Yuwen Zhang, with Mayank Rathee, Raluca Ada Popa, Henry Corrigan Gibbs.
CS 262A

December 5th, 2023

# Whisper: Private Analytics via Streaming, Sketching, and Silently Verifiable Proofs

## Background

In private aggregation, an analyst wants to compute some aggregate statistic, such as mean, standard deviation, or sum, over many client inputs. If the analyst wants to compute heavy hitters – the most frequently occurring client inputs – special techniques are required. To preserve client privacy, clients secret share their inputs between two servers, and provide hiding proofs of their well formedness that the servers need to check.

## Problem

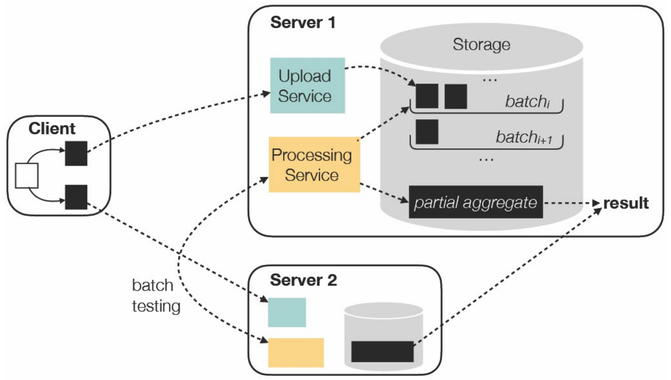
1. Existing solutions incur large server-server communication to verify these well formedness proofs.

   We desire a way to efficiently compress multiple proofs into a single succinct batch proof.

2. Existing solutions for private heavy hitters additionally require all client submissions to be present before starting proof verification and aggregation.

   We desire a space efficient, streaming friendly way to compute heavy hitters.

## System design

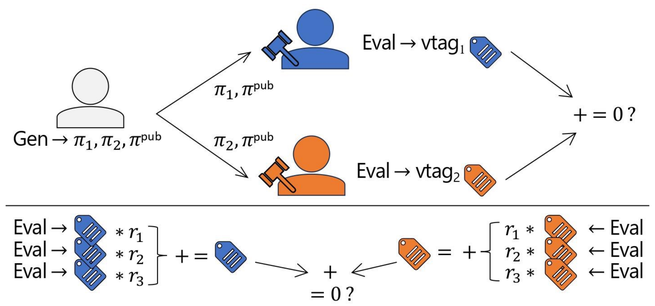



Clients invoke the upload service, where they upload their secret shared proofs and measurements into cloud storage buckets. The processing service verifies the well formedness of each batch, and outputs an aggregate.

At midnight in every global time zone, clients submit their inputs to each server's upload service.

To ensure that each honest client input ends up in the same batch, clients tag their submissions with a timestamp. Each server has a fixed window, where it groups client submissions based on these tagged timestamps.

## Silently Verifiable Proofs

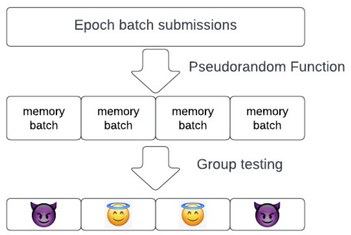



Clients generate two proof shares $\boldsymbol{\pi_1}$ and $\boldsymbol{\pi_2}$, for each individual server, as well as a public share $\boldsymbol{\pi_{pub}}$ that verifiers will use generate succinct **vtags**. vtags are valid if they sum to 0.

Many of these 0 checks can be batched together using shared randomness between the servers. Each batch will verify *only if* every tag in it is well formed, and will fail otherwise.

Note: the general idea behind these proofs is broadly applicable to many zero knowledge proofs on secret shared data.

## Detecting malicious clients




Each epoch is broken down into chunks that fit into server memory.

We use a PRF to enforce evenly distributed batches, and to protect against malicious clients

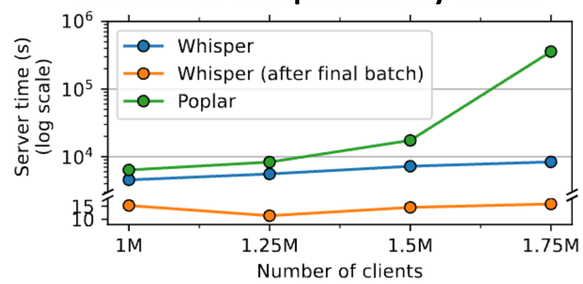
To detect failing proofs in each batch, we use a modified version of binary search.

## Sketching for heavy hitters

Instead of computing exact heavy hitters, we use sketching to compute approximate heavy hitters. Our approach is very similar to Count Sketch: each string is hashed to a certain bucket, and some local aggregation occurs within that bucket. Failure probability increases as a heavy hitter's bucket gets more and more non-heavy-hitter collisions.

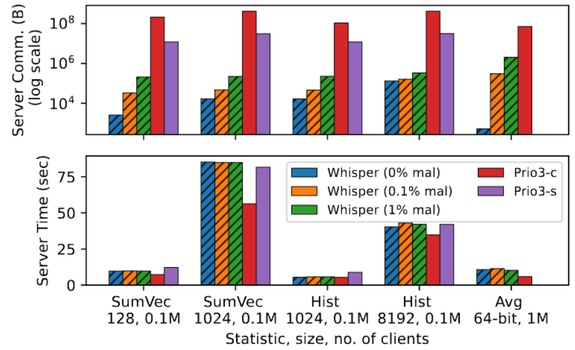
## Evaluation

### Runtime to Compute Heavy Hitters




Poplar, the current state of the art for computing heavy hitters in our trust setting, is not streaming friendly, and sees drastic slowdowns as the number of clients stops fitting in main memory. Our streaming friendly approach doesn't have this issue.

### Communication and Runtime for Common Statistics




These are some commonly deployed statistics – SumVec takes the vector sum of client measurements, ensuring that the $\ell_\infty$ norm of each measurement is less than some fixed value. Servers take slightly longer to generate vtags, but save orders of magnitude communication.

## Future Work

- End to end cost comparison to prior work, including storage and networking costs
- Other ways to validate sketches
- Applications of Silently Verifiable Proofs to more Zero Knowledge Proofs.