VeriZexe: Decentralized Private Computation with Universal Setup

Alex Luoyuan Xiong, *Espresso Systems, National University of Singapore;* Binyi Chen and Zhenfei Zhang, *Espresso Systems;* Benedikt Bünz, *Espresso Systems, Stanford University;* Ben Fisch, *Espresso Systems, Yale University;* Fernando Krell and Philippe Camacho, *Espresso Systems*

https://www.usenix.org/conference/usenixsecurity23/presentation/xiong

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 32nd USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 32nd USENIX Security Symposium.

August 9–11, 2023 • Anaheim, CA, USA

978-1-939133-37-3

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE
ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL
ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

USENIX'23 Artifact Appendix: VeriZexe: Decentralized Private Computation with Universal Setup

Alex Luoyuan Xiong[1], Binyi Chen[2], Zhenfei Zhang[3], Benedikt Bünz[4], Ben Fisch[5], Fernando Krell[6], and Philippe Camacho[7]

[1,2,3,4,5,6,7]Espresso Systems
[1]National University of Singapore
[4]Stanford University
[5]Yale University

April 24, 2023

A Artifact Appendix

A.1 Abstract

We provide the instructions to access and evaluate artifacts for performance of VeriZexe system. The artifacts contain a `veri-zexe` code base written in Rust with benchmark test suites, and a forked `snarkVM` code base[1] as the state-of-the-art to compare against. We further specify the hardware specifications under which our VeriZexe can successfully generate transaction in a reasonable time frame thanks to the massive improvements on prover time and memory usage. This demonstrates the practicality of our system even on resource-limited devices like phones and laptops.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

There should be no security risk or privacy leakage of any kind for evaluators. Executions of the artifacts have no side-effect outside of the testing folders, nor would the programs require any privileged system permission to run.

A.2.2 How to access

Our artifacts (software only) are hosted as public repositories on GitHub. Our implementation of VeriZexe system is accessible via:

https://github.com/EspressoSystems/veri-zexe/tree/42657f254c7f1353914b098dc78f5fb97408bfcd.

The primary prior work that we improve on and benchmark against is accessible via:

https://github.com/alxiong/snarkVM/tree/290c05273e3a30523335524fb682ef316cbbf414.

A.2.3 Hardware dependencies

We do not require special hardware, and the evaluation can be run on any Linux machine. To reproduce the same result, we recommend using Amazon EC2 instances:

Instance Type	vCPU	Memory	Arch	Simulating
a1.xlarge	4	8 GB	arm64	Phone
c5a.4xlarge	16	32 GB	x86_64	Laptop
c5a.16xlarge	64	128 GB	x86_64	Server

Table 1: AWS EC2 instance type and hardware spec

A.2.4 Software dependencies

Any Linux distribution will work, and we use Ubuntu 20.04 across all experiments. The only software prerequisite is: `Rust` : https://www.rust-lang.org/tools/install.

A.2.5 Benchmarks

None. No external data-set or benchmark model required.

A.3 Set-up

A.3.1 Installation

1. Install software prerequisites listed in ??.

2. Git clone both repos, `veri-zexe` at https://github.com/EspressoSystems/veri-zexe.git and `snarkVM` at https://github.com/alxiong/snarkVM.

3. For both repos (same procedure), `cd` into the repo folder, git checkout to `paper-benchmark` branch, run `cargo build`.

[1] Modified for fair comparison and faithful instantiation of the original DPC scheme

A.3.2 Basic Test

Ensure you can compile the source code and test/bench code by running:

```
cargo check && cargo test --no-run
```

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): VERIZEXE improves the state-of-the-art (snarkVM) by 9x in transaction generation time and by 3.4x in memory usage with small variability across different transaction dimensions. This is proven by the Experiment (E1 + E2) described in **??** whose results are reported in Table. 2.

(C2): VERIZEXE is the first DPC scheme to make transaction generation possible and practical in resource-limited hardware environments, such as mobile phones or consumer-grade laptops. Furthermore, we exhibit a trade-off between prover time and peak memory usage. This is proven by the Experiment (E3) desribed in **??** whose results are reported in Table. 4.

A.4.2 Experiments

[Mandatory for Artifacts Functional & Results Reproduced, optional for Artifact Available] Link explicitly the description of your experiments to the items you have provided in the previous subsection about Major Claims. Please provide your estimates of human- and compute-time for each of the listed experiments (using the suggested hardware/software configuration above). Follows an example:

(E1): *[2 human-minutes + 1.2 computer-minutes + c5a.16xlarge EC2]:* We run benchmarks on `veri-zexe` across different transaction dimensions (`2x2,3x3,4x4`) and measure all major metrics among which total transaction generation time and peak memory usage are the main targets.
Preparation: Enter into your AWS c5a.16xlarge EC2 instance, or environments of the same hardware spec (see Table. **??**).
Execution and Result: Please follow detailed instructions in `usenix-ae.md` file in the `veri-zexe` project root.

(E2): *[5 human-minutes + 15 computer-minutes + c5a.16xlarge EC2]* We run benchmarks on `snarkVM` across different transaction dimensions (`2x2,3x3,4x4`) in the same environment and measuring the same metrics.
Preparation: Enter into your AWS c5a.16xlarge EC2 instance.
Execution and Result: Please follow detailed instructions in `usenix-ae.md` file in the `snarkVM` project root.

(E3): *[5 human-minutes + 6 computer-minutes + a1.xlarge & c5a.4xlarge EC2]* We try to generate 2-input-2-output DPC transaction across different hardware environments, especially resource-limited environment simulating phones and laptops. This used to be impossible for `snarkVM` due to high memory usage and much slower prover.
Preparation: Enter into your AWS a1.xlarge and c5a.4xlarge EC2 instance.
Execution and Result: Please follow detailed instructions in `usenix-ae.md` file in the `veri-zexe` project root.

A.5 Version

Based on the LaTeX template for Artifact Evaluation V20220926. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2023/.