USENIX Security '25 Artifact Appendix: H_2O_2RAM: A High-Performance Hierarchical Doubly Oblivious RAM

Leqian Zheng
City University of Hong Kong

Zheng Zhang
ByteDance Inc.

Wentao Dong
City University of Hong Kong

Yao Zhang
ByteDance Inc.

Ye Wu
ByteDance Inc.

Cong Wang
City University of Hong Kong

A Artifact Appendix

A.1 Abstract

This artifact contains the source code and scripts needed to reproduce our results. The Functional badge only requires an Ubuntu 20.04 environment with some easily installable packages. While reproducing the results presented in our paper, an Intel Xeon(R) Platinum 8457C processor and 2TB of RAM are necessary. If the specific CPU is unavailable, the results may exhibit minor deviations from the reported data. If memory is insufficient, a partial reproduction of our work for small datasets remains feasible. In addition, completing the full set of experiments requires several weeks, as the reported results are amortized over a substantial number of data accesses to more accurately assess the overhead of hierarchical ORAM.

A.2 Description & Requirements

Note that one can run H_2O_2RAM on a "non-confidential" machine to estimate the approximate performance overhead, as our design focuses on concealing memory access patterns rather than memory confidentiality and integrity (which are handled by the confidential virtual machine).

Software Requirements: Host operating system: Debian GNU/Linux 10 with the kernel version 5.15.120+. Confidential operating system: Ubuntu 20.04 with the same kernel version. Virtualization components: Qemu, OVMF, and Kata. Required tools: CMake (version $\geq$ 3.16), g++ (support C++-20), libboost, libtbb-dev, libnlopt-dev, libgtest-dev (for tests), Google Benchmark (for benchmarks), Python3.

Hardware Requirements: An Intel Xeon(R) Platinum 8457C processor (96-core) and 2TB of RAM. Confidential virtual machine needs 64 cores and 1TB of memory.

A.2.1 Security, privacy, and ethical concerns

This artifact will not lead to any security and privacy issues. One potential ethical concern is that running full experiment sets may take several weeks and result in (possibly notable) energy consumption.

A.2.2 How to access

The artifact is available on GitHub: https://github.com/55199789/H2O2RAM/commits/66d48b9 and Zenodo https://zenodo.org/records/14648338.

A.2.3 Hardware dependencies

We use and recommend a physical server powered by dual 48-core Intel Xeon(R) Platinum 8457C processors and 2TB RAM. Intel TDX, the specific TEE we use, is widely available with 5th Gen Intel Xeon Scalable processors. Alternatively, one can use other hardware-assisted CVMs, such as AMD SEV requiring AMD processors. Note that a "nontrusted" hardware (i.e., common CPUs) could also simulate our experiments as Intel TDX introduces only minor performance overheads for memory-intensive tasks (please refer to the work). Due to corporation policy, we do not support SSH access to the machine running our experiments.

A.2.4 Software dependencies

Setting up the TDX requires physical access to the machine (e.g., configuring the BIOS). We recommend following the instructions provided in the Intel TDX Module[1] and Intel TDX[2] for properly creating a TD image. In specific, we offer a guest image for CVM that includes the full experimental setup[3]. This image is derived from Ubuntu 20.04 and is preconfigured with TD guest, CMake, Google Benchmark, Boost, OpenMP, Intel TBB, and Google Test. Inside the image, use Python3 to run our experiment scripts.

A.2.5 Benchmarks

None.

[1] https://github.com/intel/tdx-module
[2] https://github.com/canonical/tdx
[3] https://zenodo.org/records/15000727/files/demo.zip

A.3 Set-up

A.3.1 Installation

Setting up the TDX is somewhat challenging. In case of any failure, you may skip related steps and continue directly on the Ubuntu 20.04 system.

Host Environment. Please follow the instructions in Intel TDX Module and Intel TDX to setup the environment. We also provide a sample scripts to enable Intel TDX in Host OS:

```
$ git clone https://github.com/canonical
    /tdx.git
$ cd tdx && sudo ./setup-tdx-host.sh
$ sudo reboot
```

Guest Image. Build `demo.raw` following this guide or download the pre-configured version. Then, run `./start.sh`. To boot in TD mode, uncomment lines 13∼15 of `start.sh`.

Software Dependencies. Log in to the TD VM and install all required packages:

```
$ git clone https://github.com/55199789/
    H2O2RAM.git
$ cd H2O2RAM && sudo ./setup.sh
```

To compile H_2O_2RAM:

```
$ mkdir build && cd build
$ cmake .. && make -j
```

A.3.2 Basic Test

Host Environment. Run `./25.tdx_host_check.sh` in the project directory. The output should include

```
[    22.593272] tdx: TDX module initialized.
```

TD VM. Log in to the TD VM and check that the memory encryption is active by `dmesg | grep -i tdx`. The output should look like:

```
[    0.000000] tdx: Guest detected
[    3.211618] Memory Encryption Features active: Intel TDX
[    13.230680] process: using TDX aware idle routine
```

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): H_2O_2RAM outperforms EnigMap by a factor ranging from 100× to 997× for oblivious map accesses with results reported in Figure 8 (a). H_2O_2RAM is compared with GraphOS for oblivious single-source shortest path computation as shown in Figure 8 (b), with GraphOS's results *taken* directly from its paper. Note that GraphOS and EnigMap neither compare nor cite each other. This claim is proven by the experiment E1.

(C2): H_2O_2RAM's performance under different scenarios (e.g., available threads, data block sizes). This is proven by the experiment (E2) discussed in §5.1 whose results are shown in Figure 7.

(C3): More precise failure probabilities of bucket hash tables & stashless Cuckoo hash tables. This is proven by the experiment (E3) discussed in §5.1 whose results are shown in Figure 5.

A.4.2 Experiments

(E1): *[Evaluation on different oblivious tasks] [10 human-minute + 6 compute-hour + 1 TB memory]:* A one-time preprocessing process will take about several compute-days. If your memory is insufficient, partial results will be produced.
How to: Under the project folder, run `cd benchmarks/ && python3 omap_exp_script.py` and `python3 osssp_exp_script.py` to obtain our results. In case of any interruption/corruption, please retry the Python scripts. Follow the instructions in the adapted repository to reproduce EnigMap's results. While GraphOS's results are taken directly from its paper, rather than being rerun in our environment.
Results: `benchmarks/result_omap_{n}.json` will include results for oblivious maps with n data blocks. `benchmarks/results_sssp.json` will include results for oblivious single-source-shortest-path computations across different graph sizes.

(E2): *[Evaluation on different scernarios] [30 human-minute + 12 compute-hour + 1 TB memory]:* A one-time preprocessing process will take about several compute-days. If memory is insufficient, partial results will be produced.
How to: Under the project folder, run `cd benchmarks/ && nohup python3 oram_exp_script.py &`. In case of any interruption/corruption, please try `nohup python3 oram_exp_script.py &` again.
Results: `benchmarks/result_{n}_{b}_{t}.json` includes results for n data blocks of size b with t threads.

(E3): *[Concrete failure probabilities] [10 human-minutes + 1 compute-hour]:* Leverage numeric methods to compute concrete failure probabilities of a) bucket hash tables for different bucket sizes ℓ and bucket numbers m, and b) stashless Cuckoo hash tables for different table sizes n and number of hash functions k.
How to: Under the project folder, run `cd failure_probabilities/ && ./compile.sh`. Then run the compiled binaries `./fail_prob_a` and `./fail_prob_b` respectively.
Results: `./fail_prob_a` will output ℓ versus different numbers of buckets. `./fail_prob_a` will output concrete failure probabilities δ with logarithmic values to the base 2 against different n and k.

A.5 Notes on Reusability

One can directly replace `std::vector` with H_2O_2RAM in their implementation to hide memory accesses. However, H_2O_2RAM

is not thread-safe even if all threads are performing only read operations, as reads can modify H_2O_2RAM's internal states.

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/usenixsec2025/.