Project 2: PSI protocol

This project is due on **March 6, 2023** at **11:59pm CST**. We strongly recommend that you get started early. You will get 80% of your total points if you turn in up to 72 hours after the due date. Late work will not be accepted after 72 hours past the due date.

The project is split into two parts. Checkpoint 1 helps you to get familiar with the language and tools you will be using for this project. You do NOT need to make a separate submission for each checkpoint. That is, the questions for the checkpoint 2 are within Coursera, and the code for checkpoint 1 should be pushed to your personal repository by **March 6, 2023** at **11:59pm CST**. Detailed submission requirements are listed at the end of the document.

This is an individual project; you SHOULD work **individually**.

The code and other answers you submit must be entirely your own work, and you are bound by the Student Code. You MAY consult with other students about the conceptualization of the project and the meaning of the questions, but you MUST NOT look at any part of someone else's solution or collaborate with anyone else. You may consult published references, provided that you appropriately cite them (e.g., with program comments), as you would in an academic paper.

Solutions MUST be submitted electronically in the Github directory, following the submission checklist given at the end of each checkpoint. Details on the filename and submission guideline is listed at the end of the document.

Release date: February 6, 2023
Checkpoint 1 Recommended Due date: February 27, 2023
Checkpoint 2 Due date: March 6, 2023 at 11:59pm CST

Introduction

Cybersecurity incidents are an ever-growing problem for companies. Recently, the President of the United States called for steps to improve cybersecurity, including new legislation that would limit liabilities of companies that report incidents that had happened to them. In this machine problem, we will study how companies can share cybersecurity incidents with each other without revealing unnecessary information about the incidents.
Each group represents a company that has experienced 10 standard cybersecurity incidents. You seek other companies that have experienced incidents similar to the ones you have experienced. You want to learn whether other companies have experienced similar incidents, but without sharing the set of all incidents you have experienced.
You are given a list of cybersecurity incidents in the file `incident_list.txt` labeled from 1 to 25. In this machine problem, each group will select exactly 10 cybersecurity incidents randomly from the list. You will use a private set intersection (PSI) protocol to see if other groups have experienced the same incidents. Each run of the protocol requires exactly two groups: a Client group and a Server group.

Please read the assignment carefully before starting the implementation.

Objectives

- Understand the PSI protocol.
- Be able to implement part of the PSI protocol.
- Gain familiarity with Crypto APIs in Java.

Checkpoints

This machine problem is split into 2 checkpoints. Checkpoint 1 would help you get familiar with the PSI protocol. You will need to implement part of the protocol and test `Client.java` and `Server.java` on your own machine. In Checkpoint 2, you will run the protocol with your own machine in a few different settings and answer a few questions about your results.

Implementations

You are provided with a complete implementation of the Client (`ClientPhase1.java`, `ClientPhase2.java`), a partial implementation of the Server (`Server.java`), and some useful classes (`Inputs.java`, `Paillier.java`, and `StaticUtils.java`). We will guide you through the implementation of server's side of the PSI protocol in `Server.java`.

2.1 Checkpoint 1 (80 points)

This machine problem is split into 2 checkpoints. Checkpoint 1 would help you get familiar with the PSI protocol. You will need to implement part of the protocol and test `Client.java` and `Server.java` on your own machine.

2.1.1 Java 8 (0 points)

Java 8 is needed to run the code for this assignment. Check your Java environment with command `java -version` and `javac -version`. Please make sure you have Java 8 installed on your machine.

Reference

- JRE and JDK Installation (Java 8). `http://docs.oracle.com/javase/8/docs/technotes/guides/install/install_overview.html`

What to submit No submission required.

2.1.2 Implement the PSI Server (80 points)

In this section, you need to complete the implementation of the PSI Server and test it on your own machine.

2.1.2.1 Implement `Server.java`

Please place your code in the file `Server.java` starting at line 26. Detailed instructions are included as comments in the file.

Tips:

- Please refer to the slides of the Crypto Constructs lecture for more details on the PSI protocol. Note that you only need to implement the server's side of the protocol.
- Please use the provided class `Paillier.java` for the homomorphic operations.
- Please use the provided method `randomBigInt(BigInteger n)` to generate r_i mentioned in the slides.

2.1.2.2 Run the PSI protocol

In this section, you can test the protocol within yourself and act as a Server and a Client at the same time.
Before start running the protocol, please perform the following **initial steps**:

1. Compile the code using
   - Unix/Linux/Mac: `javac -cp .:flanagan.jar *.java`
   - Windows: `javac -cp ".;flanagan.jar" *.java` (note: if you are getting an error saying package flanagan.math does not exist, try javac -cp .;flanagan.jar *.java)
2. Select exactly 10 unique cybersecurity incidents from the list in file `incident_list.txt`.
3. Create another text file called `client_inputs.txt`, put the labels of the 10 incidents you selected in step 2, and separate the labels by new lines. See example `example_inputs.txt` to understand the format. DO NOT use the example `example_inputs.txt` file.
4. Repeat the above steps for the server, and create another text file called `sever_inputs.txt`. It is possible that `client_inputs.txt` and `server_inputs.txt` may share some common incidents.

The **Client** needs to perform the following steps:

1. Run the following program (Use your netid and the `client_inputs.txt` you generated in the last step as input):
   - Unix/Linux/Mac: `java -cp .:flanagan.jar ClientPhase1 <client_inputs.txt> <netid>`
   - Windows: `java -cp ".;flanagan.jar" ClientPhase1 <client_inputs.txt> <netid>`
2. This will generate two files with filename being `netid` and `ClientPK.out`
3. Email both of these files to the Server (you don't need to send an actual email, just put them in the Server's folder).
4. Wait for the server to run its program and send you the file `netid.out`.
5. After you receive the file from Server named `netid.out`, put it in the same directory as the code and run the following program:
   - Unix/Linux/Mac: `java -cp .:flanagan.jar ClientPhase2 <client_inputs.txt> <netid.out>`
   - Windows: `java -cp ".;flanagan.jar" ClientPhase2 <client_inputs.txt> <netid.out>`
6. The program will output the common incidents.

The **Server** needs to perform the following steps (Note: You must implement the server PSI sub-protocol correctly in order to perform the following steps):

1. Put the files (`ClientPK.out` and `netid`) received from the Client in the same directory as the code.
2. Run the following program (Use the `server_inputs.txt` you generated in the last step as input) :
   - Unix/Linux/Mac: `java -cp .:flanagan.jar Server <server_inputs.txt> <netid>`
   - Windows: `java -cp ".;flanagan.jar" Server <server_inputs.txt> <netid>`
3. This will generate a file with the filename `netid.out`
4. Email `netid.out` back to the Client.

The output file at Client in Step 6 should contain all the common incidents between `client_inputs.txt` and `server_inputs.txt`. You may test your code with different sets of incidents.

What to submit

- Place `Server.java`, `server_inputs.txt`, and `client_inputs.txt` in `mp2`.

2.2 Checkpoint 2 (10 points)

For this checkpoint, you'll need to test the PSI protocol on your own machine while considering both honest and malicious actors.

2.2.1 Test the Protocol

In the following tests, you should perform the role of the Client or Server at least once.

2.2.1.1 Honest Client and Honest Server

In this setting, both Client and Server group behave honestly and help each other to find the common incidents.

2.2.1.2 Malicious Client and Honest Server

In this setting, Client is ill intentioned and wants to learn as much information about the Server incidents as it can. In this case, Client is allowed to choose any incidents he wants and can run the protocol with the Server multiple times with different list of incidents.

2.2.1.3 Honest Client and Malicious Server

In this setting, Server is ill intentioned and wants to learn as much information about the Client incidents as it can. Client is honest.

What to submit No submission required.

2.2.2 Answer Questions (10 points)

For this part, please go to Coursera Week 4: "MP2: Private Set Intersection" to answer all the questions. **Please note that you can make an one-time submission only**.

What to submit

- Please go to Coursera to submit your answers.

Submission Checklist

Place the following files in `mp2` in your Github repository.

- `Server.java` (80 points) [Implementation of the PSI server]
- `server_inputs.txt`

- `client_inputs.txt`