FIT 5124 Advance Topics in Security (S1 2021)

Assignment 2: Secure Multi-Party Computation

Total marks 100

Due on 22 May 11:59:59 am (Firm!)

1 Overview

The objective of this assignment is to assess the learning outcomes in secure multi-party computation (SMC) techniques and their applications to privacy-preserving data mining and machine learning. Specifically, the tasks will evaluate your knowledge of SMC and assess your capability of using SMC primitives to build, analyse, implement, and evaluate privacy-preserving computational protocols in a comprehensive manner.

2 Submission Policy

You need to submit a report (one single PDF file) to describe what you have done and what you have observed with screen shots whenever necessary; you also need to provide explanation or codes to the observations that are related to the tasks. In your report, you are expected to answer all the questions listed in this manual. Typeset your report into .pdf format (make sure it can be opened with Adobe Reader) and name it as the format:
[Your Name]-[Student ID]-FIT5124-Assignment2,
HarryPotter-12345678-FIT5124-Assignment2.pdf.

Please upload the PDF file to Moodle. Note that the assignment is due on **22 May, Saturday, 11:59:59 am (No extension)**.

Late submission penalty: 10-point deduction per day. If you require a special consideration, the application should be submitted at least three days in advance via Monash Connect (https://www.monash.edu/connect). Zero tolerance on plagiarism: If you are found cheating, penalties will be applied, i.e., a zero grade for the unit. The demonstration video is also used to detect/avoid plagiarism. University policies can be found at https://www.monash.edu/students/academic/policies/academic-integrity

3 Using Secure Computation Techniques to Realise Privacy-preserving Data Mining and Machine Learning applications (100 marks)

Pick ***one application scenario*** of interest to you, where secure computation protocols can offer a privacy-preserving solution to the security requirements of the parties involved. Application scenarios include:

- K-means clustering
- Linear regression (10 bonus marks if you select this task)
- Neural network inference (20 bonus marks if you select this task, only inference not training)

Note that we plan to give you bonus marks for Linear Regression and Neural Network Inference due to the higher difficulty of the tasks. Namely, if you choose one of them, your final mark will be multiplied by 1.1 and 1.2, respectively.

For K-means clustering or linear regression, you can choose any dataset from UCI machine learning repository: https://archive.ics.uci.edu/ml/index.php. For the selected dataset, the number of attributes must be larger or equal than 20.

For neural network inference, the task is image classification for the FashionMNIST dataset. Pytorch already include Fashion-MNIST data/API, you can use load the dataset by using:

```
train_set = torchvision.datasets.FashionMNIST(
    root = './data/FashionMNIST',
    train = True,
    download = True,
    transform = transforms.Compose([transforms.ToTensor()])
)
test_set = torchvision.datasets.FashionMNIST(
    root = './data/FashionMNIST',
    train = False,
    transform = transforms.Compose([transforms.ToTensor()])
)
```

You can use direct links to download the dataset. The data is stored in the same format as the original MNIST data as we used before:
training set images:
http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-images-idx3-ubyte.gz

training set labels:
http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-labels-idx1-ubyte.gz
test set images:
http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10k-images-idx3-ubyte.gz
test set labels:
http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10k-labels-idx1-ubyte.gz

Some references (you may follow other references/realisations/algorithms as long as they can achieve the same functionality):

- K-means clustering: Chapter 7.3 http://infolab.stanford.edu/~ullman/mmds/ch7.pdf, “Mining of Massive Datasets”, Jure Leskovec, Anand Rajaraman, Jeff Ullman.
- Linear regression and Neural Network Inference: https://eprint.iacr.org/2017/396.pdf, “SecureML: A System for Scalable Privacy-Preserving Machine Learning”, Payman Mohassel, Yupeng Zhang.

3.1 Plaintext Algorithm Description (10 Marks)

To design a privacy-preserving protocol, the first task is to understand how the plaintext algorithm works. In this question, please describe the details of the plaintext algorithm for your selected application (step by step) with clear definitions of notations and parameters used in the description. For neural network inference, you can also draw a figure or table to illustrate the model architecture you plan to adopt.

3.2 Protocol Overview (10 Marks)

Please explain the concrete application scenario. What are the parties involved in your proposed protocol? What are the security and privacy goals of your proposed protocol? You can draw a figure to illustrate the parties involved and to describe the high-level overview of your proposed protocol.

3.3 Protocol Design (40 Marks)

Please analyse the plaintext algorithm and explain how to use SMC techniques to realise the plaintext algorithm in a privacy-preserving manner. Explain how the algorithm can be computed securely among the parties involved (please write down the protocol step by step), and explain what are the security guarantees for your proposed protocol?

25 marks for clearly presenting the design of the proposed protocols, and 10 marks for correctness of your proposed protocol, and 5 marks on clearly explaining the security guarantees of your proposed protocol.

3.4 Protocol Analysis (10 Marks)

Please conduct performance analysis for your proposed protocol. What is the complexity of communication, computation cost?

3.5 Implementation and evaluation (30 Marks)

Please implement your proposed protocol and clearly explain how to run the code on your VM in detail, so that our teaching team can verify the correctness of your implementation and reproduce the results. Please add detailed comments and print detailed logs in your implementation to foster the teaching team to check the correctness and verify your results. (20 Marks)

Please evaluate the practical performance of your protocol based on the implementation, e.g., the time cost for your protocol under different parameter settings, the communication cost between parities when running your protocol. Please also report the accuracy of your protocol, and tell whether there is an accuracy loss compared to the plaintext algorithm and explain the reason. (10 Marks)

Appendix: Arithmetic Shares to Yao's Shares (A2Y)

In the lecture, we introduced mixed protocols for efficiently realising complicated computation tasks. For example, when computing K-Means in the assignment, sorting is a necessary operation after computing distances. However, we cannot directly sort arithmetic shares since they are randomised values. Thus, we need to seek an alternative method that can sort the distances without revealing it. Intuitively, Yao's Garbled Circuit will help achieve this goal. In the lecture, we have introduced the A2Y technique that can convert the arithmetic shares to garbled circuit labels (i.e., Yao's shares). The emp-toolkit provides the corresponding class/functions to enable the A2Y method. Particularly, we can use the **Integer** class of emp-toolkit (https://github.com/emp-toolkit/emp-tool/blob/master/emp-tool/circuits/integer.h) to achieve our goal.

Assuming that we have already shared a value *a* to two-servers as $<a>_0$ and $<a>_1$, we can use the constructor of the **Integer** class to convert the arithmetic share as well as the field size (e.g., 2^{32}) to Yao's share. Then, ***the overridden operators (+, %)*** can be invoked to compute the label of *a* *(*$(<a>_0 + <a>_1)$ % Field size*)*. Finally, the sorting algorithm can be executed over Yao's garbled circuits (GC).

N.B. In week6 tutorial, we were using https://github.com/emp-toolkit/emp-sh2pc/blob/master/test/example.cpp as an example to demonstrate how to use emp-toolkit under the semi-honest two-server setting. Please refer to this again when you design a GC-based program.