Privacy Preserving Measurement
Internet-Draft
Intended status: Standards Track
Expires: 7 May 2026

M. Thomson
Mozilla
D. Cook
ISRG

3 November 2025

A Prio Instantiation for Vector Sums with an L1 Norm Bound on Contributions
draft-ietf-ppm-l1-bound-sum-00

Abstract

A Prio Verifiable Distributed Aggregation Function is defined that supports vector or histogram addition, where the sum of the values in the contribution is less than a chosen value.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://martinthomson.github.io/prio-l1-bound-sum/draft-thomson-ppm-l1-bound-sum.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-ppm-l1-bound-sum/.

Discussion of this document takes place on the Privacy Preserving Measurement Working Group mailing list (mailto:ppm@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/ppm/. Subscribe at https://www.ietf.org/mailman/listinfo/ppm/.

Source for this draft and an issue tracker can be found at https://github.com/martinthomson/prio-l1-bound-sum.

Status of This Memo

This Internet-Draft will expire on 7 May 2026.

Table of Contents

1. Introduction

Existing Prio instantiations of a Verifiable Distributed Aggregation Function (VDAF) [VDAF] all support a simple summation of measurements. From Prio3Count (Section 7.4.1 of [VDAF]), which adds measurements containing a single one or a zero value, to Prio3SumVec (Section 7.4.3 of [VDAF]), which adds measurements containing a vector where each dimension is a limited number of bits, all instantations take the same basic form.

One case that is presently not included in the suite of instantiations is the addition of vectors or histogram contributions, where each measurement has an L1 bound. The L1 norm of a vector is defined as the sum of its components. An L1 bound limits that sum to some maximum.

This document defines the Prio3L1BoundSum instantiation. This instantiation limits the L1 norm of a vector or histogram to a value that is one less than a chosen power of 2, or 2^n-1. This choice significantly reduces the size of the encoding relative to a more flexible limit.

This instantiation has similarities with other instantiations. Unlike Prio3Histogram (Section 7.4.4 of [VDAF]), in which measurements need to have an L1 norm of exactly 1, a valid measurement for Prio3L1BoundSum can have an L1 norm equal to any value between 0 and the chosen limit. Unlike Prio3MultiHotCountVec (Section 7.4.5 of [VDAF]), in which each component can only be zero or one, components in Prio3L1BoundSum can take any value up to the L1 bound as long as their sum is within that bound.

Section 3 defines the Prio3L1BoundSum VDAF.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document uses the terminology and functions defined in Section 2 of [VDAF].

3. Prio3L1BoundSum Definition

The Prio3L1BoundSum instantiation of Prio [CGB17] supports the addition of a vector of integers.

The instantiation is summarized in Table 1.

Parameter	Value
field	Field128 (Section 6.1.2 of [VDAF])
Valid	L1BoundSum(field, length, bits, chunk_length)
PROOFS	1
XOF	XofTurboShake128 (Section 6.2.1 of [VDAF])

Table 1: Prio3L1BoundSum Parameters

The function takes three parameters: length, bits, and chunk_length. The vector contains "length" components, each of which is a non-negative integer less than 2^bits.

3.1. Chunk Length Selection

The chunk_length parameter can be chosen in approximately the same way as for Prio3SumVec, as detailed in Section 7.4.3.1 of [VDAF]. The difference is that Prio3L1BoundSum involves validation of bits * (length + 1) values, which might increase the most efficient value for chunk_length.

3.2. Encoding and Decoding

The encoded form of each measurement appends a bitwise decomposition of the L1 norm (the sum of the vector components) to the encoding:

```
def encode(self, measurement: list[int]) -> list[F]:
    encoded = []
    weight = self.field(0)
    for v in measurement:
        weight += v
        encoded += self.field.encode_into_bit_vec(v, self.bits)
    w_bits = self.field.encode_into_bit_vec(weight, self.bits)
    return encoded + w_bits
```

The encoded measurement has a total length of (length + 1) * bits.

This extra information is not included in the output share that is submitted for aggregation. That is, the truncate() function emits only the core measurements.

```
def truncate(self, meas: list[F]) -> list[F]:
    return [
       self.field.decode_from_bit_vec(m)
       for m in chunks(meas, self.bits)
    ]
```

This uses a chunks(v, c) function that takes a list of values, v, and a chunk length, c, to split v into multiple lists from v, where each chunk has a length c.

The decode() function is therefore identical to that in Prio3SumVec.

```
def decode(self, output: list[F], _count) -> list[int]:
    return [x.int() for x in output]
```

3.3. Validity Circuit

The validity circuit for Prio3L1BoundSum uses an extended version of the validity circuit used by Prio3SumVec, see Section 7.4.3 of [VDAF].

The encoded measurement is checked to ensure that every component of the vector plus the added L1 norm is encoded in the specified number of bits. That is, the circuit checks that each component has a value between 0 (inclusive) and 2^bits (exclusive) by checking that each of the first "bits" bits of the value are either zero or one. This process is identical to the Prio3SumVec check, except that one additional value is checked.

The validity circuit then checks whether the added L1 norm value is consistent with the encoded vector elements. The L1 norm is checked by decoding the measurement values, including the encoded L1 norm, recomputing the L1 norm as the sum of the individual components, and subtracting the reported and computed values to confirm that they are identical.

The complete circuit is specified in Figure 1.

```
   def eval(self, meas: list[F],
            joint_rand: list[F], num_shares: int) -> list[F]:
        assert len(meas) == (self.length + 1) * self.bits
        shares_inv = self.field(num_shares).inv()
        parallel_sum = ParallelSum(Mul(), chunk_length)

        num_chunks = ceil(len(meas) / self.chunk_length)
        pad_len = self.chunk_length * num_chunks - len(meas)
        meas += [self.field(0)] * pad_len

        range_check = self.field(0)
        for (r, m) in zip(joint_rand, chunks(meas, self.chunk_length)):
            inputs = []
            for i in range(self.chunk_length):
                inputs += [
                    r**(i + 1) * m[i],
                    m[i] - shares_inv,
                ]
            range_check += parallel_sum.eval(self.field, inputs)

        components = [
            self.field.decode_from_bit_vec(m)
            for m in chunks(meas, self.bits)
        ]
        observed_weight = sum(components[:self.length])
        claimed_weight = components[self.length]
        weight_check = observed_weight - claimed_weight

        return [range_check, weight_check]
```

Figure 1: Evaluation function for Prio3L1BoundSum

4. Security Considerations

The Prio3L1BoundSum VDAF is subject to the same considerations as other Prio-based VDAFs. These considerations are detailed in Section 9 of [VDAF].

In particular, this instantiation uses Field128 to ensure robustness despite the use of joint randomness in proofs. Joint randomness increases the risk of an attacker finding a combination of invalid inputs that passes validation. A larger field increases the computational cost of finding such a combination.

5. IANA Considerations

This document registers a codepoint for Prio3L1BoundSum in the "Verifiable Distributed Aggregation Functions (VDAF)" registry as defined by Section 10 of [VDAF]. This entry contains the following fields:

Value: 0x00000007
Scheme: Prio3L1BoundSum
Type: VDAF
Reference: RFCXXXX (this document)

6. References

6.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[VDAF] Barnes, R., Cook, D., Patton, C., and P. Schoppmann, "Verifiable Distributed Aggregation Functions", Work in Progress, Internet-Draft, draft-irtf-cfrg-vdaf-15, 17 June 2025, <https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-vdaf-15>.

6.2. Informative References

[CGB17] Boneh, D. and H. Corrigan-Gibbs, "Prio: Private, Robust, and Scalable Computation of Aggregate Statistics", USENIX Symposium on Networked Systems Design and Implementation (NSDI), 2017, <https://dl.acm.org/doi/10.5555/3154630.3154652>.

Acknowledgments

Chris Patton provided extensive input into the construction of this VDAF.

Authors' Addresses

Martin Thomson
Mozilla

Email: mt@lowentropy.net

David Cook
ISRG
Email: divergentdave@gmail.com