PRIVACY ENHANCING TECHNOLOGIES – ASSIGNMENT # 2

“Secure Multiparty Computation for Privacy in Practice”

The goal of this project is to explore some of the Secure Multiparty Computation (SMC) techniques in practice. Specifically, you will compute and compare different protocols for Private Set Intersection (PSI).

The assignment is organized in 4 steps, where each step should be documented and analysed in the form of a final report. The report should follow the usual organization, with an introduction and conclusion at the extremes, with the required analysis for the 4 steps in between.

Step #1 - Protocols for Private Set Intersection: Study, Description and Comparison

In this assignment you will compare different protocols for Private Set Intersection (PSI). You will be using the program PSI from the Cryptography and Privacy Engineering Group at TU Darmstadt and wireshark, the network analyzer. PSI is a small program which implements four different Private Set Intersection (PSI) protocols:

1. a naive hashing solutions where elements are hashed and compared
2. the server-aided protocol of [4]
3. the Diffie-Hellman-based PSI protocol of [5]
4. the OT-based PSI protocol of [3]

Install wireshark (for next step) and obtain the PSI program and experiment with the *demo.exe* using each of the four protocols (only works on linux).

Note: Installing the PSI program requires old versions of software that may be cumbersome to install nowadays. You can try the following, or follow the instructions on the git repository[1]:

```
sudo apt install -y g++ make libgmp-dev libglib2.0-dev libssl-dev
git clone –recursive https://github.com/eduardo010174/PSI
cd PSI
make
```

Check how to execute the PSI code.

All four protocols should output the 3 intersecting elements: Michael.Zohner@ec-spride.de, Evelyne.Wagener@tvcablenet.be and Ivonne.Pfisterer@mail.ru.

Your first task is to study, describe and compare the four protocols in the report. Consult references [3, 4, 5] as bibliography for this step.

Step #2 - Experimentation with Protocols for Private Set Intersection

With this assignment you should have received two files: AppList0.csv and AppList1.csv. Each of these files is a list of applications installed in the smartphones of two anonymous users.

1 Alternatively, you can obtain the project with executable files already compiled in linux (may not work in every OS).

The following steps shall be taken and documented to be delivered in the form a report:

1. We will start by exploring the naive hashing solution. In this approach, each party hashes the inputs and sends the hashes to the other parties. Each party then computes the intersection with its own hashed input to find the intersections.

   Start by creating a text file named *input* with a single line with your student number (or name). Open two terminals and execute one of the following commands in each terminal: (run the receiver first -- option -r 0)

   ```
    ./demo.exe -r 0 -p 0 -f input
   ./demo.exe -r 1 -p 0 -f input
   ```

   The output should have:

   ```
   Computation finished. Found 1 intersecting elements:
   <your student number/name>
   ```

   In both terminals you should also see that the hashes of the inputs of both parties are the same. Also notice from the code that these hashes are the only data that is sent and received from the sockets (see line 86 in src/naive-hashing/naive-psi.cpp).
   Note: The default hashing algorithm used is the sha256 truncated to the 6 most significant bytes. You can verify this in linux with the command:

   ```
   echo -n “your student number/name” | sha256sum | cut -c -12
   ```
2. Create another input file, named *input2*, and write the single word “test” inside. In one terminal execute:

   ```
   ./demo.exe -r 0 -p 0 -f input
   ```

   and in the other execute:

   ```
   ./demo.exe -r 1 -p 0 -f input2
   ```

   Notice that the hashes are different and thus no intersection is found.
3. Using wireshark, check the tcp packets that are being sent between the sender and receiver. The communications can be sniffed in the loopback interface and the default port used is the 7766 (use the filter tcp.port == 7766).
   Re-run points 1. and 2. while capturing the packets with wireshark. You should find 2 packets having 6 bytes of data (they should be the last 2 packets before the connection termination phase), one from source port 7766 and other with destination port 7766. Check the data inside the packets and notice that it is the exchanged hashes.
   Note: The default hashing algorithm used is the sha256 truncated to the 6 most significant bytes. You can verify this in linux with the command:

   ```
   echo -n “your student number/name” | sha256sum | cut -c -12
   ```
4. Considering the previous three points, discuss in the report on why the naive hashing protocol might be insecure.
5. Start a new capture in wireshark and using the list of apps AppList1.csv, run the naive hashing protocol against AppList2.csv. Save the output of the intersection to a file and register it in the report. Additionally, register the number of captured packets and the total captured bytes. You can get the total captured bytes in wireshark by going to: Statistics > Capture File Properties.
6. We will now explore the server-aided protocol of [4] (option -p 1). This protocol assumes a third-party that is responsible to calculate the intersection.

Note: the code for this protocol has a bug in where it requires that the input files have the same size. Thus, check which list of apps (either AppList1.csv or AppList2.csv) has less apps and add lines to the smallest list until the number of lines is the same. You can add fake emails from the list in PSI/sample_sets/emails_alice.txt, for instance.

By using two machines, run the server (option -r 0, where r stands for role) in one of the machines by executing:

```
./demo.exe -r 0 -p 1 -f README.md -a <local server ip>
```

The input file (-f) is required even though the server does not use it (program bug). So we send any existing file such as README.md

In the other machine, in two different terminals run the clients (-r 1):

```
./demo.exe -r 1 -p 1 -f AppList1.csv -a <local server ip>
./demo.exe -r 1 -p 1 -f AppList2.csv -a <local server ip>
```

Check that the intersection in the client terminals is equal to the output from the naive protocol (obtained in step 5.). Register in your report the number of captured packets and the total captured bytes. You can get the total captured bytes in wireshark by going to Statistics > Capture File Properties.

7. In the server's output from the previous point, the data received from the clients is displayed. You should also note that this data is the same hashes as when using the naive protocol (in point 5). Given these aspects, discuss in the report on what privacy issues arise from the use of a server as a third-party.
8. Explore the Diffie-Hellman-based PSI protocol: Start a new capture in wireshark and perform the PSI between both app lists using the Diffie-Hellman-based PSI protocol of [5] (option -p 2). Notice that in this protocol, the terminal with -r 0 does not compute the intersection. Confirm that the intersection is the same as with the previous protocols. Register in your report the number of captured packets and the total captured bytes. You can get the total capture bytes in wireshark by going to: Statistics > Capture File Properties.
9. Explore the OT-based PSI protocol: Repeat the previous step using the OT-based PSI protocol of [3] (option -p 3). Notice that in this protocol, the terminal with -r 0 does not compute the intersection. Register the number of captured packets and the total captured bytes.
10. Look at the number of captured packets and the total captured bytes for the four protocols (captured in points 5., 6., 8. and 9.) and analyse them in your report from a security versus communication cost point-of-view.

Step #3 - Benchmarking Private Set Intersection Protocols

In this step you will benchmark each of the four protocols with respect to the execution time and exchanged data. To do that, you can use the program psi.exe that should be in the root of the project.

For the benchmark, you will run the *psi.exe* executable in two terminals as:

```
./psi.exe -r 0 -p <protocol> -b 16 -n <number of elements>
./psi.exe -r 1 -p <protocol> -b 16 -n <number of elements>
```

For the protocol (option -p), you will compare protocols 0, 2 and 3 (protocol 1 has a bug when benchmarking, so we won't run it). Differently from *demo.exe*, *psi.exe* does not receive an input file. Instead it creates the input given a number of elements (-n). For the number of elements, you should choose a range that allows you to test the effect for smaller datasets of dozens or hundreds of elements (with smaller steps), as well as larger datasets up to say 100000 elements (with larger steps). At the end of each execution, the terminal will give you the following outputs: required time, data sent and data received. Create an table to register these values for each of your runs and create the following plots:

- create a plot with three lines, one for each of the protocols, where the x axis is the set size and the y axis is the required time;
- create a plot with three lines, one for each of the protocols, where the x axis is the set size and the y axis is the total data exchanged (data sent + data received);

Add the table and the plots to your report and discuss the results.

Step #4 - Apply secure intersection protocols to another dataset of your choice

Define other setup/datasets, and obtain an instance for each of the group members to determine joint aspects. Some possibilities/suggestions to explore:

- list of common smartphone contacts
- common URLs of browsing history
- lists of shared clients from different companies/hotels
- detection of leaked passwords (against public lists)
- verification of clients of a gym vs list of infected people from public authorities

If you resort to personal datasets, make sure to remove any privacy-sensitive information from the datasets before including info about them in the report.

You should now apply PSI mechanisms to achieve the goal of determining common information from the setup/datasets you have defined. This should allow you to achieve the goal without compromising privacy on both ends. Make sure that you have some intersections with your group partner.

In your report, describe the input sets and intersection and justify the choice of the PSI mechanisms you have considered based on the analysis performed in previous steps.

FINAL REPORT

Write a final report that should include: the description and comparison of the four PSI protocols; discussions/analysis from points 4., 7. and 10. of step #2; the benchmark and analysis of the three PSI protocols, including the two graphs, as requested in step #3; the results of step #4, including the justification on the selection of the data type used in this step.

Submit the PDF report at the course moodle.

Final deadline: **5th May 2024.**

Assignment defenses: classes of **May 17 (TBC)**.

Evaluation Criteria

- Study, description and comparison of the four secure set intersection protocols [25%]
- Critical analysis on: [25%]
    - Why the naive hashing protocol might be insecure
    - What privacy issues arise from the use of a server as a third-party
    - Privacy and security versus communication cost
- Benchmarking, evaluation and critical analysis of the three secure set intersection protocols [25%]
- Apply secure intersection protocols to another dataset of your choice [10%]
- Assignment defense [15%]

References

[1] B. Pinkas, T. Schneider, M. Zohner. Faster Private Set Intersection Based on OT Extension. USENIX Security 2014: 797-812. Full version available at http://eprint.iacr.org/2014/447.

[2] B. Pinkas, T. Schneider, G. Segev, M. Zohner. Phasing: Private Set Intersection using Permutation-based Hashing. USENIX Security 2015. Full version available at http://eprint.iacr.org/2015/634.

[3] B. Pinkas, T. Schneider, M. Zohner. Scalable Private Set Intersection Based on OT Extension. Available at http://eprint.iacr.org/2016/930.

[4] S. Kamara, P. Mohassel, M. Raykova, and S. Sadeghian. Scaling private set intersection to billion-element sets. In Financial Cryptography and Data Security (FC'14) , LNCS. Springer, 2014.

[5] C. Meadows. A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. In IEEE S&P'86, pages 134–137. IEEE, 1986.