Constant-weight PIR: Single-round Keyword PIR via Constant-weight Equality Operators

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

https://www.usenix.org/conference/usenixsecurity22/presentation/mahdavi

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 31st USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 31st USENIX Security Symposium.

August 10–12, 2022 • Boston, MA, USA

978-1-939133-31-1

A Artifact Appendix

A.1 Abstract

Our artifact contains the implementation of constant-weight PIR as proposed in the paper titled “Constant-weight PIR: Single-round Keyword PIR via Constant-weight Equality Operators".

We use this implementation to compare constant-weight PIR with other PIR protocols. We provide an implementation of folklore PIR for comparison. We provide scripts that use this implementation to generate the tables shown in the paper.

Aside from PIR, we provide scripts that benchmark the proposed equality operators and compare them with existing ones in terms of runtime.

A.2 Artifact check-list (meta-information)

- **Compilation:** The GNU GCC compiler (version >= 6.0) is required which supports OpenMP for parallelization. This compiler is publicly available.
- **Binary:** Binaries are not included but can be easily build using the steps outlined in the README. Two executables should be generated: `main` to experiment with PIR and `benchmark_eq` to benchmark the proposed equality operators.
- **Run-time environment:** Our code has been tested for Ubuntu 20.04. Besides the specified compiler, it requires the Microsoft SEAL library [1] to be installed.
- **Hardware:** Some runtimes in the paper are parallelized over 64 and 114 threads. To achieve the same results, it is required to have hardware with similar specs. The precise specs of the hardware are noted in the paper in each section.
- **Metrics:** In our PIR implementation, we measure the runtime of each step and the total runtime as well. We also measure the upload and download communication. In the benchmarks of equality operators, we measure the runtime.
- **Output:** The specified metrics are written to file (the name of the file is generated randomly) in the directory specified in the command line.
- **Experiments:** Scripts are provided to reproduce the results in the paper. These scripts run experiments and write the results to the ‘results’ directory.
- **How much disk space required (approximately)?:** All experiments are done in memory so not much disk space is required. However, the largest experiments use more than 100 GB of memory.
- **How much time is needed to prepare workflow (approximately)?:** Assuming all the prerequisites need to be installed, the installation time should not take more than 30 mins.
- **How much time is needed to complete experiments (approximately)?:** To reproduce all the results, the experiments take over a two weeks to run on a single machine. However, the number of runs can be reduced in all the provided scripts to reduce this to a couple days. Currently, the experiments are run 10 times.
- **Publicly available:** Our artifact is publicly available on Github at `https://github.com/RasoulAM/constant-weight-pir`
- **Code licenses:** The code is published under a BSD-3 license.
- **Archived (explicitly provide DOI or stable reference):** `https://github.com/RasoulAM/constant-weight-pir/releases/tag/artifact-accepted`

[1]https://github.com/microsoft/SEAL

A.3 Description

A.3.1 How to access

The artifact is publicly available on Github at `https://github.com/RasoulAM/constant-weight-pir/releases/tag/artifact-accepted`.

A.3.2 Hardware dependencies

Some runtimes in the paper are parallelized over 64 and 114 threads. To achieve the same results, it is required to have hardware with similar specs. The precise specs of the hardware are noted in the paper in each section.

A.3.3 Software dependencies

This artifact runs on Ubuntu 20.04. It requires GNU GCC compiler (version >= 6.0) as the C++ compiler and the Microsoft SEAL library [2] to be installed. Instructions to install SEAL can be found in their repository.

A.3.4 Data sets

N/A

A.3.5 Models

N/A

A.3.6 Security, privacy, and ethical concerns

N/A

A.4 Installation

The instructions to build the repository are provided in the main README. The prerequisites such as the gcc compiler are specified in the README. Instructions on how to install the other dependencies such as SEAL and googletest are also specified.

[2]https://github.com/microsoft/SEAL

A.5 Experiment workflow

We provide scripts to run the experiments outlined in the paper. These scripts are provided in the `src/build/scripts` directory. Details regarding these scripts and instructions on how to run them are given in `src/build/README.md`

To interpret the result of the experiments, we provide scripts in `src/build/interpret-results.ipynb`

A.6 Evaluation and expected results

We use this artifact to generate the tables shown in the paper, specifically Table 4, 5, 7 and 9 (and Table 12 and 13 in the appendix) . Instructions on which script to use to generate each table is given in `src/build/README.md`

A.7 Experiment customization

We provide a command line interface to experiment with different PIR protocols. Particularly, the user can experiment with folklore PIR and constant-weight PIR. All parameters can be assigned via the command-line. Instructions on what these parameters are and how to set them are given in the README. All results can be written to file and printed to the standard output.

Our scripts are not customizable (except for the number of runs) and are only used to automatically produce the results shown in the paper.

A.8 Version

Based on the LaTeX template for Artifact Evaluation V20220119.