



FACULDADE DE  
CIÊNCIAS E TECNOLOGIA  
UNIVERSIDADE DE  
COIMBRA



departamento  
de engenharia informática  
1995 – 2020

SEGURANÇA E PRIVACIDADE

---

# Secure Multiparty Computation

---

MESTRADO EM ENGENHARIA E CIÊNCIA DE DADOS

Mariana Lopes Paulino - 2020190448

Rui Alexandre Coelho Tapadinhas - 2018283200

Coimbra - Portugal

2023/2024

# Index

<b>1. Introduction</b>	<b>3</b>
<b>2. Problem</b>	<b>3</b>
<b>3. Dataset</b>	<b>3</b>
<b>4. Protocols</b>	<b>3</b>
4.1. Naïve Hashing Protocol . . . . .	3
4.2. Testing . . . . .	4
4.2.1. Dataset with 100.000 entries: . . . . .	4
4.2.2. Dataset with 200.000 entries: . . . . .	4
4.2.3. Dataset with 300.000 entries: . . . . .	4
4.2.4. Dataset with 400.000 entries: . . . . .	4
4.2.5. Dataset with 500.000 entries: . . . . .	5
4.2.6. Dataset with 600.000 entries: . . . . .	5
4.2.7. Dataset with 700.000 entries: . . . . .	5
4.2.8. Dataset with 800.000 entries: . . . . .	5
4.2.9. Dataset with 900.000 entries: . . . . .	5
4.2.10. Dataset with 1.000.000 entries: . . . . .	5
4.3. Diffie-Hellman Protocol . . . . .	6
4.4. Testing . . . . .	6
4.4.1. Dataset with 100.000 entries: . . . . .	6
4.4.2. Dataset with 200.000 entries: . . . . .	6
4.4.3. Dataset with 300.000 entries: . . . . .	6
4.4.4. Dataset with 400.000 entries: . . . . .	7
4.4.5. Dataset with 500.000 entries: . . . . .	7
4.4.6. Dataset with 600.000 entries: . . . . .	7
4.4.7. Dataset with 700.000 entries: . . . . .	7
4.4.8. Dataset with 800.000 entries: . . . . .	7
4.4.9. Dataset with 900.000 entries: . . . . .	7
4.4.10. Dataset with 1.000.000 entries: . . . . .	8
4.5. Oblivious Transfers . . . . .	8
4.6. Testing . . . . .	8
4.6.1. Dataset with 100.000 entries: . . . . .	9
4.6.2. Dataset with 200.000 entries: . . . . .	9
4.6.3. Dataset with 300.000 entries: . . . . .	9
4.6.4. Dataset with 400.000 entries: . . . . .	10
4.6.5. Dataset with 500.000 entries: . . . . .	10
4.6.6. Dataset with 600.000 entries: . . . . .	10
4.6.7. Dataset with 700.000 entries: . . . . .	11
4.6.8. Dataset with 800.000 entries: . . . . .	11
4.6.9. Dataset with 900.000 entries: . . . . .	11
4.6.10. Dataset with 1.000.000 entries: . . . . .	12
4.7. Server Aided Protocol . . . . .	12
4.8. Testing Server Aided . . . . .	12
4.8.1. Dataset with 100.000 entries: . . . . .	12
4.8.2. Dataset with 200.000 entries: . . . . .	13

4.8.3. Dataset with 300.000 entries: . . . . .	13
4.8.4. Dataset with 400.000 entries: . . . . .	13
4.8.5. Dataset with 500.000 entries: . . . . .	13
4.8.6. Dataset with 600.000 entries: . . . . .	13
4.8.7. Dataset with 700.000 entries: . . . . .	13
4.8.8. Dataset with 800.000 entries: . . . . .	14
4.8.9. Dataset with 900.000 entries: . . . . .	14
4.8.10. Dataset with 1.000.000 entries: . . . . .	14
<b>5. Protocol Comparison</b>	<b>14</b>
5.1. Comparisons . . . . .	15
5.1.1. Time and Exchanged Data Comparisons - Demo Executable File . . . . .	15
5.1.2. Time and Exchanged Data Comparisons - PSI Executable File . . . . .	16
<b>6. Conclusion</b>	<b>17</b>
<b>7. References</b>	<b>18</b>

## 1. Introduction

Private Set Intersection is the concept related to finding the intersection between two or more private sets of data while maintaining the non-disclosure of each private dataset to the other party.

In this assignment, we were asked to use the PSI software [1] to test and benchmark the implemented protocols for Private Set Intersection in the proposed software, with a dataset we created.

The protocols used were the naive hashing protocol, the server-aided protocol, the Diffie-Hellman-based PSI protocol and the OT-based protocol.

## 2. Problem

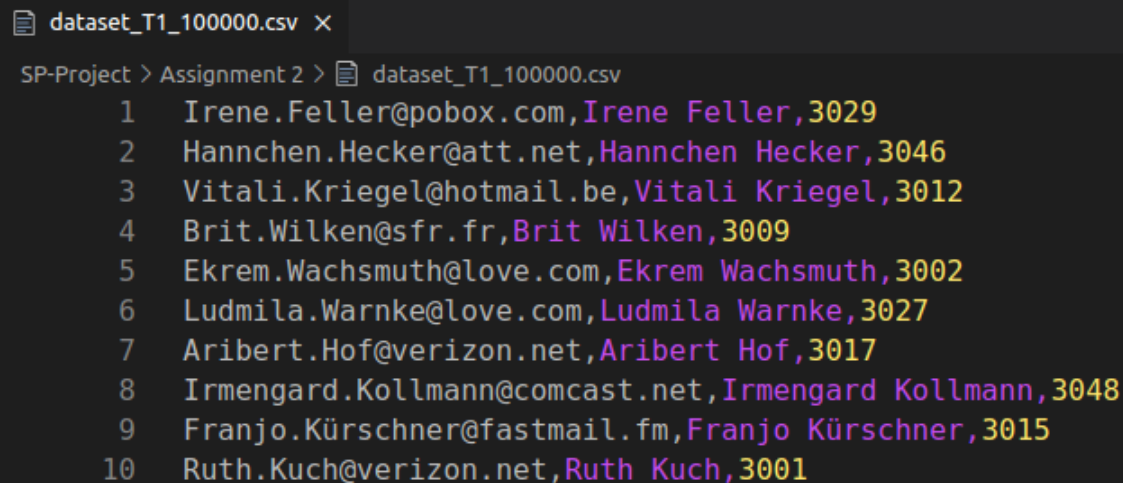
We will use the PSI protocols to test and benchmark the intersection between two datasets of two marketing departments from telecommunication companies.

## 3. Dataset

The datasets used in this assignment were generated with the help of a script present in the PSI script that was altered by us to add postal codes.

To aid with the generation of the dataset, there were 3 files containing a list of 2003 distinct given names, a list of 3422 distinct family names and 107 distinct email providers. Our script chooses a random given name, family name, email provider and a number between 3000 and 3050 to represent a random postal code in this specific interval, so there are only 51 distinct possible values. The only argument required to the script is the desired size for the dataset.

We chose to create 10 datasets of sizes starting from 100.000 increasing 100.000 each time until it reaches 1.000.000. The datasets respect the following example:



```
dataset_T1_100000.csv x
SP-Project > Assignment 2 > dataset_T1_100000.csv
1 Irene.Feller@pobox.com,Irene Feller,3029
2 Hannchen.Hecker@att.net,Hannchen Hecker,3046
3 Vitali.Kriegel@hotmail.be,Vitali Kriegel,3012
4 Brit.Wilken@sfr.fr,Brit Wilken,3009
5 Ekrem.Wachsmuth@love.com,Ekrem Wachsmuth,3002
6 Ludmila.Warnke@love.com,Ludmila Warnke,3027
7 Aribert.Hof@verizon.net,Aribert Hof,3017
8 Irmengard.Kollmann@comcast.net,Irmengard Kollmann,3048
9 Franjo.Kürschner@fastmail.fm,Franjo Kürschner,3015
10 Ruth.Kuch@verizon.net,Ruth Kuch,3001
```

## 4. Protocols

### 4.1. Naïve Hashing Protocol

The **Naïve Hashing Protocol** is one of the simplest Private Set Intersection Protocol where it simply hashes each element in each dataset after the two individuals agree on a hash function and then the resulting hashes are compared. After this comparison, if the hashes are equal, then, the elements must be in the intersection of the two sets. This protocol is not secure against malicious

parties, since they can learn information about the other part's set by carefully if they choose the hashes they submit, and, it can turn out to be slow to compute the intersection of many elements.

## 4.2. Testing

To test this protocol, we used the executable file *demo* with the following commands:

- `./demo.exe -r 0 -p 0 -f /pathToFile/dataset_T1_[size].csv -a 127.0.0.1 -t`
- `./demo.exe -r 1 -p 0 -f /pathToFile/dataset_T2_[size].csv -a 127.0.0.1 -t`

The flag **r** defines the role of each process, the flag **p** defines the protocol (0 representing the Naive Hashing protocol), **a** defines the address and **t** represents the flag to show the time and data sent and received.

### 4.2.1. Dataset with 100.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 0 -f
Time for reading elements:      0.03 s
Computation finished. Found 4 intersecting elements:
Required time: 0.1 s
Data sent:      1.0 MB
Data received:  1.0 MB
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$
```

### 4.2.2. Dataset with 200.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 0 -f
Time for reading elements:      0.05 s
Computation finished. Found 9 intersecting elements:
Required time: 0.1 s
Data sent:      1.9 MB
Data received:  1.9 MB
```

### 4.2.3. Dataset with 300.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 0 -f
Time for reading elements:      0.07 s
Computation finished. Found 26 intersecting elements:
Required time: 0.2 s
Data sent:      2.9 MB
Data received:  2.9 MB
```

### 4.2.4. Dataset with 400.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 0 -f
Time for reading elements:      0.08 s
Computation finished. Found 37 intersecting elements:
Required time: 0.2 s
Data sent:      3.8 MB
Data received:  3.8 MB
```

#### 4.2.5. Dataset with 500.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 0 -f
Time for reading elements:      0.08 s
Computation finished. Found 60 intersecting elements:
Required time: 0.3 s
Data sent:      4.8 MB
Data received:  4.8 MB
```

#### 4.2.6. Dataset with 600.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 0 -f
Time for reading elements:      0.09 s
Computation finished. Found 105 intersecting elements:
Required time: 0.3 s
Data sent:      5.7 MB
Data received:  5.7 MB
```

#### 4.2.7. Dataset with 700.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 0 -f
Time for reading elements:      0.09 s
Computation finished. Found 125 intersecting elements:
Required time: 0.3 s
Data sent:      6.7 MB
Data received:  6.7 MB
```

#### 4.2.8. Dataset with 800.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 0 -f
Time for reading elements:      0.09 s
Computation finished. Found 172 intersecting elements:
Required time: 0.4 s
Data sent:      7.6 MB
Data received:  7.6 MB
```

#### 4.2.9. Dataset with 900.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 0 -f
Time for reading elements:      0.12 s
Computation finished. Found 242 intersecting elements:
Required time: 0.5 s
Data sent:      8.6 MB
Data received:  8.6 MB
```

#### 4.2.10. Dataset with 1.000.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 0 -f
Time for reading elements:      0.13 s
Computation finished. Found 241 intersecting elements:
Required time: 0.6 s
Data sent:      9.5 MB
Data received:  9.5 MB
```

### 4.3. Diffie-Hellman Protocol

The **Diffie-Hellman** based Private Set Intersection Protocol is a cryptographic protocol that allows two parties, such as A and B to compute the intersection between their datasets without revealing their individual sets to each other. In this protocol, A and B choose their private keys. Then, each of them calculates their public keys and sends it to each other. Now, both A and B have each other's public key to encrypt the data before sending, and have their own secret key that is not shared, to decrypt the data sent.

In this protocol, the private contacts shared are available to the party that computes the intersection.

### 4.4. Testing

To test this protocol, we used the executable file *demo* with the following commands:

- `./demo.exe -r 0 -p 2 -f /pathToFile/dataset_T1_[size].csv -a 127.0.0.1 -t`
- `./demo.exe -r 1 -p 2 -f /pathToFile/dataset_T2_[size].csv -a 127.0.0.1 -t`

The flag **r** defines the role of each process, the flag **p** defines the protocol (0 representing the Naive Hashing protocol), **a** defines the address and **t** represents the flag to show the time and data sent and received.

#### 4.4.1. Dataset with 100.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 2 -f
Time for reading elements:      0.04 s
Computation finished. Found 100000 intersecting elements:
Required time:  77.0 s
Data sent:      3.5 MB
Data received:  6.6 MB
```

#### 4.4.2. Dataset with 200.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 2 -f
Time for reading elements:      0.03 s
Computation finished. Found 4 intersecting elements:
Required time:  75.9 s
Data sent:      3.5 MB
Data received:  6.6 MB
```

#### 4.4.3. Dataset with 300.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 2 -f
Time for reading elements:      0.04 s
Computation finished. Found 11 intersecting elements:
Required time:  157.9 s
Data sent:      7.1 MB
Data received:  13.2 MB
```

#### 4.4.4. Dataset with 400.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 2 -f
Time for reading elements:      0.06 s
Computation finished. Found 49 intersecting elements:
Required time: 403.8 s
Data sent:      17.6 MB
Data received:  32.9 MB
```

#### 4.4.5. Dataset with 500.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 2 -f
Time for reading elements:      0.06 s
Computation finished. Found 25 intersecting elements:
Required time: 237.4 s
Data sent:      10.6 MB
Data received:  19.7 MB
```

#### 4.4.6. Dataset with 600.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 2 -f
Time for reading elements:      0.07 s
Computation finished. Found 42 intersecting elements:
Required time: 313.7 s
Data sent:      14.1 MB
Data received:  26.3 MB
```

#### 4.4.7. Dataset with 700.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 2 -f
Time for reading elements:      0.08 s
Computation finished. Found 105 intersecting elements:
Required time: 470.4 s
Data sent:      21.2 MB
Data received:  39.5 MB
```

#### 4.4.8. Dataset with 800.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 2 -f
Time for reading elements:      0.10 s
Computation finished. Found 123 intersecting elements:
Required time: 566.7 s
Data sent:      24.7 MB
Data received:  46.1 MB
```

#### 4.4.9. Dataset with 900.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 2 -f
Time for reading elements:      0.12 s
Computation finished. Found 133 intersecting elements:
Required time: 646.1 s
Data sent:      28.2 MB
Data received:  52.6 MB
```



#### 4.4.10. Dataset with 1.000.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 2 -f
Time for reading elements:      0.12 s
Computation finished. Found 253 intersecting elements:
Required time: 736.4 s
Data sent:      31.8 MB
Data received:  59.2 MB
```

### 4.5. Oblivious Transfers

The **Oblivious Transfers** based Private Set Intersection Protocol is based on a cryptographic primitive that allows information exchange from the two parties without revealing anything, about the information that is being swapped between part A and B. This tool is a powerful one to be used in privacy preserving computation as they can be used in the implementation of protocols that allow the two users (A and B) to compute functions of their data sets without making information public to the other one.

In an OT-based PSI protocol, the parties use OTs to exchange messages that allow them to determine whether their elements are in the intersection of their sets without revealing their individual elements.

In this protocol a sender, for the purpose of example, A can send one of potentially many pieces of information to a receiver B but still remains oblivious about which piece was transferred. B can learn the content of the transferred piece without revealing to A which piece he has received. For this to happen, both parties need to agree on the protocol.

To conclude, this protocol can be more useful for large datasets than the Naïve Hashing one, apart from being more useful it is more secure against malicious parties, and so it is more complex to implement because it ensures that the OTs make it impossible to learn about each other's data, but still allow them to determine whether they have intersected elements or not.

### 4.6. Testing

To test this protocol, we used the executable file *demo* with the following commands:

- `./demo.exe -r 0 -p 3 -f /pathToFile/dataset_T1_[size].csv -a 127.0.0.1 -t`
- `./demo.exe -r 1 -p 3 -f /pathToFile/dataset_T2_[size].csv -a 127.0.0.1 -t`

The flag **r** defines the role of each process, the flag **p** defines the protocol (0 representing the Naive Hashing protocol), **a** defines the address and **t** represents the flag to show the time and data sent and received.

#### 4.6.1. Dataset with 100.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 3 -f
Time for reading elements:      0.02 s
Hashing 100000 elements with arbitrary length into into 10 bytes
Time for domain hashing:       20.47 ms
Time for Cuckoo hashing:       35.20 ms
Client: bins = 120000, elebitlen = 64 and maskbitlen = 80 and performs 120000 OTs
Time for OT extension:        -539463120198519.12 ms
Time for CRF evaluation:       0.00 ms
Time for receiving masks:      155.39 ms
Time for intersecting:         17.39 ms
Computation finished. Found 0 intersecting elements:
Required time: 0.6 s
Data sent:      7.3 MB
Data received:  2.9 MB
```

#### 4.6.2. Dataset with 200.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 3 -f
Time for reading elements:      0.02 s
Hashing 200000 elements with arbitrary length into into 10 bytes
Time for domain hashing:       35.78 ms
Time for Cuckoo hashing:       65.32 ms
Client: bins = 240000, elebitlen = 63 and maskbitlen = 80 and performs 240000 OTs
Time for OT extension:        -539463120118157.12 ms
Time for CRF evaluation:       0.00 ms
Time for receiving masks:      311.56 ms
Time for intersecting:         39.33 ms
Computation finished. Found 3 intersecting elements:
Required time: 1.0 s
Data sent:      14.7 MB
Data received:  5.7 MB
```

#### 4.6.3. Dataset with 300.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 3 -f
Time for reading elements:      0.05 s
Hashing 300000 elements with arbitrary length into into 10 bytes
Time for domain hashing:       52.89 ms
Time for Cuckoo hashing:       94.10 ms
Client: bins = 360000, elebitlen = 62 and maskbitlen = 80 and performs 360000 OTs
Time for OT extension:        -539463120077720.50 ms
Time for CRF evaluation:       0.00 ms
Time for receiving masks:      469.81 ms
Time for intersecting:         60.91 ms
Computation finished. Found 3 intersecting elements:
Required time: 1.4 s
Data sent:      22.0 MB
Data received:  8.6 MB
```

#### 4.6.4. Dataset with 400.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 3 -f
Time for reading elements:      0.07 s
Hashing 400000 elements with arbitrary length into into 10 bytes
Time for domain hashing:       71.30 ms
Time for Cuckoo hashing:       135.97 ms
Client: bins = 480000, elebitlen = 62 and maskbitlen = 80 and performs 480000 OTs
Time for OT extension:        -539463120039110.62 ms
Time for CRF evaluation:       0.00 ms
Time for receiving masks:      607.26 ms
Time for intersecting:         81.50 ms
Computation finished. Found 5 intersecting elements:
Required time:  1.8 s
Data sent:      29.3 MB
Data received:  11.5 MB
```

#### 4.6.5. Dataset with 500.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 3 -f
Time for reading elements:      0.08 s
Hashing 500000 elements with arbitrary length into into 10 bytes
Time for domain hashing:       88.69 ms
Time for Cuckoo hashing:       165.64 ms
Client: bins = 600000, elebitlen = 61 and maskbitlen = 80 and performs 600000 OTs
Time for OT extension:        -539463120009532.44 ms
Time for CRF evaluation:       0.00 ms
Time for receiving masks:      641.00 ms
Time for intersecting:         109.17 ms
Computation finished. Found 7 intersecting elements:
Required time:  2.1 s
Data sent:      36.6 MB
Data received:  14.3 MB
```

#### 4.6.6. Dataset with 600.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 3 -f
Time for reading elements:      0.09 s
Hashing 600000 elements with arbitrary length into into 10 bytes
Time for domain hashing:       104.56 ms
Time for Cuckoo hashing:       201.66 ms
Client: bins = 720000, elebitlen = 61 and maskbitlen = 80 and performs 720000 OTs
Time for OT extension:        -539463119973945.94 ms
Time for CRF evaluation:       0.00 ms
Time for receiving masks:      261.67 ms
Time for intersecting:         129.64 ms
Computation finished. Found 16 intersecting elements:
Required time:  2.5 s
Data sent:      44.0 MB
Data received:  17.2 MB
```

#### 4.6.7. Dataset with 700.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 3 -f
Time for reading elements:      0.13 s
Hashing 700000 elements with arbitrary length into into 10 bytes
Time for domain hashing:       124.32 ms
Time for Cuckoo hashing:       238.50 ms
Client: bins = 840000, elebitlen = 61 and maskbitlen = 80 and performs 840000 OTs
Time for OT extension:        -539463119925419.19 ms
Time for CRF evaluation:       0.00 ms
Time for receiving masks:      394.01 ms
Time for intersecting:         166.31 ms
Computation finished. Found 15 intersecting elements:
Required time:  2.8 s
Data sent:      51.3 MB
Data received:  20.0 MB
```

#### 4.6.8. Dataset with 800.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 3 -f
Time for reading elements:      0.09 s
Hashing 800000 elements with arbitrary length into into 10 bytes
Time for domain hashing:       139.30 ms
Time for Cuckoo hashing:       275.50 ms
Client: bins = 960000, elebitlen = 61 and maskbitlen = 80 and performs 960000 OTs
Time for OT extension:        -539463119894320.69 ms
Time for CRF evaluation:       0.00 ms
Time for receiving masks:      551.10 ms
Time for intersecting:         173.70 ms
Computation finished. Found 14 intersecting elements:
Required time:  3.0 s
Data sent:      58.6 MB
Data received:  22.9 MB
```

#### 4.6.9. Dataset with 900.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 3 -f
Time for reading elements:      0.12 s
Hashing 900000 elements with arbitrary length into into 10 bytes
Time for domain hashing:       152.08 ms
Time for Cuckoo hashing:       320.62 ms
Client: bins = 1080000, elebitlen = 60 and maskbitlen = 80 and performs 1080000 OTs
Time for OT extension:        -539463119831269.19 ms
Time for CRF evaluation:       0.00 ms
Time for receiving masks:      713.22 ms
Time for intersecting:         213.29 ms
Computation finished. Found 27 intersecting elements:
Required time:  3.4 s
Data sent:      65.9 MB
Data received:  25.8 MB
```

#### 4.6.10. Dataset with 1.000.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 3 -f
Time for reading elements:      0.12 s
Hashing 1000000 elements with arbitrary length into into 10 bytes
Time for domain hashing:       175.35 ms
Time for Cuckoo hashing:       362.21 ms
Client: bins = 1200000, elebitlen = 60 and maskbitlen = 80 and performs 1200000 OTs
Time for OT extension:        -539463119798426.69 ms
Time for CRF evaluation:       0.00 ms
Time for receiving masks:      207.08 ms
Time for intersecting:         300.83 ms
Computation finished. Found 35 intersecting elements:
Required time:  3.7 s
Data sent:      73.3 MB
Data received:  28.6 MB
```

### 4.7. Server Aided Protocol

The **Server Aided Protocol** relies on a **centralized trusted party** such as an intermediate server to make the communication between the two other users (A and B) easier. The server is responsible to ensure that the protocol is executed correctly and that the users' privacy is protected.

This protocol is particularly useful in situations where the two parties do not have a connection in common or, they do not trust each other, although they have to trust the server.

Since the server handles much of the complexity, it can be simpler to implement than other protocols, as the information is all centralized. However, relying on a trusted server can create a single point of failure, which can compromise the security of the entire protocol.

### 4.8. Testing Server Aided

To test this protocol, we used the executable file *demo* with the following commands:

- `./demo.exe -r 0 -p 1 -f randomFile.txt -a 127.0.0.1 -t`
- `./demo.exe -r 1 -p 1 -f /pathToFile/dataset_T1_[size].csv -a 127.0.0.1 -t`
- `./demo.exe -r 1 -p 1 -f /pathToFile/dataset_T2_[size].csv -a 127.0.0.1 -t`

The flag **r** defines the role of each process, the flag **p** defines the protocol (0 representing the Naive Hashing protocol), **a** defines the address and **t** represents the flag to show the time and data sent and received.

#### 4.8.1. Dataset with 100.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 1 -f
Time for reading elements:      0.04 s
Computation finished. Found 0 intersecting elements:
Required time:  2.6 s
Data sent:      1.5 MB
Data received:  0.0 MB
```

#### 4.8.2. Dataset with 200.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 1 -f
Time for reading elements:      0.05 s
Computation finished. Found 3 intersecting elements:
Required time:  1.8 s
Data sent:      3.1 MB
Data received:  0.0 MB
```

#### 4.8.3. Dataset with 300.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 1 -f
Time for reading elements:      0.05 s
Computation finished. Found 3 intersecting elements:
Required time:  0.2 s
Data sent:      4.6 MB
Data received:  0.0 MB
```

#### 4.8.4. Dataset with 400.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 1 -f
Time for reading elements:      0.07 s
Computation finished. Found 5 intersecting elements:
Required time:  5.5 s
Data sent:      6.1 MB
Data received:  0.0 MB
```

#### 4.8.5. Dataset with 500.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 1 -f
Time for reading elements:      0.08 s
Computation finished. Found 7 intersecting elements:
Required time:  5.5 s
Data sent:      7.6 MB
Data received:  0.1 MB
```

#### 4.8.6. Dataset with 600.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 1 -f
Time for reading elements:      0.10 s
Computation finished. Found 16 intersecting elements:
Required time:  5.9 s
Data sent:      9.2 MB
Data received:  0.1 MB
```

#### 4.8.7. Dataset with 700.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 1 -f
Time for reading elements:      0.09 s
Computation finished. Found 15 intersecting elements:
Required time:  6.5 s
Data sent:     10.7 MB
Data received:  0.1 MB
```



## 4.8.8. Dataset with 800.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 1 -f
Time for reading elements:      0.10 s
Computation finished. Found 14 intersecting elements:
Required time:  5.7 s
Data sent:      12.2 MB
Data received:  0.1 MB
```

## 4.8.9. Dataset with 900.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 1 -f
Time for reading elements:      0.14 s
Computation finished. Found 27 intersecting elements:
Required time:  5.2 s
Data sent:      13.7 MB
Data received:  0.1 MB
```

## 4.8.10. Dataset with 1.000.000 entries:

```
alexandre@ZenBook:~/Documents/Uni/6th year/1st Semester/SP/PSI$ ./demo.exe -r 1 -p 1 -f
Time for reading elements:      0.14 s
Computation finished. Found 35 intersecting elements:
Required time:  4.1 s
Data sent:      15.3 MB
Data received:  0.1 MB
```

## 5. Protocol Comparison

In order to analyse and evaluate the **Private Set Intersection** (PSI) protocols, we executed *psi.exe* instead of *demo.exe* in two terminals. The *psi.exe* doesn't take files as inputs because it creates its own datasets with the required number of lines to test and benchmark the different protocols available. Although the results from the *psi.exe* command were very close to our results that were run with the *demo.exe* command.

- `./psi.exe -r 0 -p x* -b 16 -n n*`
- `./psi.exe -r 1 -p x* -b 16 -n n*`

**x\*** - number of the correspondent protocol, varying between 0 and 3

- 0- Naïve Hashing Protocol
- 1- Server Aided Protocol
- 2- Diffie-Hellman Protocol
- 3- Oblivious Transfers Protocol

**n\*** - number of lines that are created for the testing file

To make correct comparisons, we have to consider the required time in seconds, the sent data and finally the received data. The last two components summed is considered to be the exchanged data.

## 5.1. Comparisons

### 5.1.1. Time and Exchanged Data Comparisons - Demo Executable File

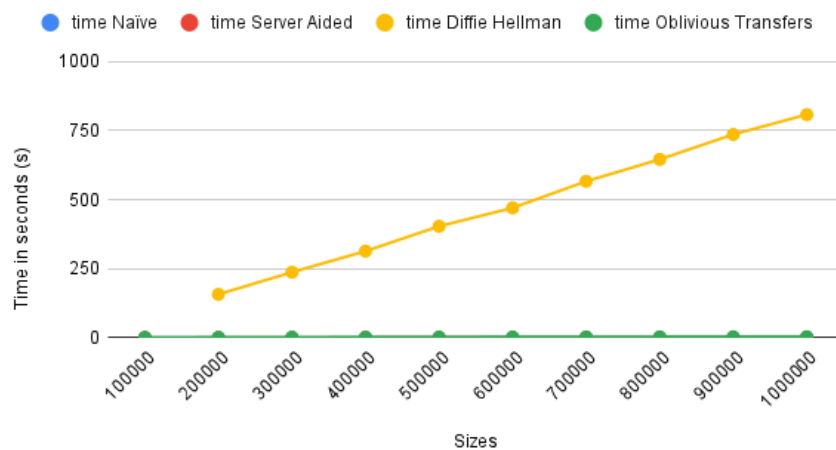
With the following table, we have stored all our data from all of our iterations with the *demo.exe* file.

Protocols	p0 Naïve Hashing			p1 Server Aided			p2 Diffie-Hellman			p3 Oblivious Transfers		
Sizes	time Naïve	data exchanged Naïve	common contacts	time Server Aided	data exchanged Server Aided	common contacts	time Diffie Hellman	data exchanged Diffie Hellman	common contacts	time Oblivious Transfers	data exchanged Oblivious Transfers	common contacts
100000	0,1	2	4	0,1	3,1	0	75,8	10,1	4	0,6	10,2	0
200000	0,1	3,8	9	0,1	6,1	3	157,2	20,3	11	1	20,4	3
300000	0,2	5,8	26	0,2	9,3	3	237,3	30,3	25	1,4	30,6	3
400000	0,2	7,6	37	0,2	12,3	5	313,6	40,4	42	1,7	40,8	5
500000	0,3	9,6	60	0,3	15,4	7	403,7	50,5	49	2	50,9	7
600000	0,3	11,4	105	0,3	18,4	16	470,3	60,7	105	2,4	61,2	16
700000	0,4	13,4	125	0,4	21,6	15	566,6	70,8	123	2,7	71,3	15
800000	0,4	15,2	172	0,4	24,6	14	646	80,8	133	2,9	81,5	14
900000	0,5	17,2	242	0,4	27,7	27	736,3	91	253	3,2	91,7	27
1000000	0,6	19	241	0,5	30,7	35	808,1	101,1	239	3,4	106,9	35

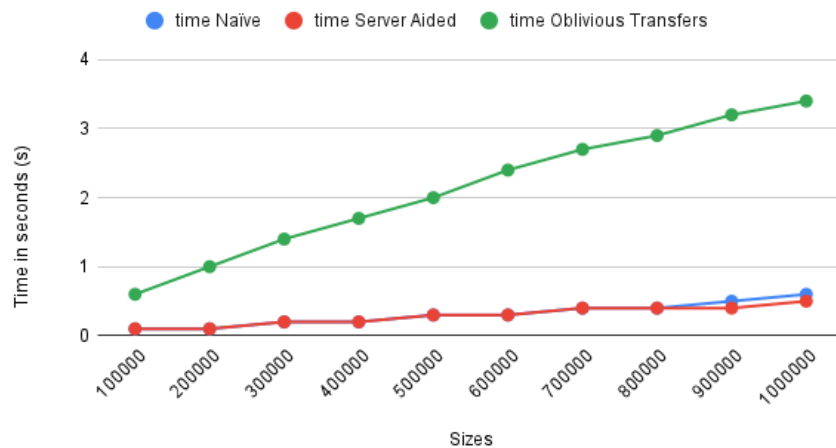
For a better visualization, we made graphics that are better to visualize all the data.

In the following images, there is a quick plot of our tables' values, but since the Diffie-Hellman protocol had times that were massive compared to the other protocols, we made a separated plot without it the Diffie-Hellman protocol.

Times with Dataset



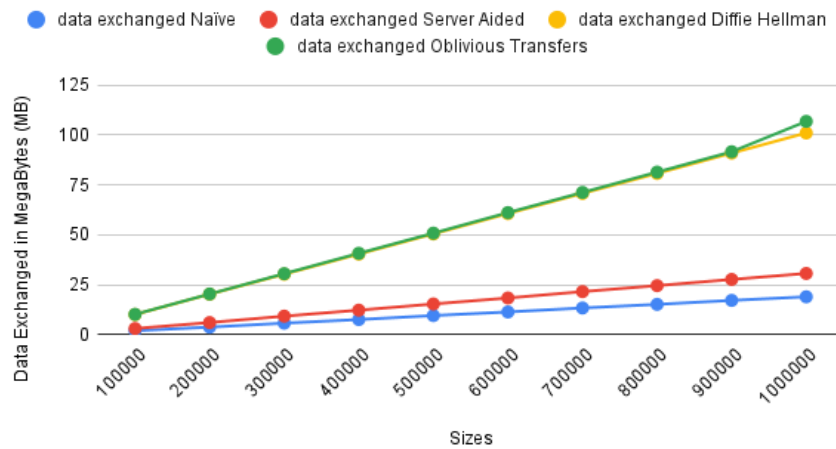
Times with Dataset without Diffie Hellman



After comparing execution time, we compared the exchanged data between client A and client B as the following image demonstrate.



Data Exchanged with Dataset



### 5.1.2. Time and Exchanged Data Comparisons - PSI Executable File

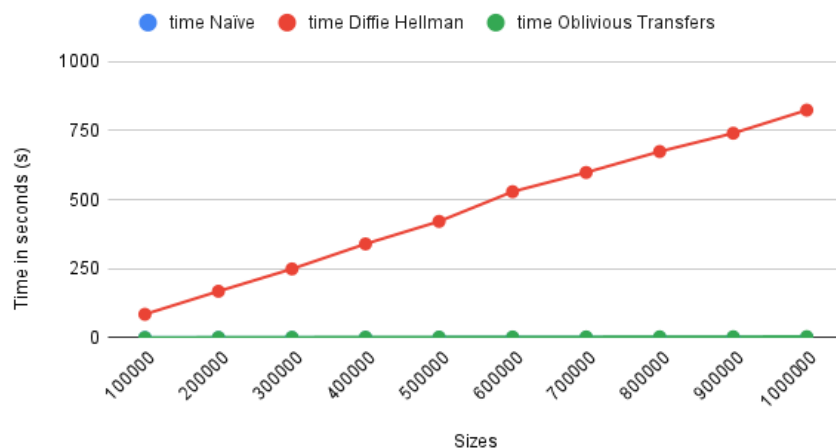
With the following table, we have stored all our data from all of our iterations with the *psi.exe* file, excluding the Server Aided Protocol, which we could not run it with the *psi.exe*.

Protocols	p0 Naïve Hashing			p1 Server Aided			p2 Diffie-Hellman			p3 Oblivious Transfers		
Sizes	time Naïve	data exchanged Naïve	common contacts	time Server Aided	data exchanged TTP	common contacts	time Diffie Hellman	data exchanged Diffie Hellman	common contacts	time Oblivious Transfers	data exchanged Oblivious Transfers	common contacts
100000	0,1	2	-	-	-	-	84,8	10,1	-	0,6	10,2	-
200000	0,1	3,8	-	-	-	-	168,2	20,3	-	1	20,4	-
300000	0,1	5,8	-	-	-	-	249,1	30,3	-	1,4	30,6	-
400000	0,2	7,6	-	-	-	-	339,7	40,4	-	1,9	40,8	-
500000	0,2	9,6	-	-	-	-	421,3	50,5	-	2,2	50,9	-
600000	0,3	11,4	-	-	-	-	529	60,7	-	2,5	61,2	-
700000	0,3	13,4	-	-	-	-	598,5	70,8	-	2,8	71,3	-
800000	0,4	15,2	-	-	-	-	674,4	80,8	-	3,2	81,5	-
900000	0,4	17,2	-	-	-	-	740,6	91	-	3,4	91,7	-
1000000	0,5	19	-	-	-	-	824,4	101,1	-	3,8	101,9	-

For a better visualization, we made graphics that are better to visualize all the data.

In the following images, there is a quick plot of our tables' values, but since the Diffie-Hellman protocol had times that were massive compared to the other protocols, we made a separated plot without it the Diffie-Hellman protocol.

Times from Benchmarks

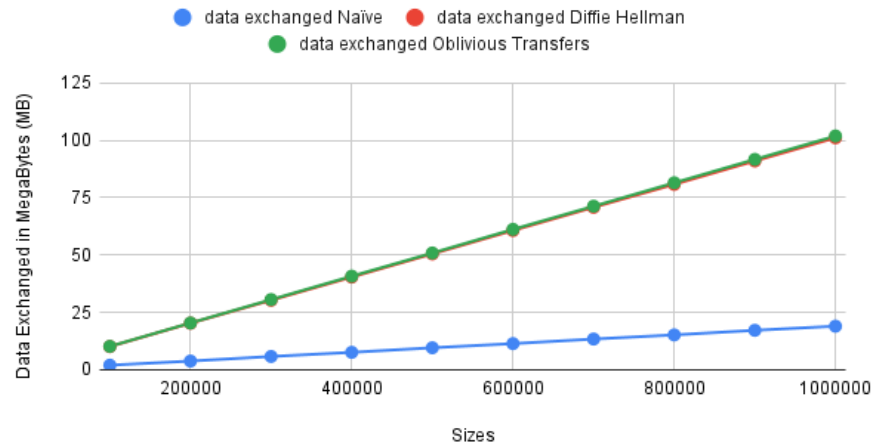


Times from Benchmarks without Diffie Hellman



After comparing execution time, we compared the exchanged data between client A and client B as the following image demonstrate.

Data Exchanged from Benchmarks



## 6. Conclusion

By understanding what each algorithm does, we can immediately understand the time that the Diffie-Hellman protocol takes is due to the amount of encryption processing that needs to be done before sending the data to the other party and then the amount of processing needed to decrypt the intersected data.

By the tests made, we can confirm that the Diffie-Hellman PSI protocol shows the lowest efficiency, followed by the Naïve Hashing protocol, then the Server-aided PSI protocol and the Oblivious Transfer based protocol show the highest efficiency.

In terms of data exchanged between the intervening parties of each protocol, we can state that the least demanding ones are the Naïve protocol and the Server-aided protocol. And the most demanding, in terms of data exchange, being the Oblivious Transfers based protocol and Diffie-Hellman based protocol.

So, we can see that we have to make a compromise between the time needed and exchanged data, while considering the security and privacy of the data being shared between the two or more parties.

## 7. References

- [1] - Private Set Intersection (PSI) - <https://github.com/encryptogroup/PSI> , accessed on December 2023
- [2] - Theoretical Classes Presentations - provided by the professor