UNIVERSIDADE D
COIMBRA

Security and Privacy

Secure Multiparty Computation

Francisco Catarino Mendes - 2019222823
Leonardo Oliveira Pereira - 2020239125
Department of Informatics Engineering
University of Coimbra

Introduction

This assignment pretends to tackle Secure Multiparty Computation (SMC). The main goal is to explore this concept, as well as the tools and libraries that implement it.

A problem is first going to be defined, which PSI (Private Set Intersection) Protocols are going to be able to solve. Then, the chosen dataset will be described, with a classification of its attributes, which involves talking about the input sets and the intersections.

Next, the PSI Protocols are going to be applied. Before that, a brief explanation of which one is going to be given, and the dataset division will also be exhibited. Once those are in order, the Naive Hashing solutions, the Diffie-Helmann-based protocol, the OT-based protocol and the solution implemented are going to be executed on the dataset. The process and the results are going to be shown there.

Finally, a benchmarking of the protocols is also needed. An evaluation and a comparison of each of the 4 protocols will be performed with respect to the execution time and exchanged data, information collected along the making of the assignment. Graphs are going to be made, and conclusions will be taken from discussing them. The assignment also contains a conclusion, to sum up all of these activities and reflect on what was learned.

1 - Characterization of the Dataset

In this section, the dataset is selected and characterized. The problem is defined and the attributes are classified.

1.1 - Problem Definition

Two marketing firms, Firm A and Firm B, are collaborating to create a joint marketing campaign. Both firms have their own lists of subscribers, which include first name, last name, gender, email, and phone number. To maximize the effectiveness of their campaign, they need to identify common subscribers between their lists. However, due to privacy concerns and competitive reasons, neither firm is willing to share its entire subscriber list with the other.

By using Private Set Intersection protocols, the firms are able to securely and privately identify the common subscribers (based on phone numbers) between their subscriber lists without revealing any other information about the subscribers to the other party, addressing privacy concerns this way. Also, email addresses, phone numbers, and other personal information are sensitive, and PSI ensures that this data is not leaked during the process.

1.2 - Classification of the attributes - Input sets and Intersections

The data collected is organized like Figure 1 has:

Giustina,Yashunin,Female,gyashunin0@aol.com,640 140 1411
Drake,Crocker,Male,dcrocker1@earthlink.net,597 310 9169
Zack,Thynn,Agender,zthynn2@devhub.com,516 112 0015
Dania,Bindin,Female,dbindin3@mediafire.com,481 532 6829
Madison,Glassman,Male,mglassman4@weebly.com,907 465 3820
Zebadiah,Litt,Male,zlitt5@youku.com,716 743 4968
Valry,Fardy,Female,vfardy6@blinklist.com,353 332 5032
Miguelita,Fernan,Female,mfernan7@youku.com,711 542 4137
Terri,Hugo,Female,thugo8@answers.com,477 292 8615
Reggis,Oakeby,Bigender,roakeby9@buzzfeed.com,553 441 3139
Aubry,Merrigan,Female,amerrigana@toplist.cz,613 421 3954
Jenn,Vinall,Female,jvinallb@wikispaces.com,535 273 0881
Lisbeth,Collie,Polygender,lcolliec@amazon.com,132 844 2269
Elyse,Reinger,Female,ereingerd@cnet.com,232 837 9626

Figure 1: Organization of the data attributes

As one can see, the dataset is composed of the following attributes:

a) **First Name -** the first name of a subscriber;

b) **Last Name -** the last name of a subscriber;

c) **Gender -** the gender of a subscriber, masculine or feminine;

d) **Email -** the email address of a subscriber;

e) **Phone Number -** the number corresponding to the contact gave by the subscriber;

These are the attributes that represent the data that is going to be exchanged or intersected in a secure and private way, using the PSI protocols to achieve that.

As for intersections, they happen because the two firms will have, once again, subscribers in common, with the same exact information being contained in both datasets. This will be better explained in section 2.2.

2 - Protocols for Private Set Intersection - Description and Application

Here, the PSI protocols are going to be put to use. Firstly, they need to be explained, so that when executing them, one can understand what is being done.

Also, along with the protocols, the tool Wireshark is going to be used to 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 (capture loopback (127.0.0.1) and using the filter tcp.port == 7766).

2.1 - Protocol Description

Private Set Intersection (PSI) protocols are cryptographic methods that allow two or more parties to compute the intersection of their datasets without revealing any other information about the datasets. These protocols ensure that only the common elements between the datasets are identified, preserving the privacy of all other data.

The primary goal of PSI protocols is to ensure that each party learns nothing beyond the intersection of their datasets. Specifically, no party learns any element that is not common to both datasets. PSI protocols are designed to be secure against various types of attacks, including passive and active adversaries. This means that even if one party tries to cheat or learn additional information, the protocol should prevent such actions. PSI protocols are also designed to minimize computational and communication overhead, making them efficient and practical for use with large datasets.

In this assignment, the following types of PSI protocols are going to be used:

Naive Hashing Solution - each party hashes their dataset elements and exchanges the hashed values. Then, the hashes are compared to find common elements. These are simple and easy to implement, but are vulnerable to hash collision attacks and require large communication overhead.

Diffie-Helmann-based Protocol - utilizes the Diffie-Hellman key exchange protocol. Each party computes a cryptographic hash of its elements and applies a Diffie-Hellman operation to obscure the values. The parties then exchange the obscured values and find the intersections by comparing them. It provides better security and can handle larger datasets more efficiently than basic hash-based methods, but is more computationally intensive due to cryptographic operations.

Oblivious Transfer (OT)-based Protocol - uses Oblivious Transfer, a cryptographic primitive, to securely transfer data between parties without revealing any information beyond the intersection. It gives strong security guarantees and does not require parties to exchange their entire dataset, but is typically more complex and computationally intensive.

Solution Implemented - talked about later, in section 2.6.

2.2 - Dataset Division

To better explain how the dataset was created, this short section was made. The dataset has two independent parts, each one corresponding to the data of one party. The data is presented in a clean and uniform way inside CSV files.

After the dataset was established, 5 versions were generated in which the main varying factor was the size. The dataset sizes determined were 5k, 10k, 15k, 20k, and 25k. The tests with the different protocols are going to be executed on all of these five versions with different data sizes.

Another important factor to mention is the amount of intersections. One made sure that as the dataset increases by 5k from version to version, the amount of common information would also increase by 2k. Therefore, for the 5k version, the number of intersections should be 1k, for the 10k version, the number of intersections should be 3k, for the 15k version, the number of intersections should be 5k, and so on until the 25k version.

2.3 - Naive Hashing Solutions

Once again, in this approach, each party hashes the inputs and sends the hashes to the other party. Each party then computes the intersection with its own hashed input to find the intersections. The default hashing algorithm used is the sha256 truncated to the 6 most significant bytes. Figure 2 displays the information gathered using the Naive Hashing Protocol:

Entries	Algorithm	Time span	Packets	Bytes	Data Sum	Repeated	Intersections
5000	P0-Naive Hashing	0,093	33	92234	92267	1 000	1 000
10000	P0-Naive Hashing	0,129	36	182432	182468	3 000	3 000
15000	P0-Naive Hashing	0,15	44	272960	273004	5 000	5 001
20000	P0-Naive Hashing	0,2	45	365806	365851	7 000	7 000
25000	P0-Naive Hashing	0,239	47	453158	453205	9 000	9 000

Figure 2: Using the Naive Hashing Protocol

As the number of entries increases from 5000 to 25000, the time span also increases from 0.093 to 0.239. This indicates that the Naive Hashing algorithm scales with the size of the dataset, though the increase in time is relatively moderate. The number of packages and the amount of data exchanged (bytes) also increases with the dataset size. Specifically, the bytes exchanged grow linearly with the number of entries, reflecting the additional computational and communication load.

Furthermore, the increase in the number of packets is not linear, suggesting that the protocol manages data efficiently up to a certain extent before additional overheads start contributing more significantly. Also, the intersection results align perfectly with the ”Repeated” entries, suggesting that the protocol accurately identifies common entries without errors or omissions.

Finally, the total data sum increases substantially with the size of the dataset, highlighting the direct relationship between data volume and computational/communication requirements.

This type of protocol appears suitable for applications where dataset sizes are in the range of thousands to tens of thousands of entries. For significantly larger datasets, more efficient protocols or optimizations might be necessary to keep communication and computation within acceptable limits.

2.4 - Diffie-Helmann-based Protocol

Regarding this protocol, where each party computes a cryptographic hash of its elements and applies a Diffie-Hellman operation to obscure the values, which are then exchanged, the following results were obtained, as Figure 3 shows:

Entries	Algorithm	Time span	Packets	Bytes	Data Sum	Repeated	Intersections
5000	P2-Diffie-Hellman-based	6,227	55	533686	533741	1 000	1 000
10000	P2-Diffie-Hellman-based	11,926	79	1098050	1098129	3 000	3 000
15000	P2-Diffie-Hellman-based	18,284	101	1596722	1596823	5 000	5 000
20000	P2-Diffie-Hellman-based	24,767	127	2128555	2128682	7 000	7 000
25000	P2-Diffie-Hellman-based	31,19	146	2659704	2659850	9 000	9 000

Figure 3: Using the Diffie-Helmann-based Protocol

As the number of entries increases from 5000 to 25000, the time span also increases significantly from 6.227 to 31.19 seconds. This indicates that the Diffie-Hellman-based algorithm scales with the size of the dataset, but the increase in time is more pronounced compared to the Naive Hashing protocol. The number of packages and the amount of data exchanged also increase with the dataset size. The bytes exchanged grow substantially with the number of entries, reflecting the higher computational and communication complexity of the Diffie-Hellman-based protocol.

Moreover, the number of packages increases significantly with the dataset size, suggesting that the protocol requires more communication overhead as the dataset grows. But despite the increased complexity, the protocol accurately identifies the intersections, with the number of intersections matching the ”Repeated” entries perfectly.

Again, the total data sum increases considerably with the size of the dataset, highlighting the heavy data processing and communication demands of the Diffie-Hellman-based protocol.

This protocol appears more suitable for scenarios where strong security is paramount, and the datasets are not excessively large. For extremely large datasets, the communication and computational costs might outweigh the benefits.

2.5 - OT-based Protocol

Concerning the OT-based protocol, which uses Oblivious Transfer to securely transfer data between parties without revealing any information beyond the intersection, the information achieved when applying it to the dataset is illustrated by Figure 4:

Entries	Algorithm	Time span	Packets	Bytes	Data Sum	Repeated	Intersections
5000	P3-OT-based	0,541	67	551744	551811	1 000	1 000
10000	P3-OT-based	0,636	87	1081280	1081367	3 000	3 000
15000	P3-OT-based	0,599	99	1610288	1610387	5 000	5 000
20000	P3-OT-based	0,666	141	2108505	2108646	7 000	7 000
25000	P3-OT-based	0,864	151	2408008	2408159	9 000	9 000

Figure 4: Using the OT-based Protocol

The time span increases from 0.541 seconds for 5000 entries to 0.864 seconds for 25000 entries. This is a relatively moderate increase compared to the P2-Diffie-Hellman-based protocol, indicating better scalability. The number of packages and the amount of data exchanged also increase with the dataset size. The bytes exchanged grow in proportion to the number of entries, but the increase is less steep compared to the Diffie-Hellman-based protocol.

Additionally, the number of packages increases with the dataset size, but the increase is less pronounced than in the Diffie-Hellman-based protocol. The protocol accurately identifies intersections, with the number of intersections matching the ”Repeated” entries perfectly.

The total data sum increases significantly with the size of the dataset, but not as drastically as with the Diffie-Hellman-based protocol. From 551744 bytes for 5000 entries to 2408008 bytes for 25000 entries, this shows a more efficient handling of data volume.

This protocol appears suitable for a wide range of applications, particularly where dataset sizes are large and efficiency is crucial. The relatively lower computational and communication overhead makes it ideal for practical use cases.

2.6 - Solution Implemented

The solution implemented uses a trusted centralized party and evaluates the performance in terms of time and exchanged data for this solution, like the other protocols. The chosen solution was the Server-Aided Protocol.

Server-Aided Protocol - leverages the assistance of an external, semi-trusted server to facilitate the intersection process between two or more parties. The server's role is to aid in the computation, making the process more efficient while maintaining the privacy of the individual datasets.

The server is assumed to be semi-trusted, meaning it follows the protocol correctly but is curious and might try to learn additional information. The protocol ensures that even if the server is curious, it cannot learn anything beyond what is allowed.

Figure 5 displays the results of the application of this protocol on the datasets:

Entries	Algorithm	Time span	Packets	Bytes	Data Sum	Repeated	Intersections
5000	P1-Server-aided	1,316	41	164012	164053	1 000	1 000
10000	P1-Server-aided	1,334	54	326120	326174	3 000	3 000
15000	P1-Server-aided	1,373	66	488162	488228	5 000	5 000
20000	P1-Server-aided	1,416	74	649940	650014	7 000	7 000
25000	P1-Server-aided	1,434	84	811850	811934	9 000	9 000

Figure 5: Using the Server-Aided Protocol

The time span increases slightly from 1.316 seconds for 5000 entries to 1.434 seconds for 25000 entries. This moderate increase indicates that the Server-aided protocol scales well with the dataset size, maintaining relatively consistent performance. This happens due to the fact that this variable, in the case of this protocol, is controlled by the tester, as the time difference between running the commands for one dataset and the other will always result in the time-span result. Therefore, it presents consistency.

The number of packages and the amount of data exchanged also increase with the dataset size. However, the increase in bytes is less steep compared to other protocols, demonstrating the efficiency of the Server-aided approach in handling larger datasets. Furthermore, the number of packages increases with the dataset size, but the increase is manageable, showing the protocol's efficiency in communication. The protocol accurately identifies intersections, with the number of intersections matching the "Repeated" entries perfectly.

Finally, the total data sum increases with the size of the dataset, but the growth is relatively moderate compared to other protocols like the Diffie-Hellman-based PSI. From 164012 bytes for 5000 entries to 811850 bytes for 25000 entries, this shows efficient handling of data volume.

This protocol is suitable for applications where efficiency and scalability are crucial, especially with large datasets. The consistent performance makes it ideal for real-world scenarios requiring secure and efficient data intersection.

3 - Benchmarking Private Set Intersection Protocols

In this final section of the assignment, the aim is to benchmark each of the four protocols with respect to the execution time and exchanged data. Graphs are going to be created: 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; and 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).

3.1 - Required time

One of the main topics to be evaluated in the application of the PSI protocols is the required time. It has already been talked about a little in the previous sections, but now some more discussions are going to take place. The following graphs were constructed, exhibited by Figures 6, 7, 8 and 9, one for each Protocol applied, based on the already acquired information in the entirety of section 2:

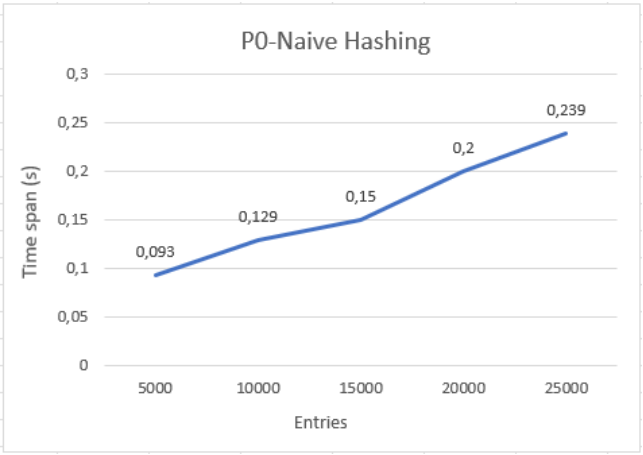

Figure 6: Required time - Graph of the Naive Hashing Protocol

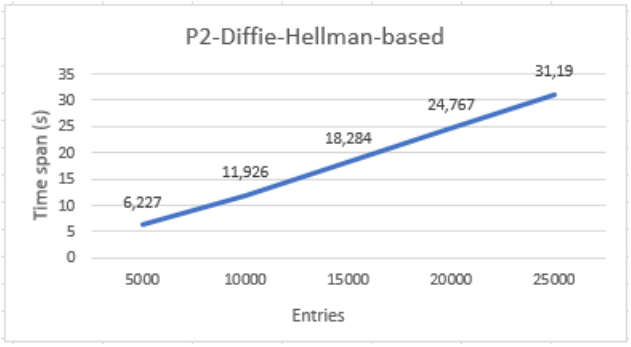

Figure 7: Required time - Graph of the Diffie-Helmann-based Protocol

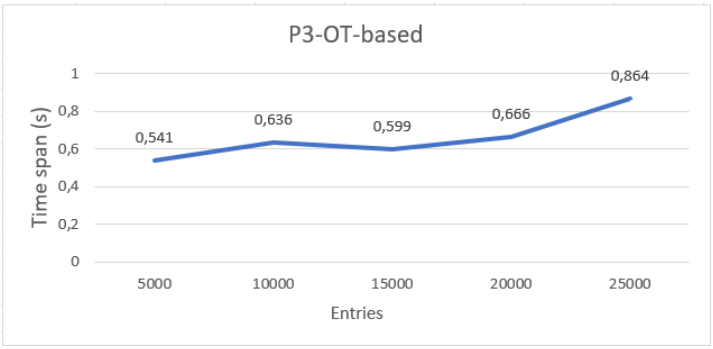

Figure 8: Required time - Graph of the OT-based Protocol

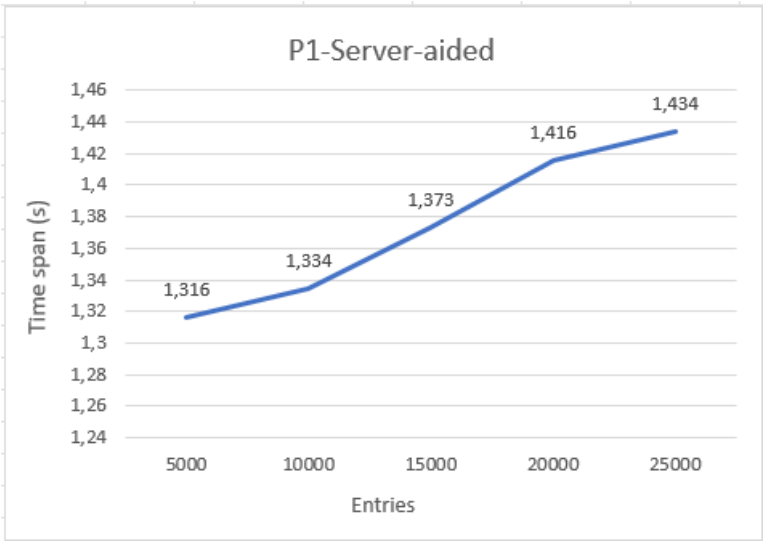

Figure 9: Required time - Graph of the Server-Aided Protocol

In terms of scalability, the Naive Hashing, and the Server-Aided protocols show the best performance, with only a moderate increase in time as dataset sizes grow. The Diffie-Hellman-based protocol shows poor scalability, with a steep increase in required time, making it less suitable for larger datasets. As for the OT-based protocol, it shows reasonable scalability, with moderate increases in time.

Moving on to efficiency, the Server-aided protocol demonstrates the highest efficiency, maintaining low time spans even for larger datasets. Once again, this happens as the time is controlled by the quickness of the tester in running the commands. The Naive Hashing Protocol is efficient for smaller datasets but may not be suitable for very large datasets. The OT-based protocol balances efficiency and security, making it a versatile option. And the Diffie-Hellman-based protocol, while secure, is less efficient and more suitable for smaller datasets.

The last point going to be discussed is suitability. For applications with large datasets and a need for efficiency, the Server-aided protocol is the best choice. For smaller datasets or when computational simplicity is desired, the Naive Hashing solutions are appropriate. For applications requiring a balance of security and efficiency across various dataset sizes, the OT-based protocol is suitable. Finally, for applications where strong security is paramount but datasets are smaller, the Diffie-Hellman-based protocol can be considered.

All of these can be proven in Figure 10, which represents the junction of all the individual graphs in a single one:

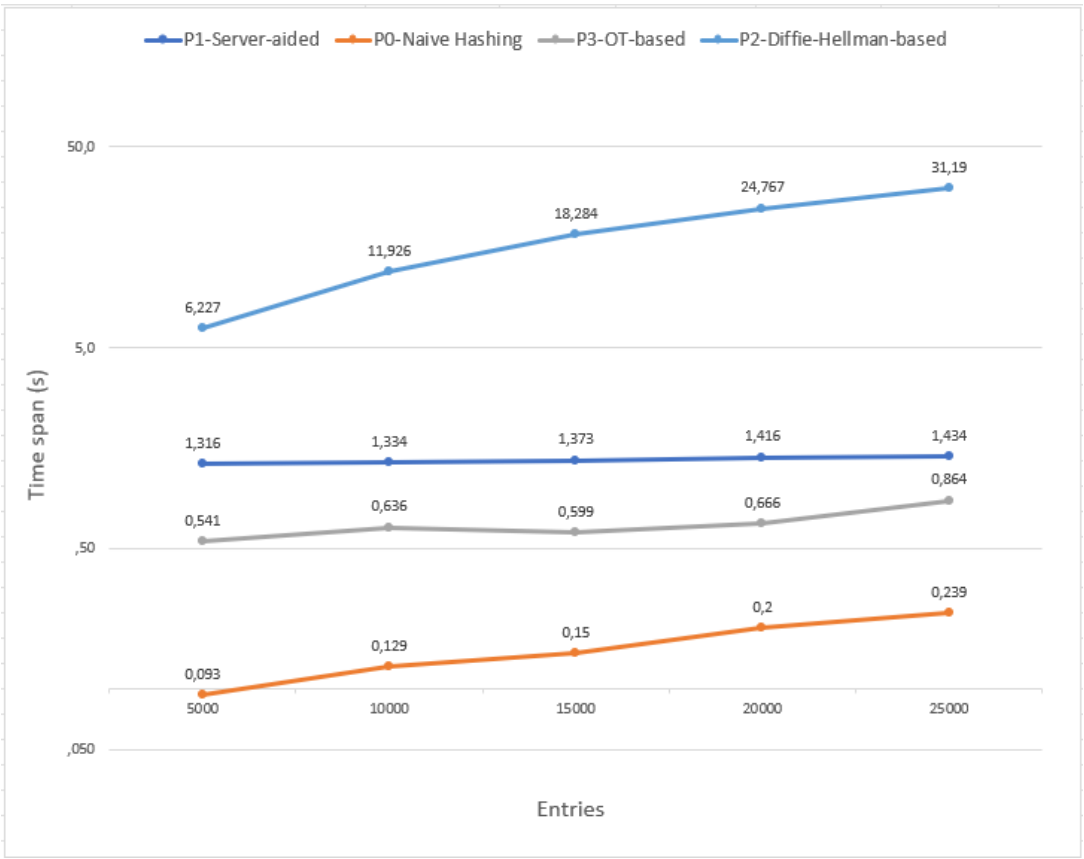

Figure 10: Required time - Graph of all the Protocols together

3.2 - Data Exchange

The other main topic evaluated is the amount of data exchanged (data sent + data received). Once again, this was already talked about in the application of the protocols, but here, more information is given, alongside the graphs of Figures 11, 12, 13, and 14, elaborated from the data gathered in section 2:

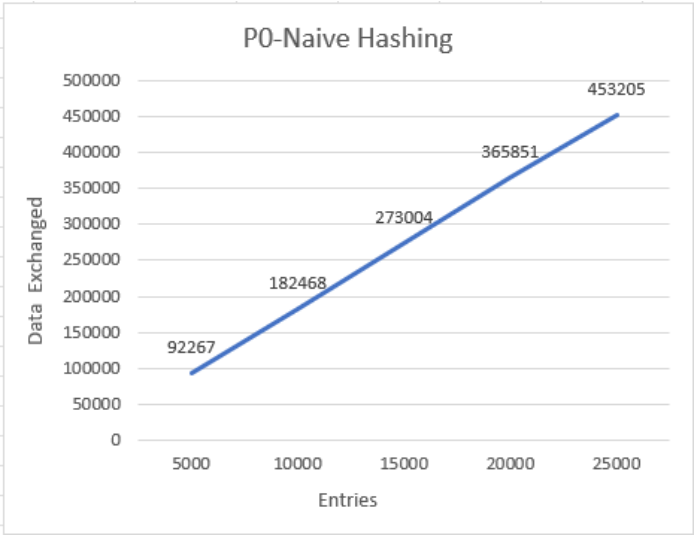

Figure 11: Data Exchanged - Graph of the Naive Hashing Protocol

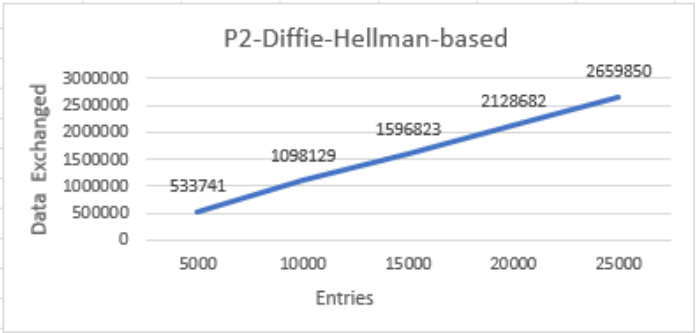

Figure 12: Data Exchanged - Graph of the Diffie-Helmann-based Protocol

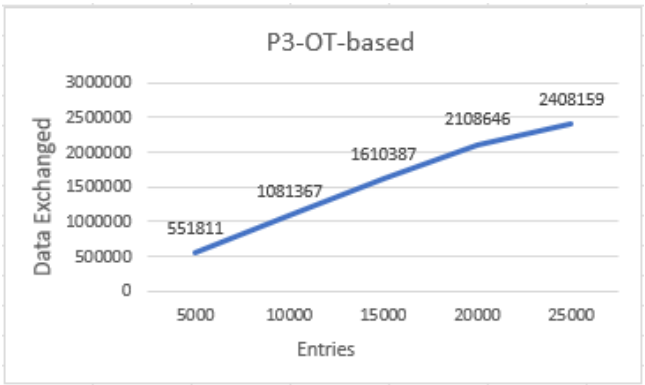

Figure 13: Data Exchanged - Graph of the OT-based Protocol

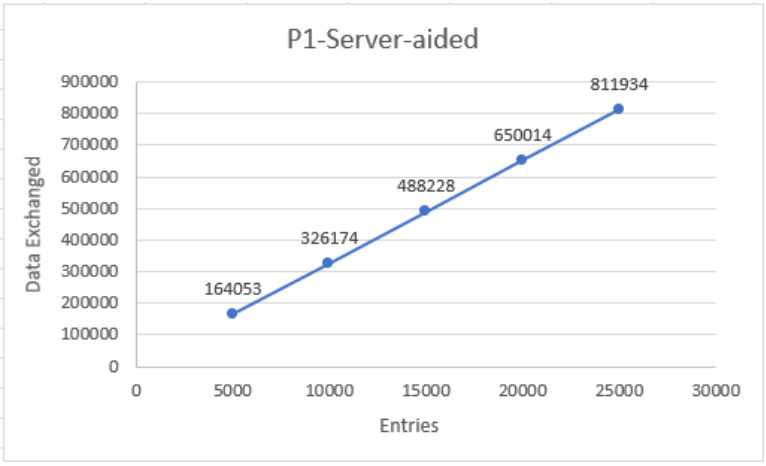

Figure 14: Data Exchanged - Graph of the Server-Aided Protocol

Talking once again about scalability, the Naive Hashing solutions, and the Server-Aided protocol show the best performance in terms of data exchanged, with linear increases indicating predictable and manageable communication costs. The Diffie-Hellman-based protocol shows poor scalability, with a steep increase in data exchanged, making it less practical for large datasets. As for the OT-based protocol, it shows moderate scalability, with a reasonable increase in data exchanged.

Regarding efficiency, the Server-Aided protocol demonstrates the highest efficiency in terms of data exchanged, maintaining low communication costs even for larger datasets. The Naive Hashing protocol is also efficient for smaller datasets, with a predictable increase in data exchanged. The OT-based protocol offers a good balance of security and efficiency, suitable for medium to large datasets. On the other hand, the Diffie-Hellman-based protocol, while secure, comes with high communication costs, making it less efficient for larger datasets.

To conclude, suitability must also be addressed again. For applications with large datasets and a need for efficient data exchange, the Server-Aided protocol is the best choice. For smaller datasets or when computational simplicity and predictable communication costs are desired, the Naive Hashing solutions are appropriate. For applications requiring a balance of security and efficiency, the OT-based protocol is suitable, and for applications where strong security is paramount but datasets are smaller, the Diffie-Hellman-based protocol can be considered.

This information can be effectively seen in the graph that groups together all of the individual protocol graphs, exhibited in Figure 15:

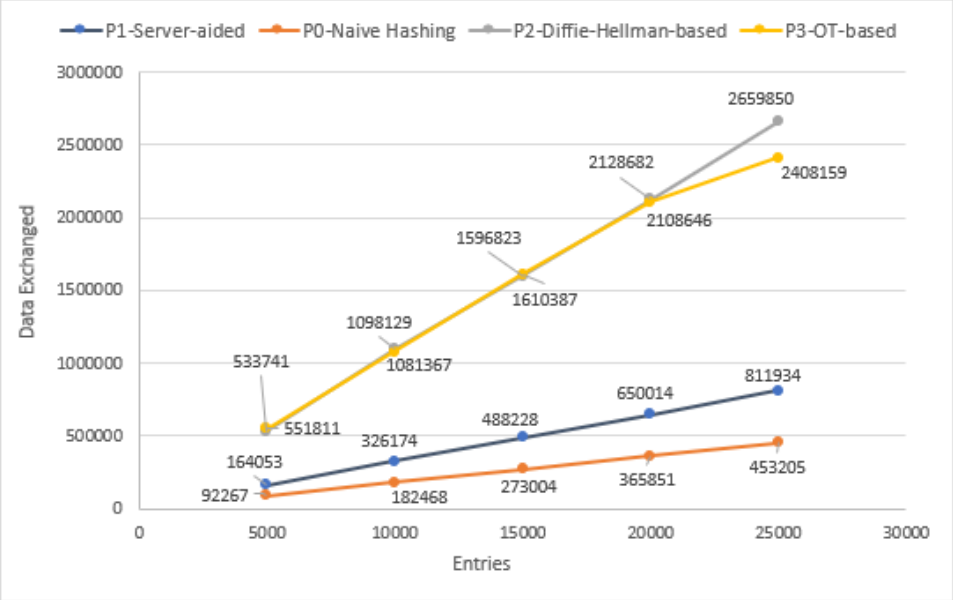

Figure 15: Data Exchanged - Graph of all the Protocols together

3.3 - Considerations about Security and Privacy

Some final considerations about the concepts of security and privacy must be made regarding the PSI protocols:

Naive Hashing Protocol - uses simple hashing to identify intersections, which provides basic security. However, it is susceptible to hash collisions and brute-force attacks. Since hashing is deterministic, an adversary who can guess or knows the hashing algorithm and has partial knowledge of the dataset can infer some information about the data.

Diffie-Hellmann-based Protocol - leverages the Diffie-Hellman key exchange mechanism, providing strong cryptographic guarantees. It is secure against passive adversaries who cannot derive the original data from the exchanged encrypted values. The use of public key cryptography ensures that even if data is intercepted, it cannot be decrypted without the private keys.

OT-based Protocol - provides strong security by ensuring that each data exchange is oblivious, meaning one party cannot learn anything about the other party's input except for the result of the intersection. Resistant to various types of attacks, including replay attacks and man-in-the-middle attacks, due to its cryptographic foundations.

Solution Implemented - Server-Aided Protocol - relies on the assumption that the server is semi-trusted. It follows the protocol but may be curious. Strong cryptographic measures are employed to ensure the server cannot infer any additional information. While the server facilitates the computation, it does not learn the actual datasets or the result of the intersection. Only the intersecting elements are known to the parties involved.

Selecting the right PSI protocol involves balancing security, privacy, and efficiency. For applications requiring strong security and privacy, Diffie-Hellman-based and OT-based protocols are recommended. For scenarios where efficiency and performance are critical, and the data is less sensitive, Naive Hashing or Server-aided protocols might be more appropriate.

3.4 - Solving the problem

Now, after all the previous work and information gathered, a final choice for our problem that was defined in the beginning needs to be made.

Given the problem of securely identifying common subscribers between two marketing firms, the solution implemented, the Server-Aided protocol, is the best-suited solution. It offers an optimal balance between efficiency, scalability, and security, making it ideal for handling large datasets while maintaining data privacy. This protocol ensures that the intersection is computed accurately and securely, without revealing additional information about the datasets to the participating firms or the semi-trusted server.

Conclusion

In this assignment, Private Set Intersection (PSI) protocols were used to solve the problem of securely identifying common subscribers between two marketing firms' subscriber lists. The dataset consisted of first names, last names, gender, email addresses, and phone numbers.

Four different PSI protocols were analyzed: Naive Hashing, Server Aided, Diffie-Hellman-based, and OT-based, evaluating their performance in terms of required time and data exchanged across various dataset sizes.

Finally, the Benchmarking of the PSI protocols was made, once again targeting the concepts of required time and data exchanged. Graphs were constructed using the information in the tables displayed in section 2, and from those, some more discussions were performed. To conclude the assignment, some considerations about the security and privacy of the PSI protocols were presented, and the final solution to the problem defined at the beginning of this work was provided.