
# USENIX Security '26 Artifact Appendix: "Heli: Heavy-Light Private Aggregation"

Ryan Lehmkuhl
MIT

Henry Corrigan-Gibbs
MIT

Emma Dauterman
Stanford

David J. Wu
UT Austin

## A Artifact Appendix

### A.1 Abstract

This artifact implements Heli, a system that lets a pair of servers collect aggregate statistics about private client-held data with-out learning anything more about any individual client's data. Like prior systems, Heli protects client privacy against a malicious server, protects correctness against misbehaving clients, and supports common statistical functions: average, variance, and more. Heli's innovation is that only one of the servers (the "heavy server") needs to do per-run work proportional to the number of clients; the other server (the "light server") does work sublinear in the number of clients, after a one-time setup phase. As a result, a computationally limited party, such as a low-budget non-profit, could potentially serve as the second server for a Heli deployment with millions of clients.

Heli relies on a new cryptographic primitive, aggregation-only encryption, that allows computing certain restricted functions on many clients' encrypted data. In a deployment with ten million clients, in which the servers privately compute the sum of 32 client-held 1-bit integers, Heli's heavy server does 240,000 core-s of work and the light server does 7 core-ms of work.

### A.2 Description & Requirements

All experiments were run on AWS `c7i.*` instances with Ubuntu 22.04 with at least 32 GB of disk storage.

Microbenchmarks were run on a single `c7i.4xlarge` instance (16 vCPUs, 32 GB RAM). They are single-threaded and use $<$ 8 GB of RAM, so can also be run on a smaller machine.

We provide two configurations for reproducing the end-to-end results:

- *Simplified configuration:* Aggregation over 100,000 clients using two `c7i.4xlarge` instances and a `c7i.large` instance. This method takes approximately 20 minutes.
- *Full configuration:* Aggregation over 10,000,000 clients using a `c7i.metal-24xl` instance, one `c7i.4xlarge` instance, and a `c7i.large` instance. This method takes approximately 2 hours.

#### A.2.1 Security, privacy, and ethical concerns

N/A.

#### A.2.2 How to access

Our artifact can be accessed via:

- Zenodo at: https://doi.org/10.5281/zenodo.17980903
- Github at: github.com/ryanleh/heli

#### A.2.3 Hardware dependencies

The artifact requires support for AES-NI CPU intrinsics and at least 8 GB of disk storage.

#### A.2.4 Software dependencies

Please see the artifact's README.md.

#### A.2.5 Benchmarks

N/A.

### A.3 Set-up

The aggregator and decryptor machines used in the end-to-end evaluation should accept TCP traffic on ports 9000 and 9001.

#### A.3.1 Installation

Please see the artifact's README.md.

#### A.3.2 Basic Test

Please see the artifact's README.md.

### A.4 Evaluation workflow

#### A.4.1 Major Claims

**(C1):** Heli's computational cost, for the light server, and total server-to-server communication depends only on the number of offline clients in each round. This is proven

by the experiment (E1) described in Section 6.1 and whose results are displayed in Figure 4.

**(C2):** Heli's computational cost for the heavy server and clients grows linearly with the number of measurements. This is proven by the experiment (E1) described in Section 6.2 and whose results are displayed in Figure 5.

**(C3):** When privately aggregating 32 1-bit integers, the AWS cost of operating a light server is five orders of magnitude smaller than operating a heavy server. This is proven by the experiment (E2 or E3) described in Section 6.3 and whose results are displayed in Table 2.

**Note (updated numbers).** After some updates to the implementation, the server numbers reported in Table 2 of the Heli paper have changed slightly. Below is the revised numbers that were confirmed during artifact evaluation:

| Number of Measurements | | Server aggregation costs: Wall Time (s) | Server aggregation costs: Egress (KB) |
|---|---|---|---|
| $\ell = 1$ | *Heavy:* | 43.9 | 3000 |
| | *Light:* | 0.007 | 0.048 |
| $\ell = 32$ | *Heavy:* | 873 | 3000 |
| | *Light:* | 0.008 | 1 |
| $\ell = 128$ | *Heavy:* | 3232 | 3000 |
| | *Light:* | 0.012 | 4 |

These changes are reflected in the full version of the paper (https://eprint.iacr.org/2026/059.pdf).

### A.4.2 Experiments

**(E1):** *[Heli Microbenchmarks] [10 human-minutes + 20 compute-minutes]*: This experiment runs the Heli microbenchmarks used in Sections 6.1 and 6.2.
**How to:** See the artifact's README.md.
**Results:** The outputted plots should visually match Figures 4 and 5.

**(E2):** *[Heli Simplified End-to-end Evaluation] [10 human-minutes + 10 compute-minutes]*: This experiment runs a simplified end-to-end Heli workflow that can approximate the server metrics in Table 2. It can be used in place of experiment E3.
**How to:** See the artifact's README.md for how to run the end-to-end experiments using the "simplified-{1,32,128}.json" config files. The aggregator and client should be run on `c7i.4xlarge` instances or equivalent (in separate regions), and the decryptor should be run on a `c7i.large` instance (in a separate region from the aggregator).
**Results:** After getting numbers, plug them into the corresponding variables at the top of `benches/estimate_e2e.py` (including details about your machine setup), the result should match Table 2.

**(E3):** *[Heli End-to-end Evaluation] [10 human-minutes + 2 compute-hours]* This experiment runs the end-to-end Heli workflow to validate the server metrics in Table 2. The aggregator should be run on an `c7i.24xlarge` instance, the decryptor should be run on a `c7i.large` instance (in a separate region from the aggregator), and the client should be run on a `c7i.4xlarge` instance.
**How to:** See the artifact's README.md for how to run the end-to-end experiments using the "full-{1,32,128}.json" config files.
**Results:** The results should match the numbers in Table 2.

## A.5 Notes on Reusability

N/A.

## A.6 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2026/.