SIMC: ML Inference Secure Against Malicious Clients at Semi-Honest Cost

Nishanth Chandran, Divya Gupta, and Sai Lakshmi Bhavana Obbattu, *Microsoft Research;* Akash Shah, *UCLA*

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

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

Secure inference allows a model owner (or, the server) and the input owner (or, the client) to perform inference on machine learning model without revealing their private information to each other. Recently, Lehmkuhl *et al.* proposed a secure inference system, MUSE, in client malicious threat model. In our paper titled "SIMC: ML Inference Secure Against Malicious Clients at Semi-Honest Cost", we design and build SIMC, a new cryptographic system for secure inference in client malicious threat model.

In this artifact, we implement our proposed system SIMC. Using this implementation, we show that SIMC has $23-29\times$ lesser communication and is up to $11.4\times$ faster than MUSE, for benchmarks considered by MUSE. SIMC obtains these improvements using a novel protocol for non-linear activation functions (such as ReLU) that has $> 28\times$ lesser communication and is up to $43\times$ more performant than MUSE.

In this article, we summarize the system requirement, installation and building process, and finally, the execution process in order to obtain the performance numbers reported in our paper.

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

- **Algorithm:** SIMC (Secure Inference Against Malicious Client) protocol.
- **Program:** Implementation in C++ (https://aka.ms/simc).
- **How much disk space required (approximately)?:** 16GB.
- **How much time is needed to prepare workflow (approximately)?:** 3 hours.
- **How much time is needed to complete experiments (approximately)?:** 20 minutes
- **Publicly available?:** Yes.
- **Code licenses (if publicly available)?:** MIT License.

A.3 Description

A.3.1 How to access

Access the github repo using link: https://aka.ms/simc (commit id: 2a5fd092b52427cc9cac55b36ec50ae43ecee6be).

A.3.2 Software dependencies

Install Eigen3, SEAL and emp-toolkit repositories. See Installation steps for more details.

A.4 Installation and Compilation

1. Create parent directory `msi-code`

   ```
   mkdir msi-code && cd msi-code
   ```

2. To install Eigen3 do

   ```
   sudo apt-get update -y
   sudo apt-get install -y libeigen3-dev
   ```

3. Follow the installation steps of [emp-toolkit/emp-sh2pc].
4. Clone this repo in the parent directory `msi-code`.
5. Install SEAL 3.64
   (a) Clone SEAL 3.6 repo in the parent directory `msi-code`.
   (b) Execute

   ```
   cd SEAL
   git checkout 3.6.4
   mkdir build && cd build
   cmake ..
   make -j
   sudo make install
   ```

6. In `msi-code`, go to `emp-tool` and do `git checkout df363bf30b56c48a12c352845efa3a4d8f75b388`.
7. Next, go to `emp-ot` in `msi-code` and do `git checkout 3b21d6314cb1e7d8dbb9bb1f1ed80261738e4f4c`.
8. For multi-threading support, go to `emp-tool` and run the following

   ```
   cmake . -DTHREADING=ON
   make -j
   sudo make install
   ```

9. Do the same for `emp-ot` repository.
10. Finally, do the same in our (`simc`) repository.

A.5 Experiment workflow

The protocol is run between two parties. Open two terminal windows and run the following test files from path `msi-code/simc`:

A.5.1 Run Neuralnet Benchmarks

1. Fully-connected Layer: In one terminal run `bin/test_msi_linearlayer 1 0.0.0.0 <port_no> 44 <neural_network>` and in other terminal run `bin/test_msi_linearlayer 2 <server_ip_address> <port_no> 44 <neural_network>`.
2. Convolution Layer: In one terminal run `bin/test_msi_convlayer 1 0.0.0.0 <port_no> 44 <neural_network>` and in other terminal run `bin/test_msi_convlayer 2 <server_ip_address> <port_no> 44 <neural_network>`.
3. Non-Linear Layer (ReLU): In one terminal run `bin/test_msi_relu_final 1 0.0.0.0 <port_no> 44 <neural_network>` 0 0 <num_threads> and in other terminal run `bin/test_msi_relu_final 2 <server_ip_address> <port_no> 44 <neural_network>` 0 0 <num_threads>.
4. Average Pool Layer: In one terminal run `bin/test_msi_average 1 0.0.0.0 <port_no> 44 <neural_network>` and in other terminal run `bin/test_msi_average 2 <server_ip_address> <port_no> 44 <neural_network>`.

Here, the first parameters 1 and 2 denote the ID of the participating party. `<server_ip_address>` denotes the ip address of the server machine and set `<neural_network>`=1 for MNIST and `<neural_network>`=2 for CIFAR-10. See Figure 1 for examples.

A.5.2 Run Neuralnet Micro-benchmarks

See Figure 2 for instructions and examples to run micro-benchmarks. Note that, for different system-configuration, different number of threads may provide best performance for given number of ReLUs.

A.6 Evaluation and expected results

To obtain performance numbers of our protocol that were used to generate the plot of Figure 7 of our paper, follow instructions in Section A.5.2.

Follow instructions in Section A.5.1, and then aggregate the observed runtime and communication cost across all the layers to obtain performance numbers of Tables 1, 2 and 3 of our paper. If the protocols are run in similar system setting as ours, the observed runtime will be similar to what has been reported in paper.

```
Fully connected Layer:
Terminal 1: bin/test_msi_linearlayer 1 0.0.0.0 31000 44 1
Terminal 2: bin/test_msi_linearlayer 2 <server_ip_address> 31000 44 1

Convolution Layer:
Terminal 1: bin/test_msi_convlayer 1 0.0.0.0 31000 44 1
Terminal 2: bin/test_msi_convlayer 2 <server_ip_address> 31000 44 1

Non-Linear Layer (ReLU):
Terminal 1: bin/test_msi_relu_final 1 0.0.0.0 31000 44 1 0 0 8
Terminal 2: bin/test_msi_relu_final 2 <server_ip_address> 31000 44 1 0 0 8

Average Pool Layer:
Terminal 1: bin/test_msi_average 1 0.0.0.0 31000 44 1
Terminal 2: bin/test_msi_average 2 <server_ip_address> 31000 44 1
```

Figure 1: Run Neuralnet Benchmarks Examples

```
Terminal 1: bin/test_msi_microbenchmark 1 0.0.0.0 31000 44 <benchmark_choice>
            <num_relus> <#threads>
Terminal 2: bin/test_msi_microbenchmark 2 <server_ip_address> 31000 44 <benchmark_choice>
            <num_relus> <#threads>


Input Parameters:
1. <server_ip_address>: IP Address of Server.
2. <benchmark_choice>: 0 - ReLU6, 1 - ReLU.
3. <num_relus>: Number of ReLUs
4. <#threads>: Number of threads


if <num_relus> <=2, set <#threads>=1,
else if <num_relus> <=4, set <#threads>=2,
else if <num_relus> <=16, set <#threads>=4,
else if <num_relus> >16, set <#threads>=8.

Example:
Terminal 1: bin/test_msi_microbenchmark 1 0.0.0.0 31000 44 0 16384 8
Terminal 2: bin/test_msi_microbenchmark 2 <server_ip_address> 31000 44 0 16384 8
```

Figure 2: Run Neuralnet Micro-benchmarks