USENIX Security '25 Artifact Appendix: Distributional Private Information Retrieval

Ryan Lehmkuhl
MIT

Alexandra Henzinger
MIT

Henry Corrigan-Gibbs
MIT

A Artifact Appendix

A.1 Abstract

A private-information-retrieval (PIR) scheme lets a client fetch a record from a remote database without revealing which record it fetched. Classic PIR schemes treat all database records the same but, in practice, some database records are much more popular (i.e., commonly fetched) than others. We introduce distributional PIR, a new type of PIR that can run faster than classic PIR—both asymptotically and concretely—when the popularity distribution is heavily skewed. Distributional PIR provides exactly the same cryptographic privacy as classic PIR. The speedup comes from a relaxed form of correctness: distributional PIR guarantees that in-distribution queries succeed with good probability, while out-of-distribution queries succeed with lower probability.

We construct a distributional-PIR scheme that makes black-box use of classic PIR protocols. On a popularity distribution built from real-world data, we show that distributional PIR reduces compute costs by 5–77× compared to existing techniques. Additionally, we present new optimizations to the state-of-the-art classical PIR protocol that decreases preprocessing and query generation time by by 128–351×. Finally, we build CrowdSurf, an end-to-end system for privately fetching tweets, and show that distributional-PIR reduces the end-to-end server cost by 8×.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

N/A

A.2.2 How to access

Our artifact is available at:

- `https://github.com/ryanleh/crowdsurf`, and
- `https://zenodo.org/records/14642111`.

For the sake of the artifact evaluation, please use the GitHub link as we'll be pushing any updates, bugfixes, etc. there.

A.2.3 Hardware dependencies

To replicate the exact numbers in the paper, you will need to access to the following AWS machines:

- a `c7i.2xlarge` instances,
- a `c7i.4xlarge` instance,
- a `r7i.4xlarge` instance, and
- a `p3.2xlarge` instance.

If you are unable to access these instances on your own, we are happy to coordinate running these machines for you, just let us know on HotCRP.

Note that running our code does not require access to these exact machines: in each experiment we describe the minimal hardware constraints necessary if you want to try and run on different machines.

A.2.4 Software dependencies

Please see the artifact's README.md.

A.2.5 Benchmarks

None.

(Two of our main results rely on a real-world popularity distribution, however, we don't have permission to publicly release these. Instead, in the artifact we have hardcoded the relevant parameters of our scheme when run on these distributions.)

A.3 Set-up

All machines used should accept TCP traffic on ports 8728 and 8729.

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): Our PIR optimizations reduce SimplePIR's preprocessing time by 116–351× and query generation time by 128–349×. This is proven by the experiment (E1) described in Section 7.1 and whose results are displayed in Figures 3 and 4.

(C2): On the real-world Twitter popularity distribution, our distributional PIR scheme reduces computational costs by 5–77× and communication costs by 8.1–95× compared to batch codes when instantiated with SimplePIR. This is proven by the experiment (E2) described in Section 7.2.1 whose results are displayed in Figure 9.

(C3): On the real-world Twitter popularity distribution, our system, CrowdSurf, allows users to privately fetch tweets for 8× less than existing techniques. This is proven by experiment (E3) described in Section 9.1 and whose results are displayed in Table 12.

A.4.2 Experiments

(E1): *LHE Microbenchmarks [10 human-minutes + 30 compute-minutes]*: This experiment should be run on a `c7i.4xlarge` AWS instance, or any Intel-based machine that supports AVX512 intrinsics with ∼ 32GB of memory.
How to: See the artifact's README.md.
Results: The reported improvement for our scheme should match the claims in C1.

(E2): *Distributional PIR Microbenchmarks [10 human-minutes + 3 compute-hours]*: This experiment should be run on a `r7i.4xlarge` AWS instance, or any instance with ∼ 128GB of memory. *(Note that one can significantly reduce the running time of this experiment by omitting the numbers for the Cuckoo-based batch code as outlined in the artifact README.)*
How to: See the artifact's README.md.
Results: The outputted numbers should visually match Figure 9.

(E3): *CrowdSurf Evaluation [1 human-hour + 1 compute-hour]*: This experiment benchmarks the costs of running a single shard of the full CrowdSurf system–the total end-to-end costs are interpolated from these costs. Recreating the numbers from the paper requires two `c7i.2xlarge` instance, and a `p3.2xlarge` instance. If desired, one could accurately estimate these results without the GPU machine (described in the abstract's README.md).
How to: See the artifact's README.md.
Preparation: All three machines will need to be launched simultaneously, allow TCP traffic on ports 8728 and 8729, and have the correct IP addresses set in the config file described in the abstract's README.md.
Results: The produced numbers should match the numbers from Table 12.

A.5 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/usenixsec2025/.