Two additional instantiations from the Tip5 hash function construction

Robin Salen

Toposware, Inc., USA
research@toposware.com

February 2, 2023

Abstract. In this short note, we propose two additional instantiations based on the design of the novel hash function Tip5 [SLST23] defined over the finite field $\mathbb{F}_p$ with $p = 2^{64} - 2^{32} + 1$, targeting 128 bits of security, and which parameterization may be better suited for generic-purpose STARK-based projects over this finite field. These instantiations, denoted Tip4 and Tip4' respectively, from their digest size, offer better performances both in native hashing, native digest compression, and in-circuit performance. In particular, Tip4 allows for an even more efficient compression, leveraging the Jive compression mode [BBC+22] to perform 4-to-1 digest compression with only one internal permutation, and Tip4' native cost is only about 12% slower than SHA3-256, the shortest gap to date between algebraic-oriented and traditional hash functions. We accompany these new instantiations with evaluation of their performances and security analysis.

Keywords: Tip5 · Hash functions · Jive · Merkle tree · zk-STARK · AIR

1 Introduction

Algebraic-oriented (AO) hash functions have seen a rising interest over the past years, with the emergence of various zero-knowledge protocols and projects built on them requiring verifiable computation of such primitives, such as in proving knowledge of Merkle Tree authentication paths. While traditional binary hash functions have been designed with hardware considerations, AO hash functions are designed such that they maintain a low multiplicative degree, allowing efficient execution within zero-knowledge protocols. This design requirement however usually induces lower native performances, as opposed to functions like SHA256 or blake2.

In the past few years, we have seen the rise of several AO hash function families, for instance [AABS+19, AGR+16, GKR+19, BBC+22] to name a few, providing generic designs to obtain instances of hash functions over finite fields satisfying the low-multiplicative degree efficiency requirement of zero-knowledge proving systems.

In parallel, there has been an increasing number of zero-knowledge protocol constructions based on FRI [BSBHR17], which removes the need for an algebraic group, hence allowing to use much smaller fields than it was previously possible. Among those fields, the "small" Goldilocks prime field, defined by $\mathbb{F}_p$, with $p = 2^{64} - 2^{32} + 1$ and originally discovered by the Polygon Zero team, enjoys really efficient modular reduction, thanks to its particular shape. In addition, this field has a large two-adicity, making it suitable for zero-knowledge protocols requiring the use of FFTs.

Since its discovery, this field has been adopted in several projects, including [Mid23, Zer23, GDH+22], and has been at the heart of several research topics, for instance in the construction of STARK-friendly elliptic curves over extension fields [SSS22, Por22]. The latter is aimed at being used within the Miden VM [Mid23].

Unlike generic constructions, some designers have opted for designs specific to some finite fields: [GKL+21] defined a novel hash function with three instances, over two commonly used fields (BN254 and BLS12-381 scalar fields), and over a specifically crafted field tailored for their construction; [AKM+22] defined a variation of the Rescue-Prime construction over the "small" Goldilocks field, and more recently, [SLST23] proposed a new construction, inspired from the work of [AABS+19] and [GKL+21], also built on the same small field.

Motivation The recent Tip5 construction from [SLST23] reduces the gap between traditional binary hash functions and AO ones. However, it is application-specific, in that:

- It targets a security level of 160 bits, as aimed to be used within the TritonVM [SV21], while most applications only require 128 bits of security.
- Consequently, its digest size consists of 5 field elements (320 bits), while applications on this field usually require only 4 elements. Having a larger digest size also increases proof size of FRI-based proving systems.
- Its capacity size (6 field elements) is relatively large, reducing the efficiency of hashing arbitrary sequences (only a capacity of 4 is necessary to achieve 128 bits security level, allowing to dedicate a larger portion for the rate).

Our Contribution In this note, we present two variations of the hash function defined in [SLST23], namely Tip4 and Tip4', achieving 128 bits of security, and offering better performances both in terms of native hashing, native digest compression, and in-circuit computation. In particular, Tip4' reduces the gap even further between existing algebraic-oriented hash functions and their traditional counterpart.

2 Background

Let's review in a nutshell the construction of Tip5 hash function designed in [SLST23]. Inspired by the SHARK design strategy of interleaving full S-Box layers with MDS layers, and by the `Reinforced Concrete` design leveraging lookups, it defines a novel sponge construction over $\mathbb{F}_p$ ($p = 2^{64} - 2^{32} + 1$). The specific instantiation has a state of 16 field elements and a rate width of 10 elements. The entire parameterization of Tip5 is available in table 1.

Its internal permutation $f : \mathbb{F}_p^{16} \to \mathbb{F}_p^{16}$ consists of $N = 5$ rounds identically constituted of three subroutines:

- an S-Box layer, operating on each state element independently. It operates a split-and-lookup S-Box denoted S on the first four elements, and a power-map S-Box denoted T on the remaining 12 elements.
- an MDS layer, consisting of a matrix-vector multiplication of the internal state by a 16×16 MDS matrix with coefficients specifically crafted for efficient product in the frequency domain in time $O(k \log k)$. We refer to [AKM+22] for a description on the technique.
- an ARK (Additive Round Constants) layer, adding to each state element an independently sampled constant in $\mathbb{F}_p$.[1]

[1] The N rounds constituting the permutation f are identical, except for the round constants they insert.

3 Alternative instantiations: Tip4 and Tip4'

We denote by Tip4 and Tip4' our two variants of the original Tip5 construction, respectively with state width 12 and 16. The complete sets of parameters defining these two instantiations are listed in 1 along with the original Tip5 parameters.

Table 1: *Summary of parameters*

Parameter	Tip5	Tip4	Tip4'
field modulus	$2^{64} - 2^{32} + 1$	$2^{64} - 2^{32} + 1$	$2^{64} - 2^{32} + 1$
number of rounds	5	5	5
state size	16	16	12
sponge rate	10	12	8
sponce capacity	6	4	4
digest length	5	4	4
power map exponent	7	7	7
number of split-and-lookups/round	4	4	4
number of power maps/round	12	12	8

The main difference from the original construction comes from the digest size and sponge capacity, both reduced to 4 field elements, minimal required value to achieve 128 bits of security. In particular, compared to Tip5:

- Tip4 has a larger rate, allowing to hash more efficiently arbitrary input sequences. In addition, it can perform 3-to-1 compression when instantiated in sponge mode, or 4-to-1 compression when using the Jive compression mode, at no extra cost. The latter approach allows to divide by two the depth of Merkle trees (and hence number of Tip4 permutations).
- Tip4' has a smaller state, while still being able to perform 2-to-1 compression with only one internal permutation call. In addition, its smaller MDS matrix allows for a faster linear layer than the one in Tip5. It also enjoys fewer power maps per round (8 instead of 12).

Similarly to Tip5, both instantiations apply the split-and-lookup S-Box S on the first 4 elements of the state (part of the rate portion).

3.1 Number of rounds

Although the number of rounds could be reduced when aiming at 128 bits security, based on the analysis in section 5, out of caution, we keep the number of rounds at 5 for both Tip4 and Tip4', similarly to the original instance.

3.2 Round constants

Round constants for each new instantiation follows the original round construction generation defined in [SLST23], consisting in hashing with Blake3 the concatenation of byte i (for round constant i) and the ASCII string "Tip4" or "Tip4'" respectively, followed by the same hash output processing mentioned in [SLST23].

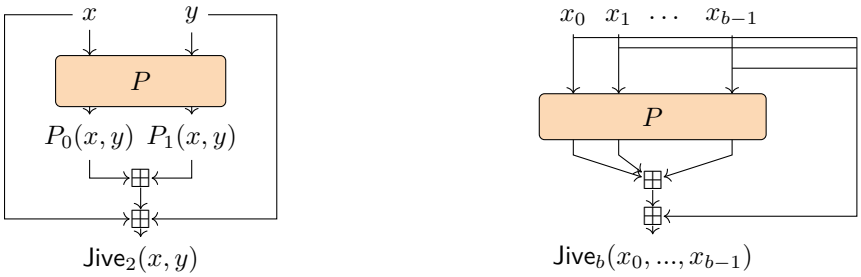
3.3 MDS matrix for Tip4'

Tip4 requires a different MDS matrix than in the original Tip5 construction, because of its smaller state. Keeping efficiency in mind, we opt for a similar approach based on selecting a circulant MDS matrix, which coefficients in the frequency domain are

scaled powers of two, for efficient matrix-vector multiplication in $O(k \log k)$ leveraging FFTs. For this, we choose the same MDS matrix as the 128-bit security instantiation of Rescue-Prime Optimized defined in [AKM+22], i.e. the circulant matrix which first row is $[7, 23, 8, 26, 13, 10, 9, 7, 6, 22, 21, 8]$.

3.4 Use of Jive mode for compression

The Jive mode, introduced in [BBC+22], allows to ignore the sponge construction when compressing digests. In addition, at no extra cost, it can perform k-to-1 compression, for $k > 2$ with still only one internal permutation call. Its construction is detailed in 1.



(a) Jive_2, *which maps* $(\mathbb{F}_q^m)^2$ *to* $\mathbb{F}_q^m$. **(b)** Jive_b, *which maps* $(\mathbb{F}_q^m)^b$ *to* $\mathbb{F}_q^m$.

Figure 1: *The* Jive *compression mode*

In the case of Tip4, using Jive to achieve 4-to-1 compression triples the size of the siblings provided in Merkle tree authentication paths, while halving the depth of those trees, for an overall increase of 50% in authentication path size (major part of FRI-based proofs). This overhead can nevertheless be amortized with several techniques, such as performing recursive proof composition, leveraging Merkle tree inclusion proofs batching, etc.

We note that Tip4' can also benefit from the Jive compression mode, if one wanted to perform 3-to-1 digest compression with only one Tip4' permutation. This approach has been done for instance in [LPP+22]. Although it is possible, FRI-based proving systems usually rely on radix-2 FFTs and 2-to-1 compression functions in Merkle trees (possibly extending them to $2k$-ary Merkle trees with a $2k$-to-1 compression functions). To this end, it makes the use of 3-to-1 compression less trivial as it would involve several modifications in core components of most proving systems based on FRI.

3.5 Arithmetization

The arithmetization of Tip4 and Tip4' is similar to what is described in [SLST23]. Leveraging similar techniques for efficient lookups across tables, we end up with an identical number of regular and extension columns for Tip4, respectively 59 and 21. Tip4' on the other end has 4 fewer regular columns, and the same number of extension columns.

In Merkle trees, arithmetizing the compression using Jive does not yield any overhead, as one can include the final Jive chunk summation when inserting the nodes of the upper layer, without impacting the overall constraint degree.

4 Performances

We highlight below the native performances of hashing and digest compression of our two variants and compare them with the original Tip5 construction. All instances have

been implemented within the winterfell Rust library, a FRI-based STARK proving system implementation, available at [Fac23].

Table 2: *Time for compressing digests*

Tip5 (2-to-1)	Tip4 (2-to-1)	Tip4 (4-to-1)	Tip4' (2-to-1)
1.17 µs	1.16 µs	1.12 µs	397 ns

Table 3: *Time for hashing r elements*

Tip5 ($r = 10$)	Tip4 ($r = 12$)	Tip4' ($r = 8$)
1.09 µs	1.07 µs	442 ns

The larger $\frac{r}{m}$ ratio in the case of Tip4 and Tip4' allows for greater hash throughput, illustrated in 4.

Table 4: *Hash throughput per second*

Tip5	Tip4	Tip4'
9.17M hashes	11.2M hashes	18.1M hashes

In particular, Tip4' is getting particularly close to SHA3-256, compressing two digests being only about 12% slower than the Winterfell SHA3-256 implementation.

4.1 In-circuit performance

While we do not directly provide prover performance estimates in this note, we still highlight the in-circuit efficiency gain brought up by those two new instances:

- Tip4: Tip4 AIR execution trace has the same dimension than the original Tip5 construction. However, leveraging the Jive compression mode allows to halve the number of internal permutations when verifying authentication paths (the major component of FRI-based proof verification). This in turns shortens the execution trace length, effectively reducing the cost of computing FFTs to evaluate the trace polynomials over the LDE domain.
- Tip4': Tip4' AIR execution trace is only slightly narrower than Tip5, however it removes the need for 4 columns of degree 7 (the highest constraint degree of the permutation).

5 Security arguments

Most of the arguments that follow are derived from section 5 of [SLST23]. We refer to that section for extended discussion on how the mentioned attacks work.

- Regarding differential attacks, the MDS matrix remains unchanged with Tip4, yielding an identical probability of differential caracteristic across two rounds of $\left(\frac{\alpha - 1}{p}\right)^{m+1-2s} = 2^{-552}$.
  For Tip4', the MDS matrix has branching factor $m + 1 = 9$, activating fewer power maps per round as we still have 4 split-and-lookup maps per round. This yield a higher probability of successful differential attacks over two rounds, at 2^{-307}, although still low enough to achieve 128 bits security.

- As for Gröbner basis attacks, assuming like in the original construction that the obtained square Macaulay matrix of size $M = \left(\binom{p-1}{m(N+1)-c-d}\right)$ is dense, we can obtain a kernel vector in time $O(M^2)$. For $N = 1$, this gives us about 2^{2913} operations for Tip4, 2^{1959} operations for Tip4' respectively.
- Leveraging linear approximations of the split-and-lookup map to perform Gröbner basis attacks doesn't yield any advantage compared to the original construction, as the number of fixed points of the split-and-lookup map over one round is identical, i.e. 240 points, yielding a probability of $\left(\frac{240}{p}\right)^{(N-2)s} \approx 2^{-673}$ when excluding the first and last round.
- Finally, fixing input and output entries of the split-and-lookup map to perform Gröbner basis attacks is also non-sufficient, as it yields a random system of equation having solutions with probability p^{r-d-Ns}, being approximately $p^{-12} \approx 2^{-768}$ for Tip4 and $p^{-16} \approx 2^{-1024}$ for Tip4' respectively.

6 Conclusion

In this short note, we have highlighted two new instances derived from the new Tip5 hash function construction, designed for FRI-based proving systems over the prime field of characteristic $p = 2^{64} - 2^{32} + 1$ and targeting 128 bits security level. These two instantiations offer faster hashing, faster compression, and better performances within proving systems than the original construction, with one of them almost closing the performance gap between AO and traditional hash functions.

References

[AABS+19] Abdelrahaman Aly, Tomer Ashur, Eli Ben-Sasson, Siemen Dhooghe, and Alan Szepieniec. Design of symmetric-key primitives for advanced cryptographic protocols. Cryptology ePrint Archive, Paper 2019/426, 2019. https://eprint.iacr.org/2019/426.

[AGR+16] Martin Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. Mimc: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. Cryptology ePrint Archive, Paper 2016/492, 2016. https://eprint.iacr.org/2016/492.

[AKM+22] Tomer Ashur, Al Kindi, Willi Meier, Alan Szepieniec, and Bobbin Threadbare. Rescue-prime optimized. Cryptology ePrint Archive, Paper 2022/1577, 2022. https://eprint.iacr.org/2022/1577.

[BBC+22] 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. Cryptology ePrint Archive, Paper 2022/840, 2022. https://eprint.iacr.org/2022/840.

[BSBHR17] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast reed-solomon interactive oracle proofs of proximity. *Electron. Colloquium Comput. Complex.*, TR17, 2017.

[Fac23] Facebook. Winterfell. Repository (fork) https://github.com/Nashtare/winterfell/tree/tip4, January 2023.

[GDH+22] Théo Gauthier, Sébastien Dan, Monir Hadji, Antonella Del Pozzo, and Yackolley Amoussou-Guenou. Topos: A secure, trustless, and decentralized interoperability protocol, 2022.

[GKL+21] Lorenzo Grassi, Dmitry Khovratovich, Reinhard Lüftenegger, Christian Rechberger, Markus Schofnegger, and Roman Walch. Reinforced concrete: A fast hash function for verifiable computation. Cryptology ePrint Archive, Paper 2021/1038, 2021. https://eprint.iacr.org/2021/1038.

[GKR+19] Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger. Poseidon: A new hash function for zero-knowledge proof systems. Cryptology ePrint Archive, Paper 2019/458, 2019. https://eprint.iacr.org/2019/458.

[LPP+22] Jianwei Liu, Harshad Patil, Akhil Sai Peddireddy, Kevin Singh, Haifeng Sun, Huachuang Sun, and Weikeng Chen. An efficient verifiable state for zk-evm and beyond from the anemoi hash function. Cryptology ePrint Archive, Paper 2022/1487, 2022. https://eprint.iacr.org/2022/1487.

[Mid23] Polygon Miden. Miden VM. Repository https://github.com/maticnetwork/miden, January 2023.

[Por22] Thomas Pornin. Ecgfp5: a specialized elliptic curve. Cryptology ePrint Archive, Paper 2022/274, 2022. https://eprint.iacr.org/2022/274.

[SLST23] Alan Szepieniec, Alexander Lemmens, Jan Ferdinand Sauer, and Bobbin Threadbare. The tip5 hash function for recursive starks. Cryptology ePrint Archive, Paper 2023/107, 2023. https://eprint.iacr.org/2023/107.

[SSS22] Robin Salen, Vijaykumar Singh, and Vladimir Soukharev. Security analysis of elliptic curves over sextic extension of small prime fields. Cryptology ePrint Archive, Paper 2022/277, 2022. https://eprint.iacr.org/2022/277.

[SV21] Alan Szepieniec and Thorkil Værge. Neptune whitepaper. Whitepaper https://neptune.cash/whitepaper/, April 2021.

[Zer23] Polygon Zero. Plonky2. Repository https://github.com/mir-protocol/plonky2, January 2023.