

USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

ZipPIR: High-throughput Single-server PIR without Client-side Storage

Rasoul Akhavan Mahdavi, Abdulrahman Diaa, and Florian Kerschbaum, *University of Waterloo*

https://www.usenix.org/conference/usenixsecurity26/presentation/mahdavi

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 35th USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 35th USENIX Security Symposium.

August 12–14, 2026 • Baltimore, MD, USA

ISBN 978-1-939133-58-8

Open access to the Artifact Appendices to the Proceedings of the 35th USENIX Security Symposium is sponsored by



جامعة الملك عبدالله للعلوم والتقنية

King Abdullah University of Science and Technology



USENIX Security '26 Artifact Appendix: "ZipPIR: High-throughput Single-server PIR without Client-side Storage"

Rasoul Akhavan Mahdavi Abdulrahman Diaa Florian Kerschbaum
University of Waterloo
{rasoul.akhavan.mahdavi, abdulrahman.diaa, florian.kerschbaum}@uwaterloo.ca

A Artifact Appendix

A.1 Abstract

Private Information Retrieval (PIR) allows a client to privately access a database without revealing which element is accessed. The limitations of existing PIR protocols conflict with the practical constraints of resource-limited clients and dynamic databases.

To address these challenges, we propose ZipPIR, a high-throughput PIR protocol that compresses LWE ciphertexts into significantly smaller Paillier ciphertexts. ZipPIR leverages the offline phase to obtain this size reduction without incurring the associated computational cost in the online phase. Moreover, under computational assumptions, ZipPIR features an almost silent offline phase, requiring no communication beyond an initial public key, enabling the server to independently generate and update hints during idle times without client interaction. ZipPIR achieves over 2 GB/s of throughput — comparable to state-of-the-art protocols such as SimplePIR — without the need for a large client-stored hint. For PIR over a 1 GB database, ZipPIR has up to 10x higher throughput than existing protocols with no client-side storage, while requiring less than 200 KB of server-side storage per client, significantly enhancing scalability for practical deployments. While prior PIR protocols using Paillier are very inefficient, ZipPIR is the first PIR protocol using Paillier that achieves throughput that is competitive with state-of-the-art PIR protocols.

Our artifact provides the prototype implementation of ZipPIR, used to compare with related work. It also contains all necessary scripts for evaluating related work and comparing the results with ZipPIR.

A.2 Description & Requirements

Our artifact requires a server running Ubuntu 24.04 or higher. The only hardware requirement for this artifact is that the server have support for AVX512. We recommend using a hardware such as an Amazon c5.4xlarge, which has the required support.

A.2.1 Security, privacy, and ethical concerns

Our artifact does not pose any risk for the evaluators.

A.2.2 How to access

Link to artifact on Gitub (recommended for functionality/reproducibility evaluation):

https://github.com/RasoulAM/ZipPIR/tree/a2ffee01accd20c51dbbc69fd2bf9de12f79c0b5

Permanent link on Zenodo:

https://doi.org/10.5281/zenodo.17907224

A.2.3 Hardware dependencies

The only hardware requirement for this artifact is that the server have support for AVX512. We recommend using a hardware such as an Amazon c5.4xlarge, which has the required support. Our evaluation does not rely on multithreading for the results, but non-critical sections of the code are parallelized to save time. We encourage the evaluators to use a machine with many cores to save time.

A.2.4 Software dependencies

Our artifact requires a server running Ubuntu 24.04 or higher. Other software dependencies can be installed using the provided setup scripts.

A.2.5 Benchmarks

None

A.3 Set-up

A.3.1 Installation

A script is provided that performs the entire setup process. This is mentioned in the README of the artifact and the script if called 'setup.sh'. The specific required dependencies are also listed in the README of the artifact.

A.3.2 Basic Test

The script 'run_test.sh' runs the protocol for test parameters. This test should write results to 'test_results.json'.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): *ZipPIR is faster than all related work listed in Table 4.*
(C2): *ZipPIR is faster than all relaed work listed in Table 5.*

A.4.2 Experiments

The entire process for reproducing the experimental evaluation is described in the 'benchmarks' directory of the artifact.

(E1): *[Table 4] [Several hours for partial results, over 20 hours of computation for full results]: This experiments compares ZipPIR with other state-of-the-art PIR protocols as a protocol without client-side storage.*
Preparation: *The steps required to install all dependencies and build the artifact are described in the main README of the artifact. (run 'setup.sh' in the root directory of the artifact)*
Execution: *The steps required to run the experiment is described in the README in the benchmarks directory of the artifact. (run 'run-table-4.sh' and 'run-exp.sh' in the 'benchmarks' directory of the artifact)*
Results: *The steps required to interpret the results are described in the README in the benchmarks directory of the artifact. (run 'python3 generate_table4.py' in the 'benchmarks' directory of the artifact). This script prints the results in the 'benchmarks/results' directory in the form of a csv file.*

(E2): *[Table 5] [Less than 30 minutes]: This experiments compares ZipPIR with other state-of-the-art PIR protocols that fall within the category of stateful PIR.*
Preparation: *The steps required to install all dependencies and build the artifact are described in the main README of the artifact. (run 'setup.sh' in the root directory of the artifact).*
Execution: *The steps required to run the experiment is described in the README in the benchmarks directory of the artifact. (run 'run-table-5.sh' in the 'benchmarks' directory of the artifact). This script prints the results in the 'benchmarks/results' directory in the form of a csv file. This only runs ZipPIR, and not related work, since there is no source code available for related work. For that, we refer the reader to the paper.*
Results: *The steps required to interpret the results are described in the README in the benchmarks directory of the artifact. (run 'python3 generate_table5.py' in the 'benchmarks' directory of the artifact). This script prints the results in the 'benchmarks/results' directory in the form of a csv file. This only runs ZipPIR, and not related work, since there is no source code available for related work. For that, we refer the reader to the paper.*

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/usenixsec2026/.