IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2025, No. 2, pp. 743–780. DOI:10.46586/tches.v2025.i2.743-780

Skyscraper: Fast Hashing on Big Primes

Clémence Bouvier[1,2], Lorenzo Grassi[2,3], Dmitry Khovratovich[4,5], Katharina Koschatko[6], Christian Rechberger[6], Fabian Schmid[6] and Markus Schofnegger[7]

[1] Université de Lorraine, CNRS, Inria, LORIA, Nancy, France, clemence.bouvier@inria.fr
[2] Ruhr University Bochum, Bochum, Germany
[3] Ponos Technology, Zug, Switzerland, lorenzo@ponos.technology
[4] Ethereum Foundation, Luxembourg, Luxembourg, khovratovich@gmail.com
[5] ABDK Consulting, Tallinn, Estonia
[6] Graz University of Technology, Graz, Austria, {firstname.lastname}@tugraz.at
[7] Fabric Cryptography, San Francisco, United States, markus.schofnegger@gmail.com

Abstract. Arithmetic hash functions defined over prime fields have been actively developed and used in verifiable computation (VC) protocols. Among those, elliptic-curve-based SNARKs require large (256-bit and higher) primes. Such hash functions are notably slow, losing a factor of up to 1000 compared to regular constructions like SHA-2/3.

In this paper, we present the hash function Skyscraper, which is aimed at large prime fields and provides major improvements compared to Reinforced Concrete and Monolith. First, the design is exactly the same for all large primes, which simplifies analysis and deployment. Secondly, it achieves a performance comparable to cryptographic hash standards by using low-degree non-invertible transformations and minimizing modulo reductions. Concretely, it hashes two 256-bit prime field (BLS12-381 curve scalar field) elements in 135 nanoseconds, whereas SHA-256 needs 42 nanoseconds on the same machine.

The low circuit complexity of Skyscraper, together with its high native speed, should allow a substantial reduction in many VC scenarios, particularly in recursive proofs.

Keywords: hash functions · zero-knowledge · circuits

1 Introduction

The area of computational integrity has seen a dynamic development in the past couple of years, mainly fueled by the need for efficient constructions for verifiable computation [BBHR18, BDFG21, CBBZ23, DP24]. In part, the development has been accelerated by blockchain applications, where often large chunks of transaction executions need to be proven quickly and, most importantly, in a succinct way.

Using new advancements in proving technology, the computational cost for creating these proofs has decreased significantly over the last couple of years, in some cases even by several orders of magnitude. Since this cost is directly related to the funds consumed in the entire process, making these methods faster and even more efficient is a crucial direction in research. In particular, the increased feasibility of these approaches is leading to many new use cases that go significantly further than merely transferring assets on a main chain. For example, this includes verifying data storage with Filecoin [Fil24], efficient rollups, private voting systems, and anonymous credentials for age verification or accessing services.

Today, the research areas in this space can mainly be split into two larger settings, namely, one considering small prime numbers of around 32 or 64 bits and one considering

Received: 2024-10-15 Accepted: 2024-12-15 Published: 2025-03-04

larger prime numbers of around 256 bits. Protocols in the latter setting mostly make use of arguments based on elliptic curve operations [Gro16, BDFG21]. They are also natively used in *on-chain* operations on various L1 blockchains (like Ethereum), being supported by native functions like elliptic curve pairings over BN254.

In many cases, the cryptographic building blocks used in these two settings are different. For example, for protocols based on coding theory arguments like FRI [BBHR18], using smaller finite fields makes sense for increased performance, which is mainly due to faster interpolations and more efficient base field operations for commitments and evaluations in general (assuming use cases where smaller fields are sufficient). This creates new challenges for the corresponding proving systems, such as the need for new fast Fourier transform (FFT) algorithms and efficient hash functions over small finite fields. These issues have been addressed in multiple recent works [HLP24, GKL$^+$24].

In protocols based on elliptic curves, a prover is forced to compute over finite fields whose sizes are proportional to the size of the elliptic curve and, thus, the final security. This means that the underlying base field for e.g. constraint evaluations has to be a comparatively large one in the order of ≈ 256 bits. It becomes then a more challenging task to achieve a fast prover [LWY$^+$23, ZHY$^+$24, LFG23, BH23].

1.1 Hash Functions in Proving Systems

Having established these two areas, it remains to explain the need for efficient, often called "circuit-friendly", hash functions in this context. They are used both directly as a building block for polynomial commitment schemes such as FRI and indirectly as random oracle replacements in various protocols. We give a non-exhaustive overview of hash function applications outside of commitment schemes in the following.

Provable Randomness. In a Fiat-Shamir transformation, challenges are derived from current transcripts in order to make a protocol non-interactive. Using hash functions that instantiate a random oracle to generate these challenges, one may want to prove succinctly that they have been evaluated in an honest way.

Recursion-Based Verifiable Computation. Recursion-based schemes are proofs about previous proofs, where each step ensures the validity of a previous one. A hash function is necessary to compress the inputs to each proof, which otherwise would blow up and make the scheme infeasible. Folding schemes [KST22, KS24] are a classical example of this technique.

Merkle Membership Proofs. Circuit-friendly hash functions over large finite fields can also be used to natively construct Merkle trees with large leaf values and prove openings efficiently using, e.g., pairings over BN254. This can also be done directly on-chain and is significantly cheaper (in terms of gas costs) than any other approach of proving Merkle tree openings with smaller primes (without an additional conversion step from one proving system to another).

Hash-Based ZK Proof Systems. Fast circuit-friendly hash functions over large fields are crucial in various hash-based ZK proof systems, for example when requiring efficient recursion. This includes StarkWare's Stone prover [Sta24] and Fractal [COS20].

Recursion Based on GKR and Similar Protocols. GKR and similar protocols can be used for recursion, and are often used together with circuit-friendly hash functions in this context, as is done e.g. in [BSB23].

In this paper we will focus on the above settings, and we propose a new hash function that is significantly more efficient than other designs [GKR+21, GKL+22], both in terms of plain performance and in terms of circuit performance.

1.1.1 Hash Functions for Large Finite Fields

Regarding hash functions, most efforts in the last couple of years have been put into efficient hash functions and new design strategies for smaller prime fields [GKL+24, SLS+23, Sal23], which is in stark contrast to the requirements of the settings discussed above. While the recent constructions Anemoi [BBC+23] and Griffin [GHR+23] offer suitable instantiations, the security of both of them is questionable for some parameter choices [BBL+24, KLR24]. The only other new construction focusing on this scenario is Reinforced Concrete [GKL+22], arguably the first one making heavy use of the lookup argument [GW20]. However, it is significantly slower (in plain performance) than more classical hash functions, mainly due to a complex interplay between lookup and base field operations in its *decomposition* step.

One may try to adapt recent fast constructions such as Tip5 and Monolith [SLS+23, GKL+24] for large finite fields, but both of them require prime numbers of a special form, which does not match the prime fields we need (e.g., BN254).

1.2 Contribution

We make a significant step forward in the performance of hash functions defined over large prime fields. Our new design Skyscraper runs within a small factor of the SHA-3 standard and is on par with recent designs over small and specially crafted prime fields [GKL+24], see Figure 1. This allows for faster verifiable computation protocols that rely on large prime field arithmetic such as folding schemes [KST22] and earlier STARK schemes like the Stone Prover.[1] Our new design removes the bottleneck of hash computations prevalent in such protocols. In concrete numbers, we achieve the following.

- **Performance.** We do not use expensive modular multiplications but only a few (6) Montgomery reductions and 18 modular additions. The permutation over $\mathbb{F}_p^2$ (and thus the 2-to-1 compression function) runs within 200ns for most 256-bit primes.

- **Circuit Size.** For 256-bit fields, an arithmetic circuit would contain 6 squarings and 128 8-bit lookups.

This performance increase has been achieved via the following ideas.

- Replacing the SPN-based scheme with a Feistel network allows using non-invertible quadratic transformations, whereas the SPN would require power maps of degree 5 and higher [GKR+21].

- High-degree components can also be non-invertible and thus identical for all possible prime domains. This allows defining Skyscraper in a uniform way for any prime.

- The multiplication over the prime field is sped up using the Montgomery fast arithmetic trick. Since the power mapping degree is at most 2, and there is no matrix multiplication, it is possible to apply the Montgomery multiplication universally throughout the scheme.

By focusing on large primes and a small number of elements in the state, we arrive at a simple design that allows for easy third-party scrutiny.

[1] https://github.com/starkware-libs/stone-prover

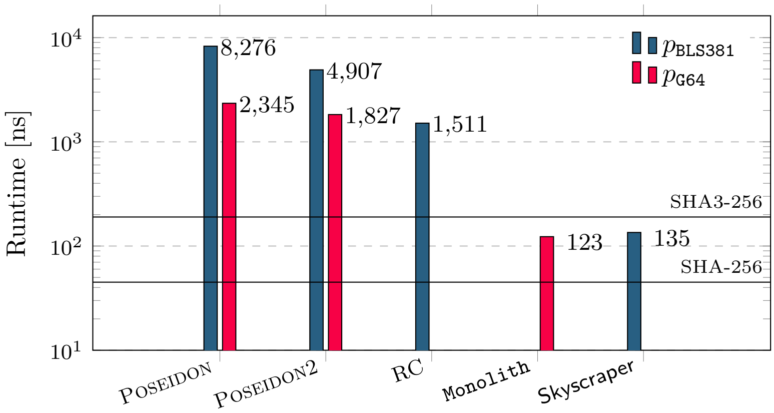


Figure 1: Native performance for arithmetization-friendly hashing of ≈ 500 bits over $\mathbb{F}_p$ for large (BLS12-381, 255 bits) and small (Goldilocks, 64 bits) primes. Setup details in Section 5.

To enhance both the accessibility and reliability of our results, we provide a reference implementation in SageMath.[2] Further, we provide a Rust codebase for benchmarks.[3]

2 Skyscraper

2.1 Domain

The Skyscraper permutation operates on prime fields $\mathbb{F}_p$, where $p \geq 2^{248}$, and with the additional condition that $m := \lceil \log_2(p)/8 \rceil$ is even.[4] We present concrete examples of primes used in practical applications in Table 1.

Let $n \in \{1, 2, 3\}$ be a positive integer. The state consists of two branches. Each branch is an element of the field extension $\mathbb{F}_{p^n} \equiv \mathbb{Z}_p[X]/\mathsf{G}(X)$, where G is an irreducible polynomial of degree n. Hence, the state is an element of $\mathbb{F}_{p^n}^2$. A summary of all parameters is given in Table 2.

Table 1: Some primes p and irreducible polynomials G of degree n considered in our practical evaluations over the field $\mathbb{F}_{p^n} \equiv \mathbb{Z}_p[X]/\mathsf{G}(X)$.

Curve	Prime p	$n = 2$	$n = 3$
BLS12-381	0x73eda753299d7d483339d80809a1d80553bda402fffe5bfeffffffff00000001	$X^2 + 5$	$X^3 + 2$
BN254	0x30644e72e131a029b85045b68181585d2833e84879b9709143e1f593f0000001	$X^2 + 5$	$X^3 + 3$
Pallas	0x40000000000000000000000000000000224698fc094cf91b992d30ed00000001	$X^2 + 5$	$X^3 + 2$
Vesta	0x40000000000000000000000000000000224698fc0994a8dd8c46eb2100000001	$X^2 + 5$	$X^3 + 2$

2.2 Modes of Operation

The Skyscraper permutation can be used within a sponge or in a 2-to-1 compression function.

[2] https://github.com/skyscraper-hash/skyscraper-sage
[3] https://extgit.isec.tugraz.at/krypto/zkfriendlyhashzoo/-/tree/master/plain_impls
[4] This is common for most applications due to performance reasons.

Table 2: Summary of parameters for Skyscraper : $\mathbb{F}_{p^n}^2 \to \mathbb{F}_{p^n}^2$.

p	Security (κ)	n	# Feistel Rounds	Chunk-size s	# Chunks m
$\geq 2^{248}$	124 bits	$\{1, 2, 3\}$	10	8 bits	$\left\lceil \frac{\log_2(p)}{s} \right\rceil$ (even)

Sponge-Based Schemes. First, Skyscraper can instantiate a sponge [BDPV07, BDPV08], and thus various symmetric constructions such as variable-length hash functions, commitment schemes, authenticated encryption, and stream ciphers. The recently proposed SAFE framework [KBM23] instructs how to handle domain separation and padding in these constructions. In a sponge, the permutation state is split into an outer part with a rate of r elements and an inner part with a capacity of c elements. We set $c = 1$, which allows for a 128-bit security level for prime fields of size $\approx 2^{256}$ and larger.

Compression Function. We also suggest a fixed-length 2-to-1 compression function, that can be used in, e.g., Merkle trees. Concretely, it takes $2n$ $\mathbb{F}_p$ elements as input and produces n $\mathbb{F}_p$ elements as output. It is defined as

$$x \in \mathbb{F}_p^{2n} \equiv \mathbb{F}_{p^n}^2 \mapsto \mathrm{Trunc}_n(\mathsf{P}(x) + x) \in \mathbb{F}_p^n \equiv \mathbb{F}_{p^n}, \tag{1}$$

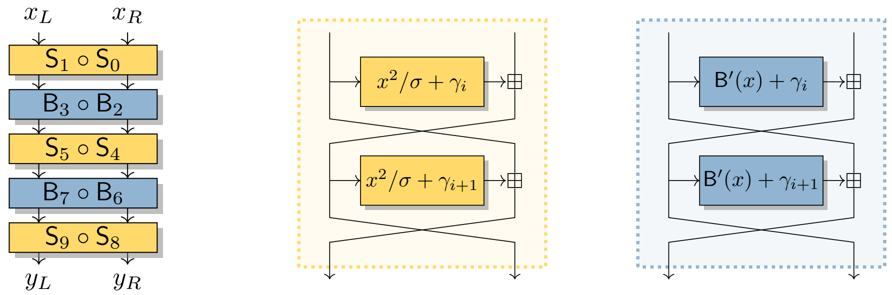
where $\mathrm{Trunc}_n(\cdot)$ yields the first n elements of the input.

2.3 Overview and Round Function

Over $\mathbb{F}_{p^n}^2$, the Skyscraper permutation is defined as a sequence of 10 consecutive functions over $\mathbb{F}_{p^n}^2$ such that

$$(x_L, x_R) \mapsto \mathsf{S}_9 \circ \mathsf{S}_8 \circ \mathsf{B}_7 \circ \mathsf{B}_6 \circ \mathsf{S}_5 \circ \mathsf{S}_4 \circ \mathsf{B}_3 \circ \mathsf{B}_2 \circ \mathsf{S}_1 \circ \mathsf{S}_0(x_L, x_R), \tag{2}$$

where S_i and B_i are illustrated in Figure 2 and defined as follows.


x_L
x_R
$\mathsf{S}_1 \circ \mathsf{S}_0$
$\mathsf{B}_3 \circ \mathsf{B}_2$
$\mathsf{S}_5 \circ \mathsf{S}_4$
$\mathsf{B}_7 \circ \mathsf{B}_6$
$\mathsf{S}_9 \circ \mathsf{S}_8$
y_L
y_R
$x^2/\sigma + \gamma_i$
$x^2/\sigma + \gamma_{i+1}$
$\mathsf{B}'(x) + \gamma_i$
$\mathsf{B}'(x) + \gamma_{i+1}$

(a) Overview of Skyscraper.

(b) Details of $\mathsf{S}_{i+1} \circ \mathsf{S}_i$.

(c) Details of $\mathsf{B}_{i+1} \circ \mathsf{B}_i$.

Figure 2: Overview of the design.

Square Operation. The operation $\mathsf{S}_i : \mathbb{F}_{p^n}^2 \to \mathbb{F}_{p^n}^2$ is a Feistel round with the round function being the squaring operation followed by the addition of a pseudo-random constant γ_i, i.e.,

$$(x_L, x_R) \mapsto \left(x_R + \frac{(x_L)^2}{\sigma} + \gamma_i, x_L \right), \tag{3}$$

where σ is a Montgomery constant s.t. ($2^{256} \mod p$) for $p < 2^{256}$ and ($2^{512} \mod p$) for other p.

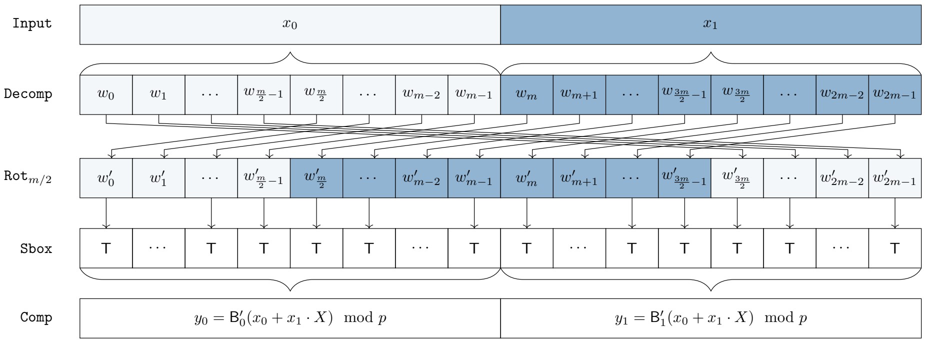

Input
x_0
x_1
Decomp
w_0 w_1 $\cdots$ $w_{\frac{m}{2}-1}$ $w_{\frac{m}{2}}$ $\cdots$ w_{m-2} w_{m-1} w_m w_{m+1} $\cdots$ $w_{\frac{3m}{2}-1}$ $w_{\frac{3m}{2}}$ $\cdots$ w_{2m-2} w_{2m-1}
$\mathtt{Rot}_{m/2}$
w'_0 w'_1 $\cdots$ $w'_{\frac{m}{2}-1}$ $w'_{\frac{m}{2}}$ $\cdots$ w'_{m-2} w'_{m-1} w'_m w'_{m+1} $\cdots$ $w'_{\frac{3m}{2}-1}$ $w'_{\frac{3m}{2}}$ $\cdots$ w'_{2m-2} w'_{2m-1}
Sbox
T $\cdots$ T T T T $\cdots$ T T $\cdots$ T T T T $\cdots$ T
Comp
$y_0 = \mathsf{B}'_0(x_0 + x_1 \cdot X) \mod p$
$y_1 = \mathsf{B}'_1(x_0 + x_1 \cdot X) \mod p$

Figure 3: The Bar layer $\mathsf{B}' : \mathbb{F}_{p^n} \to \mathbb{F}_{p^n}$ for $n = 2$ in detail, including the decomposition, the rotation, the S-box, and the composition.

Bars Operation. The `Bars` operation $\mathsf{B}_i : \mathbb{F}_{p^n}^2 \to \mathbb{F}_{p^n}^2$ is defined as

$$(x_L, x_R) \mapsto (x_R + \mathsf{B}'(x_L) + \gamma_i, x_L), \tag{4}$$

where γ_i is a pseudo-random round constant and the function $\mathsf{B}' : \mathbb{F}_{p^n} \to \mathbb{F}_{p^n}$, illustrated in Figure 3, is defined as follows.

1. `Decomp`. Interpret $x \in \mathbb{F}_{p^n}$ as $(x_0, x_1, \ldots, x_{n-1}) \in \mathbb{F}_p^n$, then split each $x_j \in \mathbb{F}_p$ into $m = \lceil \log_2(p)/8 \rceil$ 8-bit chunks, i.e.,

$$x \mapsto (w_0, w_1, \ldots, w_{n \cdot m-1}) \in (\mathbb{F}_2^8)^m \tag{5}$$

such that $x_j = \sum_{\ell=0}^{m-1} 2^{8(m-1-\ell)} \cdot w_{j \cdot m+\ell}$ for all $0 \leq j < n$.

2. $\mathtt{Rot}_{m/2}$. Apply a cyclic rotation to w by $m/2$ elements such that

$$w'_k \leftarrow w_{(k+m/2) \bmod n \cdot m}, \qquad 0 \leq k < n \cdot m. \tag{6}$$

(Recall that $m/2$ is an integer by assumption on p.)

3. `Sbox`. Apply an (invertible) 8-bit operation T to each w'_k independently:

$$z_k \leftarrow \mathsf{T}(w'_k), \qquad 0 \leq k < n \cdot m, \tag{7}$$

where $\mathsf{T} : \mathbb{F}_2^8 \to \mathbb{F}_2^8$, borrowed from `Monolith` and [Dae95], is defined as

$$\mathsf{T}(v) = \big(v \oplus \big((\overline{v} \lll 1) \odot (v \lll 2) \odot (v \lll 3)\big)\big) \lll 1. \tag{8}$$

4. `Comp`. Map $(z_0, z_1, \ldots, z_{m \cdot n})$ to $(y_0, \ldots, y_{n-1})$ via

$$y_j = \left(\sum_{\ell=0}^{m-1} 2^{8(m-1-\ell)} \cdot z_{j \cdot m+\ell} \right) \bmod p \tag{9}$$

and interpret the vector as $y \in \mathbb{F}_{p^n}$.

Concrete examples for $n \in \{1, 2, 3\}$ are given in Table 3.

Round Constants. We generate the round constants $\gamma_i \in \mathbb{F}_{p^n}$ represented as

$$\gamma_i = \alpha_{i,0} + \alpha_{i,1} \cdot X + \cdots + \alpha_{i,n-1} \cdot X^{n-1}, \tag{10}$$

coefficient-wise for $i \in \{1, \ldots, 8\}$, $j \in \{0, \ldots, n-1\}$ such that

$$\alpha_{i,j} = \text{SHA-256}(\underbrace{(i-1)n + j}_{\text{32-bit, big-endian}} \;||\; \underbrace{\mathsf{Skyscraper}}_{\text{28-byte string}}) \mod p. \tag{11}$$

The round constants γ_0 and γ_9 are set to zero.

Table 3: Examples of $\mathsf{B}' : \mathbb{F}_{p^n} \to \mathbb{F}_{p^n}$ for the 15-bit prime $p = 28657$, $n \in \{1, 2, 3\}$, and $m = 2$. The extension field modulus is $X^n + 5$.

Steps	$n = 1$	$n = 2$	$n = 3$
$x \in \mathbb{F}_{p^n}$	`17cd`	`1e83 + 142b`$\cdot X$	`09ce + 4aae`$\cdot X$ `+ 2d7c`$\cdot X^2$
$x \in \mathbb{F}_p^n$	`(17cd)`	`(1e83,142b)`	`(09ce,4aae,2d7c)`
1. `Decomp`	`(17,cd)`	`(1e,83,14,2b)`	`(09,ce,4a,ae,2d,7c)`
2. `Rot`$_{m/2}$	`(cd,17)`	`(83,14,2b,1e)`	`(ce,4a,ae,2d,7c,09)`
3. `Sbox`	`(d3,0e)`	`(17,28,46,bc)`	`(d9,94,1d,1a,fa,12)`
4. `Comp`	`(631d)`	`(1728,46bc)`	`(69a3,1d1a,1a30)`
$y \in \mathbb{F}_{p^n}$	`631d`	`1728 + 46bc`$\cdot X$	`69a3 + 1d1a`$\cdot X$ `+ 1a30`$\cdot X^2$

2.4 Security Level

We claim at least 124 bits of security against collision and preimage attacks on both the sponge hash function and the 2-to-1 compression function instantiated by the Skyscraper permutation for all primes larger than 2^{248}.[5]

For the inner Skyscraper *permutation* we also claim security against CICO attacks. By definition, a permutation $\mathsf{P} : \mathbb{F}_p^t \to \mathbb{F}_p^t$ is *secure against the CICO attack* if there is no algorithm that finds $x, y \in \mathbb{F}_p^{t-1}$ such that $\mathsf{P}(0, x) = (0, y)$ faster than p calls to P or its inverse [BDPV11].[6] Note that the CICO security directly translates to the short-message preimage resistance: a preimage attack of complexity T yields a CICO solver with the same complexity. As this reduction is one-directional, we effectively strengthen our security claims with the CICO security. Note that if Skyscraper is defined over the extension field, for the CICO security we consider an equivalent representation over the base field.

We do not, however, claim security against finding more sophisticated relations between inputs and outputs of Skyscraper permutation, such as zero sums and zero-sum partitions.

3 Design Rationale

3.1 Skyscraper Core Ideas

When designing a new function, we tried to

- improve the performance of hashing over large prime fields,
- make the design as simple as possible to minimize the number of cryptanalysis vectors, and
- use lookups/S-boxes to increase the algebraic degree and become less prone to algebraic attacks.

In order to achieve these goals, we constructed the function based on the following principles.

1. Feistel structure instead of substitution-permutation network (SPN) to allow for non-invertible nonlinear mappings;
2. Alternate S-boxes and squarings as Feistel round functions to withstand attacks;
3. Montgomery multiplication instead of regular squaring;

[5] Technically, we could claim 248 bits of security against preimage attacks in the compression mode, but such a claim is useless as most protocols do not claim security beyond 128 bits anyway.

[6] The actual position of the zero value does not matter, and we place it upfront without loss of generality.

4. Keeping the same algorithm for wide states by switching to extension fields;
5. 8-bit, software-efficient S-boxes to have both small lookup tables and high native performance;
6. Rearranging S-box outputs using a single shift to get a high degree over the base field;

In the following, we detail our approach and argue each principle separately.

3.2 Feistel Instead of SPN

We have looked into the instances of Poseidon over certain domains and observed that the power mapping impedes the performance significantly. One reason is that there are few tricks that boost multiplication over prime fields, where the prime does not have any special form. One of the tricks available, the Montgomery multiplication (see the next subsection), scales poorly as the degree of the power mapping grows. As the minimal degree for x^d to be an invertible transformation is often 5 or 7 (based on popular prime number choices such as BN254), it has been natural to seek a construction that does not need invertibility. We have also learned from MiMC and `Monolith` [GKL+24] that the Feistel structure provides sufficient cryptographic strength if used with care, even though those designs do not employ the Montgomery technique for performance.

In order to get a high algebraic degree quickly, we alternate a low-degree Feistel round function with a high-degree one which is based on S-boxes. From a different standpoint, we alternate between a statistically strong and a statistically weak round function, thus having a strong transformation overall.

3.3 Montgomery Multiplication

In modular arithmetic, multiplication and subsequent reduction are among the most expensive operations. This cost highly depends on the modulus and the reduction technique. In Skyscraper (see Section 2), we work on large prime moduli. One efficient optimization is the Montgomery modular multiplication [Mon85](i.e., multiplication with subsequent Montgomery reduction). Given $c = a \cdot b$ for $a, b \in \mathbb{F}_p$ and $\sigma > p$, where σ and p are coprime, the reduction computes $c \cdot \sigma^{-1} \mod p$, for $0 \leq c < \sigma \cdot p$. While classical modular reduction computes $c' \equiv c \mod p$ as $c' = c - \lfloor \frac{c}{p} \rfloor \cdot p$, the Montgomery reduction replaces the expensive trial division by p with multiplication by a constant and a reduction and division by σ (see Algorithm 1). This algorithm is thus most efficient if σ is set as a multiple of machine words (i.e., $\sigma = 2^{256}$ for the primes we are interested in) as reductions are then free and divisions represent bit shifts.

Features. As mentioned above, the Montgomery reduction computes $c' = c\sigma^{-1} \mod p$. Usually, this effect is compensated by transforming a value into its Montgomery representation by computing

$$a' = a \cdot \sigma = \texttt{MontReduce}(a \cdot \sigma^2). \tag{12}$$

The Montgomery multiplication of two values a', b' in this representation,

$$c' = a \cdot b \cdot \sigma = \texttt{MontReduce}(a' \cdot b'), \tag{13}$$

leads to a result c' again in this representation. As one must transform into and out of the Montgomery representation only once for a given multiplication chain, the performance amortizes with the number of multiplications.

Algorithm 1: `MontReduce`.

```
Input: $c \in \{0, \ldots, p\sigma - 1\}$
    1. $\tilde{c} \leftarrow (c \cdot \underbrace{((-p)^{-1} \mod \sigma)}_{\text{precomputed}}) \mod \sigma$
    2. $r \leftarrow \frac{(c+\tilde{c}p)}{\sigma}$
    3. $r \leftarrow \begin{cases} r - p & r \geq p \\ r & \text{else} \end{cases}$
    4. Return $r \equiv c\sigma^{-1} \mod p$
```

Our design (see Figure 2) has interleaved evaluations of the squaring and `Bars`. While evaluating `Bars` on $(x'_L, x'_R) = (x_L\sigma, x_R\sigma)$ would retain its high nonlinearity, the function description would fundamentally change, losing the advantages of our chunk representation. As a result of these considerations, we include the constant σ^{-1} in the definition of the squaring operation. This step removes the cost of transforming the representation while retaining the desired structure of `Bars`.

3.4 Working with a Wider State

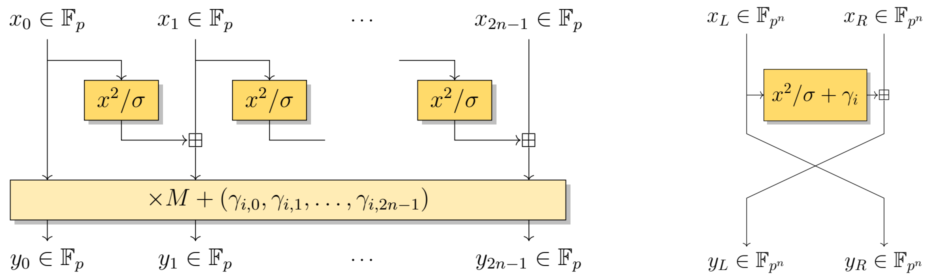
As some applications need a high-rate sponge, e.g. for a high-arity Merkle tree, we have had to scale the basic version, which deals with two $\mathbb{F}_p$ elements, to a larger state. Going to extension fields is a natural choice, and the `Bars` operation scales naturally to those, as it works on the byte level and is, to a large extent, agnostic to whether the outer field is an extension. However, for the squaring operation, this change is less trivial. There exist at least three options:

1. Treat the extension field element as a vector and replace a single squaring operation with an SPN, which can be non-invertible.
2. Again, treat the input as a vector and for $n > 1$ replace the squaring Feistel round function with a `Monolith`-like combination of a Type-3 Feistel and an MDS matrix.
3. Treat the input as an atomic element and use the squaring operation over the extension field $\mathbb{F}_{p^n}$ (see Figure 4).

The first option seems fragile, as for non-invertible SPNs we do not get nice statistical bounds, whereas an invertible SPN would be too expensive in computation due to a higher-degree power mapping. Thus we opted for the latter two.

As a baseline for comparison, we implement the extension field squaring where the two main design decisions are the squaring algorithm and reduction polynomial $\mathsf{G}(X)$. We base our implementation on the reduction algorithms from [DhSD06] and employ $\mathsf{G}(X) = X^n + \beta$. The concrete choice of β depends on the prime p s.t. $\mathsf{G}(X)$ is irreducible in $\mathbb{F}_p$. For the primes we considered in our implementations, we provide the irreducible polynomials in Table 1. For a possible MDS matrix combined with the Type-3 Feistel approach we decided to sample matrices with small entries s.t. matrix multiplication can be implemented with F_p additions only. With this approach, the smallest circulant MDS matrices are $\mathtt{circ}(1, 1, 2, 3)$ and $\mathtt{circ}(1, 4, 2, 4, 3, 1)$, which we implemented with 13 and 42 additions, respectively.

The results of implementing the latter two scaling options are presented in Table 4. We see that the last option, i.e. the extension field squaring, is faster by about 30%. One


$x_0 \in \mathbb{F}_p$
$x_1 \in \mathbb{F}_p$
$\cdots$
$x_{2n-1} \in \mathbb{F}_p$
x^2/σ
x^2/σ
x^2/σ
$\times M + (\gamma_{i,0}, \gamma_{i,1}, \ldots, \gamma_{i,2n-1})$
$y_0 \in \mathbb{F}_p$
$y_1 \in \mathbb{F}_p$
$\cdots$
$y_{2n-1} \in \mathbb{F}_p$
$x_L \in \mathbb{F}_{p^n}$
$x_R \in \mathbb{F}_{p^n}$
$x^2/\sigma + \gamma_i$
$y_L \in \mathbb{F}_{p^n}$
$y_R \in \mathbb{F}_{p^n}$

(a) T3FM

(b) ExtSq

Figure 4: The two options considered for the squaring during the design phase. The `Bars` operation remains as in Section 2.3.

Table 4: Comparing the impact of different strategies for a wide state of size $t = 2n$ elements in $\mathbb{F}_p$. The squaring operation (Sq) becomes either a classical Feistel with input $(x_L, x_R) \in \mathbb{F}_{p^n}^2$ (denoted by ExtSq), or a Type-3 Feistel followed by an MDS matrix multiplication with input $\mathbf{x} \in \mathbb{F}_p^{2n}$ (denoted by T3FM). The running time is in nanoseconds, setup details are given in Section 5.

	$t = 2$	$t = 4$		$t = 6$	
	Sq ($\mathbb{F}_p^2$)	ExtSq ($\mathbb{F}_{p^2}^2$)	T3FM ($\mathbb{F}_p^4$)	ExtSq ($\mathbb{F}_{p^3}^2$)	T3FM ($\mathbb{F}_p^6$)
`Bars` Feistel round	6		12		19
Square Feistel round	13	60	96	135	198
Skyscraper Perm.	135	320	497	722	887

could calculate that even though the $2n - 1$ base field squarings in the Type-3 options are faster than a single extension field squaring, the follow-up matrix multiplication consumes all that advantage and much more than that.

Regarding the benchmarks of the full Skyscraper permutation, we have to interpret the numbers according to the actual rate of hashing. Assuming the sponge has capacity 1, we hash $n - 1$ $\mathbb{F}$ elements per permutation call. Thus, in the Feistel-MDS combination, the hashing rate is 166 ns/element for rate 3 and 178 ns/element for rate 5. In the extension field squaring case, the numbers drop to 106 ns/elt and 147 ns/elt, respectively.

Eventually we conclude that the squaring in the extension field is the best strategy for small extensions and have fixed our design accordingly.

3.5 High Degree Component: Similarities and Differences of `Bars` among `Reinforced Concrete`, `Monolith`, Tip5, and Skyscraper

We had to use a high-degree component to counteract algebraic attacks. After we decided on the Feistel structure, it has become natural to alternate a high-degree round function with a low-degree one. We were aware of the approach in Tip5 and `Monolith` to get a high-degree function from a decomposition-S-box-composition structure, but the requirements to cover arbitrary prime fields and no invertibility condition in place have opened new design vectors for us, which we detail below.

Invertibility. The first obvious difference relies on the invertibility of `Bars`. While the `Bars` functions are invertible for `Reinforced Concrete`, `Monolith`, and Tip5, this is not the case in Skyscraper. The main consequence of this is that Skyscraper requires a Feistel construction to guarantee the invertibility of the entire scheme.

Table 5: Properties of Bars for different constructions. "No condition on $\mathbb{F}_p$" means that the construction does not require an assumption on the prime that defines the field. "Simple decomposition" is related to the simplicity of the decomposition phase.

	Invertible	No condition on $\mathbb{F}_p$	Simple decomposition
Reinforced Concrete	✓	✓	✗
Tip5, Monolith	✓	✗	✓
Skyscraper	✗	✓	✓

No Condition on $\mathbb{F}_p$. Next, we point out that the definition of Bars requires no condition on $\mathbb{F}_p$ in the case of Reinforced Concrete and Skyscraper.[7] Instead, in the case of Monolith (and similarly for Tip5), it is required that the bit representation of p is defined via substrings of 0s and 1s bits of lengths at least 3 (see [GKL+24, Def. 3] for more details) in order to achieve the invertibility condition. It is obvious that not all primes satisfy this condition.

Simple Decomposition. As we have seen before, the Bars function is composed of several steps, one of which is the decomposition. Both in the case of Monolith, Tip5, and Skyscraper, this decomposition is simple and intuitive. Any $x \in \mathbb{F}_p$ is decomposed in $(x_0, x_1, \ldots, x_{m-1}) \in \mathbb{Z}_{2^{n_0}} \times \mathbb{Z}_{2^{n_1}} \times \cdots \times \mathbb{Z}_{2^{n_{m-1}}}$ for suitable $n_0, n_1, \ldots, n_{m-1}$ (usually $n_0 = n_1 = \cdots = n_{m-1} = 8$) such that

$$x = \sum_{i=0}^{m-1} x_i \cdot \prod_{j \le i} 2^{n_j} . \tag{14}$$

In the case of Reinforced Concrete, any x is decomposed into $(x_0, x_1, \ldots, x_{m-1}) \in \mathbb{Z}_{p_0} \times \mathbb{Z}_{p_1} \times \cdots \times \mathbb{Z}_{p_{m-1}}$ for some integers $p_0, p_1, \ldots, p_{m-1}$ such that

$$x = \sum_{i=0}^{m-1} x_i \cdot \prod_{j \le i} p_j . \tag{15}$$

While the authors of Reinforced Concrete give concrete examples of such a decomposition for some primes, it is currently unclear how to find $p_0, p_1, \ldots, p_{m-1}$ for a generic prime p in order to guarantee efficiency and security of Bars.

We summarize the main differences in the Bars functions between Skyscraper and Reinforced Concrete, Monolith, and Tip5 in Table 5.

3.6 Number of Rounds

We will see from Section 4 that a combination of two Bars rounds plus two squaring rounds protects from all existing attacks. As some attacks, like CICO, provide additional freedom to the attacker, we can assume that they may guess/fix the output of one more Bars round. We surround this structure with one more Bars and two more squaring rounds at each side as a security margin and obtain a total of 10 rounds.

4 Security

In this section, we present the security analysis of Skyscraper.

[7] To be precise, in the case of Skyscraper, we required that $\lceil \log_2(p)/8 \rceil$ is even for simplifying the presentation only. However, it is possible to generalize the scheme for the case in which $\lceil \log_2(p)/8 \rceil$ is odd – the only thing that changes is the rotation of the $\mathbb{F}_2^8$-chunks in Bars.

4.1 Properties

We start by discussing some properties of the functions that instantiate Skyscraper.

4.1.1 Statistical Properties of $x \mapsto x^2$

Here we list some statistical properties of $x \mapsto x^2$. The *maximum differential probability* of a function F over $\mathbb{F}_{p^n}$ is defined as

$$\mathrm{DP}_{\max}(\mathsf{F}) = \max_{\delta_I \neq 0, \delta_O} \frac{|\{x \in \mathbb{F}_{p^n} \mid \mathsf{F}(x + \delta_I) - \mathsf{F}(x) = \delta_O\}|}{p^n}. \tag{16}$$

Lemma 1. *The maximum differential probability of $x \mapsto x^2$ over $\mathbb{F}_{p^n}$ is $1/p^n$.*

Proof. It is trivial to check that $(x + \delta_I)^2 - x^2 = \delta_O$ is satisfied by $x = \frac{\delta_O - \delta_I^2}{2\delta_I}$ only, which implies the result. □

Let $\odot$ be the inner product over $\mathbb{F}_p^n$: $\mathbf{a} \odot \mathbf{b} = \sum a_i \cdot b_i$. The *maximum linear correlation* of a function F over $\mathbb{F}_p^n$ is defined as

$$\mathrm{LC}_{\max}(\mathsf{F}) = \max_{\mathbf{a} \neq 0, \mathbf{b} \in \mathbb{F}_p^n} \frac{|\{\mathbf{x} \in \mathbb{F}_p^n \mid \mathbf{b} \odot \mathsf{F}(\mathbf{x}) = \mathbf{a} \odot \mathbf{x}\}|}{p^n}. \tag{17}$$

Lemma 2. *The maximum linear correlation of $x \mapsto x^2$ over $\mathbb{F}_{p^n}$ is $2/p$.*

Proof. The function $x \mapsto x^2$ has degree 2 as a multivariate polynomial over $(x_1, \ldots, x_n) \in \mathbb{F}_p^n$. Therefore, for any $\mathbf{a}, \mathbf{b} \in \mathbb{F}_p^n$ the linear correlation condition $\mathbf{b} \odot \mathsf{F}(\mathbf{x}) = \mathbf{a} \odot \mathbf{x}$ is a degree-2 equation. By the Schwartz–Zippel lemma, the number of solutions to it is at most $(2 \cdot p^{n-1})/p^n = 2/p$. □

4.1.2 Algebraic Properties of Bars and of $x \mapsto x^2$

We first recall the following results from Monolith [GKL+24].

Lemma 3 (Lemma 1 in [GKL+24])**.** *Let $p \geq 3$ be a prime number, and let $\mathcal{F}_{sq}$ denote the squaring function $x \to x^2$ over $\mathbb{F}_p$. Let F_{sq} be any interpolant of $\mathcal{F}_{sq}$ over $\mathbb{F}_2^{\lceil \log_2 p \rceil}$, i.e., for any $a < p$ and its bit representation a we have that $\mathsf{F}_{sq}(\mathsf{a})$ is the bit representation of $\mathcal{F}_{sq}(a)$. Then F_{sq} has (multivariate) degree at least d, where d is the maximum positive integer such that $d < \log_2 \sqrt{p}$ and $\lceil 2^{d-0.5} \rceil$ is odd.*

Lemma 4 (Lemma 2 in [GKL+24])**.** *Let F be a function that maps $\mathbb{F}_p$ to itself with a differential $\Delta_I \mapsto \Delta_O$ holding with probability $0 < \alpha < 1$, i.e., $|\{x \in \mathbb{F}_p \mid F(x + \Delta_I) = F(x) + \Delta_O\}| = p \cdot \alpha$. Then we have $\deg(F) > \alpha \cdot p$, where $\deg(F)$ is the degree of F as a polynomial over $\mathbb{F}_p$.*

The following property directly follows from its analogue in Monolith [GKL+24].

Proposition 1. *The `Bars` function has degree at least $63/256 \cdot p$ over $\mathbb{F}_{p^n}$.*

Proof. Indeed, when viewed as a function over $\mathbb{Z}_{256}$, `Sbox` admits the differential $\{+1\} \mapsto \{+2\}$ for 63 inputs out of 256. Taking the chunk rotation into account, the output difference becomes $\{+2^{4m+1}\}$. Setting a zero input difference in all other `Sbox` applications, it follows that the differential $1 \mapsto 2^{4m+1}$ holds for $63/256$ of all possible inputs from $\mathbb{F}_{p^n}$. The degree bound follows immediately by applying Lemma 4. □

For the next result, we need to introduce the following definition.

Definition 1. Given $x \in \mathbb{F}_{p^n}$, we denote its bit representation by $\widetilde{x} \in \mathbb{F}_2^{n \cdot \lceil \log_2 p \rceil}$. Let F be a function over $\mathbb{F}_{p^n}$ and F be a function over $\mathbb{F}_2^{n \cdot \lceil \log_2 p \rceil}$. We say that F is an interpolant of F if

$$\forall x \in \mathbb{F}_{p^n} : \qquad \widetilde{F(x)} = \mathsf{F}(\widetilde{x}) .$$

Lemma 5. *Let F be the mapping $x \mapsto x^2$ over $\mathbb{F}_{p^n}$ with $p > 2^{246}$. Then any interpolant of F over $\mathbb{F}_2^{n \cdot \lceil \log_2 p \rceil}$ has multivariate degree at least 123.*

Proof. Let F be some interpolant of F over $\mathbb{F}_2^{n \lceil \log_2 p \rceil}$. By fixing the first $(n-1)\lceil \log_2 p \rceil$ variables to 0, we get the function F_n, which is an interpolant to the squaring function over $\mathbb{F}_p$. From Lemma 3, F_n has degree at least d if $\lceil 2^{d-0.5} \rceil$ is odd. As $\lceil 2^{123-0.5} \rceil =$ 7 519 249 036 500 140 985 782 305 389 925 783 029 is odd, the function F_n has degree at least 123. As $\deg(\mathsf{F}) \geq \deg(\mathsf{F}_n)$, we get the lemma statement. □

4.1.3 Linear Approximation of Bars

In order to derive useful linear approximations for `Bars`, we start by considering simplified variants of it.

Case: $\boldsymbol{x \mapsto x \oplus \big((\overline{x} \lll 1) \odot (x \lll 2) \odot (x \lll 3)\big)}$. First, let us note that the map $\mathsf{T} : x \mapsto x \oplus \big((\overline{x} \lll 1) \odot (x \lll 2) \odot (x \lll 3)\big)$ over $\mathbb{F}_2^8$ has exactly 48 fixed points, including 34 with 0 as the most significant bit. This results from Lemma 6 and Lemma 7.

Lemma 6. *Let $\mathcal{E}_\tau$ be the set defined by*

$$\{x \in \mathbb{F}_2^\tau \mid x_0 = 0 \textit{ and } \forall i \in \{0, \ldots, \tau - 1\}, x_i = 1 \Rightarrow x_{i+1 \bmod \tau} = x_{i-1 \bmod t} = 0\}.$$

Then, the cardinal of $\mathcal{E}_\tau$ is $|\mathcal{E}_\tau| = \mathrm{Fib}(\tau + 1)$, where $\mathrm{Fib}(\tau + 1)$ is the $(\tau + 1)$-th term in the Fibonacci sequence.

Proof. Let us show by induction on $\tau \geq 2$ that there are $\mathrm{Fib}(\tau)$ elements with LSB (least significant bit) equal to 0 (i.e. $x_{\tau-1} = 0$), and $\mathrm{Fib}(\tau - 1)$ elements with LSB equal to 1.

- **For $\boldsymbol{\tau = 2}$.** We have
$$\mathcal{E}_2 = \{(0, 0), (0, 1)\}$$
so that $\mathrm{Fib}(2) = 1$ element is with LSB $x_1 = 0$, and $\mathrm{Fib}(1) = 1$ element is with LSB $x_1 = 1$.
- **For $\boldsymbol{\tau + 1}$.** By definition of the set $\mathcal{E}_\tau$, if $x_{\tau-1} = 1$, then $x_\tau = 0$, and if $x_{\tau-1} = 0$, then $x_\tau \in \{0, 1\}$. By induction hypothesis, we have $\mathrm{Fib}(\tau - 1) + \mathrm{Fib}(\tau) = \mathrm{Fib}(\tau + 1)$ elements with LSB equals to 0, and $\mathrm{Fib}(\tau)$ elements with and LSB equals to 1.

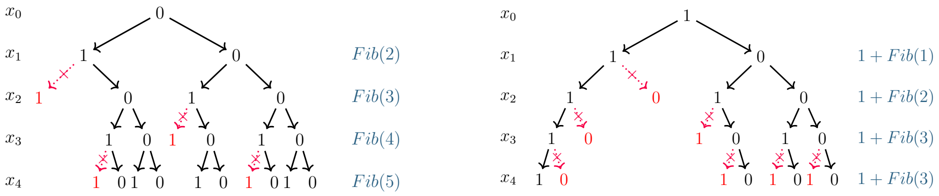
It follows that the cardinal of the set $\mathcal{E}_\tau$ is $\mathrm{Fib}(\tau - 1) + \mathrm{Fib}(\tau) = \mathrm{Fib}(\tau + 1)$. □

Lemma 7. *The fixed points of the map $\mathsf{T} : x \mapsto x \oplus \big((\overline{x} \lll 1) \odot (x \lll 2) \odot (x \lll 3)\big)$ over $\mathbb{F}_2^t$, with $\gcd(t, 3) = 1$ (necessary condition for the invertibility), are exactly the $\mathrm{Fib}(t + 1) + \mathrm{Fib}(t - 1) + 1$ following elements:*

$$\{\mathbf{1}\} \;\cup\; \{x \in \mathbb{F}_2^t \mid \forall i \in \{0, \ldots, t - 1\}, x_i = 1 \Rightarrow x_{i+1 \bmod t} = x_{i-1 \bmod t} = 0\}.$$

Proof. First, let us observe that the fixed points of T satisfy the following system of equations:

$$\forall i \in \{0, \ldots, t - 1\} : \qquad x_i \oplus (\overline{x_{i+1 \bmod t}} \cdot x_{i+2 \bmod t} \cdot x_{i+3 \bmod t}) = x_i ,$$



(a) MSB $= 0$.

(b) MSB $= 1$.

Figure 5: Counting fixed points for T when $t = 5$.

which is equivalent to

$$\forall i \in \{0, \ldots, t-1\} : \qquad \overline{x_i} \cdot x_{i+1 \bmod t} \cdot x_{i+2 \bmod t} = 0\,.$$

In particular, it means that if there exists $i \in \{0, \ldots, t-1\}$ such that $x_i = x_{i+1 \bmod t} = 1$, then $\mathbf{1} = (1, \ldots, 1)$ is the only solution to the system. Indeed, w.l.o.g., let us assume that $x_0 = x_1 = 1$, then we have

$$\begin{cases} 0 \cdot 1 \cdot x_2 & = 0 \\ 0 \cdot x_2 \cdot x_3 & = 0 \\ \ldots & \\ \overline{x_{t-3}} \cdot x_{t-2} \cdot x_{t-1} & = 0 \\ \overline{x_{t-2}} \cdot x_{t-1} \cdot 1 & = 0 \\ \overline{x_{t-1}} \cdot 1 \cdot 1 & = 0. \end{cases}$$

From the last equation we obtain $\overline{x_{t-1}} = 0$, that is, $x_{t-1} = 1$. Then, it follows that $x_{t-2} = 1$, and so on until $x_2 = 1$, meaning that $(1, \ldots, 1)$ is the only solution. It remains to show that all points such that if $x_i = 1$ then $x_{i+1 \bmod t} = 0$ and $x_{i-1 \bmod t} = 0$ are solutions. This follows from noting that in any equation

$$\overline{x_i} \cdot x_{i+1 \bmod t} \cdot x_{i+2 \bmod t} = 0$$

either $x_{i+1 \bmod t} = 0$ or $x_{i+2 \bmod t} = 0$ so that the equation is always satisfied. We deduce that all fixed points are exactly

$$\{\mathbf{1}\} \,\cup\, \{x \in \mathbb{F}_2^t \mid \forall i \in \{0, \ldots, t-1\}, x_i = 1 \Rightarrow x_{i+1 \bmod t} = 0, x_{i-1 \bmod t} = 0\}.$$

The number of such points then follows from Lemma 6. Indeed, if the MSB (Most Significant Bit) is 0 (i.e., $x_0 = 0$), then by a direct application of the lemma with $\tau = t$, we have $\mathrm{Fib}(t+1)$ fixed points. Then let us assume that the MSB is 1. If $x_1 = 1$, then we necessarily have $x = (1, \ldots, 1)$. Else we have $x_1 = x_{t-1} = 0$, and we apply Lemma 6 for $\tau = t - 2$, so that we obtain $\mathrm{Fib}(t-1)$ fixed points. It follows that the map T has exactly $\mathrm{Fib}(t+1) + \mathrm{Fib}(t-1) + 1$ fixed points. The procedure to count the fixed points is depicted in Figure 5. □

By approximating the Fibonacci term $\mathrm{Fib}(t) = \frac{\Phi^t - (-\Phi)^{-t}}{\sqrt{5}}$ where $\Phi = \frac{\sqrt{5}+1}{2} \approx 1.61803$ is the golden section number we get

$$1 + \frac{\Phi^{t+1} - (-\Phi)^{-(t+1)}}{\sqrt{5}} + \frac{\Phi^{t-1} - (-\Phi)^{-(t-1)}}{\sqrt{5}}$$

fixed points. We give concrete examples in Table 6.

Remark 1. We point out that our new bound given in Lemma 7 improves the one previously given in [GKL$^+$24, Sect. 5.3] for `Monolith`, and equal to $(7/4)^t$. For a concrete comparison, we refer to Table 6.

Table 6: Number of fixed points for T defined over $\mathbb{F}_2^t$.

t bits	Practical # fixed points	Theoretical # fixed points	Formula $(7/4)^t$
8	48	48	88
10	124	124	269
16	2208	2208	7738

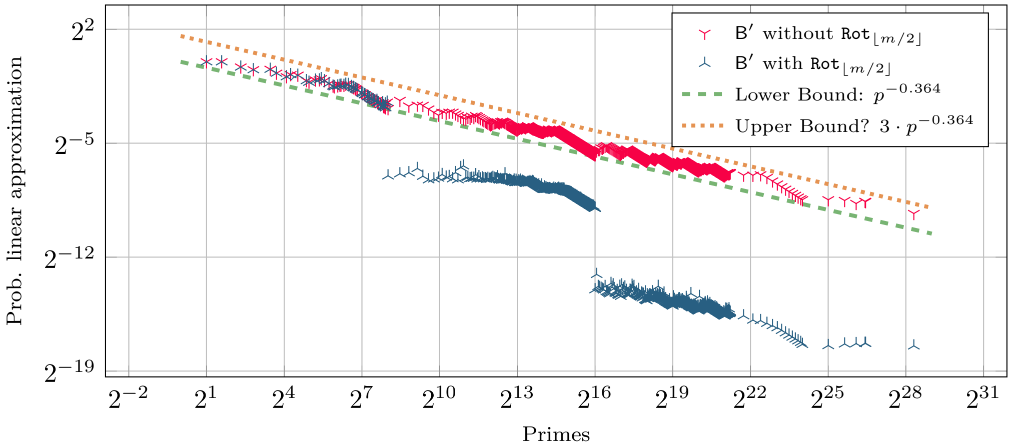


Figure 6: Probabilities of the linear approximation $x \mapsto 2 \cdot x$ $(n = 1)$. The orange line corresponds to the linear approximation for the case without rotation, while the purple one for the case with rotation. Note that the lower and the upper bounds hold for the orange line only. (For experiments, we used $\texttt{Rot}_{\lfloor m/2 \rfloor}$ to cover the cases $m \in \{1, 2, 3\}$.)

Case: $\boldsymbol{x \mapsto \big(x \oplus \big((\overline{x} \lll 1) \odot (x \lll 2) \odot (x \lll 3)\big)\big) \lll 1}$. Next, the operation $\cdot \lll 1$ applied on $x \mapsto x \oplus \big((\overline{x} \lll 1) \odot (x \lll 2) \odot (x \lll 3)\big)$ over $\mathbb{F}_2^8$ corresponds to the double operation over $\mathbb{Z}_{2^8}$ when the most significant bit is 0. It follows that $x \mapsto \big(x \oplus \big((\overline{x} \lll 1) \odot (x \lll 2) \odot (x \lll 3)\big)\big) \lll 1$ is well approximated by $x \mapsto 2 \cdot x$ over $\mathbb{Z}_{2^8}$ when x is a fixed point of $x \mapsto x \oplus \big((\overline{x} \lll 1) \odot (x \lll 2) \odot (x \lll 3)\big)$. Due to the previous result, this happens with probability at least

$$\frac{34}{256} \approx 2^{-2.913} \, .$$

Considering now all the chunks over $\mathbb{F}_{p^n}$, we get

$$\left(2^{-2.913}\right)^{m \cdot n} = \left(2^{-2.913}\right)^{\lceil \log_2(p)/8 \rceil \cdot n} \approx \left(2^{-2.913}\right)^{\log_2(p)/8 \cdot n} = p^{-0.364 \cdot n}$$

as a lower bound for the linear approximation of B′ (without the rotation $\texttt{Rot}_{m/2}$) by the mapping $x \mapsto 2 \cdot x$. For example, for $p \approx 2^{254}$, this corresponds to a probability of $2^{-92.5 \cdot n}$.

This lower bound has been practically tested – results given in Figure 6 (green line). An upper bound is more complicated to establish. Figure 6 suggests that the bound must be close to $3 \cdot p^{-0.364 \cdot n}$ (red line).

Still, we recall that the $\texttt{Bars}$ operation includes a rotation $\texttt{Rot}_{m/2}$ of the chunks. This rotation has the effect to destroy the linear approximation $x \mapsto 2 \cdot x$, as it is possible to observe in Figure 6 (purple line).

About $\texttt{Rot}_{m/2}$. Having said that, we cannot exclude a priori that other linear approximations with better probabilities exist. Concretely, assume $n = 1$, and consider the case in which all lookup tables $\texttt{Sbox}$ are defined by the identity map over $\mathbb{F}_2^8$. In such a case,

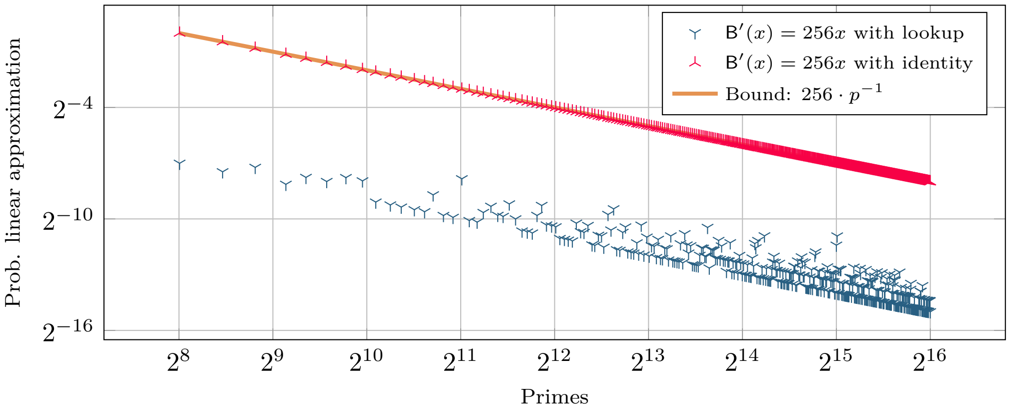


Figure 7: Probabilities of the linear approximation $x \mapsto 2^8 \cdot x$ ($n = 1$). The cyan line corresponds to the linear approximation for the case without lookup, while the magenta one for the case with lookup.

the `Bars` function reduces to a rotation of the chunks (and so of the bits). This implies that the `Bars` function admits $x \mapsto 2^{8\cdot m/2} \cdot x = 2^{4\cdot m} \cdot x$ as linear approximation for all the points x such that $0 \leq x < 2^{4\cdot m}$ (unless $p = 2^{4\cdot m} + 1$, in which case $0 \leq x \leq 2^{4\cdot m}$). This implies a probability equal to

$$\frac{2^{4m}}{p}\,.$$

For example, for $p \approx 2^{254}$, this corresponds to 2^{-126}. Note that

$$\frac{2^{4m}}{p} = \frac{2^{4\cdot\lceil\log_2(p)/8\rceil}}{p} \approx \frac{2^{4\cdot\log_2(p)/8}}{p} = \frac{p^{0.5}}{p} = p^{-0.5}\,,$$

which is smaller than $p^{-0.364}$ for the previous linear approximation.

One can prove that this bound is actually *tight*, i.e., there is no better linear approximation. Indeed, the entire domain $\mathbb{F}_p^n$ splits into classes of form $\mathcal{C}_A = \{2^{4m}A + 0, 2^{4m}A + 1, \ldots, 2^{4m}A + 2^{4m-1}\}$ for every $A < p/2^{4m}$ where the mapping is actually linear:

$$x \mapsto 2^{4m} \cdot x + A(2^{8m} - 1)\,.$$

As all these mappings are different, we effectively get that $p^{-0.5}$ is the tight linear approximation bound.

In the general case $n \geq 1$, the probability becomes $p^{-0.5\cdot n}$, since it is possible to use the previous linear approximation independently on each chunk. As before, we expect that this probability drops down when the “real” `Bars` functions (hence, `Sbox` are not the identity maps) are considered.

Conclusion. By summarizing the previous results, we get that

- `Bars` without rotation admits the linear approximation $x \mapsto 2 \cdot x$, which holds with probability $p^{-0.364\cdot n}$;
- the rotation of `Bars` (instantiated with identity maps instead of `Sbox`) admits the linear approximation $x \mapsto 2^{4\cdot m} \cdot x$, which holds with probability $p^{-0.5\cdot n}$.

By combining these two results, we get for `Bars` the following linear approximation:

$$x \mapsto \underbrace{2^{4\cdot m}}_{\text{rotation}} \cdot \underbrace{(2 \cdot x)}_{\texttt{Bars}\text{ without rotation}} = 2^{4\cdot m+1} \cdot x\,,$$

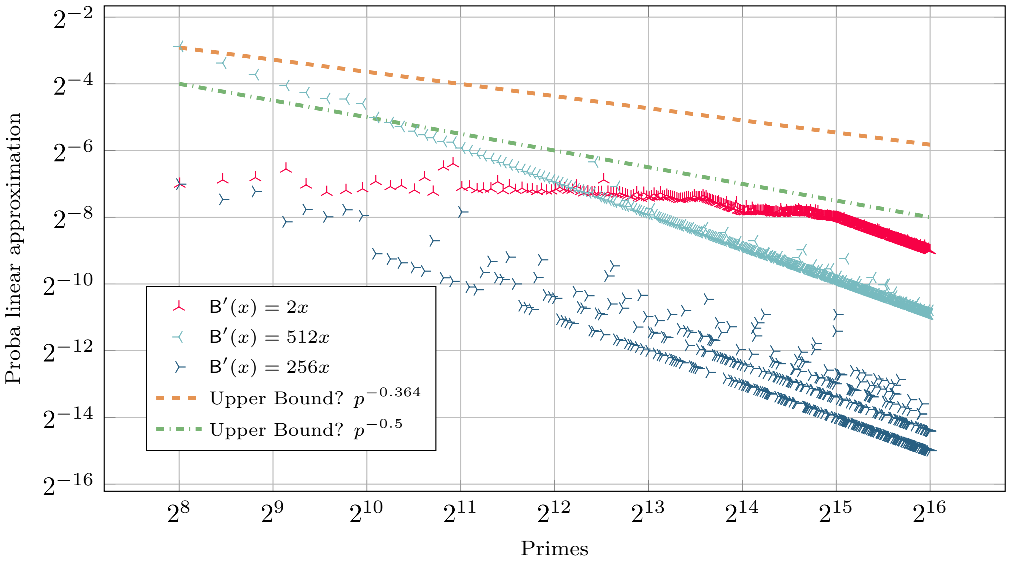


Figure 8: Case $n = 1$, $2^8 < p < 2^{16}$, and $m = 2$. Probabilities of the linear approximations $x \mapsto 2^8 \cdot x$ (magenta), $x \mapsto 2^9 \cdot x$ (blue) and $x \mapsto 2 \cdot x$ (purple) for `Bars`. Note that the theoretical linear approximation $x \mapsto 2 \cdot x$ obtained for the case of `Bars` without rotation is an upper bound for both.

whose probability is lower bounded by $p^{-0.364 \cdot n} \cdot p^{-0.5 \cdot n} = p^{-0.864 \cdot n}$. Practical results for the case $2^8 < p < 2^{16}$ (for which we have two 8-bit chunks, hence, the linear approximation $x \mapsto 2^{2 \cdot 4+1} \cdot x = 2^9 \cdot x$) are given in Figure 8. As it is possible to observe in there, the best linear approximation we found is either $x \mapsto 2 \cdot x$ or $x \mapsto 2^{4 \cdot m+1} \cdot x$.

The problem of identifying the best linear approximation for `Bars` is open for future research. In Figure 8 we observe that the bound $p^{-0.5 \cdot n}$ holds for sufficiently big primes (i.e. $p \geq 2^{10}$). At the current state, based on our theoretical and on our practical results, and assuming $p \geq 2^{248}$, we conjecture the following.

Conjecture 1. *There exists no linear approximation of* `Bars` *with probability higher than* $p^{-0.5 \cdot n}$.

4.2 Security against Statistical Attacks

Here we show that our design is secure against statistical attacks.

Remark 2. Since the `Bars` layer is not supposed to have good statistical properties, we simply assume that the attacker can skip it with probability 1. Equivalently, we consider a "weaker" version in which `Bars` are omitted. If such a construction is secure against statistical attacks, then Skyscraper is secure as well.

Remark 3. We do **not** consider impossible differential [BBS99, Knu98] and zero-linear cryptanalysis [BW12], since they do not lead to any collision and/or preimage attack on the sponge/compression hash function.

4.2.1 Differential and Linear Cryptanalysis

Differential cryptanalysis [BS90, BS93] exploits the probability distribution of a non-zero input difference leading to an output difference after a given number of rounds. Since Skyscraper is an iterated cipher, a cryptanalyst considers ordered sequences of differences over consecutive rounds called differential characteristics/trails. Assuming the independence of the rounds, the Differential Probability (DP) of a differential trail is the product of the DPs of its one-round differences.

In the case of Skyscraper, the maximum DP of $x \mapsto x^2$ is $1/p^n$, as shown in Lemma 1. As it is well known (see e.g. [DKLS20]), in a Feistel construction, only one of two consecutive rounds is active. Indeed, the attacker can impose the difference of the first round in order not to activate the consecutive rounds. It follows that 4 rounds are largely sufficient to prevent differential cryptanalysis since the probability of any differential characteristic is upper bounded by $p^{-2n} \le 2^{-4\cdot\kappa}$ where $n \ge 1$. This also takes into account the fact that more characteristics can be used simultaneously in order to set up a differential attack.

Linear cryptanalysis [Mat93] exploits the existence of linear approximations. For primitives over binary fields, the attack makes use of the high correlations between sums of input bits and sums of output bits. The generalization of this attack over prime fields has been proposed in [BSV07, DGGK21]. Due to the low correlation of the map $x \mapsto x^2$ (see Lemma 2), we claim that our scheme is secure against this approach (in the same way it is secure against differential attacks).

4.2.2 Fixed Points

As in `Monolith`, the `Bars` layer of Skyscraper has some fixed points. From [GKL+24, Sect. 5.3], the only fixed points of $x \mapsto \big(x \oplus \big((\overline{x} \lll 1) \odot (x \lll 2) \odot (x \lll 3)\big)\big) \lll 1$ over $\mathbb{F}_2^8$ are $\mathbf{0}$ and $\mathbf{1}$. Hence, without the rotation $\texttt{Rot}_{m/2}$ of the chunks, the number of fixed points of `Bars` is at most $2^{n\cdot m} - 1 = 2^{n\cdot\lceil\log_2(p)/8\rceil} - 1$, where -1 is due to the fact that not all chunks can be equal to 1s.

However, due to the rotation $\texttt{Rot}_{m/2}$, the number of fixed points is reduced to $2^{m/2} - 1$ (again, -1 is due to the fact that not all chunks can be equal to 1s),[8] since no more than $m/2$ 8-bit chunks are free to take any value among $\mathbf{0}$ and $\mathbf{1}$ if one aims to guarantee that a point is unchanged after the rotation.

It follows that the probability of a point in $\mathbb{F}_p^n$ to be fixed for `Bars` is

$$\frac{2^m}{p^n} = \frac{2^{\lceil\log_2(p)/8\rceil}}{p^n} \approx \frac{2^{\log_2(p)/8}}{p^n} = p^{-n+1/8} \le p^{-7/8}\,.$$

As a concrete example, for $p \approx 2^{254}$, this corresponds to a probability of approximately $2^{31-254\cdot n} \le 2^{-223}$. For comparison, we remember that the probability that a point is fixed for `Monolith` is 2^{-216}. Due to the fact that the only fixed points of $x \mapsto x^2$ are 0 and 1, and due to the random constant additions, we conjecture that this does not represent a threat to the security of Skyscraper.

4.2.3 Truncated Differential and (Invariant) Subspace Trail Cryptanalysis

Truncated differential attacks [Knu94] and (invariant) subspace trail attacks [GRR16] are used mostly against primitives that have incomplete diffusion over a few rounds. This is not the case here, since full diffusion is achieved in two consecutive rounds.

4.2.4 Rebound Attacks

Rebound attacks [LMR+09, JNP12] have been widely used to analyze the security of various types of hash functions against shortcut collision attacks since the beginning of the SHA-3 competition. It starts by choosing internal state values in the middle of the computation and then computing in the forward and backward directions to arrive at the inputs and outputs. It is useful to think of it as having central (often called “inbound”)

[8] Only for completeness, we point out that in the general case in which $m/2$ is not an integer, the number of fixed points after the rotation $\texttt{Rot}_{\lfloor m/2\rfloor}$ is reduced to $2^{\gcd(\lfloor m/2\rfloor, n\cdot m)} - 1$.

and the above-mentioned “outbound” parts. In the attack, solutions to the inbound phase are first found and then filtered in the outbound phase.

Since any (truncated) differential characteristic on 2-round Skyscraper has a probability that is much lower than the security level, we can immediately conclude that the rebound attack is not a threat to our design.

4.3 Security against Algebraic Attacks

Here we provide evidence that our design withstands algebraic attacks.

Remark 4. We do **not** consider higher-differential [Knu94] and zero-sum partition [BCC11, BDPA11] attacks, since they do not lead to any collision and/or preimage attack on the sponge/compression hash function. In more detail, our choice is motivated by the gap present in the literature between the number of rounds of the internal permutation that can be covered by a zero-sum partition and by the number of rounds in the corresponding sponge hash function that can be broken, e.g. via a preimage or a collision attack. As a concrete example, consider the case of Keccak: While 24 rounds of Keccak-f can be distinguished from a random permutation using a zero-sum partition [BCC11] (that is, full Keccak-f), preimage/collision attacks on Keccak can only be set up for up to 6 rounds of Keccak-f [GLST22].

4.3.1 Interpolation Attack: Density of the Interpolating Polynomial

Interpolation attacks [JK97] exploit the (low) algebraic degree of a component to reconstruct polynomials efficiently describing the output of a primitive in dependence of its inputs.

Interpolation of B'. Let $L(F, x, \mathcal{D})$ denote the (unique) univariate Lagrange polynomial in x of the function F, with domain $\mathcal{D}$, that is,

$$L(F, x, \mathcal{D}) := \sum_{i\in\mathcal{D}} F(i) \cdot \prod_{j\in\mathcal{D}, i\neq j} \frac{x-j}{i-j} \tag{18}$$

with $\deg(L(F, x, \mathcal{D})) \leq |\mathcal{D}| - 1$. The Lagrange interpolation of the function $\mathsf{B}' : \mathbb{F}_{p^n} \to \mathbb{F}_{p^n}$, as defined in Section 2.3, has degree at least $63/256 \cdot p$, as we proved in Proposition 1.

Still, calculating the actual interpolating polynomial of B' over $\mathbb{F}_p^n$, or even assessing its concrete degree and density, is infeasible in practice for p^n sufficiently large. Thus, we conduct experiments on smaller finite fields with $n = 1$ and chunk size $s = 4$ (which permits the same formula for T, given in Equation (8)), and report the deviation between the actual degree and density from the theoretical maximum values. In total, we interpolated B' over more than 700 different primes p (13 7-bit, 23 8-bit, 122 11-bit, 196 12-bit, 428 14-bit). Notably, in all cases, the univariate Lagrange interpolation polynomial was of maximum degree and high density, as shown in Figure 9.

In particular, our experiments suggest that B' has the same behavior as a random polynomial over $\mathbb{F}_p$ for p sufficiently large.

Lemma 8 (Asymptotic density of a random polynomial over $\mathbb{F}_p$)**.** *In a random univariate polynomial $F : \mathbb{F}_p \to \mathbb{F}_p$ of degree $p - 1$, all monomials are present with high probability. In particular, the probability that exactly k out of p monomials have a zero coefficient is given by*

$$\mathcal{B}\left(k\,;\, p, \frac{1}{p}\right) \leq \frac{1}{k!} \cdot e^{-1} \tag{19}$$

for p sufficiently large.

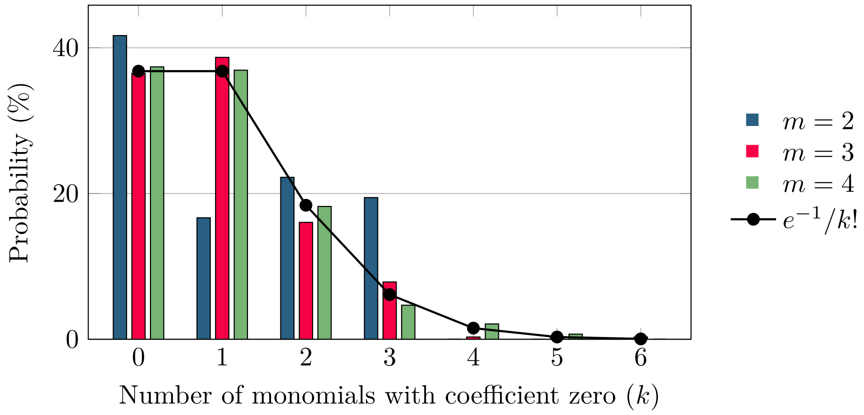


Figure 9: Density of the interpolation polynomial of $\mathsf{B}' : \mathbb{F}_{p^n} \to \mathbb{F}_{p^n}$ $(n = 1)$.

Proof. In a random univariate polynomial $F : \mathbb{F}_p \to \mathbb{F}_p$ of degree $p - 1$, each monomial x^i for $0 \leq i \leq p - 1$ has a probability of $\frac{1}{p}$ of being not present. Let η denote the number of monomials with coefficient zero. Then $\mathrm{Prob}(\eta = k)$ is given by the Binomial distribution

$$\mathrm{Prob}(\eta = k) = \mathcal{B}\left(k\,;\,p, \frac{1}{p}\right) = \binom{p}{k} \cdot \left(\frac{1}{p}\right)^k \cdot \left(1 - \frac{1}{p}\right)^{p-k} .$$

The binomial coefficient $\binom{p}{k}$ is always upper bounded by $p^k/k!$, since $\binom{p}{k} = \prod_{j=0}^{k}(p-j)/k! \leq p^k/k!$. Hence:

$$\mathrm{Prob}(\eta = k) \leq \frac{p^k}{k!} \cdot \left(\frac{1}{p}\right)^k \cdot \left(1 - \frac{1}{p}\right)^{p-k} = \frac{1}{k!} \cdot \left(1 - \frac{1}{p}\right)^p \approx \frac{e^{-1}}{k!} ,$$

where the last approximation holds for $p \gg 1$. □

Conclusion. Due to the algebraic properties of B', the multivariate polynomial describing the Skyscraper permutation to be of degree p in each variable and thus renders the interpolation attack infeasible. Thus, mounting an interpolation attack on Skyscraper is infeasible. The same conclusion holds for the meet-in-the-middle (MitM) version of the attack.

4.3.2 Gröbner Basis Attacks

In this section, we attempt to solve the CICO problem as it is formulated in Section 2.4. We take an algebraic representation of Skyscraper and formulate the problem as a set of equations (a concrete choice of equations will be detailed below). Then the process is generic as follows.

Given a system of n_e multivariate polynomial equations in n_v variables over a field $\mathbb{F}$ generating a zero-dimensional ideal $\mathcal{I}$, the Gröbner basis attack consists of the following steps.

1. Compute the Gröbner basis of $\mathcal{I}$ with respect to a total degree order using algorithms such as Faugère's F4 [Fau99] or F5 [Fau02] algorithm.
2. Apply a basis conversion algorithm, such as the FGLM algorithm [FGLM93], to convert the total degree order Gröbner basis into a Gröbner basis with respect to an elimination order.

3. Recover the solutions to the polynomial equation system by iteratively solving (systems of) univariate equations. See [CLO15, Chapter 3] or [BBLP22, §3.1] for more details.

The complexity of the aforementioned matrix-based algorithms to compute a total degree order Gröbner basis can be bounded by

$$\mathcal{O}\left(n_e \cdot \binom{n_v + d_{\mathrm{reg}}}{d_{\mathrm{reg}}}^{\omega}\right) \tag{20}$$

operations in $\mathbb{F}_p$, where $2 \leq \omega \leq 3$. Here, d_{reg} denotes the *degree of regularity*, as defined in [BSGL20, §A 2.2.1]. In general, the degree of regularity is as difficult to compute as computing the Gröbner basis itself. Thus, a common approach is to perform small-scale experiments on reduced versions of a primitive and extrapolate d_{reg} from the experimental data points to large-scale instances, which are computationally intractable.

In the following, we describe an algebraic model of Skyscraper over $\mathbb{F}_p^2$, that is, for the simplest case $n = 1$. We point out that we subsequently only focus on step 1 of a Gröbner basis attack and show that already the complexity of this step by far exceeds the targeted security level.

Generalized construction. For the purpose of analysis, it is informative to consider a generalized version of Skyscraper. Let q be the number of consecutive squaring components S_i, let b be the number of consecutive Bars components B_i, and let N be the number of times the pattern "$q\times$ Square, $b\times$ Bars" is repeated. In particular, for our design given in Equation (2), we consider the instance $(q, b, N) = (2, 2, 2)$, yielding a total of $R = (q + b) \cdot N + q = 10$ rounds.

Algebraic Model for B'. Let s denote the chunk size in the decomposition Decomp, and let $m = \left\lceil \frac{\log_2(p)}{s} \right\rceil$ denote the number of s-bit chunks in the decomposition of x. Further, let $p_0 = p \gg ((m-1) \cdot s)$ denote the integer value of the most significant chunk of p's decomposition, and let $r = \left\lfloor \frac{m}{2} \right\rfloor$ denote the number of elements for the left rotation.

To model $y = \mathsf{B}'(x)$, we introduce m variables $w_0, \ldots, w_{m-1} \in \mathbb{F}_p$. The following constraints need to hold:

1. Range constraints: To ensure that $w_0, \ldots, w_{m-1}$ are in the correct range, that is, they fit into a maximum of s bits, we introduce the following m constraints:

$$0 = \prod_{i=0}^{p_0 - 1} (w_0 - i), \qquad \text{and} \qquad 0 = \prod_{i=0}^{2^s - 1} (w_k - i) \quad \forall\, 1 \leq k < m. \tag{21}$$

   Note that for the most significant chunk, while in practice $w_0 \leq p_0$, we require that $w_0 < p_0$, excluding some solutions and thus giving the attacker an advantage. In total, the range constraints are modeled as a system of $m - 1$ polynomial equations of degree 2^s and 1 polynomial of degree $p_0 < 2^s$. The polynomials have high density.

2. Composition constraint: Step 1 in the definition of B' can be modeled using a single equation linear in the input x:

$$x = \sum_{k=0}^{m-1} b_k \cdot w_k, \tag{22}$$

   where $b_k := 2^{s(m-1-k)}$ for $0 \leq k < m$.

3. Model $\mathtt{Rot}_r$, $\mathtt{Sbox}$, and $\mathtt{Comp}$. Using Lagrange interpolation, as defined in Equation (18), steps 2 - 4 in the definition of B' can be modeled using a single equation of degree $\max\{\deg(y), 2^s\}$:

$$y = \sum_{k=0}^{m-1} b_k \cdot L(\mathsf{T}, w_{k-r \bmod m}, \mathcal{D}_k), \tag{23}$$

where the respective interpolation domains are given by

$$\mathcal{D}_k = \begin{cases} [0, p_0 - 1] & \text{if } k = m - r, \\ [0, 2^s - 1] & \text{otherwise.} \end{cases}$$

Treating the input x and output y as constants, the component $\mathsf{B}' : \mathbb{F}_p \to \mathbb{F}_p$ is modeled as a system of $m + 2$ polynomial equations in m variables.

Algebraic Model for Skyscraper. The Skyscraper-CICO problem consists of finding $x_R, y_L \in \mathbb{F}_p$ such that

$$\mathsf{Skyscraper}(0, x_R) = (y_L, 0).$$

Skyscraper-CICO can be modeled as an overdetermined system of $n_e = (m + 2) \cdot b \cdot N + 1$ equations in $n_v = (m + 1) \cdot b \cdot N + 1$ variables, which arises as follows:

- *Variables*: We introduce 1 variable x_R modeling the unconstrained input to the Feistel construction. For each of the $b \cdot N$ operations $\mathsf{B}_i : \mathbb{F}_p^2 \to \mathbb{F}_p^2$, we introduce one variable b_i to model the output of B', as well as m variables to model the decomposition.

- *Equations*: We apply the round function until we reach a $\mathtt{Bars}$ component B_i. In particular, the input to B' in B_i is given by a polynomial of degree at least 2^{q-1}, and the output is given by the variable b_i. Then B' is modeled as a system of $m + 2$ polynomial equations. This strategy is repeated until the output of the Feistel construction is reached. Finally, one additional equation is needed to model the constrained output $y_R = 0$.

An example for the instance $(q, b, N) = (2, 1, 1)$ is given in Figure 10.

Small-Scale Experiments. In order to perform practical experiments, we consider reduced versions of Skyscraper. In particular, we investigate chunk sizes of $s = 4$ and $s = 8$ (which both permit the same formula for T, given in Equation (8)) and vary (q, b, N) to get an overview of the complexity of the Gröbner basis attack against Skyscraper-CICO. We achieved practical results for primes with ≤ 16 bits, summarized in Tables 14 to 16.

In particular, we observed that the degree of regularity is given by $d_{\mathrm{reg}} = 2^s$ for primes of size $\approx 2 \cdot s$ bits, regardless of the total number of $\mathtt{Bars}$ and rounds. Notably, this is exactly the maximum degree that appears in the algebraic model. Further, d_{reg} increases with the prime size, more precisely with the number of chunks m in the decomposition.

Conjecture 2 (from experiments)**.** *A lower bound for the degree of regularity d_{reg} when solving instances of Skyscraper-CICO is given by*

$$d_{\mathrm{reg}} \geq 2^s. \tag{24}$$

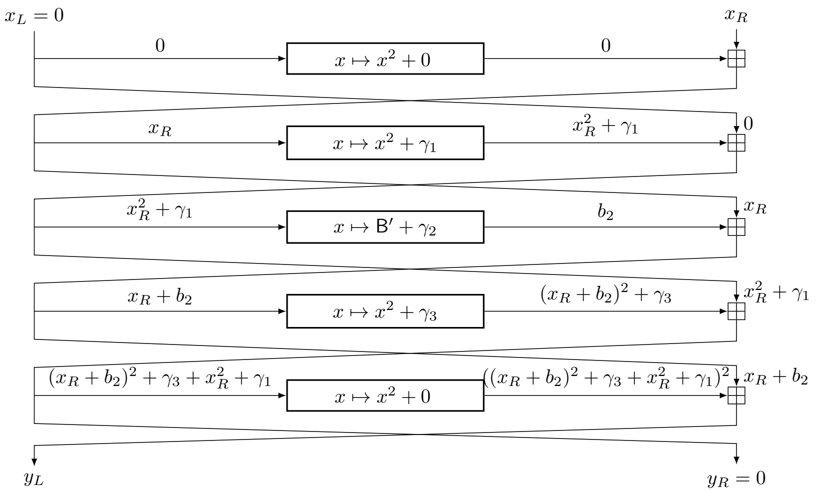

$x_L = 0$
x_R
0
$x \mapsto x^2 + 0$
0
x_R
$x \mapsto x^2 + \gamma_1$
$x_R^2 + \gamma_1$
0
$x_R^2 + \gamma_1$
$x \mapsto \mathsf{B}' + \gamma_2$
b_2
x_R
$x_R + b_2$
$x \mapsto x^2 + \gamma_3$
$(x_R + b_2)^2 + \gamma_3$
$x_R^2 + \gamma_1$
$(x_R + b_2)^2 + \gamma_3 + x_R^2 + \gamma_1$
$x \mapsto x^2 + 0$
$((x_R + b_2)^2 + \gamma_3 + x_R^2 + \gamma_1)^2$
$x_R + b_2$
y_L
$y_R = 0$

Figure 10: Modeling Skyscraper-CICO for the instance $(q, b, N) = (2, 1, 1)$.

Conclusion. We conclude by giving a security estimate for the Gröbner basis attack on Skyscraper-CICO for primes of size ≈ 256 bits. We fix a chunk size of $s = 8$ bits, yielding $m = 32$ chunks for the decomposition. Using the conjectured lower bound for d_{reg} yields the following complexity estimate:

$$\mathcal{C}_{\text{GB}} = n_e \cdot \binom{n_v + d_{\text{reg}}}{d_{\text{reg}}}^{\omega} = (34bN + 1) \cdot \binom{33bN + 1 + 2^8}{2^8}^{\omega}. \tag{25}$$

The respective complexity estimates for different values of b and N are given in Table 7. One can see that for optimistic values $\omega = 2$ and even for $\omega = 1$, the attack cost is prohibitive for Skyscraper (instance $(q, b, N) = (2, 2, 2)$). This is evidence that the Skyscraper design is secure against Gröbner basis attacks.

Table 7: Complexity of computing a Gröbner basis for Skyscraper-CICO over $\mathbb{F}_p$ with $\log_2 p \approx 256$. For every instance, we note the number of Feistel rounds in parentheses, calculated as $R = (2 + b) \cdot N + 2$.

$\mathcal{C}_{\text{GB}}$ (R)	$N = 1$	$N = 2$	$N = 3$
$b = 1$	$2^{147.4\omega+5.1}$ (5)	$2^{233.7\omega+6.1}$ (8)	$2^{300.5\omega+6.6}$ (11)
$b = 2$	$2^{233.7\omega+6.1}$ (6)	$\mathbf{2^{355.9\omega+7.0}}$ **(10)**	$2^{445.1\omega+7.6}$ (14)

4.3.3 Further Considerations

Algebraic Attacks over $\mathbb{F}_2$. As for the case of Monolith (see [GKL+24, Sect. 6.4] for details), even if Bars has a low-degree over $\mathbb{F}_2$, we claim that algebraic attacks against Skyscraper that exploit the binary representation over $\mathbb{F}_2$ are not efficient in general. The reason is the high degree of the $\mathbb{F}_2$ function corresponding to the squaring operation $x \mapsto x^2$, as proved in Lemma 5. For this reason, in the following, we limit ourselves to focus on algebraic attacks that exploit the $\mathbb{F}_p$ representation of Skyscraper.

Algebraic Attacks via Low-degree Approximation of Bars. Algebraic attacks exploit the simple algebraic description of the attacked construction to break it. In Skyscraper, Bars are the only functions with a complex algebraic expression. If a simple algebraic description of it that holds with high probability exists, then the attacker can potentially exploit it to set up a probabilistic algebraic attack on Skyscraper. As we showed in Section 4.1, any linear approximation of `Bars` holds with a probability that is too low to be useful for such an attack. We conjecture that the same occurs for any other low-degree approximation (e.g., quadratic approximations) of `Bars` as well.

Resistance against FreeLunch Attack and Resultants. Other algebraic attacks include the FreeLunch attack [BBL+24] and the resultant approach [YZY+25]. These attacks are particularly efficient in the case of symmetric primitives with simple algebraic descriptions, such as *Rescue*, Griffin, and Anemoi. As for the case of `Reinforced Concrete`, `Monolith`, and Tip5, since there is substantial evidence that `Bars` do not have any low-degree approximations, we expect that neither the FreeLunch attack nor the resultant approach would succeed against Skyscraper.

4.4 Attacks in the Quantum Setting

We claim that Skyscraper provides the same resistance against quantum attacks as other sponge-based hash functions with similar capacity parameters. It is known that the sponge framework provides about $(c\log(p))/3$ bits of quantum security [Unr21], which is arguably sufficient given the high actual costs of quantum attacks. Indeed, the best-known quantum attack against hash functions (and many other symmetric primitives) is quantum search with Grover's algorithm [Gro96], which, given a quantum circuit of the attacked primitive, exhaustively (in the quantum domain) searches for a solution. The algorithm's result stabilizes after around $I \in \mathcal{O}(2^{n/\gamma})$ iterations for preimage and collision search and an input size of n bits, where $2 \leq \gamma \leq 3$ [CNS17].

Regarding non-blackbox attacks, some works appeared recently on some hash functions [HS21, HS20]. We are not aware of any application of these attacks to the full version of Skyscraper.

5 Benchmarks

In this section, we discuss the plain performance of Skyscraper, i.e., its efficiency when evaluated outside of an arithmetic circuit. This aspect has been a crucial motivation for the new design since, in many scenarios, it is important to compute Merkle trees as fast as possible [COS20, Eth22].

As a baseline of discussion, we performed extensive benchmarks comparing Skyscraper to other hash functions relevant within the domain. We include our plain implementations in the open-source benchmarking framework of `Monolith`.[9] They provide implementations of Poseidon, Poseidon2, `Reinforced Concrete`, and Griffin. Further, they merged hash implementations for Tip5 and Tip4'[10] and added SHA3-256 and SHA-256 from `RustCrypto`[11]. On top of this, we merge the implementation of the Anemoi hash function into the framework.[12] All our tests are run on an AMD Ryzen 9 7900X with a single thread with 5 seconds of warmup time and averaged over 10 000 executions. Due to the small time scale, an individual execution of a hash function may deviate depending on the system load, and these results serve as a basis for comparison.

[9] https://extgit.isec.tugraz.at/krypto/zkfriendlyhashzoo/-/tree/master/plain_impls

[10] https://github.com/Neptune-Crypto/twenty-first

[11] https://github.com/RustCrypto/hashes

[12] https://github.com/anemoi-hash/anemoi-rust

Table 8: Hashing ≤ 512 bits of data (8 elements for the 64-bit prime and 2 elements for BLS12-381) with different hash functions. For the category Other, we follow the specification.

		$p =$ BLS12-381		$p = 2^{64} - 2^{32} + 1$		Other
		State t	time [ns]	State t	time [ns]	time [ns]
[SAD20]	*Rescue*-Prime	2	235 630	8	12 819	–
[BBC+23]	Anemoi	2	82 297	–	–	–
[GHR+23]	Griffin	3	64 763	8	1948	–
[AGP+19]	GMiMC	2	12 339	8	1827	–
[GKR+21]	Poseidon	2	8276	8	2345	–
[GOPS22]	Neptune	2	7637	8	2600	–
[GKS23]	Poseidon2	2	4907	8	1827	–
[GKL+22]	Reinforced Concrete	3	1511	–	–	–
[SLS+23]	Tip5	–	–	16	609	–
[Sal23]	Tip4’	–	–	12	305	–
[GKL+24]	Monolith	–	–	8	123	–
Ours	Skyscraper	2	135	–	–	–
[ND15]	SHA3-256	–	–	–	–	196
[ANWW13]	Blake2s	–	–	–	–	80
[NIS02]	SHA-256	–	–	–	–	42

5.1 A Single Permutation Call

In our first evaluation, we started by comparing the latency of a single permutation call of different hash functions. In Table 8, one can see the results of hashing ≤ 512 bits for different hash functions. This is an approximation, as we compare hash functions working over different domains. For the large prime setting, we evaluate hashing 2 elements of $\mathbb{F}_p$ where $p =$ BLS12-381 fully representing 509 bit. For the small prime setting, we evaluate the hash functions for 8 elements of $\mathbb{F}_p$ where p is the Goldilocks Prime $p = 2^{64} - 2^{32} + 1$, representing a total of 511 bits. Finally, the traditional hashes SHA-256, SHA3-256, and Blake2s do not work over a prime field and can thus represent 512 bits. Another caveat in this comparison is the mode of operation. For some hashes, the minimum state size is larger than the input size. There, we instantiate these hash functions in sponge mode. For the large prime, we instantiate Reinforced Concrete and Griffin with a fixed state $\mathbb{F}_p^3$ in sponge mode. For the small prime, Tip5 and Tip4’ are in sponge mode with respective state sizes of 16 and 12. Overall, we chose the minimum state size that hashes at least 509 bits with the best permutation latency.

While there are shortcomings in the comparison, as outlined above, Table 8 clearly highlights the advantage of Skyscraper. Very often, the choice of the underlying prime field is not based on the hashing performance but on the underlying use case. Then, in the case of a large prime, there was simply no way of benefiting from the performance of many of the novel circuit-friendly hash functions. Skyscraper reduces this gap significantly.

5.2 Hashing Performance with Larger Inputs

Next to the 2-to1 compression outlined above, we evaluate the performance of Skyscraper when hashing larger inputs. Comparing in this setting removes the disadvantage of larger states and looks at amortized performance. This time we chose the state size to maximize the amortized performance of the respective hash functions. All evaluated functions are instantiated in sponge mode. For Skyscraper, we showcase multiple state settings dictated by the degree of the underlying extension field. Given the previous benchmarks, there are some unexpected results in Table 9. For big primes, Anemoi has a Sponge instance

Table 9: Hashing 1 Mbit in sponge mode (with the only exception of SHA-256 used in Compression mode).

			$p =$ `BLS12-381`		$p = 2^{64} - 2^{32} + 1$		Other
			`State` t	time [μs]	`State` t	time [μs]	time [μs]
[SAD20]	*Rescue*-Prime		3	495 460	12	42 035	–
[BBC$^+$23]	Anemoi		4	12 864		–	–
[GHR$^+$23]	Griffin		3	140 550	12	4391	–
[AGP$^+$19]	GMiMC		2	52 616	12	6661	–
[GKR$^+$21]	Poseidon		3	26 375	12	6607	–
[GOPS22]	Neptune		4	19 563	12	7213	–
[GKS23]	Poseidon2		3	13 043	12	2780	–
[GKL$^+$22]	`Reinforced Concrete`		3	3411		–	–
[SLS$^+$23]	Tip5			–	16	1000	–
[Sal23]	Tip4’			–	12	586	–
[GKL$^+$24]	`Monolith`			–	12	61	–
Ours	Skyscraper	$\mathbb{F}_p$	2	711		–	–
		$\mathbb{F}_{p^2}$	4	657		–	–
		$\mathbb{F}_{p^3}$	6	880		–	–
[ND15]	SHA3-256		–	–	–	–	180
[ANWW13]	Blake2s		–	–	–	–	138
[NIS02]	SHA-256		–	–	–	–	47

with state size 4, which has a drastically better-amortized runtime than its compression. Similarly, for small primes, Griffin amortized better with its Sponge state size. For Skyscraper, we confirm the preliminary benchmarks of Table 4 showing that $\mathbb{F}_{p^2}$ has the best-amortized performance. Further, Skyscraper remains the fastest hash function in the big prime setting. However, the gap between Skyscraper and both `Monolith` and SHA-256 widens. For `Monolith`, there are better amortization effects with the larger state size, while SHA-256 has generally much stronger performance optimizations.

5.3 Merkle Tree Evaluation

Another significant area of application of arithmetization-oriented hash functions is Merkle tree evaluation. We describe a Merkle tree with a compression function and an *arity* a, which defines the number of child nodes each parent node has. Given a set of n input nodes $\in \{0,1\}^l$ and a hash function that maps $\{0,1\}^{al} \mapsto \{0,1\}^l$, a Merkle tree evaluation recursively compresses the input nodes until only a final node is left. This final node is the root of the tree. As a result, the tree has $d = \log_a n$ layers. Verifying that an element is part of the tree requires only $N_V \in \mathcal{O}(d)$ hashes, substantially improving performance in many applications. Further, reducing the required hashes for verification enhances the efficiency of zero-knowledge-proof systems that rely on Merkle tree constructions. Some protocols rely on Merkle trees at their very heart (e.g., the FRI protocol uses Merkle trees to encode their codewords).

In Table 10, we give an overview of the plain evaluation performance for a Merkle tree with different hash functions and arities. Similar to the tables above, we define $l \approx 256$. In other words, a node is equivalent to an input element $\mathbb{F}_p$ for the large primes considered. Further, for the Goldilocks prime, a node is represented as four $\mathbb{F}_p$ elements, and finally, in the other hash functions, a node represents 256 bits.

The choice of arity in a Merkle tree highly depends on the use case, as it directly impacts the tree’s depth and the required hash verifications per level. We evaluated the hash functions with their respective state sizes and the arities these state sizes permit. Our experiments focus on arities 2 and 4, as these correspond to the arities of Skyscraper

Table 10: Merkle-Tree with $2^{20} \approx 256$-bit input elements. The *Arity* describes the number of 256-bit inputs per permutation call.

		$p =$ `BLS12-381`		$p = 2^{64} - 2^{32} + 1$		Other
		`State` t	time [ms]	`State` t	time [ms]	time [ms]
Arity = 2						
[SAD20]	*Rescue*-Prime	2	250 170	8	13 072	–
[GHR+23]	Griffin	3	74 402	8	1877	–
[AGP+19]	GMiMC	2	13 938	8	1562	–
[GKR+21]	Poseidon	2	9578	8	2006	–
[GOPS22]	Neptune	2	8925	8	2223	–
[GKS23]	Poseidon2	2	5608	8	978	–
[GKL+22]	`Reinforced Concrete`	3	1826		–	–
[SLS+23]	Tip5		–	16	590	–
[Sal23]	Tip4’		–	12	275	–
[GKL+24]	`Monolith`		–	8	182	–
[ND15]	SHA3-256	–	–	–	–	264
[ANWW13]	Blake2s	–	–	–	–	88
[NIS02]	SHA-256	–	–	–	–	66
Ours	Skyscraper $\mathbb{F}_p$	2	220		–	–
Arity = 4						
[GOPS22]	Neptune	4	4808	–	–	–
Ours	Skyscraper $\mathbb{F}_{p^2}$	4	208		–	–
[ND15]	SHA3-256	–	–	–	–	95
[ANWW13]	Blake2s	–	–	–	–	58
[NIS02]	SHA-256	–	–	–	–	37

in $\mathbb{F}_p$ and $\mathbb{F}_{p^2}$, respectively, making them especially relevant for our context. In particular, we evaluated all hash functions for arity two as it is the classical Merkle tree setting. We evaluated those hash functions with arity four that have a large enough state size. Larger arities can be achieved by securely increasing the state size, which requires care and might not be captured by the security analysis of all hash functions. For Skyscraper, increasing the state size works by increasing the degree of the extension field. However, as our experiments above have shown, the increased arity is penalized for higher degrees.

The results of the Merkle tree evaluation follow a similar pattern as the one in the table with the 512-bit compression. We see that Skyscraper is the best-performing hash in the category of large prime fields. Further, we see that Skyscraper is the first hash function on large primes that is competitive compared to small primes, with only `Monolith` being faster here. It is again important to note that Griffin and `Reinforced Concrete` suffer from a minimum state size of three for the large primes. These hash functions might perform better in scenarios with a dataset size of 3^k for some k. We can see that Griffin comparably improves significantly with a fitting state size in the smaller prime field. Given the category of Other hash functions, Skyscraper is again faster than SHA3 but slower than Blake2s and SHA2.

For Arity 4, we provide a small sample of data for hash functions that already support the larger state size. We see that Skyscraper improves its performance over the case for two inputs, resulting in the highest performance for large primes. However, compared to the improvements for the other hashes, it falls behind as the overhead of the extension field operations diminishes the potential speedup. In scenarios where Arity 4 is required or applicable, the degree 2 extension can be recommended. Otherwise, Skyscraper in $\mathbb{F}_p$ provides a good middle ground.

6 Circuit Performance

Here we discuss the circuit performance of Skyscraper, i.e., its efficiency when represented as an arithmetic circuit. For this purpose we assume a general-purpose proving system with the Plonkish arithmetization [GWC19] and support for lookup arguments (e.g., [Hab22]). This characterization applies to various well-known protocols, however, our design can also be represented using different arithmetizations such as R1CS or AIR.

As is customary in the area, we focus on representing the round function of Skyscraper as an arithmetic circuit.

Canonical Decomposition Proof. There is one crucial difference between previous lookup-based hash function designs (e.g. `Reinforced Concrete` or `Monolith`) and Skyscraper. Indeed, let us focus on the lookup step of the round function. There we decompose (i.e., split) a field element x into various ι-bit chunks, such that a combination of these chunks sums up to x. After applying the lookups, the results are composed together into an output field element, which is essentially the inverse operation of the first step. Unfortunately, however, one main problem arises when combining this approach with arbitrary fields. In particular, it has to be guaranteed that the chunks received from the first step are the correct decomposition of $x \in \mathbb{N}$ rather than $x + p \in \mathbb{N}$ for a prime field $\mathbb{F}_p$. Otherwise, a cheating prover may use the (wrong) decomposition for $x + p \in \mathbb{N}$ and apply incorrect lookup operations. Without any countermeasures, this does not violate the constraint system since clearly $x \equiv x + p \mod p$. This issue can be solved efficiently in previous hash function designs, where the considered prime numbers have advantageous forms (e.g., $2^{31} - 1$ or $2^{64} - 2^{32} + 1$), allowing for simple constraints based on the bit representation of the prime number in order to avoid this attack. In particular, the structure of consecutive `0` and `1` bits is exploited, and the chunk boundaries align with these sequences of bits (we refer to [GKL+24] for more details).

When using arbitrary prime numbers, as is the case for Skyscraper, the sequences of consecutive `0` and `1` bits in their representation are shorter in general, making this solution significantly less efficient. We solve this issue by applying a *canonical decomposition proof* to the trace elements, which allows to prove that the correct decomposition is used for arbitrary prime numbers and without adding too much overhead. We explain our approach in App. B and use it in the circuit description below.

6.1 Square Operation

The first component, which merely consists of the squaring operation in a Feistel network, requires a single degree-2 constraint reflecting the squaring and (linear) round constant addition itself. Considering a state of two elements, these constraints are applied to three trace elements, namely the two input elements of the round and the new output element (note that one of the output elements is equal to one of the input elements).

Similar to `Monolith`, our design allows for a tradeoff between the number of variables and the degree of the constraints. Indeed, one may decide to use two constraints, one of degree 2 and one of degree 4, to map the two consecutive Feistel rounds. In this case, we need four trace elements, where one of the output elements is described as a degree-2 function of the input trace elements and the other is described as a degree-4 function of the input trace elements.

6.2 Bars Operation

The `Bars` operation makes heavy use of lookup tables and also uses a canonical decomposition proof. We summarize the circuit cost of these components below.

Table 11: Circuit performance of `Bars`. All constraints are linear.

Witnesses	Constraints	Lookups
$8 + 3m$	10	$2m$

- First, the input element x gets split into two chunks x_1 and x_2 such that $x = x_2 \,||\, x_1$. These two chunks are used in a binary comparison, adding the two trace elements s_1, s_2 and a linear constraint for both of them. In particular,

  $$s_2 = p_2 - x_2 \quad \text{and} \quad s_1 = p_1 - x_1,$$

  where p_1 and p_2 are the lower and upper half of the prime p, respectively, and thus publicly known values (see also App. B).
  (Cost: 5 witnesses, 3 linear constraints)

- We then show that s_1 and s_2 are each within an allowed range. To do this, we split them into lookup-sized chunks, needing m chunks in total.
  (Cost: m witnesses, 2 linear constraints), m lookups

- Finally, we apply the lookup operations to x_1 and x_2 (and hence to x). Note that this also concludes the first step, implicitly showing that the chunks x_1 and x_2 are within their allowed range. We arrive at the new elements y_1, y_2, and eventually y.
  (Cost: $3 + 2m$ witnesses, 5 linear constraints, m lookups)

The circuit cost of `Bars` is summarized in Table 11.

6.3 Concrete Instance for $p \approx 2^{256}$

We consider an instance with two state elements and $\log_2(p) \approx 256$ (e.g., BN254). Assuming 16-bit lookup tables, we have $m = 16$. We further assume a maximum constraint degree of 2, which means that we handle the Feistel rounds separately, requiring a total of 3 witnesses and 3 degree-2 constraints for two consecutive Feistel rounds (ignoring the input, which is added separately in the first round and coming from the public input or the preceding `Bars` operation). Taking into account the cost for `Bars` from Table 11, we list the estimated circuit performance of all components of Skyscraper in Table 12. Comparing it to `Reinforced Concrete`, we can see that the number of witnesses and lookups is significantly reduced for $m = 16$. Moreover, the area-degree product is significantly less than `Reinforced Concrete`. Even when opting for 8-bit lookup tables (i.e., $m = 32$, essentially doubling the number of witness elements and lookups in `Bars`), Skyscraper is still more efficient in the circuit.

6.4 Proof System Implementation

In addition to the theoretical discussion, we provide a proof system implementation of Skyscraper in the Halo2 [BGH19] framework. This framework builds on the Plonkish arithmetization style and allows for custom gates and lookup tables. Further, with application in Zcash, it is an established framework fit for a draft implementation.

As indicated above, there is a high degree of freedom when choosing the concrete polynomial constraints resulting in very different metrics of performance. We have to balance the degree of individual constraints vs the number of rows in the table. Further, we can increase the number of polynomials (i.e., columns of our table) to reduce the required rows for a computation. These considerations are highly use case dependant so we went for a balanced approach to showcase the general feasibility of a proof system implementation.

Table 12: Circuit performance of all components of Skyscraper ($n = 1$), where we assume the BN254 setting. This also includes a comparison with Reinforced Concrete, Poseidon, Poseidon2, and *Rescue*-Prime, where d is the smallest positive integer such that $\gcd(d, p-1) = 1$ (e.g., $d = 5$ for BN254).

Component	Witnesses	Constraints (deg.)	Lookups	A-D product
Double squaring	3	3 (2)	0	–
All squarings	9	9 (2)	0	–
Bars	56	10 (1)	32	–
All Bars	224	40 (1)	128	–
Skyscraper	233	49 $(1,2)$	128	1398
Reinforced Concrete	378	24 $(1,3,d)$	267	5670
Poseidon, Poseidon2	80	80 (d)	0	1200
Rescue-Prime	42	42 (d)	0	630
Anemoi	42	42 (d)	0	630
Griffin	42	42 $(3,d)$	0	630

Table 13: Concrete cost of a single permutation in the Halo2 ZKP library.

	m	Prove [ms]	Verify [ms]
Poseidon	–	41.0	1.6
Skyscraper	16	5076.0	85.0
	32	98.0	2.4

Concretely, our arithmetization in Halo2 requires 4 advice columns and 5 public fixed columns. The advice columns hold the left and right state, respectively, and the composition and decomposition are stored in two columns. The fixed columns are filled with the lookup table pairs, the round constants, and the selector bits. We provide two settings for decomposition, one evaluating a LUT of 16 bits and $m = 16$ chunks and one with a LUT of 8 bits and $m = 32$ chunks. These settings require 74 and 138 rows, respectively. However, for a Plonkish lookup, the key-value pairs are stored each in one of the fixed columns requiring at least $2^k + 4m$ rows. Since the row count is a power of two, they require 2^9 and 2^{17} rows, respectively. The performance given in Table 13 shows this tradeoff. In other words, we could fit 884 executions of Skyscraper into the proof with a 16-bit LUT since $2^k + 884 \cdot 4m + s < 2^{17}$, where s is a required security buffer for the LUT constraint.

In summary, Skyscraper offers a proof system performance at half the speed of a Poseidon implementation for one permutation, but with potential for amortization with large batches of hash evaluations. We emphasize that this also includes Merkle path openings and hashing longer inputs, two prominent settings where multiple permutation calls have to be proven consecutively.

Acknowledgements

Lorenzo Grassi has been supported by the Ethereum Foundation via the Academic Grant Program 2024, and by the German Research foundation (DFG) within the framework of the Excellence Strategy of the Federal Government and the States – EXC 2092 CaSa. Further, this work is partly supported by the European Union under the project Confidential6G with Grant agreement ID: 101096435. Clémence Bouvier was supported by the European Research Council (ERC, grant number 101097056 "SymTrust").

References

[AGP+19] Martin R. Albrecht, Lorenzo Grassi, Léo Perrin, Sebastian Ramacher, Christian Rechberger, Dragos Rotaru, Arnab Roy, and Markus Schofnegger. Feistel structures for MPC, and more. In *ESORICS 2019*, volume 11736 of *LNCS*, pages 151–171, 2019.

[ANWW13] Jean-Philippe Aumasson, Samuel Neves, Zooko Wilcox-O'Hearn, and Christian Winnerlein. BLAKE2: Simpler, smaller, fast as MD5. In *ACNS 2013*, volume 7954 of *LNCS*, pages 119–135, 2013.

[BBC+23] Clémence Bouvier, Pierre Briaud, Pyrros Chaidos, Léo Perrin, Robin Salen, Vesselin Velichkov, and Danny Willems. New design techniques for efficient arithmetization-oriented hash functions: Anemoi permutations and Jive compression mode. In *CRYPTO 2023*, volume 14083 of *LNCS*, pages 507–539. Springer, 2023.

[BBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/46, 2018. https://ia.cr/2018/046.

[BBL+24] Augustin Bariant, Aurélien Boeuf, Axel Lemoine, Irati Manterola Ayala, Morten Øygarden, Léo Perrin, and Håvard Raddum. The algebraic FreeLunch: Efficient Gröbner basis attacks against arithmetization-oriented primitives. In *CRYPTO 2024*, volume 14923 of *LNCS*, pages 139–173. Springer, 2024.

[BBLP22] Augustin Bariant, Clémence Bouvier, Gaëtan Leurent, and Léo Perrin. Algebraic attacks against some arithmetization-oriented primitives. *IACR Trans. Symmetric Cryptol.*, 2022(3):73–101, 2022.

[BBS99] Eli Biham, Alex Biryukov, and Adi Shamir. Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. In *EUROCRYPT 1999*, volume 1592 of *LNCS*, pages 12–23, 1999.

[BCC11] Christina Boura, Anne Canteaut, and Christophe De Cannière. Higher-order differential properties of Keccak and Luffa. In *FSE 2011*, volume 6733 of *LNCS*, pages 252–269, 2011.

[BDFG21] Dan Boneh, Justin Drake, Ben Fisch, and Ariel Gabizon. Halo Infinite: Proof-carrying data from additive polynomial commitments. In *CRYPTO 2021*, volume 12825 of *LNCS*, pages 649–680. Springer, 2021.

[BDPA11] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Note on zero-sum distinguishers of Keccak-f, 2011. https://keccak.team/files/NoteZeroSum.pdf.

[BDPV07] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Sponge functions. ECRYPT Hash Workshop 2007, 2007. http://www.csrc.nist.gov/pki/HashWorkshop/PublicComments/2007_May.html.

[BDPV08] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the indifferentiability of the Sponge construction. In *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 181–197, 2008.

[BDPV11] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. The KECCAK reference, 2011. https://keccak.team/files/Keccak-reference-3.0.pdf.

[BGH19] Sean Bowe, Jack Grigg, and Daira Hopwood. Halo: Recursive proof composition without a trusted setup. Cryptology ePrint Archive, Paper 2019/1021, 2019. https://ia.cr/2019/1021.

[BH23] Gautam Botrel and Youssef El Housni. Faster Montgomery multiplication and multi-scalar-multiplication for SNARKs. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(3):504–521, 2023.

[BS90] Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. In *CRYPTO 1990*, volume 537 of *LNCS*, pages 2–21. Springer, 1990.

[BS93] Eli Biham and Adi Shamir. *Differential Cryptanalysis of the Data Encryption Standard*. Springer, 1993.

[BSB23] Alexandre Belling, Azam Soleimanian, and Olivier Bégassat. Recursion over public-coin interactive proof systems; faster hash verification. In *CCS 2023*, pages 1422–1436. ACM, 2023.

[BSGL20] Eli Ben-Sasson, Lior Goldberg, and David Levit. STARK friendly hash – survey and recommendation. Cryptology ePrint Archive, Paper 2020/948, 2020. https://ia.cr/2020/948.

[BSV07] Thomas Baignères, Jacques Stern, and Serge Vaudenay. Linear cryptanalysis of non binary ciphers. In *SAC 2007*, volume 4876 of *LNCS*, pages 184–211. Springer, 2007.

[BW12] Andrey Bogdanov and Meiqin Wang. Zero correlation linear cryptanalysis with reduced data complexity. In *FSE 2012*, volume 7549 of *LNCS*, pages 29–48. Springer, 2012.

[CBBZ23] Binyi Chen, Benedikt Bünz, Dan Boneh, and Zhenfei Zhang. HyperPlonk: Plonk with linear-time prover and high-degree custom gates. In *EUROCRYPT 2023*, volume 14005 of *LNCS*, pages 499–530. Springer, 2023.

[CLO15] David A. Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms*. Springer, 4 edition, 2015.

[CNS17] André Chailloux, María Naya-Plasencia, and André Schrottenloher. An efficient quantum collision search algorithm and implications on symmetric cryptography. In *ASIACRYPT 2017*, volume 10625 of *LNCS*, pages 211–240. Springer, 2017.

[COS20] Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. Fractal: Post-quantum and transparent recursive proofs from Holography. In *EUROCRYPT 2020*, volume 12105 of *LNCS*, pages 769–793. Springer, 2020.

[Dae95] Joan Daemen. Cipher and hash function design strategies based on linear and differential cryptanalysis. Doctoral Dissertation, K.U. Leuven, 1995. https://cs.ru.nl/~joan/papers/JDA_Thesis_1995.pdf.

[DGGK21] Christoph Dobraunig, Lorenzo Grassi, Anna Guinet, and Daniël Kuijsters. Ciminion: Symmetric encryption based on Toffoli-gates over large finite fields. In *EUROCRYPT 2021*, volume 12697 of *LNCS*, pages 3–34. Springer, 2021.

[DhSD06] Augusto Jun Devegili, Colm Ó hÉigeartaigh, Michael Scott, and Ricardo Dahab. Multiplication and squaring on pairing-friendly fields. Cryptology ePrint Archive, Paper 2006/471, 2006. https://ia.cr/2006/471.

[DKLS20] Orr Dunkelman, Abhishek Kumar, Eran Lambooij, and Somitra Kumar Sanadhya. Counting active S-boxes is not enough. In *INDOCRYPT 2020*, volume 12578 of *LNCS*, pages 332–344. Springer, 2020.

[DP24] Benjamin E. Diamond and Jim Posen. Polylogarithmic proofs for multilinears over binary towers. Cryptology ePrint Archive, Paper 2024/504, 2024. https://ia.cr/2024/504.

[Eth22] Ethereum Foundation. ZKEVM introduction. https://github.com/privacy-scaling-explorations/zkevm-specs/blob/master/specs/introduction.md, 2022.

[Fau99] Jean-Charles Faugère. A new efficient algorithm for computing Gröbner bases (F4). *Journal of Pure and Applied Algebra*, 139(1):61–88, 1999.

[Fau02] Jean-Charles Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). In *ISSAC 2002*, page 75–83. ACM, 2002.

[FGLM93] Jean-Charles Faugère, Patrizia M. Gianni, Daniel Lazard, and Teo Mora. Efficient computation of zero-dimensional Gröbner bases by change of ordering. *J. Symb. Comput.*, 16(4):329–344, 1993.

[Fil24] Filecoin. Filecoin. https://filecoin.io/, 2024. Accessed: 2024-10-13.

[GHR$^+$23] Lorenzo Grassi, Yonglin Hao, Christian Rechberger, Markus Schofnegger, Roman Walch, and Qingju Wang. Horst meets Fluid-SPN: Griffin for zero-knowledge applications. In *CRYPTO 2023*, volume 14083 of *LNCS*, pages 573–606. Springer, 2023.

[GKL$^+$22] Lorenzo Grassi, Dmitry Khovratovich, Reinhard Lüftenegger, Christian Rechberger, Markus Schofnegger, and Roman Walch. Reinforced Concrete: A fast hash function for verifiable computation. In *CCS 2022*, pages 1323–1335. ACM, 2022.

[GKL$^+$24] Lorenzo Grassi, Dmitry Khovratovich, Reinhard Lüftenegger, Christian Rechberger, Markus Schofnegger, and Roman Walch. Monolith: Circuit-friendly hash functions with new nonlinear layers for fast and constant-time implementations. *IACR Trans. Symmetric Cryptol.*, 2024(3):44–83, 2024.

[GKR$^+$21] Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger. Poseidon: A new hash function for zero-knowledge proof systems. In *USENIX 2021*, pages 519–535. USENIX Association, 2021.

[GKS23] Lorenzo Grassi, Dmitry Khovratovich, and Markus Schofnegger. Poseidon2: A faster version of the Poseidon hash function. In *AFRICACRYPT 2023*, volume 14064 of *LNCS*, pages 177–203. Springer, 2023.

[GLST22] Jian Guo, Guozhen Liu, Ling Song, and Yi Tu. Exploring SAT for cryptanalysis: (Quantum) collision attacks against 6-round SHA-3. In *ASIACRYPT 2022*, volume 13793 of *LNCS*, pages 645–674. Springer, 2022.

[GOPS22] Lorenzo Grassi, Silvia Onofri, Marco Pedicini, and Luca Sozzi. Invertible quadratic non-linear layers for MPC-/FHE-/ZK-friendly schemes over F_p^n: Application to Poseidon. *IACR Trans. Symmetric Cryptol.*, 2022(3):20–72, 2022.

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *STOC 1996*, pages 212–219. ACM, 1996.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In *EUROCRYPT 2016*, volume 9666 of *LNCS*, pages 305–326. Springer, 2016.

[GRR16] Lorenzo Grassi, Christian Rechberger, and Sondre Rønjom. Subspace trail cryptanalysis and its applications to AES. *IACR Trans. Symmetric Cryptol.*, 2016(2):192–225, 2016.

[GW20] Ariel Gabizon and Zachary J. Williamson. plookup: A simplified polynomial protocol for lookup tables. Cryptology ePrint Archive, Paper 2020/315, 2020. https://ia.cr/2020/315.

[GWC19] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: Permutations over Lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019. https://ia.cr/2019/953.

[Hab22] Ulrich Haböck. Multivariate lookups based on logarithmic derivatives. Cryptology ePrint Archive, Report 2022/1530, 2022. https://ia.cr/2022/1530.

[HLP24] Ulrich Haböck, David Levit, and Shahar Papini. Circle STARKs. Cryptology ePrint Archive, Report 2024/278, 2024. https://ia.cr/2024/278.

[HS20] Akinori Hosoyamada and Yu Sasaki. Finding hash collisions with quantum computers by using differential trails with smaller probability than birthday bound. In *EUROCRYPT 2020*, volume 12106 of *LNCS*, pages 249–279. Springer, 2020.

[HS21] Akinori Hosoyamada and Yu Sasaki. Quantum collision attacks on reduced SHA-256 and SHA-512. In *CRYPTO 2021*, volume 12825 of *LNCS*, pages 616–646. Springer, 2021.

[JK97] Thomas Jakobsen and Lars R. Knudsen. The interpolation attack on block ciphers. In *FSE 1997*, volume 1267 of *LNCS*, pages 28–40. Springer, 1997.

[JNP12] Jérémy Jean, María Naya-Plasencia, and Thomas Peyrin. Improved rebound attack on the finalist Grøstl. In *FSE 2012*, volume 7549 of *LNCS*, pages 110–126. Springer, 2012.

[KBM23] Dmitry Khovratovich, Mario Marhuenda Beltrán, and Bart Mennink. Generic security of the SAFE API and its applications. In *ASIACRYPT 2023*, volume 14445 of *LNCS*, pages 301–327. Springer, 2023.

[KLR24] Katharina Koschatko, Reinhard Lüftenegger, and Christian Rechberger. Exploring the six worlds of Gröbner basis cryptanalysis: Application to Anemoi. *IACR Trans. Symmetric Cryptol.*, 2024(4):138–190, 2024.

[Knu94] Lars R. Knudsen. Truncated and higher order differentials. In *FSE 1994*, volume 1008 of *LNCS*, pages 196–211. Springer, 1994.

[Knu98] Lars R. Knudsen. *Deal : a 128-bit Block Cipher*, volume 151 of *Reports in informatics (trykt utg.)*. Department of Informatics, University of Bergen, 1998.

[KS24] Abhiram Kothapalli and Srinath T. V. Setty. HyperNova: Recursive arguments for customizable constraint systems. In *CRYPTO 2024*, volume 14929 of *LNCS*, pages 345–379. Springer, 2024.

[KST22] Abhiram Kothapalli, Srinath T. V. Setty, and Ioanna Tzialla. Nova: Recursive zero-knowledge arguments from folding schemes. In *CRYPTO 2022*, volume 13510 of *LNCS*, pages 359–388. Springer, 2022.

[LFG23] Guiwen Luo, Shihui Fu, and Guang Gong. Speeding up multi-scalar multiplication over fixed points towards efficient zkSNARKs. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(2):358–380, 2023.

[LMR+09] Mario Lamberger, Florian Mendel, Christian Rechberger, Vincent Rijmen, and Martin Schläffer. Rebound distinguishers: Results on the full Whirlpool compression function. In *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 126–143. Springer, 2009.

[LWY+23] Tao Lu, Chengkun Wei, Ruijing Yu, Chaochao Chen, Wenjing Fang, Lei Wang, Zeke Wang, and Wenzhi Chen. cuZK: Accelerating zero-knowledge proof with a faster parallel multi-scalar multiplication algorithm on GPUs. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(3):194–220, 2023.

[Mat93] Mitsuru Matsui. Linear cryptanalysis method for DES cipher. In *EUROCRYPT 1993*, volume 765 of *LNCS*, pages 386–397. Springer, 1993.

[Mon85] Peter L. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44(170):519–521, 1985.

[ND15] NIST and Morris J. Dworkin. SHA-3 Standard: Permutation-based hash and extendable-output functions. Technical Report FIPS 202, National Institute of Standards and Technology (NIST), 2015.

[NIS02] NIST. Secure Hash Standard. Technical Report FIPS 180-2, National Institute of Standards and Technology, August 2002.

[SAD20] Alan Szepieniec, Tomer Ashur, and Siemen Dhooghe. Rescue-Prime: a standard specification (SoK). Cryptology ePrint Archive, Report 2020/1143, 2020. https://ia.cr/2020/1143.

[Sal23] Robin Salen. Two additional instantiations from the Tip5 hash function construction, 2023.

[SLS+23] Alan Szepieniec, Alexander Lemmens, Jan Ferdinand Sauer, Bobbin Threadbare, and Al-Kindi. The Tip5 hash function for recursive STARKs. Cryptology ePrint Archive, Paper 2023/107, 2023.

[Sta24] StarkWare. Stone Prover. https://github.com/starkware-libs/stone-prover, 2024. Accessed: 2025-01-06.

[Unr21] Dominique Unruh. Compressed permutation oracles (and the collision-resistance of sponge/SHA3). Cryptology ePrint Archive, Paper 2021/062, 2021.

[YZY+25] Hongsen Yang, Qun-Xiong Zheng, Jing Yang, Quanfeng Liu, and Deng Tang. A new security evaluation method based on resultant for arithmetic-oriented algorithms. In *ASIACRYPT 2024*, volume 15490 of *LNCS*, pages 457–489. Springer, 2025.

[ZHY+24] Xudong Zhu, Haoqi He, Zhengbang Yang, Yi Deng, Lutan Zhao, and Rui Hou. Elastic MSM: A fast, elastic and modular preprocessing technique for multi-scalar multiplication algorithm on GPUs. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2024(4):258–284, 2024.

A Experimental Results: Gröbner Basis Attacks

Table 14: Practical results of the Gröbner basis attack against Skyscraper-CICO for the generalized instance $(q, b) = (2, 1)$, as described in Section 4.3.2, and a chunk size of $s = 4$.

Instance				Alg. model			Gröbner basis computation (step 1)				
p	m	N	R	n_v	n_e	$d_{\max}$	time [s]	mem. [MB]	d_{reg}	Degrees $\mathrm{GB}_{\mathrm{DRL}}$	#Sol.
37	2	1	5	4	5	16	1	33	16	1-1-1-1	1
79	2	1	5	4	5	16	1	33	16	2-1-1-1	2
89	2	1	5	4	5	16	1	33	16	2-2-2-1-1	3
127	2	1	5	4	5	16	1	33	16	1-1-1-1	1
149	2	1	5	4	5	16	1	33	16	2-2-2-1-1	3
233	2	1	5	4	5	16	1	33	16	1-1-1-1	1
251	2	1	5	4	5	16	1	33	16	1-1-1-1	1
563	3	1	5	5	6	16	1	46	17	1-1-1-1-1	1
641	3	1	5	5	6	16	1	46	17	1-1-1-1-1	1
1361	3	1	5	5	6	16	2	68	18	2-1-1-1-1	2
1423	3	1	5	5	6	16	2	68	18	1-1-1-1-1	1
1741	3	1	5	5	6	16	2	103	18	1-1-1-1-1	1
2129	3	1	5	5	6	16	3	121	20	2-1-1-1-1	2
2917	3	1	5	5	6	16	9	228	22	2-2-2-1-1-1	3
2999	3	1	5	5	6	16	6	196	22	1-1-1-1-1	1
11681	4	1	5	6	7	16	100	997	24	1-1-1-1-1-1	1
12227	4	1	5	6	7	16	119	1178	24	2-1-1-1-1-1	2
30133	4	1	5	6	7	16	1079	5082	26	1-1-1-1-1-1	1
30631	4	1	5	6	7	16	1046	5114	26	1-1-1-1-1-1	1
49559	4	1	5	6	7	16	18940	20201	29	1-1-1-1-1-1	1
52223	4	1	5	6	7	16	18928	20136	29	1-1-1-1-1-1	1
56711	4	1	5	6	7	16	21135	23445	30	1-1-1-1-1-1	1
43	2	2	8	7	9	16	1	33	16	1-1-1-1-1-1-1	1
83	2	2	8	7	9	16	3	136	16	2-2-2-1-1-1-1	3
109	2	2	8	7	9	16	3	174	16	1-1-1-1-1-1-1	1
113	2	2	8	7	9	16	5	170	16	1-1-1-1-1-1-1	1
211	2	2	8	7	9	16	12	685	16	2-1-1-1-1-1-1	2
251	2	2	8	7	9	16	8	353	16	1-1-1-1-1-1-1	1
647	3	2	8	9	11	16	165	1888	17	1-1-1-1-1-1-1-1-1	1
719	3	2	8	9	11	16	159	1888	17	1-1-1-1-1-1-1-1-1	1
1153	3	2	8	9	11	16	19602	53679	18	2-1-1-1-1-1-1-1-1	2
1543	3	2	8	9	11	16	38561	106476	18	1-1-1-1-1-1-1-1-1	1
1861	3	2	8	9	11	16	49714	237205	19	2-2-2-1-1-1-1-1-1	3
127	2	3	11	10	13	16	238	4747	16	1-1-1-1-1-1-1-1-1-1	1
223	2	3	11	10	13	16	392	4003	16	1-1-1-1-1-1-1-1-1-1	1
229	2	3	11	10	13	16	495	5194	16	1-1-1-1-1-1-1-1-1-1	1
239	2	3	11	10	13	16	503	5194	16	1-1-1-1-1-1-1-1-1-1	1

B Efficient High-Degree Functions over Arbitrary Fields

Assume we have a prime number p and set $n = \lceil \log_2(p) \rceil$. In the following, we give an efficient decomposition rule for arbitrary prime numbers. The result will be a provable decomposition

$$(x_1, x_2, \ldots, x_m)$$

of a value $x \in \mathbb{F}_p$, where the prover can efficiently show that

$$x = \sum_{i=1}^{m} 2^{\iota(i-1)} x_i < p$$

Table 15: Practical results of the Gröbner basis attack against Skyscraper-CICO for the generalized instance $(q, b) = (2, 1)$, as described in Section 4.3.2, and a chunk size of $s = 8$.

Instance				**Alg. model**			**Gröbner basis computation (step 1)**				
p	m	N	R	n_v	n_e	$d_{\max}$	time [s]	mem. [MB]	d_{reg}	Degrees GB_{DRL}	#Sol.
10973	2	1	5	4	5	256	103	1665	256	1-1-1-1	1
12277	2	1	5	4	5	256	207	2033	256	1-1-1-1	1
12809	2	1	5	4	5	256	574	3660	256	2-1-1-1	2
17033	2	1	5	4	5	256	483	3407	256	1-1-1-1	1
25057	2	1	5	4	5	256	2604	11395	256	2-2-2-2-2-2-1	4
28837	2	1	5	4	5	256	1348	7956	256	1-1-1-1	1
45943	2	1	5	4	5	256	3282	17224	256	1-1-1-1	1
51647	2	1	5	4	5	256	4210	21075	256	1-1-1-1	1
52541	2	1	5	4	5	256	4312	21810	256	1-1-1-1	1

Table 16: Practical results of the Gröbner basis attack against Skyscraper-CICO for the generalized instance $(q, b) = (2, 2)$, as described in Section 4.3.2, and a chunk size of $s = 4$.

Instance				**Alg. model**			**Gröbner basis computation (step 1)**				
p	m	N	R	n_v	n_e	$d_{\max}$	time [s]	mem. [MB]	d_{reg}	Degrees GB_{DRL}	#Sol.
37	2	1	6	7	9	16	1	33	16	2-2-2-1-1-1-1-1	3
109	2	1	6	7	9	16	2	79	16	1-1-1-1-1-1-1	1
163	2	1	6	7	9	16	3	146	16	1-1-1-1-1-1-1	1
191	2	1	6	7	9	16	4	149	16	1-1-1-1-1-1-1	1
587	3	1	6	9	11	16	243	3661	17	1-1-1-1-1-1-1-1-1	1
1237	3	1	6	9	11	16	7550	31871	18	2-1-1-1-1-1-1-1-1	2
1361	3	1	6	9	11	16	18974	56732	18	1-1-1-1-1-1-1-1-1	1
1399	3	1	6	9	11	16	18912	56828	18	1-1-1-1-1-1-1-1-1	1
2297	3	1	6	9	11	16	51103	194271	20	1-1-1-1-1-1-1-1-1	1
2953	3	1	6	9	11	16	123706	385124	22	1-1-1-1-1-1-1-1-1	1

over $\mathbb{N}$ for a bucket size of ι bits, i.e., $\log_2(x_i) < \iota$ for $i \in \{1, \dots, m\}$.

For simplicity, but w.l.o.g., assume that $\frac{n}{2} \in \mathbb{N}$ so that we can consider a decomposition into two buckets of the same size. Now, let $p = 2^{n/2} p_2 + p_1$, where $\lceil \log_2(p_i) \rceil = n/2$. Let us abbreviate this by $p = p_2 \mid\mid p_1$. An input $x \in \mathbb{F}_2^n$, allegedly $x \in \mathbb{F}_p$, is split in two $(n/2)$-bit chunks x_i, i.e., $x = x_2 \mid\mid x_1$. The goal of the next steps is to prove that the decomposition of x as an integer is a canonical field element representation, i.e., $2^{n/2} x_2 + x_1 < p$. We apply the following technique.

1. First, we compute $s_2 = p_2 - x_2$ and $s_1 = p_1 - x_1$ and store these two results in the trace.

2. We then show that the decomposition of s_2 only needs $(n/2)$ bits, which is easy to do with range checks. Moreover, no overflow can occur, since any $(n/2)$-bit value is clearly a canonical representation in the n-bit field. The main idea behind this step is that for $a, b \in \mathbb{F}_p$ (where $n = \lceil \log_2(p) \rceil$ and a, b are at most $(n/2)$ bits large each), the value $a - b$ is only representable by a $(n/2)$-bit decomposition in the field if $a - b \geq 0 \implies a \geq b$.

3. Since we have a valid $(n/2)$-bit decomposition of s_2, we now know that $p_2 \geq x_2$, i.e., the upper half of the prime p is larger than or equal to the upper half of the input x.

4. The final step is to apply the same approach to the lower halves if $p_2 = x_2$ (i.e., test if $p_1 > x_1$).

We prove the assumptions in the second and third step.

Lemma 9. *Let $a, b \in \mathbb{F}_p$ for $p \geq 5$ and let $n = \lceil \log_2(p) \rceil$. Further, let $\log_2(a) < n/2$ and $\log_2(b) < n/2$. Then $\log_2(a - b) < n/2$ if and only if $a \geq b$ (as integers).*

Proof. Clearly, $a \geq b \implies \log_2(a-b) < n/2$. For simplicity, set $\tilde{p} \approx \sqrt{p}$. Let us now assume $a < b$. The possible values are in the range $\{p - \tilde{p}, \dots, p-2, p-1\}$. The largest value (as an integer) is $p-1$, and note that $\log_2(p-1) \leq n/2$. The smallest value is $p - \tilde{p}$. Since $\tilde{p} \approx \sqrt{p}$ and $p \geq 5$, subtracting $\tilde{p}$ from p will change the bit representation in the most significant half. This statement is clearly the same for all values inbetween. Indeed, we can even conclude that the field representation of $a - b$ will have at least one bit set in the most significant half of its bit representation if $a < b$. □

Note that the ι-bit decompositions for x_1 and x_2 do not require additional witnesses, since we later need them anyway for the lookup applications. Indeed, the lookup applications implicitly include range checks for those, so we do not need any additional range checks here either. We only need additional range checks for s_i.

Probabilistic Alternative. When dealing with large primes (e.g., $n \approx 256$), we may be able to ignore the final step and just show that $p_2 \geq x_2$ and $p_2 \neq x_2$. This results in a small class of circuits (around $\frac{L}{2^{n/2}}$ of them for L large-word lookups) being impossible to prove. However, it would allow for the following efficient description of the proof in the circuit.

1. We first show that $p_2 \geq x_2$ using the approach above. In the circuit it means showing that a $(n/2)$-bit decomposition is sufficient to represent s_2.

2. We then show that $p_2 \neq x_2$ by storing a witness z such that $z(p_2 - x_2) = 1$ (i.e., $z = (p_2 - x_2)^{-1}$).

For our prime sizes this approach results in an overhead of only 50% regarding range checks and lookup operations.