Privacy-Preserving Voter Fraud Detection Using Private Set Intersection (PSI)

Benjamim Moreira 2020261856
José Cunha 2021223719
José Filipe 2021216675

12th November 2024

1 Introduction

In an increasingly digital world, data security has become a critical priority. As more sensitive information is processed and shared online, the need to protect this data from unauthorized access is more essential than ever, especially in critical sectors like electoral systems. Election security has emerged as a major concern, with the rise of digital voting systems and the need to prevent voter fraud while respecting voter privacy. One area of particular interest is detecting individuals who may attempt to vote in multiple states, a form of electoral fraud that can compromise the integrity of election results.[3]

To address these concerns, Private Set Intersection (PSI) protocols have become a valuable tool for enhancing security while preserving privacy. PSI enables multiple parties—in this case, different state election agencies—to compare datasets and identify overlapping elements, such as voters who appear in more than one state's voting database, without exposing other data. This allows states to collaborate in detecting potential fraud while respecting privacy and legal constraints, making PSI an essential tool for election security in the digital age.[3]

This work was conducted in a virtual machine environment, where we simulated the Linux operating system and implemented four Private Set Intersection (PSI) protocols to evaluate their efficiency.

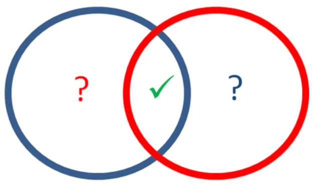

Figure 1: Private Set Intersection

2 PSI Protocol Variants

There are four main variants of the PSI protocol, each with specific strengths and applications:

1. **Naive Hashing Protocol (-p 0)**: This is a basic PSI protocol where both parties independently hash their datasets and exchange the hashed values. Then, each party checks for matches based on these hash values. While straightforward and easy to implement, this protocol can be vulnerable to frequency attacks and may become inefficient with large datasets.[2, 4]
2. **Server-Aided Protocol (-p 1)**: This protocol introduces a third-party server that assists in computing the PSI between the two parties. The server facilitates the intersection process while maintaining privacy, ensuring that only overlapping elements are identified. By leveraging a trusted server, this protocol can reduce the computational load on both parties, though it relies on the assumption of a semi-trusted third party.[1, 5]

3. **Diffie-Hellman-Based Protocol (-p 2)**: This protocol applies the Diffie-Hellman key exchange method, allowing both parties to independently encrypt their data using each other's public keys. By leveraging the properties of Diffie-Hellman, both parties can identify shared items without exposing the non-intersecting elements. This protocol offers stronger privacy guarantees but can be computationally expensive, especially for large datasets [**5**].
4. **Oblivious Transfer (OT)-Based Protocol (-p 3)**: This protocol relies on the Oblivious Transfer (OT) technique, allowing one party to learn only the intersecting elements from the other party's dataset without learning anything else. This makes it especially useful in cases where only one party requires knowledge of the intersection. OT-based PSI provides enhanced privacy for applications involving sensitive data [**3**].

Each PSI protocol variant provides distinct advantages, making it possible to select the best-suited method based on dataset size, computational power, and privacy needs. In this work, we use PSI to simulate a collaborative analysis between state election agencies, demonstrating how PSI protocols enable privacy-preserving intersections in real-world applications.

3 Problem Definition

In this project, we simulate the use of Private Set Intersection (PSI) protocols to securely identify voters who may have voted in multiple states, a form of electoral fraud that undermines the democratic process. This type of verification is essential due to the decentralized nature of the U.S. electoral system. In our simulation, each state maintains its own list of registered voters who cast ballots. Due to privacy and legal constraints, states cannot share their complete voter lists directly with each other. Instead, they need a secure method to find overlapping voter IDs without exposing their full datasets.

The PSI protocol allows state election agencies to identify voters registered in multiple states without exposing the complete details of their databases. By identifying only the intersections between lists, agencies can flag suspicious cases and investigate further, preserving voter privacy while enhancing the integrity of election results.[3].

4 Selection of the Dataset

The dataset was carefully constructed to simulate lists of registered voters who cast ballots in two states. Instead of names, each list uses unique alphanumeric identifiers (e.g., `ZWGGUMTHHF`) representing each voter, creating a dataset that resembles actual voter registries without revealing personal information.

To test the PSI protocol's performance across different dataset sizes, we created five versions of the dataset with varying sizes: 10,000, 20,000, 50,000, 100,000, and 500,000 records. Each version includes two independent lists, one for each state, with approximately 10% overlap to simulate potential cases of duplicate voting across state lines. The remaining 90% are unique to each list, representing voters who only cast ballots in one state.

This dataset structure enables us to observe how effectively PSI can identify duplicate voter IDs across varying dataset scales, demonstrating the protocol's practical utility in privacy-preserving election security and collaboration across state election agencies.

5 Execution time

Samples	Protocol 0	Protocol 1	Protocol 2	Protocol 3
10,000	0.5	7.9	34.2	1.7
20,000	1.4	9.4	61.3	1.5
50,000	2.8	8.8	150	2
100,000	6	11	321	2.5
5,000,000	34.4	46.1	1294	15.6

Table 1: Execution time (s) for the different protocols depending on the number of samples

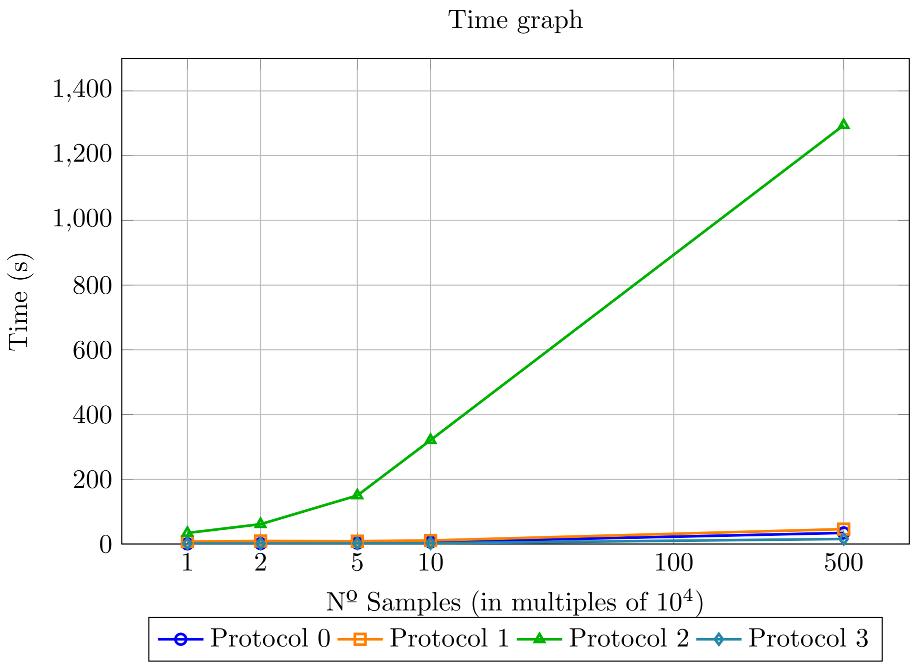

Figure 2: Graph of the time required for the different Protocols as a function of the number of samples

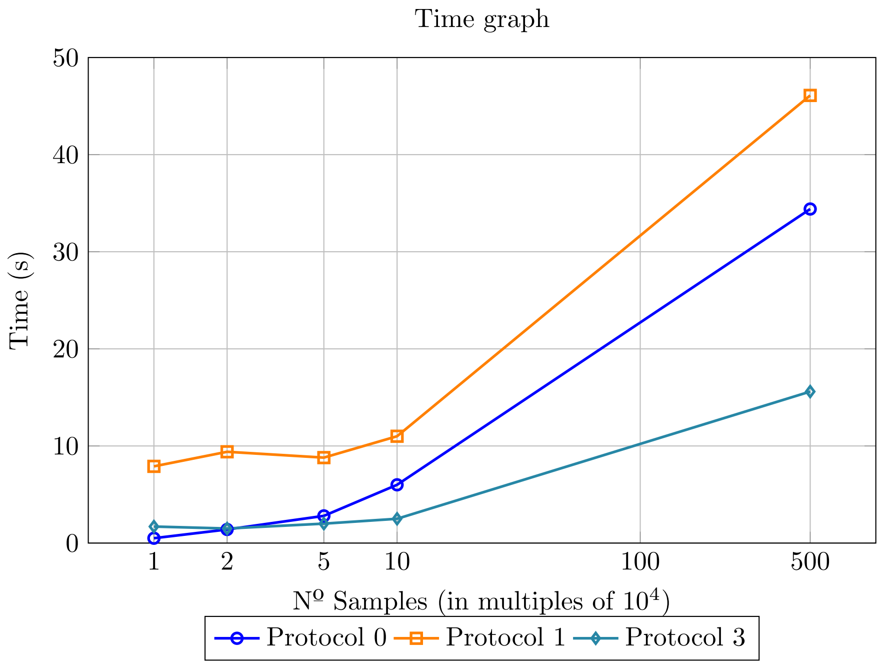

Figure 3: Time graph of the different Protocols as a function of the number of samples (without Protocol 2)

6 Data exchanged

Samples	**Protocol 0**	**Protocol 1**	**Protocol 2**	**Protocol 3**
10000	0,2	0,2	0,11	1,1
20000	0,4	0,3	2	2
50000	0,8	0,8	5,1	5
100000	2	1,5	10,1	10,2
5000000	9,6	7,7	50,5	50,9

Table 2: Table of the exchanged date (MB) of the different Protocols according to the number of samples

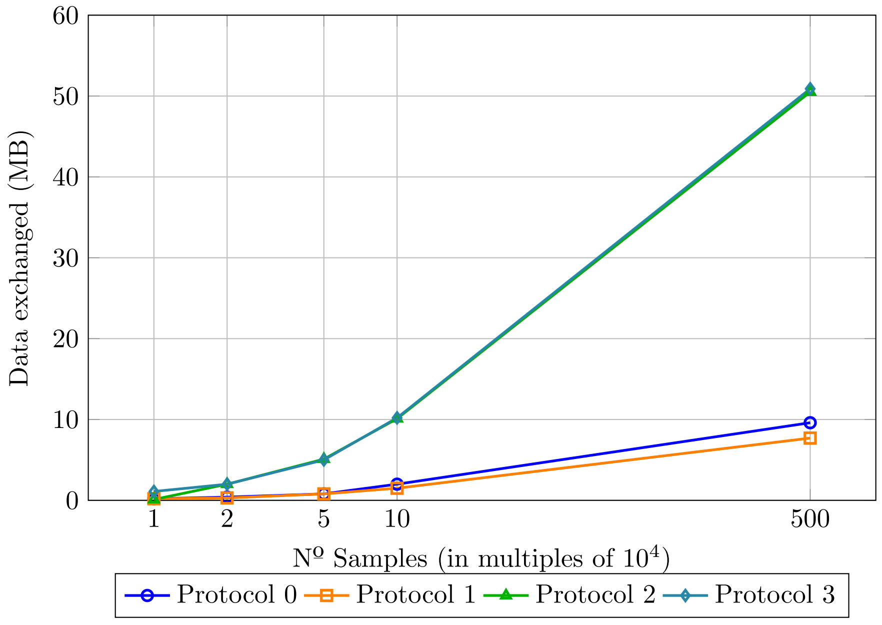

Figure 4: Graph of the date exchanged (MB) as a function of the number of samples

7 Results and Discussion

7.1 Time Analysis

The time performance across protocols varies widely, with key differences in scalability and efficiency that make some protocols more suited for larger datasets. Additionally, each protocol will be analyzed to gain deeper understanding.

• **Protocol 0 (Naive Hashing)**:Protocol 0 demonstrates a linear growth in time consumption, starting at 0.5 seconds for 10,000 samples and reaching 34.4 seconds for 5,000,000 samples. This protocol remains relatively efficient across increasing dataset sizes due to its simple hashing mechanism, making it suitable for scenarios where moderate security is sufficient and time efficiency is needed.

• **Protocol 1 (Server-Aided)**: Protocol 1 requires coordination with an intermediary server, which results in higher time consumption than Protocol 0, especially with smaller datasets. It begins at 7.9 seconds for 10,000 samples and scales to 46.1 seconds for 5,000,000 samples. The server-aided structure adds an overhead but allows the protocol to handle large datasets more consistently, making it suitable for moderate time-sensitive applications when a trusted server is available.

• **Protocol 2 (Diffie-Hellman-Based)**:Protocol 2, using the Diffie-Hellman key exchange, is the most time-intensive. It starts at 34.2 seconds for 10,000 samples and escalates significantly to 1,294 seconds (over 21 minutes) for 5,000,000 samples. The complex Diffie-Hellman operations lead to higher time costs, especially with larger datasets, making this protocol less suitable for high-volume, time-sensitive applications but valuable for smaller datasets where security is prioritized over time efficiency.

• **Protocol 3 (Oblivious Transfer-Based)**:Protocol 3 shows excellent time efficiency across all dataset sizes, requiring only 1.7 seconds for 10,000 samples and scaling up to 15.6 seconds for 5,000,000 samples. This makes Protocol 3 ideal for large-scale, time-sensitive applications. Its use of Oblivious Transfer provides strong privacy protection while keeping computational requirements low, making it well-suited for applications where quick processing is essential.

7.1.1 Summary of Time Efficiency

The protocols vary in time efficiency, guiding their ideal applications.
• **Most Time-Efficient**:Protocol 3 (Oblivious Transfer-Based) demonstrates the highest time efficiency, making it the best choice for large datasets requiring rapid processing.
• **Moderately Time-Efficient**:Protocol 0 (Naive Hashing) also performs well time-wise, especially for small to medium datasets due to its simpler approach.
• **Higher Overhead:** :Protocol 1 (Server-Aided) shows moderate performance with added server coordination overhead, performing best when server interaction is feasible.
• **Most Time-Intensive** :Protocol 2 (Diffie-Hellman-Based) has the longest processing times, making it more suited for smaller datasets where security is paramount.

7.2 Data Exchanged Analysis

The amount of data exchanged also varies significantly across protocols, highlighting each protocol's efficiency and suitability based on data transmission requirements.
• **Protocol 0 (Naive Hashing)**: This protocol has a linear increase in data exchanged, from 0.2 MB at 10,000 samples to 9.6 MB at 5,000,000 samples. Its simplicity in hashing all entries for transmission can lead to inefficiencies for high-volume data, making it less suited for very large datasets where data exchange is a primary concern.
• **Protocol 1 (Server-Aided)**: Protocol 1 shows consistently lower data exchange than Protocol 0, particularly as dataset sizes grow. Starting at 0.2 MB for 10,000 samples and reaching 7.7 MB at 5,000,000 samples, it leverages the server to limit data transfers between parties. This efficiency in data exchange makes Protocol 1 a strong choice for large datasets when a semi-trusted server is available.
• **Protocol 2 (Diffie-Hellman-Based)**:Protocol 2 is relatively efficient with smaller datasets (0.11 MB for 10,000 samples) but becomes data-intensive with larger ones, reaching 50.5 MB for 5,000,000 samples. The increased data transmission requirements are due to the encryption demands of Diffie-Hellman, making it less efficient for large-scale applications where data exchange is critical.
• **Protocol 3 (Oblivious Transfer-Based)**:Protocol 3 follows similar data exchange trends as Protocol 2, reaching 50.9 MB with 5,000,000 samples. Although slightly more data-intensive than Protocols 0 and 1, it offers high privacy protections that may justify the additional data exchange in sensitive applications.

7.2.1 Summary of Data Efficiency

• **Most Data-Efficient**:Protocol 1 (Server-Aided) is the most data-efficient protocol, especially in high-scale scenarios where data exchange must be minimized.
• **Moderately Data-Efficientt**: Protocol 0 (Naive Hashing) follows closely, making it viable for moderate data efficiency needs.
• **Higher Data Exchange**:Protocols 2 and 3 (Diffie-Hellman and Oblivious Transfer) are more data-intensive, but their strong privacy guarantees may justify their use in applications requiring heightened data protection.

8 Conclusion

In conclusion, the analysis of Private Set Intersection (PSI) protocols highlights their varying strengths in terms of time efficiency and data management for electoral security. Protocol 3 (Oblivious Transfer-Based) is the best choice for time-sensitive applications with large datasets due to its rapid processing capabilities and strong privacy protection. Protocol 1 (Server-Aided) excels in data efficiency, making it suitable for large-scale scenarios where minimizing data exchange is critical. Protocol 0 (Naive Hashing) provides a balanced option for small to medium datasets, while Protocol 2 (Diffie-Hellman-Based) is best for high-security needs, despite being the most time-intensive and data-heavy. Ultimately, the selection of a protocol should align with specific requirements, ensuring the integrity of elections in the digital age.

References

[1] Whitfield Diffie and Martin E. Hellman. “New Directions in Cryptography”. In: *IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654. DOI: 10.1109/TIT.1976.1055638.

[2] Shimon Even, Oded Goldreich, and Abraham Lempel. “Efficient Oblivious Transfer Protocols”. In: *Journal of Cryptology* 14.3 (2001), pp. 289–302. DOI: 10.1007/s001450100083.

[3] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. “Privacy-Preserving Set Intersection”. In: *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing*. 2010, pp. 114–123. DOI: 10.1145/1806689.1806713.

[4] Hugo Krawczyk. “Cryptographic Hash-Based Authentication Protocols”. In: *Journal of Computer Security* 2.2-3 (1994), pp. 109–126. DOI: 10.3233/JCS-1994-2302.

[5] Benny Pinkas et al. “Efficient Private Matching and Set Intersection”. In: *EUROCRYPT*. 2004, pp. 1–19. DOI: 10.1007/978-3-540-24676-3_1.