SP1 V4 Turbo: Memory Argument via Elliptic Curve based Multiset Hashing

Gyumin Roh and Ron Rothblum, Succinct

December 2024

1 Introduction

SP1 Version 4 utilizes a new memory consistency argument based on ellpitic curve based multiset hash functions. We describe here the multiset hash function and the choice of elliptic curve.

The goal of a memory consistency argument in a zkVM, is to ensure that when data is read from a given memory address, the value seen there is the value that was last written to the same address. In SP1 we follow the off-line memory checking approach of Blum *et al.* [BEG+94], as proposed in [Set20].

The core of this argument is a multiset equality checking procedure. As the reduction from memory consistency to multiset equality is covered in detail in the standard textbook by Thaler [Tha22, Section 6], here we focus on the multiset equality test.

Organization. In Section 2 we describe Multiset Hashing and review the construction. In Section 3 we discuss our choice of elliptic curve.

2 Multiset Hashing

Multiset hashing, introduced by Clarke, Devadas, van Dijk, Gassend and Suh [CDvD+03] (building on Bellare and Micciancio [BM97]) allows one to hash a large (multi-)set into a short string so that it is computationally difficult to find two sets that hash to the same value. The hashing is performed in an incremental manner – one element at a time. A key property is that the hash value obtained is independent of the order in which elements were hashed. Thus, we can compare two sets that were generated in a different element order.

Multiset hash functions enable a very efficient multiset equality test. As compared to other approaches, a multiset hash allows for memory checking even in an IVC (interactive verifiable computation) scenario – namely, as the execution is run, one can simultaneously update the hash value without waiting for the computation to be finalized.[1]

[1] In contrast, other approaches use verifier randomness to check the memory. This randomness would typically be generated via an application of Fiat-Shamir, which necessitates waiting for the entire computation to be run before starting the memory consistency check. Another approach based on folding was recently proposed by [AS24].

2.1 Definition

A multiset hash[2] [CDvD+03] , for a universe $\mathcal{U}$, is a family of efficiently computable compressing functions $\mathcal{H} = \{h_k : \mathcal{P}(\mathcal{U}) \to \{0,1\}^\lambda\}_{k\in\{0,1\}^*}$, where λ denotes the security parameter, equipped with the following efficient algorithms:

1. Empty Set: given k, returns $h_k(\emptyset)$.
2. Increment: given k, $h(S)$, where $S \subseteq \mathcal{U}$, and $u \in \mathcal{U}$, returns $h(S \cup \{u\})$.

We require that finding multiset collisions be intractable. Formally, for every polynomial-time adversary A it holds that:

$$\Pr_{\substack{k \\ S,T\leftarrow A(k)}} \left[S \neq T \wedge h_k(S) = h_k(T)\right]$$

is negligible (in the length of k).

Remark 1 (On Unkeyed Functions)**.** *Similarly to definitions of collision resistant hash functions, the goal of the key in the above definition is to protect again non-uniform attacks in which the adversary has hard-coded collisions. Similarly to practical hash function constructions, below we sometimes consider keyless hash functions, which allow for security against uniform adversaries.*

2.2 Hash-to-Group Based Construction

Let $(G, +)$ be a cyclic group, written in additive notation. Let $\{f_k : \mathcal{U} \to G\}_k$ be a collection of functions. Consider the multiset hash function defined as:

$$h_k(S) = \sum_{s\in S} f_k(s).$$

Notice that the emptyset and increment functionalities can be easily implemented – the hash of the empty set is the group's identity element $\mathbf{0}_G$, and to add an element x to a multiset S, just compute $h(S) + h(x)$.

In [CDvD+03, Section 5] (building on [BM97]) it was shown that the construction above is multiset collision resistant, assuming that discrete log is hard in G, and $f_k : \mathcal{U} \to G$ is modeled as a random function.

2.3 An Elliptic Curve Based Construction

Following [MTA16] we consider an implementation of the hash-to-group based approach of Section 2.2, when the group G is the set of points (x, y) on an elliptic curve $y^2 = x^3 + Ax + B$ (together with the special point at infinity) relative to a large finite field $\mathbb{F}$ and where $A, B \in \mathbb{F}$. The hash function that we use is Poseidon2 [GKS23] over a BabyBear field. To hash an element into a elliptic curve point, we use Poseidon2 to get the x coordinate of the point. As this may not be a valid x coordinate of the elliptic curve, we add a 8-bit tweak, which will also be hashed in. This is explained to be secure in [Gro24]. Also, as explained in [Gro24], we constrain that the sign of the y coordinate can not be flipped, by using appropriate range checks.

[2] Our definition is somewhat simplified as compared to that in [CDvD+03]. In particular, while we do allow for a keyed function, we do not allow the hash function itself to be randomized, and so the equality tester considered in [CDvD+03] becomes redundant.

3 Elliptic Curve Parameterization

SP1's arithmetization uses the BabyBear field: $\mathbb{F}_p$ with $p = 15 \cdot 2^{27} + 1$. Given that, it is convenient to use an elliptic curve over the degree $d = 7$ extension of the aforementioned BabyBear field.

Selected Parameters. We choose the representation $\mathbb{F}_{p^7} = \mathbb{F}_p[z]/(z^7 - 2z - 5)$ (it can be confirmed that the above polynomial is irreducible over the base field), and the elliptic curve is

$$y^2 = x^3 + 2x + 26z^5.$$

This curve has prime order

$$r = 134062710381075636479997415343722979822569397998875702343109881667.$$

Note that if $A, B \in \mathbb{F}_p$, then the elliptic curve will have the curve over $\mathbb{F}_p$ as a subgroup, so the group order will not be a prime. We have selected A, B with simplest form for efficient computation.

Basic Tests. To ensure the security of the scheme, we run the basic checks described by Pornin [Por22]. We also run the basic checks described in [BL24].

We verify the requirements put forth in [Por22, Section 2] using exact numbers and further check the parameters described by Joux and Vitse [JV13, Theorem 1].

Targeted Security Level. Throughout, we target 100 bits of security. We remark that a higher security level such as 128 is out of reach since:

- The underlying STARK proof that we use a priori only has roughly 102 bits of security.
- We are using a curve of order 217 bits, so our security is at most 108.5 bits

3.1 Analysis

First of all, the extension degree $k = 7$ is clearly prime, as required. Roughly speaking, Pornin [Por22] describes the original attack due to Gaudry [Gau04] that has running time:

$$2^{2k(k-1)} \cdot p^{2-2/k},$$

where p is the size of the base field and k is the degree of extension. For example, for $p \approx 2^{64}$ and $k = 5$, the computational complexity is around

$$2^{2 \cdot 5 \cdot 4} \cdot 2^{64 \cdot (2-2/5)} \approx 2^{142}$$

which matches the numbers presented in [Por22].

For our parameterization, with $p \approx 2^{31}$ and $k = 7$, this turns out to be

$$2^{2 \cdot 7 \cdot 6} \cdot 2^{31 \cdot (2-2/7)} \approx 2^{137},$$

which is significantly higher than our desired security threshold.

Since Pornin's paper considers 64 bit fields, it has the luxury of skipping the details of Joux and Vitse's [JV13] variant of the attack, as it already has at least an overhead of $\mathcal{O}(p^2)$, which is too costly when $p \approx 2^{64}$.

In contrast, in our setting, we cannot skip this analysis. Their result [JV13, Theorem 1] yields an attack with complexity

$$(k-1)!\left(2^{(k-1)(k-2)}e^k k^{-1/2}\right)^{\omega} p^2,$$

where ω denotes the matrix multiplication exponent. Conservatively taking $\omega = 2$, this gives us roughly the following cost of their attack, which against suffices for our purposes

$$720 \cdot \left(2^{6.5} \cdot e^7 \cdot 7^{-1/2}\right)^2 2^{31.2} \approx 2^{148}.$$

Thus, direct attacks on the 217-bit elliptic curve work best in this regime of parameters.

3.2 Embedding Degree

If the elliptic curve has size r, it is important that the embedding degree e, the minimum positive integer such that $p^e \equiv 1 \pmod{r}$ is large, to avoid MOV-type attacks. We can check that this e is large by using standard order-checking algorithms in SageMath. For additional safety, we also check that this is true for our quadratic twist, which is described below.

3.3 Twist Security

While not crucial in our current setting (especially since we will use all x, y coordinates as our system is extremely sensitive to positive/negative y selections) in some situations regarding elliptic curve (to be exact, using a Montgomery-ladder) it is important that the quadratic twist of the elliptic curve also has good order for cryptographic purposes and so we check that our curve's quadratic twist also has prime size. The size of our curve's quadratic twist is

$$r' = 2p^7 + 2 - r = 13406271038107563647999741534372238595341189033680335057720635001 7$$

3.4 CM Discriminant

We have checked that the CM Discriminant of the curve is sufficiently large, at least 200 bits.

References

[AS24] Arasu Arun and Srinath T. V. Setty. Nebula: Efficient read-write memory and switchboard circuits for folding schemes. *IACR Cryptol. ePrint Arch.*, page 1605, 2024.

[BEG+94] Manuel Blum, William S. Evans, Peter Gemmell, Sampath Kannan, and Moni Naor. Checking the correctness of memories. *Algorithmica*, 12(2/3):225–244, 1994.

[BL24] Daniel J. Bernstein and Tanja Lange. Safecurves: choosing safe curves for elliptic-curve cryptography. https://safecurves.cr.yp.to/index.html, accessed December 2024.

[BM97] Mihir Bellare and Daniele Micciancio. A new paradigm for collision-free hashing: Incrementality at reduced cost. In Walter Fumy, editor, *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, volume 1233 of *Lecture Notes in Computer Science*, pages 163–192. Springer, 1997.

[CDvD+03] Dwaine E. Clarke, Srinivas Devadas, Marten van Dijk, Blaise Gassend, and G. Edward Suh. Incremental multiset hash functions and their application to memory integrity checking. In Chi-Sung Laih, editor, *Advances in Cryptology - ASIACRYPT 2003, 9th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, November 30 - December 4, 2003, Proceedings*, volume 2894 of *Lecture Notes in Computer Science*, pages 188–207. Springer, 2003.

[Gau04] Pierrick Gaudry. Index calculus for abelian varieties and the elliptic curve discrete logarithm problem. *IACR Cryptol. ePrint Arch.*, page 73, 2004.

[GKS23] Lorenzo Grassi, Dmitry Khovratovich, and Markus Schofnegger. Poseidon2: A faster version of the poseidon hash function. In Nadia El Mrabet, Luca De Feo, and Sylvain Duquesne, editors, *Progress in Cryptology - AFRICACRYPT 2023 - 14th International Conference on Cryptology in Africa, Sousse, Tunisia, July 19-21, 2023, Proceedings*, volume 14064 of *Lecture Notes in Computer Science*, pages 177–203. Springer, 2023.

[Gro24] Jens Groth. Memory checking in IVC-based zkVMs. ZKSummit 12, available at https://www.youtube.com/watch?v=kzSYNFh4uQ0, 2024.

[JV13] Antoine Joux and Vanessa Vitse. Elliptic curve discrete logarithm problem over small degree extension fields. *J. Cryptol.*, 26(1):119–143, 2013.

[MTA16] Jeremy Maitin-Shepard, Mehdi Tibouchi, and Diego F. Aranha. Elliptic curve multiset hash. *CoRR*, abs/1601.06502, 2016.

[Por22] Thomas Pornin. EcGFp5: a specialized elliptic curve. *IACR Cryptol. ePrint Arch.*, page 274, 2022.

[Set20] Srinath T. V. Setty. Spartan: Efficient and general-purpose zksnarks without trusted setup. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part III*, volume 12172 of *Lecture Notes in Computer Science*, pages 704–737. Springer, 2020.

[Tha22] Justin Thaler. Proofs, arguments, and zero-knowledge. *Found. Trends Priv. Secur.*, 4(2-4):117–660, 2022.