USENIX Security '25 Artifact Appendix: Available Attestation: Towards a Reorg-Resilient Solution for Ethereum Proof-of-Stake

Mingfei Zhang
Shandong University
mingfei.zh@outlook.com

Rujia Li[⋆]
Tsinghua University
rujia@tsinghua.edu.cn

Xueqian Lu
Independent Researcher
xueqian.lu@bitheart.org

Sisi Duan[⋆†]
Tsinghua University
duansisi@tsinghua.edu.cn

A Artifact Appendix

A.1 Abstract

We present available attestation, a provable reorg-resilient solution for Ethereum Proof-of-Stake. In this work, we show that the majority of the known attacks on Ethereum PoS are some form of reorganization attacks. In practice, most of these attacks can be launched even if the network is synchronous (there exists a known upper bound for message transmission and processing). Different from existing studies that mitigate the attacks in an ad-hoc way, we take a systematic approach and provide an elegant yet efficient solution to reorganization attacks. We implement our modified protocol and five reorg attacks in our artifact and evaluate them using a local-built testnet of 16,384 clients. The results show that our protocol is resilient to five reorganization attacks and highly efficient. This artifact shows how to reproduce the results in Section 8 of our paper.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

All experiments are conducted on a local testnet (running on a single local machine). No experiments are conducted on the live Ethereum network. This repository does not uncover new vulnerabilities but instead analyzes known malicious reorganization attacks.

A.2.2 How to access

The artifact can be accessed by cloning our public Github project. All the scripts, container images, source codes, and sample output files can be accessed via the stable URL: https://zenodo.org/records/15205897.

[⋆] Corresponding author.
[†] Sisi is also with Zhongguancun Laboratory, Shandong Institute of Blockchains, BNRist, and State Key Laboratory of Cryptography and Digital Economy Security.

A.2.3 Hardware dependencies

The experiments do not require any specialized hardware. Our test environment is a computer with a 4-core CPU, 16 GB of RAM, 100 GB of storage, and a 100 Mbps network connection.

A.2.4 Software dependencies

Our experiments run inside Docker containers. Make sure you install Docker following the official documentation, and verify that your Docker Engine version is at least 24.
We use Python for data processing and plotting. Make sure you have Python 3.10 (or a later version) installed. Then install the required packages via the following command::

```
pip3 install -r requirements.txt
```

A.2.5 Benchmarks

None.

A.3 Set-up

A.3.1 Installation

After installing Docker and Python3, run the following steps:

1. Git clone the repository:
```
git clone https://github.com/tsinghua-
cel/available-attestation
```
2. Enter the cloned repository directory (denoted as `$HOME`):
```
cd available_attestation/
```
3. Switch to the stable version:
```
git checkout
94b76d76126481031e5aaa524cf19ced6414a182
```
4. Build the docker image:
```
./build.sh
```

A.3.2 Basic Test

After building the docker image, run the basic test by the command:

```
./runtest.sh basic
```

The following outputs are expected:

```
[+] Running 19/19
 - Network basic_meta          Created      0.1s
 - Container execute5          Started      0.4s
 - Container execute3          Started      0.4s
 - Container execute1          Started      0.4s
 - Container ethmysql          Started      0.4s
 - Container execute2          Started      0.4s
 - Container execute4          Started      0.2s
 - Container beacon-2          Started      0.6s
 - Container attacker-1        Started      0.4s
 - Container beacon-3          Started      0.4s
 - Container beacon-1          Started      0.6s
 - Container beacon-4          Started      0.4s
 - Container beacon-5          Started      0.6s
 - Container validator-4       Started      0.6s
 - Container validator-3       Started      0.8s
 - Container strategy          Started      0.8s
 - Container validator-1       Started      0.8s
 - Container validator-2       Started      0.8s
 - Container validator-5       Started      0.8s
```

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): *The number of reorg blocks for the modified protocol is zero. This is validated by the experiments (E1), which reproduce the results described in Figure 13, Section 8 of the paper.*

(C2): *The throughput of the vanilla protocol and the modified protocol are almost the same. This is proven by the experiments (E2), which reproduce the results described in Figure 14, Section 8 of the paper.*

(C3): *The latency of the vanilla protocol and the modified protocol are almost the same. This is proven by the experiments (E3), which reproduce the results described in Figure 15, Section 8 of the paper.*

A.4.2 Experiments

(Overview) *We conduct three types of experiments: reorg resilience experiments (E1), throughput experiments (E2), and latency experiments (E3). Each experiment is performed for both the vanilla Ethereum PoS protocol and the modified Ethereum PoS protocol.*

(E1): *[25 compute-hour] The experiments involve five reorganization attacks (i.e., ex-ante reorg attack, sandwich reorg attack, unrealized justification reorg attack, justification withholding attack, and staircase attack). These experiments last for 12.5 hours on each protocol. The entire experimentation time thus lasts for 25 hours.*

How to: *run the experiment with the command under* `$HOME` *:*

```
./runtest.sh reorg
```

The experiments will run reorganization attacks for 9000 seconds on each protocol (approximately 25 hours in total). The output should be similar to that in the basic test. At the end of the experiment, the output will display:

```
test done and all data in
$HOME/results/reorgtest, report in
$HOME/reorgs.png
```

Results: *After completion, the result can be found in the* `$HOME` *directory. The number of reorg blocks for the modified protocol should align with the data shown in Figure 13 in the paper (i.e., the number of reorg blocks in the modified protocol is zero).*

Note: *Experiment E1 will run for 25 hours. If one does not want to wait for such a long period of time, run the following commands to conduct each attack separately. Each attack lasts for about five hours.*

Run the modified exante reorg attack:

```
./runtest.sh 1
```

Run the sandwich reorg attack:

```
./runtest.sh 2
```

Run the unrealized justification reorg attack:

```
./runtest.sh 3
```

Run the justification withholding reorg attack:

```
./runtest.sh 4
```

Run the staircase attack:

```
./runtest.sh 5
```

(E2): *[1.3 compute-hour] The experiments are used for assessing the throughput of both the vanilla protocol and the modified protocol.*

How to: *run the experiment with the command under* `$HOME` *:*

```
./runtest.sh tps
```

An experiment lasts for 40 minutes for each protocol. The entire experiment lasts for 1.3 hours in total. The output should be similar to that in the basic test. At the end of the experiment, the output will display:

```
test done and all data in $HOME/results/tps,
report in $HOME/tps.png.
```

Results: *After completion, the results can be found in the* `$HOME` *directory. The throughput of the modified protocol should align with the data shown in Figure 14 in the paper (i.e., the throughput of the vanilla protocol and the modified protocol are almost the same).*

(E3): *[20 compute-minutes]: The experiments test the latency of both the vanilla protocol and the modified protocol.*

How to: *run the experiment with the command under* `$HOME` *:*

```
./runtest.sh latency
```

The experiments evaluate the computation time across *both the vanilla protocol and the modified protocol. The output should be similar to that in the basic test. At the end of the experiment, the output will display:*

```
test done and all data in
$HOME/results/blockcost, report as bellow.
```

Results: *After completion, the results can be found in the* `$HOME` *directory. The latency of the modified protocol should align with the data shown in Figure 15 in the paper (i.e., the latency of the vanilla protocol and the modified protocol are almost the same).*

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/`.