –in general – is instead not possible for a (pseudo-)random permutation.

In more details, each output word of an SPN permutation can be represented as a multivariate polynomial where the variables are the inputs to each S-Box.

[24] For completeness, we recall that given a set of texts $\{x_i\}_{i \in I}$, the word x^j is *active* if $x_i^j \neq x_l^j$ for each $i \neq l$, constant if $x_i^j = x_l^j$ for each i, l, and balanced if $\bigoplus_i x_i^j = 0$.

Consider a permutation input where χ input words are unknown to us, and the other $t - \chi$ words are known:

$$\chi \text{ unknown input words} \qquad \text{and} \qquad t - \chi \text{ known input words.}$$

A (rough) estimation of the number of monomials of the interpolation polynomial (and so of the complexity of the attack) is given by

$$(D_3(r) + 1)^{\chi},$$

As a result, by requiring that the number of monomials be close to the number of possible input values $2^{\chi n}$, the number of rounds must be at least $r \simeq n \cdot \log_3(2)$.

However, just reaching the full degree is not sufficient to prevent the interpolation attack. First, the polynomial should be dense to guarantee that most monomials occur in it. As showed in [24], the interpolation polynomial is dense when working in $\mathbb{F}_p$. The situation is instead different when working in $\mathbb{F}_{2^n}$, where one needs at least $1 + \lceil \log_3(2^n - 1) \rceil + \lceil \log_2(t) \rceil$ rounds in order to guarantee that E_k is *dense*.

Since S-Box$^{-1}(x) = x^{1/3} = x^{(2^{n+1}-1)/3}$ has an higher degree than S-Box$(x) = x^3$, we do not expect the attack performs better when considering the backward direction instead of the forward one.

Secondly, we consider the algebraic degree not at round r but at round $r - 1$ to account for partial S-Box case where the degree increase is delayed for $t - 1$ words by 1 round. As a result, the total number of rounds R must satisfy [25]

$$R \geq 1 + \left\lceil n \cdot \log_3(2) \right\rceil + \Phi(t)$$

to thwart the interpolation attack where

$$\Phi(t) = \begin{cases} \log_2(t) & \text{working in } \mathbb{F}_{2^n} \\ \log_3(t) & \text{working in } \mathbb{F}_p \end{cases}$$

Security up to $2^M \leq 2^N$. For completeness, we present the number of rounds necessary to provide security up to 2^M (that is, data and computational cost of the attacker upper bounded by 2^M).

Using the same argumentation given before, the number of rounds must satisfy

$$(3^{r-\Phi(t)-1} + 1)^{\chi} \approx 2^{\min(M, n\cdot\chi)}$$

that is $r \geq 1 + \Phi(t) + \min\{n, M/\chi\} \cdot \log_3(2)$. The maximum number of attacked rounds is achieved for $\chi = 1$. As a result, we have $R_P + R_F \geq \left(1 + \left\lceil \log_3(2) \cdot \min(M, n) \right\rceil\right) + \Phi(t)$.

[25] We emphasize that in this analysis we do not take into account the cost to construct the interpolation polynomial, which is (in general) non-negligible.

Gröbner Basis Attack. We consider the Gröbner Basis Attack in the same setting: some permutation inputs are unknown and the rest are known to the attacker. Given some words of the permutation output, they have to find the unknowns.

For generic systems, the complexity of computing a Gröbner basis for a system of $\mathfrak{N}$ polynomials f_i in $\mathfrak{V}$ variables is $\mathcal{O}\left(\binom{\mathfrak{V}+D_{reg}}{D_{reg}}^{\omega}\right)$ operations over the base field $\mathbb{F}$ [19], where D_{reg} is the *degree of regularity* and $2 \leq \omega < 3$ is the linear algebra constant. We note that the memory requirement of these algorithms is of the same order as the running time. The degree of regularity depends on the degrees of the polynomials d and the number of polynomials $\mathfrak{N}$. When $\mathfrak{V} = \mathfrak{N}$, we have the simple closed form

$$D_{reg} := 1 + \sum_{i=0}^{\mathfrak{N}-1} (d_i - 1), \tag{18}$$

where d_i is the degree of the i-th polynomial f_i in the polynomial system we are trying to solve (see [6] for details). In the over-determined case, i.e., $\mathfrak{V} < \mathfrak{N}$, the degree of regularity can be estimated by developing the Hilbert series of an ideal generated by generic polynomials $\langle f_0, \ldots, f_{\mathfrak{N}-1} \rangle$ of degrees d_i (under the assumption that the polynomials behave like generic systems). Closed-form formulas for D_{reg} are known for some special cases, but not in general.

Full-permutation equations. In the first case we derive equations, one by word, for the entire r-round permutation. We consider the case when the number χ of unknown input variables equals the number of known output variables. Then we get χ equations of degree $D_3(r) = 3^r$ of χ variables, so the degree of regularity is

$$D_{reg} = 1 + \chi(3^r - 1) = 3^r - \chi + 1.$$

The attack complexity can be estimated by

$$\binom{\mathfrak{V} + D_{reg}}{D_{reg}}^2 \approx \binom{\chi 3^r}{\chi}^2 \approx \frac{(3^r)^{2\chi} e^{2\chi}}{2\pi\chi},$$

where 2π is due to Stirling's approximation. If we target a security level of M bits, the number of rounds to be attacked is calculated as

$$\begin{aligned} &\frac{(3^r)^{2\chi} e^{2\chi}}{2\pi\chi} \leq 2^{\min(M, n\chi)} \\ &\implies (3^r)^{2\chi} e^{2\chi} \leq 2\pi\chi 2^{\min(M, n\chi)} \\ &\implies r \leq \frac{\log_2(2\pi\chi)}{2\chi \log_2(3)} - \frac{\log_2(e)}{\log_2(3)} + \frac{\min(M, n\chi)}{2\chi \log_2(3)}. \end{aligned}$$

Since the maximum number of attacked rounds is achieved for $\chi = 1$ and since $\frac{\log_2(2\pi)}{2\log_2(3)} - \frac{\log_2(e)}{\log_2(3)} < 0$, the security is provided by chosen

$$r > \frac{\min(M, n)}{2\log_2(3)} = \frac{1}{2\log_2(3)} \cdot \min(M, n) \leq 0.32 \cdot \min(M, n).$$

Equations for each S-Box. Here we consider equations of degree 3 for each S-Box, which relate its inputs and outputs. Given χ unknown permutation inputs and χ known outputs, we get $(t-1)R_F + R_P + \chi$ unknown S-Boxes, and for each we use 1 variable (for its input). In total, we get $(t-1)R_F + R_P$ equations for the S-Box inputs in all rounds, and χ equations for the last-round outputs. Denoting $q = (t-1)R_F + R_P + \chi$, the degree of regularity is $D_{reg} = 1 + 2q$. The attack complexity can be estimated by

$$\binom{\mathfrak{V} + D_{reg}}{D_{reg}}^2 \approx \binom{3q}{q}^2 = \left(\frac{(3q)!}{q! \cdot (2q)!}\right)^2 \approx \frac{2^{5.5q}}{\left(\frac{4\pi q}{3}\right)} \approx 2^{5.4q},$$

where we used Stirling's approximation for the factorials. We also note that the last approximation only holds true for $q \geq 85$. As q denotes the number of variables used in the attack – which is the same as the number of S-boxes in our construction – this bound is sufficient to provide security for our proposed instantiations. On the other hand, we do not claim that this approximation provides security for every possible instantiation, in particular when using comparatively small state sizes. We will apply the same technique later when evaluating the security of x^5-Poseidon and x^{-1}-Poseidon.

If we target a security level of M bits, the number of rounds to be attacked is calculated as

$$\begin{aligned} & 2^{5.4((t-1)R_F + R_P + \chi)} \leq 2^{\min(M, n\chi)} \\ \implies & 5.4((t-1)R_F + R_P + \chi) \leq \min(M, n\chi) \\ \implies & (t-1)R_F + R_P + \chi \leq 0.18 \cdot \min(M, n\chi). \end{aligned}$$

Since the maximum number of rounds to be attacked is achieved for $\chi = 1$, the security is provided by chosen

$$(t-1)R_F + R_P > 0.18 \cdot \min(M, n) - 1.$$

Combining the two strategies together, we get the following conditions:

$$R_F + R_P \geq 0.32 \cdot \min(M, n), \tag{19}$$

$$(t-1)R_F + R_P \geq 0.18 \cdot \min(M, n) - 1. \tag{20}$$

Higher-Order Differential Attack. A well-known result from the theory of Boolean functions is that if the algebraic degree of a vectorial Boolean function $f(\cdot)$ (like a permutation) is d, then the sum over the outputs of the function applied to all elements of a vector space $\mathcal{V}$ of dimension $\geq d+1$ is zero (as is the sum of all inputs, i.e., the elements of the vector space). The same property holds for affine vector spaces of the form $\{v + c \,|\, v \in \mathcal{V}\}$ for arbitrary constant c

$$\bigoplus_{v \in \mathcal{V} \oplus c} v = \bigoplus_{v \in \mathcal{V} \oplus c} f(v) = 0.$$

This is the property exploited by higher-order diff. attack [33].

Working at word level, the number of rounds (R_F+R_P) given by the interpolation attack provides security also against higher-order differential attacks. Indeed, for the interpolation attack it is required that the degree d after r rounds satisfies $d \geq 2^N$. Instead, for higher-order differentials (working at word level), it is sufficient that $d \geq N + 1$. The conclusion follows immediately.

What happens working – instead – on a bit level? To prevent such attacks, ideally we would like to be able to make a statement such as "After r rounds there is no output bit and no input subspace of dimension d' s.t. the derivative of the polynomial representation of the output bit with respect to this subspace is the zero polynomial." To achieve such a goal, we need to estimate the *growth of the boolean degree*. First of all, the degree of the S-Box $f(x) = x^3$ in its algebraic representation in $\mathbb{F}_{2^n}$ is only 2. Thus, clearly the boolean degree of the permutation after r rounds is bounded from above by 2^r. It is furthermore generally bounded from above by $N - 1$ as it is a permutation.

However, it turns out that the boolean degree grows slower than expected because the monomial x^{2^k} is a linear transformation in $\mathbb{F}_2^n$, and a high degree in $\mathbb{F}_{2^n}$ may not imply a high degree in $\mathbb{F}_2^n$. Nevertheless we assume[26] that the boolean degree of f is at least q if f over $\mathbb{F}_{2^n}$ contains a monomial x^d where d has Hamming weight q. From the interpolation attack details we know that after $r + \log_2 t$ rounds the polynomial of f is dense and thus contains most of the monomials of degree 3^r and smaller. We now recall that for integer d there are at least $\log d$ integers smaller than d with Hamming weight $\lfloor \log_2 d \rfloor - 1$. Therefore for a polynomial in 1 variable, that is dense up to degree d, the boolean degree is at least $\lfloor \log_2 d \rfloor - 1$. For a polynomial in χ variables that is dense up to total degree d, we can find a monomial with degree up to d/χ in each variable, so the boolean degree would be $\chi \cdot (\lfloor \log_2(d/\chi) \rfloor) - \chi$. Thus if bits in χ input words are unknown, the boolean degree after r rounds can be lower bounded as $\chi \cdot (\lfloor \log_2(3^{r-\log_2 t}/\chi) \rfloor) - \chi$. As long as the degree is smaller than $\min(M, \chi n)$, we get a valid attack. Therefore we have the condition for the number of attacked rounds:

$$\log_2(3^{r-\log_2 t}/\chi) - 2 \leq \min\{M/\chi, n\}.$$

For $M < n$ the maximum number of rounds is reached for $\chi = 1$, whereas for $M \geq n$ the maximum is reached for $\chi = M/n$. Eventually we get that at most

$$r = 0.63 \min(M, n) + 2 + \log_2 t$$

rounds can be attacked.

[26] This assumption is due a recent result found by C. Cid, M. Eichlseder, L. Grassi, R. Lüftenegger, C. Rechberger, M. Schofnegger and Q. Wang – *Private Communication.*

Higher-Order Differential Attacks on $\mathbb{F}_p$. Here we emphasize an important difference between the higher-order differential attack on $\mathbb{F}_{2^n}$ and on $\mathbb{F}_p$. Given a function $f(\cdot)$ of degree d, the sum over the outputs of the function applied to all elements of a vector space $\mathcal{V}$ of dimension $\geq d+1$ is zero.

The crucial point here is that the previous result holds if $\mathcal{V}$ is a (sub)space, and not only a generic set of elements. While $\mathbb{F}_{2^m}$ is always a subspace of $\mathbb{F}_{2^n}$ for each $m \leq n$, the only subspaces of $\mathbb{F}_p$ are $\{0\}$ and $\mathbb{F}_p$. It follows that the biggest subspace of $(\mathbb{F}_p)^t$ has dimension t, with respect to the biggest subspace of $(\mathbb{F}_{2^n})^t$, which has dimension $n \cdot t = N$.

As a result, in the case in which a permutation is instantiated over $\mathbb{F}_p$, a lower degree (and hence a smaller number of rounds) is sufficient to protect it from the higher-order differential attack with respect to the number of rounds for the $\mathbb{F}_{2^n}$ case. In more details, the number of rounds necessary to protect our design against the interpolation attack are sufficient in order to guarantee security against this attack also.

Zero-Sum Distinguishers. The fact that some inner primitive in a hash function has a relatively low degree can often be used to construct higher-order diff. distinguishers, or *zero-sum structures*. This direction has been investigated e.g. in [13] for two SHA-3 candidates, Luffa and Keccak. More generally, a zero-sum structure for a function $f(\cdot)$ is defined as a set Z of inputs z_i that sum to zero, and for which the corresponding outputs $f(z_i)$ also sum to zero, i.e. $\bigoplus_i z_i = \bigoplus_i f(z_i) = 0$. For an iterated function, the existence of zero sums is usually due either to the particular structure of the round function or to a low degree. 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.

By using the *inside-out* technique, here we investigate the minimum number of rounds of $x^3 - \text{POSEIDON}^\pi$ sufficient to prevent zero-sum structures.

Definition 1 (Zero-sum Partition [13]). *Let P be a permutation from $\mathbb{F}_{2^n}$ to $\mathbb{F}_{2^n}$. A zero-sum partition for P of size $K = 2^k < 2^n$ is a collection of 2^k disjoint sets $\{X_1, X_2, ..., X_k\}$ with the following properties:*

- $X_i = \{x_1^i, ..., x_{2^{n-k}}^i\} \subset \mathbb{F}_{2^n}$ *for each* $i = 1, ..., k$ *and* $\bigcup_{i=1}^{2^{n-k}} X_i = \mathbb{F}_{2^n}$,
- $\forall i = 1, ..., 2^k$: *the set X_i satisfies zero-sum* $\bigoplus_{j=1}^{2^k} x_j^i = \bigoplus_{j=1}^{2^k} P(x_j^i) = 0$.

We focus on creating zero-sum partitions of the permutation $P(\cdot)$ of the form $P(\cdot) = R_r \circ ... \circ R_1(\cdot)$, where all R_i are permutations over $\mathbb{F}_2^n$. Remember that for the permutation in a hash function, one can exploit any state starting from an intermediate state. Thus, assume one can find a set of texts $X = \{x_i\}_i$ and

a set of texts $Y = \{y_i\}_i$ with the property $\bigoplus_i R_{r-1} \circ ... \circ R_{s+1}(y_i) = 0$ and $\bigoplus_i R_s \circ ... \circ R_1(x_i) = 0$ for a certain s. Working with the intermediate states (remember that there is no secret material), the idea is to choose texts in $X \bigoplus Y$: the inputs p_i are defined as the $(r-s)$-round decryptions of $X \bigoplus Y$, while the corresponding outputs c_i are defined as the s-round encryptions of $X \bigoplus Y$. This results into a zero-sum partition $\{p_i\}$ for the permutation P.

To avoid such an attack, we require that $R_{r-1} \circ ... \circ R_{s+1}(\cdot)$ and $R_s \circ ... \circ R_1(\cdot)$ have maximum degree. About the forward direction of the permutation, one can simply reuse the result already proposed for the higher-order differential discussed in the previous section, i.e. we need $0.63 \cdot M + \log_2 t$ rounds to achieve the full boolean degree.

About the backward direction we limit ourselves to recall here that the algebraic degree of S-Box$(x) = x^{1/3}$ (i.e. the inverse S-Box) is $(n+1)/2$. This implies that 2 rounds are sufficient to prevent such attack in this direction.

Proposition 1. *The algebraic degree of S-Box$^{-1}(x) = x^{1/3} = x^{(2^{n+1}-1)/3}$ is $(n+1)/2$ (remember that n is odd).*

Proof. We prove this result by induction. For $n = 3$, it follows that S-Box$^{-1}(x) = x^{1/3} = x^5$. Since $x^5 = x^4 \cdot x$ and since x^4 is a linear operation in $GF(2^n)$, the result follows immediately.

Assume the result is true for $n - 1 = 2n' + 1$. Here we show that it works for $n = 2n' + 3$. Observe that

$$\frac{2^{n+1}-1}{3} = \frac{2^{2n'+4}-1}{3} = 2^{2n'+2} + \frac{2^{2n'+2}-1}{3}$$

thus

$$x^{\frac{2^{n+1}-1}{3}} = x^{2^{2n'+2}} \cdot x^{\frac{2^{2n'+2}-1}{3}}.$$

Since the exponent of the first term on the r.h.s. is a power of 2, it is linear in $GF(2^n)$. By the induction assumption, the second term has algebraic degree $(n-1)/2$. It follows that the algebraic degree is $(n+1)/2$.

C Security Analysis – x^5-Poseidon

For some practical applications, we need to work with a prime p s.t. $p = 1 \mod 3$. Since the cubic function x^3 is invertible if and only if $p = 2 \mod 3$, we need to change the S-Box for this particular case.

We decided to work with S-Box$(x) = x^5$ which is invertible if and only if $p \neq 1 \mod 5$. Since the analysis for this case is similar to the one just given for the cubic case, we limit ourselves here to briefly discuss the number of rounds necessary to guarantee security in GF(p) (due to our target application).

C.1 Statistical Attacks

Differential Cryptanalysis. As before, HadesFifth$^{\pi}$ instantiated by S-Box$(x) = x^5$ is secure against statistical attacks if and only if

$$R_F^{stat} \geq 6.$$

The main difference here is due to differential and linear attacks. In particular, since[27] $DP_{max}(\text{S-Box}(x) = x^5) = 4/p$ (or equivalently 2^{-n+2} in $\mathbb{F}_{2^n}$), it follows that the minimum number of rounds necessary to guarantee security against linear and differential attacks is given by

$$R_F = \begin{cases} 6 & \text{if } 2t + 2 < N + \lceil \log_2(p) \rceil - M \\ 10 & \text{if } 2t + 2 \geq N + \lceil \log_2(p) \rceil - M \end{cases}$$

for a security level up to $2^M \leq 2^N$ (that is, in the case in which the data and the computational cost of the attacker is upper bounded by 2^M).

Linear Cryptanalysis. Similar considerations hold for linear cryptanalysis.

Rebound Attacks. Due to the same argumentation in order to provide security of HadesCubic instantiated by S-Box$(x) = x^3$ against the rebound attack, 6 rounds provide security also HadesFifth instantiated by S-Box$(x) = x^5$ against the rebound attack.

C.2 Algebraic Attacks

Interpolation Attack. Due to the previous analysis, the number of rounds necessary to prevent the interpolation attack is given by

$$R_F + R_P \geq R^{inter}(N, t) \equiv 1 + \left\lceil \log_5(2) \cdot \min\left\{ \lceil \log_2(p) \rceil; M \right\} \right\rceil + \lceil \log_5(t) \rceil$$

working in $\mathbb{F}_p$. In particular, note that the degree of the encrypted function after r rounds is well approximated by 5^{r-1}, and where $\log_5(t)$ more rounds are necessary to guarantee that the polynomial is sparse. Since the degree of S-Box$^{-1}(x) = x^{1/5}$ is much higher:

$$\frac{1}{5} \mod p \equiv \begin{cases} \frac{4 \cdot p - 3}{5} & \text{if } p \mod 5 = 2 \\ \frac{2 \cdot p - 1}{5} & \text{if } p \mod 5 = 3 \\ \frac{3 \cdot p - 2}{5} & \text{if } p \mod 5 = 4 \end{cases}$$

the same number of rounds guarantee security in the case in which the attacker is performed in the decryption direction.

[27] Note that $(x + \Delta_I)^5 - x^5 = \Delta_O$ is an equation of degree 4, hence there are at most 4 different solutions.

Higher-Order Diff. Attack. Since we present HadesFifth-Hash instantiated by S-Box$(x) = x^5$ just over GF(p), we refer to previous discussion against higher-order diff. attacks over $\mathbb{F}_p$, and we limit ourselves to remember that the number of rounds necessary to guarantee security against the interpolation attack is also also sufficient to guarantee security against higher-order diff. attacks.

Gröbner Basis Attack. Using exactly the same analysis as for x^3-Poseidon, we get the following conditions:

$$R_F + R_P \geq 0.21 \cdot \min(M, n), \tag{21}$$

$$(t-1)R_F + R_P \geq 0.14 \cdot \min(M, n) - 1. \tag{22}$$

D AET Complexity of Friday

Friday [5] is a recent STARK-friendly symmetric hash function introduced by Ashur and Dhooghe and based on a new block cipher Jarvis. It is presented in several instances with different security levels. Friday-128 offers 64 bits of collision resistance, and Friday-256 offers 128 bits of collision resistance.

Here we compute the numbers for Friday in order to compare it with our design. Note, however, that Jarvis and consequently Friday are vulnerable to algebraic attacks [3] and thus the stated round number would have to be increased in order to restore security. This would also result in a higher cost in this setting.

The proposed version of Friday uses 10 and 14 rounds for a block size of 128 and 256 bits, respectively. An inverse S-Box (in $GF(2^{128})$, $GF(2^{256})$) is followed by two transformations of degree 4 in the field and a constant addition. We have to use 5 registers per round: S-Box input, S-Box output, a temporary register to store the information if the input is zero, the output of the first degree-4 transformation, and the round constant. All these variables are linked by constraints of degree not more than 4. One can optimize it by adding two more intermediate variables for the linear transformations and reducing the degree to 2. In total, we get $T = 10(14), w = 7, d = 2$:

$$\text{Prover Operations in Friday-128} = 8 \cdot 7 \cdot 2 \cdot 10 \log 70 = 6865 \tag{23}$$

$$\text{Prover Operations in Friday-256} = 8 \cdot 7 \cdot 2 \cdot 14 \log 98 = 10372 \tag{24}$$

or 54 (respectively, 41) operation per bit. Note that these operations are done in bigger fields than our $GF(2^{63})$ so the actual time difference is much bigger.

E Compact Constraints for STARKs and SNARKs

In this section we show how to generate constraints for S-Boxes that depend on only a few variables. This is useful when t is relatively small compared to R_P.

Let us denote the outputs of the ARK transformation in round r by $A_r^1, A_r^2, \ldots, A_r^t$. Let us also denote the inputs to the MDS matrix in round r by $B_r^1, B_r^2, \ldots, B_r^t$. We obtain that in full rounds

$$S(A_r^i) = B_r^i,$$

whereas in partial rounds

$$S(A_r^t) = B_r^t, \quad A_r^i = B_r^i, i < t.$$

The S-Box inputs then will be

$$\begin{aligned} A_r^i \text{ for } i < t: &\quad r \in [1; R_F/2] \cup [R_F/2 + 1 + R_P; R_F + R_P] \\ A_r^t: &\quad r \in [R_F/2 + 1; R_F/2 + R_P]. \end{aligned} \tag{25}$$

It is obvious that the equations above plus the MDS and ARK affine transformations

$$M\mathbf{B}_r + \mathbf{K}_r = \mathbf{A}_{r+1}. \tag{26}$$

fully determine the permutation. Therefore, using $2Rt$ variables $\{A_r^i, B_r^i\}$ we can describe the permutation using constraints of degree equal to the S-Box degree d.

As we have mentioned, one can do better by substituting equations (26) for all B_r^i thus getting equations on only Rt variables $\{A_r^i\}$. In the same way we can get rid of A_r^i that are not inputs to S-Boxes, thus leaving with equations on $R_F t + R_P$ variables. However, this procedure creates equations with too many variables as S-Box outputs in the first partial rounds now depend on the S-Box inputs of all future partial rounds. Our goal is to construct compact equations on the same variables. We are going to work with two consecutive segments of t partial rounds each, let them be rounds from r to $r + 2t - 1$. We proceed as follows:

1. Express A_j^t, $j \in [r + t; r + 2t - 1]$ as affine functions of

$$A_{r+t}^1, A_{r+t}^2, \ldots, A_{r+t}^t, B_{r+t}^t, B_{r+t+1}^t, \ldots, B_{r+2t-1}^t.$$

2. Using Gaussian elimination, express $(A_{r+t}^1, A_{r+t}^2, \ldots, A_{r+t}^t)$ as affine functions of

$$A_{r+t}^t, A_{r+t+1}^t, \ldots, A_{r+2t-1}^t, B_{r+t}^t, B_{r+t+1}^t, \ldots, B_{r+2t-1}^t.$$

3. Express B_j^t, $j \in [r; r + t - 1]$ as affine functions of

$$A_{r+t}^1, A_{r+t}^2, \ldots, A_{r+t}^t, A_{r+1}^t, A_{r+2}^t, \ldots, A_{r+t}^t.$$

4. Using Gaussian elimination, express $(A_{r+t}^1, A_{r+t}^2, \ldots, A_{r+t}^t)$ as affine functions of

$$A_{r+1}^t, A_{r+2}^t, \ldots, A_{r+t}^t, B_r^t, B_{r+1}^t, \ldots, B_{r+t-1}^t.$$

5. Combine items 2 and 4 and get a system of t affine equations that link

$$A^t_{r+t}, A^t_{r+t+1}, \ldots, A^t_{r+2t-1}, B^t_{r+t}, B^t_{r+t+1}, \ldots, B^t_{r+2t-1},$$
$$A^t_{r+1}, A^t_{r+2}, \ldots, A^t_{r+t}, B^t_r, B^t_{r+1}, \ldots, B^t_{r+t-1}.$$

6. Substitute $B^t_j \leftarrow S(A^t_j)$ and get a system of t degree-d polynomial constraints on $(A^t_{r+t}, A^t_{r+t+1}, \ldots, A^t_{r+2t-1}, A^t_{r+1}, A^t_{r+2}, \ldots, A^t_{r+t})$.

The resulting system P_1 of polynomial constraints does not depend on r except for the constant term, which is determined by round constants. We do not have a formal proof that the systems composed at steps 1 and 3 have rank t but experiments show that it is the case for all matrices we tried.

We thus get the following system of constraints for the entire permutation:

- For the first group of full rounds: $t(R_F/2-1)$ constraints of degree d that link $(A^1_r, A^2_r, \ldots, A^t_r, A^1_{r+1}, A^2_{r+1}, \ldots, A^t_{r+1}), r \in [1; R_F/2-1]$. The constraints depend on r in the constant terms only.
- Bridging the last full and t first partial rounds: t constraints of degree d that link $(A^1_{R_F/2}, A^2_{R_F/2}, \ldots, A^t_{R_F/2}, A^t_{R_F/2+1}, A^2_{R_F/2+2}, \ldots, A^t_{R_F/2+t})$.

  To get them, we express $A^t_{R_F/2+1}, A^2_{R_F/2+2}, \ldots, A^t_{R_F/2+t}$ as affine functions of $(A^1_{R_F/2}, A^2_{R_F/2}, \ldots, A^t_{R_F/2}, B^t_{R_F/2},$
  $B^2_{R_F/2+1}, \ldots, B^t_{R_F/2+t-1})$ and then substituting B as degree-d functions.
- For all R_P partial rounds: divide them into groups of t rounds and use the system P_1 (derived above) to link them consecutively, thus R_P-t constraints in total.
- Bridging the last t partial and round $(R_P+R_F/2+1)$ (full one): t constraints of degree d.
- For the last group of full rounds: $t(R_F/2-1)$ constraints of degree d.

This totals to $tR_F + R_P - t$ constraints of degree d.

F Concrete instances for some field sizes

Table 5: A range of different parameter sets for STARKAD$^\pi$ and POSEIDON$^\pi$ instantiated by S-Box$(x) = x^3$ (with security margin).

Security M	**Text Size** $N = n \times t$	**S-Box Size** (n or $\log_2 p$)	**# S-Boxes** (t)	$\mathbf{R_F}$	$\mathbf{R_P}$	**Field**	**Cost** Eq. (4)
128	1536	768	2	8	82	$\mathbb{F}_p$	98
128	1536	384	4	8	83	$\mathbb{F}_p$	115
128	1536	256	6	8	84	$\mathbb{F}_p$	132
128	1536	192	8	8	84	$\mathbb{F}_p$	148
128	1536	96	16	8	64	$\mathbb{F}_p$	192
128	1512	63	24	8	45	$\mathbb{F}_{2^n}$	237
128	1551	33	47	8	25	$\mathbb{F}_{2^n}$	401
128	1581	31	51	8	24	$\mathbb{F}_{2^n}$	432
256	1536	768	2	8	169	$\mathbb{F}_p$	185
256	1536	384	4	8	170	$\mathbb{F}_p$	202
256	1536	256	6	8	170	$\mathbb{F}_p$	218
256	1536	192	8	8	127	$\mathbb{F}_p$	191
256	1536	96	16	8	64	$\mathbb{F}_p$	192
256	1512	63	24	8	45	$\mathbb{F}_{2^n}$	237
256	1551	33	47	8	25	$\mathbb{F}_{2^n}$	401
256	1581	31	51	8	24	$\mathbb{F}_{2^n}$	432

Table 6: A range of different parameter sets for Poseidon$^{\pi}$ instantiated by S-Box$(x) = x^5$ (with security margin).

Security M	**Text Size** $N = n \times t$	**S-Box Size** (n or $\log_2 p$)	**# S-Boxes** (t)	$\mathbf{R_F}$	$\mathbf{R_P}$	**Field**	**Cost** Eq. (4)
128	1536	768	2	8	55	$\mathbb{F}_p$	71
128	1536	384	4	8	56	$\mathbb{F}_p$	88
128	1536	256	6	8	56	$\mathbb{F}_p$	104
128	1536	192	8	8	57	$\mathbb{F}_p$	121
128	1536	96	16	8	43	$\mathbb{F}_p$	171
256	1536	768	2	8	114	$\mathbb{F}_p$	130
256	1536	384	4	8	116	$\mathbb{F}_p$	148
256	1536	256	6	8	116	$\mathbb{F}_p$	164
256	1536	192	8	8	86	$\mathbb{F}_p$	150
256	1536	96	16	8	43	$\mathbb{F}_p$	171