Starting and Editing the Price Finder

Lukas Helminger, Fabian Schmid

January 2020

Introduction

Over the past half-year, we developed a multi-party computation (MPC) use case. We discussed the use case more thoroughly in the project handout. Hence, we direct anyone there interested in the inner workings of the use case and the framework we used.

In this guide, we will focus on quick installation, running, and editing of the project.

Installation

As mentioned in the introduction, we have a framework dependency. It is, however, not necessary to edit the framework dependent code to redefine the pricing function. If the framework code stays unchanged, the last stable release can be used. The project can be installed by the following steps:

```
cd <project-directory>
mvn clean install
```

This will install the project and its dependencies into the local maven repository. After that (or any subsequent changes to the program), use the following command.

```
make install
```

After that, the executable jar file can be found in `Application/target/demo.jar`
For a guide on how to install the newest FRESCO release as a basis please refer to the project handout.

Running the Demo

The project comes with several predefined demonstrator setups. First, let us give a quick introduction.

Since this is a Multi Party Computation, we have to run several parties at once. All the demos are run in parallel on the same machine. To seperate the individual log output, for each process a subfolder is created in the servers directory. Note that the Host (i.e., the vendor or infineon) always has to have id equal to one. All the demo executions can be found in the Makefile.

move:

```
mkdir -p servers;
cd servers;
mkdir -p servers/server1;
cp Application/target/demo.jar servers/server1;
```

After this has been done for each party, each party can be executed in the following way:

```
cd servers/server1 && java -jar demo.jar -l -i 1 -H
-p 1:localhost:8081 -p 2:localhost:8082 -p 3:localhost:8083
-Dspdz.modBitLength=128 -Dspdz.maxBitLength=10 -Dspdz.preprocessingStrategy=DUMMY
-A 1 -T 1 --price 5 --volume 100 > log.txt 2
```

- The first line enters the directory and executes the program. Then, it sets logging as active (deactivate by deleting `-l`). With `-i 1` the id is set to one. This id has to be unique. With `-H` it is declared that this user is in fact the host of the protocol.
- In the second line, all parties are specified, here localhost has to be replaced by the specific IP addresses of the clients. It is always necessary to specify your own address, so the network manager can listen to the correct port.
- The third line specifies the security parameters of FRESCO. Here the first two parameters should be kept the same, while the last one can also be set to `Strategy=MASCOT`. **Caution:** The protocol can only fulfill the security claims, if MASCOT is used as a preprocessing strategy. Dummy should only be used to test the implementation.
- In the final line there are the use case specific parameters. `-A` specifies the amount, this feature is deactivated in this version. `-T` states the date, when the amount ordered is due. `--price` specified the suggested price for the amount. `--volume` specifies the amount requested in the transaction. The final statement piped the output into the logfile.

Modifying the Pricing Function

The main entry point for any changes should always be the PriceFinder class in the Application package.

First the client input is aggregated. In the JSON file which is in the servers/server1 folder, there is the list of items which the host provides. For each date in that list, we sum up all client input that is due until then. The remainder is then added to the next entry from the JSON file. After this sequential summation is done, the aggregate values are opened to the host. In other words, the host only knows how much is requested at each of the dates he proposes.

In the PriceFinder class there is a priceFinder function which will be executed in the end of the protocol. There one has access to the ATPManager object which holds all the above mentioned information. In essence, there are three different lists, which are of interest.

- DEBUGMinCost: For each offer by infenion where there is actually a requested amount, the minimum price is stored in this list. More precisely, the amount requsted up to this point times the price defined in the JSON file.
- DEBUGCost: This defines the aggregate price proposed by the clients. I.e., the volume requested times their individual price suggested.
- DEBUGLeftOver: The amount of volume left after each date in the JSON. There is currently a minor issue in FRESCO, where values opened to just one party have no sign conversion. Hence, negative values are represented as huge integer values. At the moment there is just a hotfix in the priceFinder function since FRESCO aims to solve this problem in the next release.

To see exactly which functions are executed on which clients, one has to take a look at the `IFXMPC` and `IFXHOST` classes. There in the build computation functions of the application interface the main routines are called. For further information on this please refer to the Project Handout.