

ASIACRYPT '24 Artifact Appendix: LogRobin++

Carmit Hazay
Bar-Ilan University, Israel
Carmit.Hazay@biu.ac.il

David Heath
University of Illinois Urbana-Champaign, USA
daheath@illinois.edu

Vladimir Kolesnikov
Georgia Institute of Technology, USA
kolesnikov@gatech.edu

Muthuramakrishnan Venkitasubramaniam
Ligero Inc., USA
muthu@ligero-inc.com

Yibin Yang
Georgia Institute of Technology, USA
yyang811@gatech.edu

A Artifact Appendix

A.1 Abstract

The artifact includes code for all benchmarks presented in the paper. It mainly includes the interactive ZK schemes in our paper: Robin++, LogRobin, LogRobin++. It also includes the baseline Robin.

This document describes how one can use our code to reproduce *all* results in Section 5 of the proceedings paper.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

None.

A.2.2 How to access

GitHub link:

<https://github.com/gconeice/logrobinplus>

We will maintain new versions, and this appendix will be included and updated accordingly in our repository.

A.2.3 Hardware dependencies

Our repository can be executed on a single machine to emulate ZK Prover \mathcal{P} and ZK Verifier \mathcal{V} using a localhost network. However, our results were tested using two standalone machines: one for \mathcal{P} , and another for \mathcal{V} .

We tested our code on two machines, each having $\leq 16\text{GiB}$ memory. In particular, we used two Amazon Web Services (AWS) EC2 **m5.xlarge** machines.

We only tested it over x86_64 CPUs, but we believe ARM CPUs (i.e., Apple M1) should also work.

A.2.4 Software dependencies

We tested our code on a clean installation of Ubuntu 22.04. Our repository includes simple scripts to install everything starting from a clean installation.

We depend on Robin¹, which is developed based on the EMP-toolkit² (in particular, the VOLE functionalities inside). Our scripts will help you set it up properly.

We use Linux command `tc` to simulate the network with a certain bandwidth.

The prerequisites (on Ubuntu) include: `git`, `cmake`, `clang`, `build-essential`, `libssl-dev`, and `software-properties-common`.

A.3 Set-Up

You can simply download our repository and type “**bash setup.sh**”. In the meantime, you can enable parallel compilation by using the “**-para**” flag.

Note: for cloud deployment on AWS, ensure that the security groups are configured to allow inbound/outbound traffic on the corresponding port (e.g., 12345).

A.3.1 Installation

You can simply download our repository and type “**bash install.sh**”. In the meantime, you can enable parallel compilation by using the “**-para**” flag.

A.3.2 Basic toy test

Note that we have two machines — \mathcal{P} and \mathcal{V} . For \mathcal{P} and \mathcal{V} , goto the folder `build`. Let `ip` denote the machine \mathcal{P} 's IP address, and set environment variable ‘`IP=ip`’ on \mathcal{V} 's machine.

\mathcal{P} executes:

¹<https://github.com/gconeice/stacking-vole-zk>

²<https://github.com/emp-toolkit>

```
./bin/test_rep_bool_logrobinplus_ro 1 12345
localhost 1 10 10000000
 $\mathcal{V}$  executes:
./bin/test_rep_bool_logrobinplus_ro 2 12345
$IP 1 10 10000000
```

If everything goes through, you should see execution times and the #bytes sent on \mathcal{P} and \mathcal{V} . On the other hand, if something goes wrong, you will see the corresponding error messages. (The experiment can take a few seconds.)

A.3.3 Expected executable files

You should generate the following executable files located in build/bin/:

```
test_(rand/rep)_(bool/arith)_(log)robin(plus)_(it/ro)
```

with the following meanings:

- rand or rep stands for executing B different or identical circuits (branches).
- bool or arith stands for executing the Boolean or arithmetic circuits.
- (log)robin(plus) stands for:
 - robin: the baseline Robin protocol.
 - logrobin: our LogRobin protocol.
 - robinplus: our Robin++ protocol.
 - logrobinplus: our LogRobin++ protocol.
- it or ro stands for our information-theoretic or random-oracle-based variants.

All these executable files take the following input:

```
PARTY PORT IP LOG_BRANCH_SIZE #CIR_IN #CIR_MULT
```

A.4 Evaluation Workflow

Please set environment variable ‘IP=ip’ on \mathcal{V} ’s machine, where ip is the machine \mathcal{P} ’s IP address.

A.4.1 Major Claims

- (C1): The performance of our protocols with $B = 2^{22}$, $n_{in} = 10$, $n_x = 100$ is illustrated/reported in Table 2. This is proven by the experiment (E1) described in Section 5.2.
- (C2): The performance of our protocols with $B = 2$, $n_{in} = 10$, $n_x = 10^7$ is illustrated/reported in Table 3. This is proven by the experiment (E2) described in Section 5.2.
- (C3): The communication of our LogRobin++ v.s. the baseline Robin in the VOLE-hybrid model with $B = 2^4 \cdot 2^{16}$, $n_{in} = 10$, $n_x = 100$ is illustrated/reported in Figure 7. This is proven by the experiment (E3) described in Section 5.3.

- (C4): The communication of our LogRobin++ v.s. the baseline Robin in the VOLE-hybrid model with $B = 2$, $n_{in} = 10$, $n_x = 1 \times 10^6 - 10 \times 10^6$ is illustrated/reported in Figure 8. This is proven by the experiment (E4) described in Section 5.3.

A.4.2 Minor Claims

- (C5): The performance of our protocols between different and identical circuits is illustrated/reported in Table 4. This is proven by the experiment (E5) described in Section 5.4.
- (C6): The performance of our protocols between IT and RO variants is illustrated/reported in Table 5. This is proven by the experiment (E6) described in Section 5.5.

A.4.3 Experiments

Note: All our figures/tables are plotted based on the data in the Excel file benchmark_summary.xlsx. Therefore, we will show how one can reproduce the numbers in this Excel file and then how to transform them into figures/tables.

- (E1): [Table 2] [10 human-minutes + 10 compute-minutes of two machines + 16GB memory each machine/party]: Please cd to the folder build.

Preparation: For both machines, let the name of the network card be ens5; please set up the network as follows:

1. DEV=ens5 (change ens5 accordingly)
2. If there exists a previous old setting, initialize it:

```
sudo tc qdisc del dev $DEV root
```
3. sudo tc qdisc add dev \$DEV root handle 1: tbf rate 10Mbit burst 100000 limit 10000 (resp. 1Gbit)
4. sudo tc qdisc add dev \$DEV parent 1:1 handle 10: netem

Recall that the intended network is either 10 Mbps or 1 Gbps. You can use iperf to check it.

Data in Excel: Please refer to the Sheet1.

Execution: For each network setting, the following execution needs to be executed repeatedly (i.e., twice).

- The baseline Robin on Boolean:

\mathcal{P} machine:

```
./bin/test_rep_bool_robin_ro 1 12345
localhost 22 10 100
```

\mathcal{V} machine:

```
./bin/test_rep_bool_robin_ro 2 12345
$IP 22 10 100
```

- Our protocol Robin++ on Boolean:

\mathcal{P} machine:

```
./bin/test_rep_bool_robinplus_ro 1
12345 localhost 22 10 100
```

\mathcal{V} machine:

```
./bin/test_rep_bool_robinplus_ro 2
12345 $IP 22 10 100
```

- Our protocol LogRobin on Boolean:
 \mathcal{P} machine:
./bin/test_rep_bool_logrobin_ro 1 12345
localhost 22 10 100
 \mathcal{V} machine:
./bin/test_rep_bool_logrobin_ro 2 12345
\$IP 22 10 100
- Our protocol LogRobin++ on Boolean:
 \mathcal{P} machine:
./bin/test_rep_bool_logrobinplus_ro 1
12345 localhost 22 10 100
 \mathcal{V} machine:
./bin/test_rep_bool_logrobinplus_ro 2
12345 \$IP 22 10 100
- The baseline Robin and our protocols Robin++, LogRobin and LogRobin++ on arithmetic: simply change the bool to arith in the above instructions to perform the corresponding experiments.

Results: The time outputted on \mathcal{V} 's terminals reflects the "Time(s)" column in Table 2 (Excel and the paper). Additionally, the " $\mathcal{P} \rightarrow \mathcal{V}$ " column in Table 2 and Excel reflects the communication in \mathcal{P} 's terminal (in Byte); the " $\mathcal{V} \rightarrow \mathcal{P}$ " column in Table 2 and Excel reflects the communication in \mathcal{V} 's terminal (in Byte).

Post-processing: The "Total" and "Impr." columns in Table 2 can be computed from the other data trivially.

(E2): [Table 3] [10 human-minutes + 10 compute-minutes of two machines + 16GB memory each machine/party]: Please cd to the folder build.

Preparation: For both machines, let the name of the network card be ens5; please set up the network as follows:

1. DEV=ens5 (change ens5 accordingly)
2. If there exists a previous old setting, initialize it:
sudo tc qdisc del dev \$DEV root
3. sudo tc qdisc add dev \$DEV root handle 1: tbf rate 10Mbit burst 100000 limit 10000 (resp. 1Gbit)
4. sudo tc qdisc add dev \$DEV parent 1:1 handle 10: netem

Recall that the intended network is either 10 Mbps or 1 Gbps. You can use iperf to check it.

Data in Excel: Please refer to the Sheet1.

Execution: For each network setting, the following execution needs to be executed repeatedly (i.e., twice).

- The baseline Robin on Boolean:
 \mathcal{P} machine:
./bin/test_rep_bool_robin_ro 1 12345
localhost 1 10 10000000
 \mathcal{V} machine:

```
./bin/test_rep_bool_robin_ro 2 12345
$IP 1 10 10000000
```

- Our protocol Robin++ on Boolean:
 \mathcal{P} machine:
./bin/test_rep_bool_robinplus_ro 1
12345 localhost 1 10 10000000
 \mathcal{V} machine:
./bin/test_rep_bool_robinplus_ro 2
12345 \$IP 1 10 10000000
- Our protocol LogRobin on Boolean:
 \mathcal{P} machine:
./bin/test_rep_bool_logrobin_ro 1 12345
localhost 1 10 10000000
 \mathcal{V} machine:
./bin/test_rep_bool_logrobin_ro 2 12345
\$IP 1 10 10000000
- Our protocol LogRobin++ on Boolean:
 \mathcal{P} machine:
./bin/test_rep_bool_logrobinplus_ro 1
12345 localhost 1 10 10000000
 \mathcal{V} machine:
./bin/test_rep_bool_logrobinplus_ro 2
12345 \$IP 1 10 10000000
- The baseline Robin and our protocols Robin++, LogRobin and LogRobin++ on arithmetic: simply change the bool to arith in the above instructions to perform the corresponding experiments.

Results: The time outputted on \mathcal{V} 's terminals reflects the "Time(s)" column in Table 3 (Excel and the paper). Additionally, the " $\mathcal{P} \rightarrow \mathcal{V}$ " column in Table 3 and Excel reflects the communication in \mathcal{P} 's terminal (in Byte); the " $\mathcal{V} \rightarrow \mathcal{P}$ " column in Table 3 and Excel reflects the communication in \mathcal{V} 's terminal (in Byte).

Post-processing: The "Total" and "Impr." columns in Table 3 can be computed from the other data trivially.

(E3): [Figure 7] [10 human-minutes + 20 compute-minutes of two machines + 16GB memory each machine/party]: Please cd to the folder build.

Preparation: In this experiment, we need to count the costs in the VOLE-hybrid model. Hence, please uncomment the Line 91 in:

- /test/rep/bool_robin_ro.cpp
- /test/rep/arith_robin_ro.cpp.
- /test/rep/bool_logrobinplus_ro.cpp
- /test/rep/arith_logrobinplus_ro.cpp

Then, recompile them. I.e., execute "make -j" in the folder build. We remark that only experiments E3 and E4 need to uncomment these lines.

Data in Excel: Please refer to the Sheet2.

Execution: For each $b \in [4, 16]$, execute:

- The baseline Robin on Boolean:
 \mathcal{P} machine:

```
./bin/test_rep_bool_robin_ro 1 12345
localhost b 10 100
 $\mathcal{V}$  machine:
./bin/test_rep_bool_robin_ro 2 12345
$IP b 10 100
```

- Our protocol LogRobin++ on Boolean:

```
 $\mathcal{P}$  machine:
./bin/test_rep_bool_logrobinplus_ro 1
12345 localhost b 10 100
 $\mathcal{V}$  machine:
./bin/test_rep_bool_logrobinplus_ro 2
12345 $IP b 10 100
```

- The baseline Robin and our protocol LogRobin++ on arithmetic: simply change the bool to arith in the above instructions to perform the corresponding experiments.

Results: The “ $\mathcal{P} \rightarrow \mathcal{V}$ ” rows in Excel reflect the communication in \mathcal{P} ’s terminal (in Byte); the “ $\mathcal{V} \rightarrow \mathcal{P}$ ” rows in Excel reflect the communication in \mathcal{V} ’s terminal (in Byte).

Post-processing: Note that the plots in Figure 7 are generated by the *total* communications.

(E4): [Figure 8] [10 human-minutes + 20 compute-minutes of two machines + 16GB memory each machine/party]: Please cd to the folder build.

Preparation: In this experiment, we need to count the costs in the VOLE-hybrid model. Hence, please uncomment the Line 91 in:

- /test/rep/bool_robin_ro.cpp
- /test/rep/arith_robin_ro.cpp.
- /test/rep/bool_logrobinplus_ro.cpp
- /test/rep/arith_logrobinplus_ro.cpp

Then, recompile them. I.e., execute “make -j” in the folder build. We remark that only experiments E3 and E4 need to uncomment these lines.

Data in Excel: Please refer to the Sheet2.

Execution: For each $C \in \{1 \times 10^6, 2 \times 10^6, \dots, 9 \times 10^6, 10 \times 10^6\}$, execute:

- The baseline Robin on Boolean:

```
 $\mathcal{P}$  machine:
./bin/test_rep_bool_robin_ro 1 12345
localhost 1 10 C
 $\mathcal{V}$  machine:
./bin/test_rep_bool_robin_ro 2 12345
$IP 1 10 C
```

- Our protocol LogRobin++ on Boolean:

```
 $\mathcal{P}$  machine:
./bin/test_rep_bool_logrobinplus_ro 1
12345 localhost 1 10 C
 $\mathcal{V}$  machine:
./bin/test_rep_bool_logrobinplus_ro 2
12345 $IP 1 10 C
```

- The baseline Robin and our protocol LogRobin++ on arithmetic: simply change the bool to arith in the above instructions to perform the corresponding experiments.

Results: The “ $\mathcal{P} \rightarrow \mathcal{V}$ ” rows in Excel reflect the communication in \mathcal{P} ’s terminal (in Byte); the “ $\mathcal{V} \rightarrow \mathcal{P}$ ” rows in Excel reflect the communication in \mathcal{V} ’s terminal (in Byte).

Post-processing: Note that the plots in Figure 8 are generated by the *total* communications.

(E5): [Table 4] [10 human-minutes + 20 compute-minutes of two machines + 16GB memory each machine/party]: Please cd to the folder build.

Preparation: For both machines, let the name of the network card be ens5; please set up the network as follows:

1. DEV=ens5 (change ens5 accordingly)
2. If there exists a previous old setting, initialize it:
sudo tc qdisc del dev \$DEV root
3. sudo tc qdisc add dev \$DEV root handle 1: tbf rate 10Mbit burst 100000 limit 10000 (resp. 1Gbit)
4. sudo tc qdisc add dev \$DEV parent 1:1 handle 10: netem

Recall that the intended network is either 10 Mbps or 1 Gbps. You can use iperf to check it.

Data in Excel: Please refer to the Sheet3.

Execution: For each network setting, the following execution needs to be executed repeatedly (i.e., twice).

- The baseline Robin on arithmetic:

```
 $\mathcal{P}$  machine:
./bin/test_rep_arith_robin_ro 1 12345
localhost 10 10 100000
 $\mathcal{V}$  machine:
./bin/test_rep_arith_robin_ro 2 12345
$IP 10 10 100000
```

- Our protocol Robin++ on arithmetic:

```
 $\mathcal{P}$  machine:
./bin/test_rep_arith_robinplus_ro 1
12345 localhost 10 10 100000
 $\mathcal{V}$  machine:
./bin/test_rep_arith_robinplus_ro 2
12345 $IP 10 10 100000
```

- Our protocol LogRobin on arithmetic:

```
 $\mathcal{P}$  machine:
./bin/test_rep_arith_logrobin_ro 1
12345 localhost 10 10 100000
 $\mathcal{V}$  machine:
./bin/test_rep_arith_logrobin_ro 2
12345 $IP 10 10 100000
```

- Our protocol LogRobin++ on arithmetic:

```
 $\mathcal{P}$  machine:
```

```
./bin/test_rep_arith_logrobinplus_ro 1  
12345 localhost 10 10 100000
```

\mathcal{V} machine:

```
./bin/test_rep_arith_logrobinplus_ro 2  
12345 $IP 10 10 100000
```

- The baseline Robin and our protocols Robin++, LogRobin and LogRobin++ on B different circuits (or branches): simply change the `rep` to `rand` in the above instructions to perform the corresponding experiments.

Results: The time outputted on \mathcal{V} 's terminals reflects the "Time(s)" column in Table 4 (Excel and the paper). In particular, the "Different" column reflects the experiments with `rand`, and the "Identical" column reflects the experiments with `rep`.

(E6): [Table 5] [10 human-minutes + 20 compute-minutes of two machines + 16GB memory each machine/party]: Please `cd` to the folder `build`.

Data in Excel: Please refer to the `Sheet1`. In particular, see the "IT" half on the right.

How to: Simply change the `ro` to `it` in the E1 and E2 instructions to perform the corresponding experiments.