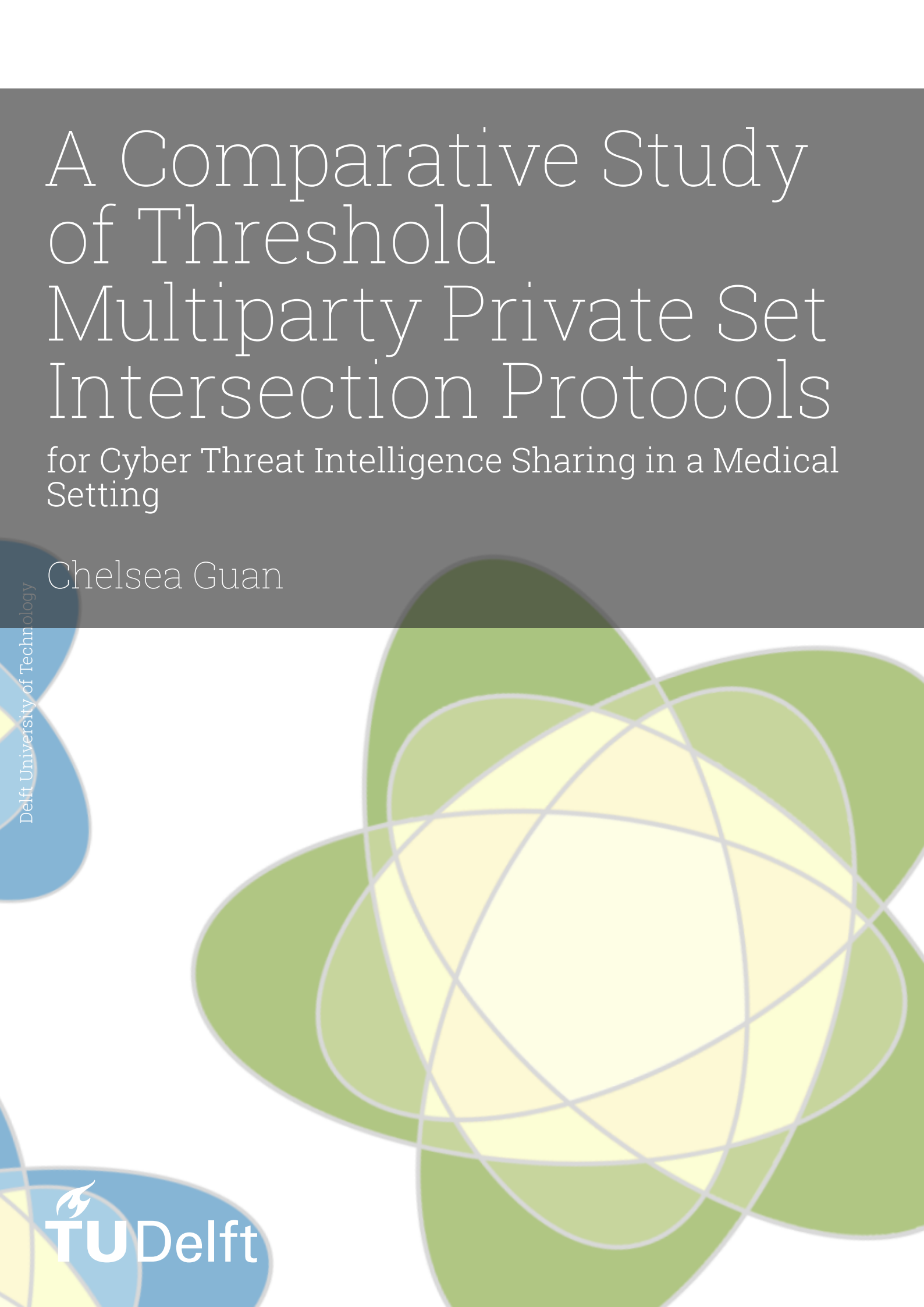
A Comparative Study of Threshold Multiparty Private Set Intersection Protocols

for Cyber Threat Intelligence Sharing in a Medical Setting

Chelsea Guan

Delft University of Technology



TU Delft

A Comparative Study of Threshold Multiparty Private Set Intersection Protocols

for Cyber Threat Intelligence Sharing in a Medical Setting

by

Chelsea Guan

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday July 2, 2024 at 10:00 AM.

Student number:	5695481
Project duration:	November 14, 2023 – July 2, 2024
Thesis committee:	Dr Z. Erkin, TU Delft, supervisor
	Dr G. Migut, TU Delft

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

Thesis contents were enhanced with the assistance of ChatGPT

TU Delft

Abstract

Within the field of *cyber threat intelligence* (CTI), healthcare institutions are one of the most targeted organizations by cybercriminals. To mitigate future attacks on their digital infrastructures, healthcare institutions can collaborate and exchange security logs. These logs include data such as IP addresses, malware hashes, and other indicators of compromise. By identifying shared elements across different datasets, threats that are harmful to a greater number of organizations—and thus, pose a more significant risk—can be highlighted to detect common attack patterns. These attack patterns could then provide insight into understanding how cyber criminals operate on a larger scale. However, disclosing locally collected CTI could compromise a hospital's security posture and reputation since it reveals vulnerabilities or attack techniques used by hackers. Furthermore, this cyber threat data is often sensitive. That is why the threat data needs to be shared in a privacy-preserving manner. *Multiparty private set intersection* (MPSI) is a solution that allows parties to find the intersection of all their sets without learning anything of the other inputs. Although, in many cases, the condition that an element be present in all sets is too restrictive. A potential threat is still worth investigating even if it only appears in a portion of sets. Therefore, we focus on *threshold multiparty private set intersection* (T-MPSI). However, not all T-MPSI schemes perform equally depending on the context. Our goal is to determine what makes a T-MPSI protocol effective for application in medical CTI sharing. To do so, we analyze four state-of-the-art T-MPSI protocols in terms of security, theoretical communication and computational complexities, and practical runtime performance.

Preface

This thesis marks the end of my two short yet memorable years at TU Delft, during which I was lucky enough to meet many wonderful people.

First, I wish to extend my heartfelt gratitude to my thesis supervisor, Dr. Erkin, for his guidance and encouragement throughout the process. It was during his Security and Cryptography course in my first quarter in Delft that I developed an interest for the field, ultimately leading me to write my thesis under his supervision.

Additionally, I would like to thank the entire CYS group for fostering such a pleasant environment for learning and socializing with the guest talks, birthday cakes and coffee chats in Echo CS offices. I am particularly grateful to Jorrit, my daily supervisor, for discussing ideas with me and providing valuable feedback. Thanks also to Tianyu, Florine and Jelle for their helpful suggestions over these past months. And, of course, my thesis experience would not have been as enjoyable without the company of the other master's students, Dāvis, Prakhar, Vojta, Juno, Dan.

Special thanks to Alex, Bogdan, Matteo, Bobe, Roy and Hendy for making my university experience here extra fun. I hope we can still have our lunches, dinners and various outings together post-graduation.

Last but not least, I want to express my deepest appreciation to my friends and family back home for their unconditional love, my parents and grandparents especially, whose unwavering support has been the cornerstone of all my successes.

Chelsea Guan
Delft, June 2024

And a small shoutout to my dog for being my dog.

Contents

1 Introduction

With the rapidly changing cyber threat landscape, advancements in *cyber threat intelligence* (CTI) must continue to be made to keep up with the evolving tactics of attackers. Threats range from ransomware attacks that lock access to critical patient records to phishing campaigns aimed at stealing sensitive data. One of the most targeted sectors is healthcare. In the past year in the EU, incidents concerning healthcare centres account for 8% of the total number, making it the third most attacked sector after public administration and targeted individuals [1]. The digitalization of medical files, the use of networked medical devices—internet of medical things—and the increasing reliance on cloud services have collectively enhanced the efficiency and accessibility of healthcare services [2]. This digital transformation has also expanded the attack surface for criminals. Cyber attacks on hospitals and other similar institutions are not just prevalent but increasingly sophisticated, leveraging the vulnerabilities in the interconnected ecosystems of hospitals [2, 3]. Furthermore, health data is especially targeted since they have more monetary value than other types of data. Indeed, financial gain was the objective of 83% of the threat actors, most of which are cyber criminals [4]. Medical records contain enough personal information to commit identity theft, allowing malicious users to access healthcare services and medications, open bank accounts, as well as acquire passports in the victims' names [4, 3].

1.1. Cyber Threats to the Medical Domain

The strategy primarily employed by these criminals is ransomware, a type of malware designed to block access to a computer system or to encrypt files on the system until a ransom is paid to the culprit. A notable ransom attack, WannaCry, occurred in 2017. The perpetrators exploited a Windows vulnerability to infect over 200,000 computers across at least 16 healthcare institutions, encrypting their data in exchange for ransom [5]. Recently, the phenomenon of ransomware-as-a-service has become increasingly problematic as it enables even more attackers to target hospitals. In this business model, developers of ransomware market their code to other individuals with malevolent intentions but who lack the technical skills to write the malicious software themselves. These buyers or subscribers are denoted affiliates [6].

Distributed denial-of-service attacks are also a prime tactic threat actors use. These attacks disrupt the normal traffic of a victim by leveraging multiple compromised systems—e.g., computers and IoT devices—to send traffic to a single victim. This generates a level of demand that overwhelms the victim's ability to respond, leading to service degradation or complete service unavailability for legitimate users [7]. Consequently, a distributed denial-of-service attack on software-enabled medical equipment or on websites critical for coordinating vital operations

can thus interrupt business continuity by preventing patients and staff from accessing these essential resources [4].

These attacks have detrimental effects on society. Firstly, as mentioned previously, they lead to the disruption of critical services, thereby hindering access to electronic medical records, diagnostic services, and other essential healthcare operations. Such disruptions can cause delays in patient care, lead to the cancellation of procedures, and in severe instances, may jeopardize patient safety [4]. Secondly, breaches resulting in stolen confidential files are especially harmful in the health sector because of the sensitivity of healthcare data. A patient's medical record contains personal information like their name, date of birth, details of their insurance provider, as well as health and genetic data. Once such private information is exposed, it becomes impossible to reclaim privacy or to mitigate the psychological and social damages inflicted [8]. Finally, the financial losses incurred by such cyber threats are substantial. These include the ransom payment, attack remediation costs, incident handling costs, legal fees, etc. [4].

1.2. Cyber Threat Intelligence

Therefore, the role of CTI in protecting hospitals has never been more important. CTI involves the collection, analysis, and dissemination of information about potential or current cyber threats and vulnerabilities [9]. The enhanced situational awareness supplied by this data allows for proactive defence, risk management and rapid incident response. It provides insights into evolving threats, enabling organizations to prioritize security measures and allocate resources effectively [10]. IBM [11] identifies three main types of threat data:

- **Tactical threat intelligence** revolves around *indicators of compromise* (IoCs) such as infected IP addresses, malware hashes, domain names of botnet command and control servers, and email headers indicative of a phishing attack. In addition to helping configure security tools and systems which block or detect the latest known threats, it also assists threat-hunting teams in uncovering hidden adversaries.
- **Operational/Technical threat intelligence** focuses on the *tactics, techniques, and procedures* (TTPs) employed by cyber criminals. This includes information about their attack methods, their objectives and the security weaknesses they leverage. This contextual data helps organizations anticipate and prepare for specific types of threats.
- **Strategic threat intelligence** involves trends in cyber threats, geopolitical developments affecting cyber security, emerging technologies as well as their potential security implications. It provides a high-level view of the threat landscape for executives without a technical background, thereby enabling them to make informed decisions about resource allocation, risk management, and compliance.

Hospitals can even further improve their preemptive and reactive cyber security measures by sharing CTI, allowing organizations to make informed decisions to protect patient data and critical healthcare services. As cyber security threats evolve, the ability to share known threats becomes more important, to the point where stakeholders can be held accountable for failing to share information that could have prevented a breach. The primary goal of CTI sharing is to foster situational awareness among stakeholders, enabling them to collectively mitigate future attacks and strengthen their defences against the ever-changing cyber threat landscape [12].

By identifying shared elements across different security logs, threats that are harmful to a greater number of organizations—and thus pose a more significant risk—are highlighted to detect common attack patterns. For instance, multiple institutions may keep a list of suspected malicious IP addresses. By comparing these suspicious IPs, they are able to narrow down their search for the culprit. Once they do, they can collectively block this IP and enhance their

defences against the associated malware. Another example is when several organizations share TTPs, they find that the same vulnerability in a common software was exploited. Thanks to this insight, they can prioritize patching that vulnerability to avoid potential breaches against this specific attack vector.

However, information related to CTI is often personal and protected. For instance, under the General Data Protection Regulation, a set of data protection rules in the European Union and the European Economic Area, an IP address is considered personally identifiable information [13]. Disclosing locally collected CTI could compromise a hospital's security posture by unintentionally advertising vulnerabilities in their systems or attack techniques used by hackers, potentially inviting further exploitation by other malicious actors aware of these weaknesses. Moreover, public knowledge of past breaches may erode patient trust, affecting a hospital's reputation for safeguarding sensitive health information [14]. For these reasons, the exchange of threat data must be done in a privacy-preserving manner to avoid leaking any additional information.

Anonymization is one of many ways to integrate privacy protections into CTI data handling. Anonymization describes the process of modifying a dataset so that individual records cannot be linked back to a single person. This is achieved by removing or altering personally identifiable information such as names, addresses, social security numbers, and other unique identifiers. One technique is K-anonymity which ensures that each record is indistinguishable from at least $K - 1$ others, making it harder for attackers to re-identify individuals based on the published information [15]. The prime objective of anonymization is to prevent the identification of these individuals from shared data. *Differential privacy* (DP) is another technique to ensure the privacy of individuals within a dataset [16]. It provides strong guarantees that the output of a computation cannot be used to infer much about any individual's data in the dataset by adding noise to the data or the computation's results. Once the CTI records have been anonymized or noise has been added, organizations can exchange this data without compromising individual privacy. While anonymization and DP effectively protect the privacy of the data subjects whose personal information is included in the dataset, it is equally important to protect the privacy of the data owners. Data owners are the organizations which collect, manage, and share threat intelligence. If adversaries can trace the owner of the shared CTI information, they might exploit this knowledge for targeted attacks. Identifying the sources of CTI also exposes these entities to reputational damage if the information is leaked.

A solution that considers the privacy of the data owners is *secure multiparty computation* (MPC). In MPC, various parties perform calculations over their private inputs without revealing said inputs [17, 18]. MPC would allow multiple hospitals' CTI teams to pool their threat intelligence to find only the common risks without disclosing any additional vulnerabilities unique to their respective datasets, thereby preserving the privacy of the data owners. The privacy of the data subjects is taken into account as well, since when executed correctly, MPC should only expose records belonging to every participating hospital's CTI dataset, i.e., records relating to the shared threat. Other users' data should remain confidential. By differentiating and addressing the privacy needs of data subjects and owners, comprehensive protection can be achieved, ensuring both individual privacy and organizational security.

MPC computes precise results without altering the input data. In contrast, because anonymization and DP must modify the input datasets to ensure privacy before performing computations over them, they produce approximate results, reducing utility. The accuracy of the intersection set is crucial because it ensures that only relevant threat intelligence is identified and acted upon, minimizing false positives and focusing efforts where they are most needed. MPC also provides robust security guarantees security in different adversarial models depending on the trustworthiness of parties. In the semi-honest model, participating parties are curious, i.e., they

try to find out as much about the other private sets as they can, but honest, i.e., they do so while following the protocol faithfully [19]. In the malicious model, corrupt parties may deviate from the protocol in an attempt to learn more about the other party's set. They can abort the protocol early and substitute their local inputs [19]. Designing protocols that are secure in the malicious model is generally more complex and computationally expensive than those secure under the semi-honest model. Still, they provide stronger security guarantees, making them essential for applications requiring high levels of trust and security.

1.3. Multi-Party Private Set Intersection

Multiparty private set intersection (MPSI), a type of MPC, is a cryptographic protocol that allows multiple parties to collaboratively compute the intersection of their private sets without revealing any additional information about the other elements of the sets [20]. It is an expansion of the two-party *private set intersection* (PSI) first introduced in 2004 by Freedman et al. [21]. Since then, MPSI has evolved significantly in terms of efficiency, scalability, and security. Early schemes were computationally intensive and not scalable for large sets. Over the years, researchers have introduced more efficient protocols that reduce computation and communication costs, making MPSI practical for large datasets of up to 2^{20} items between 32 parties [22].

One of its potential applications is in targeted advertising, where many stores may wish to collaborate in a joint promotional campaign [23, 20]. They would need to determine the customers they have in common while ensuring the privacy of other clients is not compromised. Other applications of MPSI include setting a suitable meeting time between private calendars [24], securely identifying mutual contacts in messaging apps [25], private contact tracing of infectious diseases [26, 22], and of course, usages in cyber security such as CTI incident information sharing [25] or collaborative intrusion detection [22].

The requirement in MPSI that an element must be present in all sets to appear in the output may not fit every scenario. In our application of CTI sharing, this condition is overly restrictive. Consider, for example, five hospitals privately sharing CTI in order to determine which of the suspected IPs are most likely to be malicious. If the same IP address appears in every set, then there is a high probability that it is a threat. Furthermore, even if it is not found in all the sets but in a large enough number, e.g., four out of the five datasets, then it is still worth investigating. Therefore, in this situation, the participating parties should calculate the intersections of subgroups of four sets rather than the absolute intersection. This type of intersection of subgroups is called *threshold MPSI* (T-MPIS). T-MPSI is an MPSI protocol which returns items present in a minimum number of sets as specified by the threshold, T [27, 28, 26]. In some papers, it is referred to as over-threshold MPSI (OT-MP-PSI) [28], d-and-over MPSI [23] or quorum PSI (qPSI) [29]. Note that there are many other definitions of T-MPSI which complicates research in the field. Alternatively, the threshold can denote the minimal cardinality of the intersection [30, 31] where the result of the intersection is only outputted if the size is large enough. The threshold can also denote the maximal difference between private set sizes [32]. Both these definitions are convenient for cases where the intersection is only worth computing if the parties have enough in common, e.g., in ride-sharing, where a group of people will only opt to carpool if the route they would share is long enough. To avoid confusion, from now on, we refer to these other types of threshold PSI protocols as T'-MPSI.

Even within this definition of T-MPSI, there are multiple perspectives to computing the output intersection:

- **T-MPSI$_{\text{individual}}$**: A party i wishes to obtain the elements of their set i that are also present in at least a threshold $T-1$ of the other $n-1$ parties' sets [27, 23, 26, 29]. This is denoted as "threshold contribution" by Kissner and Song in [27].

- **T-MPSI$_{\text{all}}$**: Similar to T-MPSI$_{\text{individual}}$, except that at the end of the protocol, every party learns which elements are in their private set and at least $T-1$ other sets [28]. It is equivalent to computing the T-MPSI$_{\text{individual}}$ for every party i.
- **T-MPSI$_{\text{collective}}$**: All players learn which elements appear at least a threshold number of times T in the combined private input of the players [27]. This is denoted as "perfect threshold" by Kissner and Song in [27].

1.4. Application in a Hospital Setting

Although many constructions are proposed for T-MPSI, it remains unclear which would be best suited for the specific use case of CTI sharing among healthcare institutions. Implementing a T-MPSI protocol that is convenient for CTI sharing between hospitals necessitates addressing requirements unique to the healthcare environment. Therefore, several assumptions are made for practicality. We begin by defining specific roles for the entities involved. The participating parties in the T-MPSI protocol are hospitals—more specifically, the IT and security teams hired by those hospitals, which are presumed trustworthy. Given this assumption, it suffices that the construction be secure under a semi-honest threat model. Another requirement is that the T-MPSI protocol should perform efficiently in a star network topology. Implementing the protocol in a star topology is desirable since most systems are designed with a client-server architecture facilitating deployment. In this communication structure, each assistant shares a bidirectional communication channel with the leader but none with the other assistants [24]. This minimizes the number of communication channels needed. The maximum redundancy offered by mesh topology when malicious parties refuse to communicate is not necessary since parties are trusted hospitals. The exchange of CTI between healthcare institutions would occur between a smaller number of parties, m, but each party would have a large number of elements, n, the elements being the CTI data comprising of IoCs, TTPs, etc. Consequently, the solution should scale well when n is much bigger than m. Lastly, to ensure broad accessibility among health centres, we require that the protocol be able to run on a system with the computational power of a high-end laptop.

1.5. Research Challenge

Design choices such as the security model, the set representation and the algorithm for computing the intersection all affect the usability of the T-MPSI protocol in a medical setting. The specific demands of healthcare CTI sharing underline the need for an in-depth analysis that evaluates the performance and security of these protocols. This leads to our research question:

> *How can we identify the best method to efficiently compute the threshold intersection of hospitals' CTI datasets in a privacy preserving manner to identify recurring cyber threats?*

Although there have been many T-MPSI protocols introduced in recent years [26, 29, 28, 33, 34], there have not been—to the best of our knowledge—any dedicated works comparing various T-MPSI constructions. This lack of comparison studies makes it more difficult to identify a suitable protocol for our specific use case. Furthermore, a problem in the field of threshold PSI is the lack of a standardized definition, which makes finding relevant research difficult.

1.6. Contributions

To address this gap, we conduct a comparison study of the state-of-the-art T-MPSI protocols to highlight each protocol's key features and its applicability to a hospital setting. Our contributions are as follows:

1. We present a qualitative analysis of four state-of-the-art T-MPSI protocols, taking into consideration the unique circumstances and stakeholders of the medical setting.
2. We conduct a more in-depth practical performance evaluation of two state-of-the-art T-MPSI protocols, [28] and [26] by implementing the protocols and performing runtime experiments under chosen parameters.
3. We find that [28] is more efficient for smaller numbers of parties and thresholds, $m = 5, 6$ and $T = 2, 3$, and [26] is more efficient for larger number of parties and thresholds, $m = 7, 8, 9$ and $T = 4, 5$. However, no existing T-MPSI protocol is efficient enough to be applied in CTI sharing since threat datasets contain millions of rows, but the compared protocols scale poorly for numbers of elements beyond $n = 4096$.
4. As of now, T-MPSI$_{\text{individual}}$ and T-MPSI$_{\text{collective}}$ protocols are not directly comparable in terms of computational and communication complexities. Therefore, we show a generic composition that can be applied to any T-MPSI$_{\text{individual}}$ type protocol to turn it into a T-MPSI$_{\text{collective}}$ type protocol. Unifying both variants of T-MPSI would broaden the operational scope of the T-MPSI$_{\text{individual}}$ protocols since we demonstrate that the execution times of the modified protocol are still practical.

1.7. Outline

The remainder of the thesis is structured as follows. Chapter 2 covers the necessary background to understand the cryptographic protocols and their security guarantees. Chapter 3 provides an overview of the previous works in comparing various MPSI and T-MPSI schemes. Chapter 4 details the methodology for conducting the qualitative comparison of state-of-the-art T-MPSI constructions. Chapter 5 discusses the qualitative analysis for four of these constructions. Chapter 6 goes over the implementation details of two out of the four protocols. Chapter 7 compares them in terms of runtime speed. Chapter 8 demonstrates how we can extend any T-MPSI$_{\text{individual}}$ protocol into a T-MPSI$_{\text{collective}}$ protocol. Finally, Chapter 9 discusses the results, proposes possible future works in the domain and concludes the thesis.

2

Preliminaries

We now introduce the necessary background and definitions for understanding the T-MPSI protocols we are comparing. It includes an overview of different security considerations under semi-honest and malicious models, methods for representing sets in computations, and the cryptographic primitives that enable secure multiparty computation.

2.1. Security Models

This section introduces the security models used to evaluate the safety and integrity of cryptographic protocols. The security model defines the assumptions under which the system is analyzed, outlines the capabilities of potential adversaries and establishes the goals of security measures—all of which are essential for assessing the applicability and strength of a protocol in various real-world scenarios.

2.1.1. Semi-Honest Security

The semi-honest security model is also referred to as ”honest-but-curious” or ”passive” security. In this security model, adversaries might collude to attempt to learn as much information as they can, i.e., curious, but they do so while following the protocol faithfully, i.e., honest [19, 35]. Although semi-honest parties correctly perform all computational instructions and communication requirements, they still try to derive extra knowledge about the honest parties' private sets by analyzing the timing, size, and pattern of messages they legitimately receive during the protocol execution, including intermediate computations and received messages.

In this model, the focus is on ensuring that even though parties may attempt to learn additional information, the protocol itself limits the leakage of sensitive data.

2.1.2. Malicious Security

In the malicious security model, corrupt parties may deviate from the protocol in an attempt to learn more about the other parties' set. They can abort the protocol early, alter their local inputs or perform any other actions that could potentially compromise the integrity or confidentiality of the computation [19]. For instance, a dishonest party could alter their input by adding fake elements that they speculate might be in the honest parties' sets. This would allow them to verify their guesses based on the intersection results, effectively learning information outside the scope of the protocol. Protocols designed under this model incorporate mechanisms to detect and mitigate such behaviours.

Designing secure protocols in the malicious model is generally more complex and computationally expensive than those secure under the semi-honest model, but they provide stronger

security guarantees, making them essential for applications requiring high levels of trust and security [26].

2.2. Set Representations

In this section, we present common set representations used in the cryptographic protocols which we will be evaluating.

2.2.1. Bloom Filters

Bloom filters (BFs) are a probabilistic data structure designed for rapid set membership queries with a constant time complexity, independent of the number of elements in the set. Conceived by Burton Howard Bloom in 1970 [36], the main advantage of a BF is its extreme space efficiency compared to other data structures like hash tables, lists, or trees. However, this comes at the cost of potentially yielding false positives. This error rate can be adjusted; a larger bit array and more hash functions reduce the probability of false positives but consume more space and use more computation during insertions and queries. BFs do not produce false negatives—if any bit is not set to 1, it indicates that the element was never added.

A BF, denoted as $BF = (BF[0], \ldots, BF[j], \ldots, BF[m-1])$, represents a set S with a maximum of n elements within a binary vector of length m. It is implemented using a bit array of m bits, all initially set to 0. k randomly chosen hash functions are used to uniformly map some set element to one of the m positions. These hash functions are denoted as $h_1, h_2, \ldots, h_k$, where each $h_i : \{0,1\}^* \to \{0, 1, \ldots, m-1\}$.

The key functionalities of BFs are as follows:

- **Adding an element x**: The element x is hashed by each of the k hash functions, and the corresponding m positions in the bit array are set to 1. The insertion process ensures that the BF can represent the presence of all elements in S without storing the elements themselves, thereby achieving significant space savings (space-efficient solution).
- **Verifying the presence of an element**: To query an item, we hash it with the same hash functions, and the bits at the resulting positions are checked. If any of the bits are 0, the element is definitely not in the set. If all are 1, the element might be in the set with a probability of $p_{k,n,m}$, where $p_{k,n,m}$ is the false positive rate for a BF with k hash functions and at most n elements per bit array of size m.

A false positive occurs when all k bits for a non-existent element are set to 1. We can estimate the upper bound for the probability of this event occurring. Bay et al. use the upper bound estimation of the false positive probability introduced by Bose et al. [37], which is what we will present here.

We define p as the probability that a particular bit in the BF is 1, which is expressed by the following equation,

$$p \stackrel{\text{def}}{=} Pr(B_i = 1)1 - \left(1 - \frac{1}{m}\right)^{kn}. \tag{2.1}$$

The upper bound false positive probability ϵ can thus be modelled by

$$\epsilon_{upper} = p^k \left(1 + O\left(\frac{k}{p}\sqrt{\frac{\ln m - k \ln p}{m}}\right)\right), \tag{2.2}$$

where k is the number of hash functions, n is the number of elements, m is the size of the bit array and p is the probability that a certain bit is set to 1, as observed above.

The lower bound is represented by

$$\epsilon_{lower} = p^k. \tag{2.3}$$

Then, the false positive rate of a BF, $p_{k,n,m}$, is

$$p^k \leq p_{k,n,m} \leq p^k \left(1 + O\left(\frac{k}{p}\sqrt{\frac{\ln m - k \ln p}{m}}\right)\right). \tag{2.4}$$

Since we want to limit the size of the false positive rate, we ensure that m and k follow the following relations:

- Lower bound for bit array size m:

$$m \geq n \log_2 e \cdot \log_2 \frac{1}{\epsilon} \tag{2.5}$$

  where e is the base of the natural logarithm.

- Optimal number of hash functions k:

$$k = \frac{m}{n} \ln 2 \tag{2.6}$$

If we decide to set m to its smallest possible value, i.e., the lower bound from 2.5, k would be equal to:

$$k = \log_2 \frac{1}{\epsilon} \tag{2.7}$$

For example, $k = 40$ hash functions are required to achieve a maximum false positives rate of $\epsilon = 2^{-40}$.

Bay et al. also utilize *encrypted Bloom filters* (EBFs), as described by Davidson et al. in [38]. EBFs use cryptographic methods to encrypt each bit. More formally, given the BF of a set S, $BF[j]$, its EBF is $EBF[j] = Enc_{pk}(BF[j])$, where pk represents the public key of a *public key encryption* (PKE) scheme. Public key cryptography uses two keys, a public key and a private key, for encryption and decryption processes [39]. The public key is shared openly, allowing anyone to encrypt messages, while the private key is kept secret by the recipient to decrypt these messages securely.

2.2.2. Cuckoo Hashing

The Cuckoo hashing scheme [40] uses two or more hash functions and typically has two tables for storing items. It provides worst-case constant lookup time, as well as amortized constant time for insertions and deletions.

Cuckoo hashing scheme is named after the cuckoo bird's behaviour of hijacking other birds' nests and removing the latter's eggs from their nests. When inserting an element, the algorithm checks the first hash function's position. If it is empty, the element is placed there. If not, the element that was there is moved to an alternative position according to its second hash function, potentially displacing another element in turn. This process continues until all elements have found a place, or a loop is detected. In the latter case, the table is rebuilt with new hash functions or expanded to make room for more elements.

Its approach to collision resolution ensures that the hash table remains with high space efficiency and balanced load. However, the insertion process can be complex due to potential cyclic displacements, requiring careful implementation and occasionally necessitating a rehash of the entire table.

Cuckoo hashing is a highly efficient hashing scheme used to resolve collisions in hash tables with guaranteed high performance in lookup, deletion, and insertion operations. It employs multiple (K) hash functions, denoted as $h_1, h_2, \ldots, h_K$, which map m elements into β bins.

Cuckoo hashing uses K random hash functions $h_1, \ldots, h_K : \{0,1\}^\sigma \to [\beta]$ to map m elements into β bins.
The algorithm for insertion into a bin is presented below:

- **Insertion into an empty bin:** When inserting an element x, hash functions are applied to determine potential bins for placement:
  - The element is placed in the first available bin $h_i(x)$ for $i \in [K]$ according to lexicographic order.
  - If no bins are empty, a random bin is selected among $h_i(x)$. The current item in the bin is evicted and replaced with x.
- **Handling evictions:** If an item is evicted, the displaced item is then inserted using the same method, potentially causing a series of evictions. This process continues until all items are successfully placed or a set threshold of relocations is reached. If this threshold is exceeded, it results in a failure to insert, indicating that an element could not be accommodated after multiple attempts.

This failure signifies the existence of an element that did not map to any of the bins. Some variants of Cuckoo hashing maintain a set or a ”stash” to store such elements. Stash-less cuckoo hashing is where no special stash is maintained. Our protocols can be extended to the setting with stash. While traditional Cuckoo hashing schemes may employ a stash to handle elements that cannot be placed after multiple relocations, stash-less variants eliminate this feature. Such adaptations are critical in maintaining operational efficiency and are commonly used in modern protocols, particularly in MPSI applications.

Experimental analyses have shown that the failure probability of Cuckoo hashing can be significantly low when parameters are chosen correctly. For instance, with $K = 3, 4, 5$ hash functions and β values of $1.27m, 1.09m, 1.05m$ respectively, the failure probabilities can be maintained at extremely low levels (e.g., at most 2^{-40}).

2.3. Cryptographic Tools

This section explains the encryption schemes and other cryptographic techniques used in the T-MPSI protocols that we will analyze.

2.3.1. Secret Sharing

Secret sharing is a method for distributing a secret amongst a group of participants, each of whom is allocated a share of the secret. The original secret can be reconstructed only when a sufficient threshold number of shares are combined. Below this threshold, no information about the secret can be learned. In particular, Shamir’s secret sharing [41] is a commonly used secret sharing scheme based on the mathematical concept of polynomial interpolation, specifically Lagrange interpolation.

The steps of secret sharing are as follows:

1. **Defining a secret:** To distribute a secret S, choose a random polynomial $f(x)$ of degree $t-1$ such that $f(0) = S$. The coefficients of the polynomial are chosen randomly from a

finite field $\mathbb{F}_p$, where p is a prime number larger than the total number of shares and the secret.

$$f(x) = S + a_1 x + a_2 x^2 + \cdots + a_{k-1} x^{k-1}. \tag{2.8}$$

2. **Generating shares:** Generate n shares from the polynomial by evaluating $f(x)$ at n distinct non-zero points. Each share corresponds to a point on the polynomial,$(x, f(x))$ for participants $i = 1, 2, \ldots, n$.

3. **Reconstructing the secret:** The secret S can be reconstructed by any threshold t or more shareholders by interpolating the polynomial $f(x)$ at $x = 0$ using their shares. The core of Shamir's method lies in the polynomial interpolation formula:

$$S = f(0) = \sum_{i=1}^{t} y_i \prod_{j \neq i} \frac{x_j}{x_j - x_i} \mod p, \tag{2.9}$$

where (x_i, y_i) are the shares used in the reconstruction.

2.3.2. Homomorphic Encryption

Homomorphic encryption(HE) is a type of encryption that allows computations to be carried out on ciphertexts. The result is an encrypted result which, when decrypted, corresponds to the output of operations performed on the plaintexts. This characteristic allows for the secure processing of confidential information without giving access to the underlying data. Partially homomorphic encryption is a type of HE that allows unlimited operations, but only of a single kind. For example, additive homomorphic encryption supports the addition operation on ciphertexts.
More formally, homomorphic encryption can be defined as follows. For any two plaintexts m_1 and m_2, and a binary operation $\circ$ defined over the plaintext space, there exists a corresponding operation $\star$ defined over the ciphertext space such that:

$$Dec(Enc(m_1) \star Enc(m_2)) = m_1 \circ m_2. \tag{2.10}$$

The Paillier cryptosystem, introduced by Pascal Paillier in 1999 [42], is a popular asymmetric encryption algorithm known for its homomorphic properties, specifically its additive homomorphic encryption scheme. This allows for the encrypted sums of plaintexts to be computed directly on the ciphertexts. It also supports the multiplication of an encrypted number by a plaintext number without decryption. Furthermore, it is a probabilistic scheme, generating different ciphertexts for the same plaintext, thereby making it secure against chosen plaintext attacks.

The main advantage of the Paillier cryptosystem is its simplicity and robustness in supporting additive operations on encrypted data. It is computationally more efficient than fully homomorphic schemes for tasks that require additive properties only.

It relies on the decisional composite residuosity assumption which states that given a composite n and an integer z, it is hard to determine if z is an n-residue modulo n^2. In other words, it is hard to find a number y such that

$$z \equiv y^n \mod n^2. \tag{2.11}$$

We provide the core steps of the Paillier encryption and decryption processes below:

1. **Key generation**:

- Select two large prime numbers p and q.
- Compute $N = p \times q$ and $\lambda = \text{lcm}(p-1, q-1)$.
- Select a random integer g where g is in the set $\mathbb{Z}^*_{n^2}$.
- Ensure that g has an order that is a multiple of n in $\mathbb{Z}^*_{n^2}$.
- Publish the public key $pk = (N, g)$ and keep the secret key $sk = (p, q)$ private.

2. **Encryption**: To encrypt a plaintext message x using the Paillier cryptosystem, compute

$$\text{Enc}(x) = g^x r^N \mod N^2 , \tag{2.12}$$

where g is a generator in the group, r is a random number less than N, and N is part of the public key.

3. **Decryption**: Given a ciphertext C, it is decrypted using

$$\text{Dec}(C) = \left(\frac{L(C^\mu \mod N^2)}{L(g^\mu \mod N^2)} \right) \mod N, \tag{2.13}$$

where L is a function defined as $L(x) = \frac{x-1}{N}$, and μ is related to the private key or an intermediate calculation necessary for decryption.

For scenarios requiring distributed trust, such as in Bay et al.'s T-MPSI protocol, threshold Paillier cryptosystem [43] can be useful. It enables multiple parties to participate in the encryption and decryption processes without any individual possessing the whole private key. Threshold Paillier involves the following steps:

1. **Key generation**:
   - Compute $N = pq$ from two prime numbers p and q such that
     - $p = 2p' + 1$ and $q = 2q' + 1$ where p' and q' are two other primed not equal to p nor q, and
     - $\gcd(n, \phi(n))$.
   - Set $M = p'q'$.
   - Randomly choose $\beta \in \mathbb{Z}^*_n$.
   - Randomly choose $(a, b) \in \mathbb{Z}^*_n \times \mathbb{Z}^*_n$.
   - Set $g = (1+n)^a b^n \mod n^2$.
   - Compute $\Delta = \ell!$ where ℓ is the number of servers.
   - The public key is $(g, N, \theta = aM\beta mod N)$.
   - The private key, $\beta \times M$, is shared among the participants using Shamir secret sharing scheme. A secure polynomial sharing scheme.

2. **Encryption**: To encrypt a message X, perform the following:

$$C = g^X r^N \mod N^2, \tag{2.14}$$

where r is a random element from $\mathbb{Z}^*_n$.

3. **Share decryption**: Each server i computes its part of the decryption,

$$C_i = C^{2\Delta sk_i} \mod N^2, \tag{2.15}$$

where C is the ciphertext and $\Delta = \ell!$.

4. **Combining algorithm**: Let S be a subset of t different C_i's. We use the following equation to combine elements of S back into the original message X:

$$X = L\left(\left(\prod_{i\in S} C_i^{2\lambda_{0,i}^S}\right) \mod N^2\right)\frac{1}{4\Delta^2\theta} \mod N, \tag{2.16}$$

where $\lambda_{0,i}^S = \Delta\prod_{i'\in S\setminus\{i\}}\left(\frac{i'}{i-i'}\right) \in \mathbb{Z}$ and $L(u) = \frac{u-1}{N}$.

2.3.3. Oblivious Transfer

Oblivious transfer (OT) is a cryptographic protocol that enables a sender to transmit one of potentially many pieces of information to a receiver, without knowing which specific piece the receiver obtained [44]. The most common form, 1-out-of-2 OT, involves the sender having two messages, M_1 and M_2, and the receiver wishing to learn either M_1 or M_2 without revealing their choice to the sender. The sender remains oblivious to which message the receiver chooses while ensuring that the receiver can only learn one of the messages. In PSI, OT facilitates the secure and private exchange of data elements, ensuring that each party learns only the intersection of their sets and no additional information. By using OT, parties can perform complex set operations while maintaining data privacy and security, making it a foundational tool in privacy-preserving computations.

2.3.4. Oblivious Pseudorandom Functions

While the first two-party *oblivious pseudorandom function* (OPRF) was documented in an article by Naor and Reingold in 1997 [45], the term ”OPRF” was introduced in a paper by Freedman et al. in 2005 [46]. An OPRF builds on top of *pseudorandom function* (PRF) to allow a client to obtain the evaluation of a function on a specific input, without revealing the input or the function's output to the server performing the computation. This ensures that the server remains ”oblivious” to both the input and the output. The function output appears random and indistinguishable from a truly random function by any efficient statistical test, assuming the function's key remains secret.

A high-level general description of an oblivious evaluation is as follows [28, 47]: A PRF is defined by a function F which takes as inputs an element x and a secret key sk_{PRF}, producing an output $y = F(x, sk_{PRF})$. The server possesses the secret key sk_{PRF} while the client possesses an element x for which they wish to obtain the output y. The client first blinds the input x, transforming it into x^* using a blinding factor b. This transformation ensures that x^* appears unrelated to x. The client sends x^* to the server. The server computes the PRF on the blinded input, $y^* = F(x^*, sk_{PRF})$, and sends y^* back to the client. The client uses the blinding factor b to reverse the transformation on y^*, obtaining $y = F(x, sk_{PRF})$. This guarantees that the client learns $y = F(x, sk_{PRF})$ without knowing sk_{PRF} nor $F(x', sk_{PRF})$ for other inputs, and that the server does not deduce anything regarding x nor y.

3

Related Work

The exploration of PSI protocols has evolved significantly, with numerous studies aiming to enhance the privacy and efficiency of these cryptographic protocols. Despite the abundance of protocols available, there are limited comparative studies on the topic. This chapter discusses those few previous works that compare different PSI protocols.

3.1. Comparative Studies of PSI Protocols

Morales et al. [48] conduct a *systematic literature review* (SLR) to address several questions related to the efficiency, security, and practical applicability of various PSI protocols. Their study is significant as it categorizes and compares 76 papers based on their security models, primitives, and performance metrics, offering a valuable reference for researchers deciding which protocol to implement in different scenarios.

The study categorizes PSI protocols based on their operational principles. Classical PSI enables two parties to determine their set intersection, but the paper also expands into multi-party settings, where communication topologies significantly influence protocol efficiency. Specialized variants include PSI cardinality, which solely discloses the intersection's size; size-hiding-PSI, which conceals the cardinality of participating parties' sets; and authorized PSI which mandates prior authorization for participation. Additionally, outsourced-PSI and verifiable-delegated-PSI introduce third-party computation facilitators, enhancing practicality for complex topologies and reducing direct computational burdens on the primary parties involved. The authors also briefly mention threshold PSI, giving the schemes by [49, 50] as examples, but their definition of "threshold" differs from ours. Their threshold refers to the minimal intersection set size required to reveal the items in the intersection, whereas ours refers to the minimal subset of the private inputs which must contain the item for the intersection to be revealed.

The SLR describes the PSI schemes based on their set representation, cryptographic building blocks and hashing techniques. From the building blocks predominantly used in these protocols, HE is identified as an easy construction for PSI but is less efficient compared to OT and OPRF-based PSI, which present more competitive performance. Other building blocks like generic public key, commutative encryption, and pairings have limitations regarding protocol capacity or performance.

Various metrics, such as computation, communication, and memory usage, are employed to assess if PSI protocols are ready for realistic applications, detailing the pros and cons of existing protocols.

The review by Morales also identifies several open challenges in the field of PSI. Firstly, performance optimization remains a focus in the development of PSI protocols, particularly

through the refinement of hashing techniques and acceleration of OT processes. Secondly, while many existing protocols offer robust security against semi-honest adversaries, the challenge persists in designing efficient proofs of correct behaviour that safeguard against malicious actors without considerably impacting performance. In particular, techniques like zero-knowledge proofs and commitment schemes must incur additional overhead to maintain security. Additionally, enhancing the efficiency and scalability of MPSI protocols is essential, especially as the computational and communication demands escalate with the addition of more parties. These improvements are vital for advancing the practicality and application of PSI technologies in various real-world scenarios.

In summary, the study by Morales et al. serves as a reference for understanding the current landscape of PSI research. It provides a comprehensive overview of the developments in this field, highlighting the diversity of approaches and the range of applications impacted by PSI technologies.

Another comprehensive comparison study is conducted by Vos et al. in the form of a *systematization of knowledge* (SoK) on collusion-resistant MPSI protocols in the semi-honest model [24]. They limit their research scope to protocols that do not rely on external parties. If they did, a third party would receive all private sets, calculate the intersection, and then share the results, which means they would have access to all data, compromising the privacy of the input sets. To mitigate this, some T-MPSI constructions assume the third party is non-colluding. However, the reliability of this assumption and the inability to easily extend such protocols lead Vos to focus on protocols without any external parties. They also restrict their evaluations to the semi-honest model due to its commonality in research and the observation that the fundamental insights of a protocol generally do not change when moving to the malicious model.

Since significant aspects of MPSI protocols are more influenced by the types of set encodings used——such as polynomial roots, bitsets, garbled Bloom filters, and oblivious key-value stores——than by the cryptographic primitives themselves, they are the primary focus of the study.

Their main contribution is the categorization of MPSI protocols based on the operational principles used to construct them. The three constructions are explained below:

- **Private homomorphic set representations**: Sets are encoded to fully conceal the original inputs. These encoded sets can be homomorphically combined into a single representation, which does not reveal any original set information and can safely be disclosed to the leader for plaintext membership queries. Essentially, encode sets, combine all sets homomorphically into one in a private manner, decrypt, then query the result in plaintext. Querying set representation means checking if an element is in the set. Private homomorphic set representations include bit sets, hash sets, polynomial roots, etc.

- **Leaky homomorphic set representations**: The homomorphic operation $\odot$ used to combine the sets might inadvertently leak information about the original inputs. Therefore, the resulting set representation must remain concealed from all parties. The leader must perform membership queries privately, ensuring that only the intersection is disclosed. Encode sets, combine all sets homomorphically into one in a private manner, perform private membership queries, then decrypt. Examples of leaky homomorphic set representations are Bloom filters.

- **Aggregatable membership queries**: This method involves no homomorphic operation over the set representations. Instead, the leader employs two-party protocols to query each participating party's set separately and determine membership. The leader then

aggregates these results to reveal only the intersection. Oblivious programmable PRFs and oblivious key-value stores constitute aggregatable membership queries.

After introducing the three constructions, Vos et al. present a survey of the current state-of-the-art MPSI protocols for each type of construction, comparing them based on their cryptographic building blocks, their communication and computational complexities and other key features of the MPSI schemes. A table is used to clearly summarize all the mentioned protocols with important columns such as the set encoding used, the worst case complexities, the network topology, the collusion threshold, whether there exists an extension for the malicious security model, etc.

The SoK also identifies several common pitfalls in MPSI protocols. One issue is the leakage from set representations, where certain encoding techniques, such as Bloom filters, may unintentionally reveal information if not carefully managed. Another concern is unsafe randomness during aggregation. Incorrect usage of randomness can create vulnerabilities that allow colluding parties to infer private information. Additionally, adapting protocols designed for semi-honest settings to malicious models often presents challenges. This transition increases complexity and introduces new types of potential attacks, requiring significant modifications to ensure security.

The contribution of Vos et al. not only provides a detailed classification and evaluation of existing protocols but also sets a clear agenda for addressing the critical challenges in this field.

3.2. Comparisons of T-MPSI Protocols

Comprehensive comparative studies specifically focusing on T-MPSI are sparse. Most available comparisons are embedded within the introduction or related work sections of papers proposing new protocols. In these situations, the goal of the comparison is not to provide an in-depth analysis of the field by contrasting various protocols. Rather, it is to supply context or to benchmark the author's proposed solutions against existing approaches.

Many authors, such as Badrinarayanan et al. [51] and Chandran et al. [29] describe a few threshold PSI protocols in the introduction and related work of their papers. However, they use different definitions of threshold than the one we defined in Section 1.3. Recall that we refer to these types of threshold PSI as "T'-MPSI". In the T'-MPSI protocols presented by Badrinarayanan, the participating parties may only compute the intersection if their sets differ by a threshold amount of items or less. In the ones presented by Chandran, the parties may only output the intersection if its cardinality is above the threshold. This can be confusing, especially in the case of Chandran, since one of their main contributions is a T-MPSI solution called quorum PSI, but they discuss T'-MPSI protocols instead of T-MPSI in their related work.

Yu et al. published a paper introducing a new T-MPSI protocol [30]. Their related work presents many types of MPSI schemes, including threshold PSI. However, the information about the different variants of MPSI is poorly organized. The description of the T-MPSI protocol proposed by Mahdavi et al. [28] is intertwined amongst T'-MPSI protocols such as the ones by Ghosh et al. [50] and Branco et al. [52] without clarifying that these definitions of threshold PSI all differ. Mahdavi relates the threshold to the number of sets participating in the intersection, whereas Ghosh and Branco relate it to the number of elements of the intersection.

Similarly, in their paper on T'-MPSI where the threshold represents the minimum cardinality of the resulting intersection set, Mohanty et al. [53] goes over many previous works in the domain of threshold PSI, including [27, 50, 51, 28, 26]. However, Mohanty et al. do not clarify that their concepts of threshold PSI all vary and treat them as if they were comparable.

The related work of Kerschbaum et al.'s paper [54] contains a section on specialized PSI protocols, but the part on T-MPSI protocols is brief, only mentioning the research done by

Kissner and Song [27], as well as, Mahdavi et al. [28].

The problem with these comparisons is that they are not the focus of the paper and are only meant to provide background information on their proposed protocol. Thus, they oftentimes mention many variants of MPSI, not only T-MPSI. Being only one section of a paper also means that they do not go into enough detail analyzing the T-MPSI schemes.

4

Methodology for Qualitative Comparison

This chapter describes the selection process of state-of-the-art T-MPSI protocols since the first step to a proper comparison study is establishing a body of literature that is both exhaustive and relevant. To demonstrate the thoroughness of our approach, we outline the steps of our methodology below. The challenge in compiling relevant papers arises from the numerous definitions of T-MPSI. Various forms of MPSI are referred to as T-MPSI despite significant differences in what the ”threshold” denotes and the outcomes they yield. For example, in some contexts, the threshold may indicate the minimum number of datasets that must contain an item for it to appear in the intersection, whereas in others, it might refer to the minimum or maximum size of the intersection itself. Consequently, a meticulous review of the literature is essential to identify all T-MPSI protocols pertinent to our study.

4.1. Gathering Relevant Studies

Our comparison study begins by identifying the state-of-the-art papers in the domain of T-MPSI that would be best suited for application to the hospital setting. The first step is to familiarize ourselves with the keywords by locating relevant academic articles using Google Scholar[1]. Google Scholar’s ability to automatically include related search terms means it provides broader search capabilities than digital libraries that look for exact keywords. However, Google Scholar also includes non-peer-reviewed sources, such as theses, preprints, and technical reports, which means it requires careful screening to ensure the academic rigour of selected sources. Therefore, Google Scholar functions more as a preliminary tool to compile a list of appropriate keywords for subsequent exploration in a library database rather than as the sole repository for collecting articles.

We initiate the search with the following key terms:

> *threshold multiparty private set intersection*

This yields about 17,800 results on Google Scholar. We screen titles and abstracts to shorten that list. For example, many papers focus on MPSI instead of T-MPSI, so we excluded them. We are left with papers such as [26] by Bay et al. Through reading the Bay et al. paper, we learn that the ”threshold” aspect of T-MPSI can also be denoted ”d-and-over” or ”over-

[1] `https://scholar.google.com/`

threshold". Furthermore, some authors spell "multiparty" as a hyphenated compound word, "multi-party". Our search string is expanded to include those synonyms:

> *(threshold OR d-and-over OR over-threshold) (multiparty OR multi-party) private set intersection*

We then examine the Connected Papers[2] graphs for the found articles. Through this method, we identify another paper on T-MPSI, where the authors refer to T-MPSI as "quorum MPSI": "Efficient Linear Multiparty PSI and Extensions to Circuit/Quorum PSI" by Chandran et al. [29] Adding this synonym to our keyword search results in the string:

> *(threshold OR d-and-over OR over-threshold OR quorum) (multiparty OR multi-party) private set intersection*

Moreover, we notice that certain researchers do not explicitly refer to their protocols as "multiparty" in the title, even though they can be applied to more than two parties, e.g., "Threshold Private Set Intersection with Better Communication Complexity" by Ghosh et al. [55]. This leads to a second search string which does not include "multiparty":

> *(threshold OR d-and-over OR over-threshold OR quorum) private set intersection*

The next step is to execute our search string in a library database. For the selection of papers for our comparison study, we choose the Scopus digital library since it covers a wide range of academic papers, including computer science [56]. On Scopus, the syntax for our searches is equivalent to the following:

> - *(threshold OR d-and-over OR over-threshold OR quorum) (multiparty OR multi-party) private AND set AND intersection)*
> - *(threshold OR d-and-over OR over-threshold OR quorum) private AND set AND intersection*

We apply our queries to search within titles, abstracts and keywords. Note that all our searches are restricted to articles published in English. Additionally, although works indexed on Scopus are generally considered legitimate sources, to further ensure the academic integrity of the chosen texts, we verified the reliability of their sources, e.g., the conference proceedings or journal where the paper was published.

4.2. Selection Criteria for T-MPSI Studies

Following the initial collection of data, we reviewed each paper's abstract, introduction and conclusion of each document to gauge the relevance of the study to our specific research focus on T-MPSI protocols. It allowed us to filter out studies that, despite promising titles or keywords, did not substantively address the nuances and requirements specific to the threshold-based set intersection criteria as defined in our study parameters. The results of which are summarized in table A.1 in Appendix A.

We are left with the following works in T-MPSI after filtering out the ones that do not fit our general criteria:

[2] `https://www.connectedpapers.com/`

- [23] by Miyaji and Nishida
- [28] by Mahdavi et al.
- [29] by Chandran et al.
- [26] by Bay et al.
- [33] by Ma et al.

Our last strategy to find more relevant research it to investigate the articles which cite the papers that we have gathered, as well as the ones which are cited by them. This leads to a text referenced by Mahdavi et al. [27]. Although it is a technical report, we still examine this work due to the credibility of the authors, Kissner and Song as well as the university, Carnegie Mellon in the United States. Through this method, we find another paper written by Ruan et al. [34].

To narrow down the options even further, we formulate these selection criteria for a T-MPSI protocol to use for medical CTI sharing:

- **Honest majority security model**: We require that the protocols be resistant to at least $n/2 - 1$ semi-honest colluding parties.
- **No third parties**: The protocol should not call for an external party to aid in the calculation of the intersection. All participating parties should be healthcare institutions.
- **No server-aided model**: Requiring that the server cannot collude is too constricting; therefore, we exclude server-aided models in our comparison.
- **At most one server**: We limit the number of parties with the role of leader/server to at most one.
- **Large balanced sets**: Since we are dealing with CTI data, we expect all parties to possess large datasets, so the protocol should be able to handle sets with many elements.

These additional requirements led to the exclusion of three more articles, summarized in table 4.1.

Table 4.1: Overview of works in T-MPSI which do not fit with our specified requirements

Work	First author	Reason for excluding
[23]	Miyaji	Requires that the server does not collude. The work by Bay et al. [26] is an improved version of Miyaji and Nishida.
[34]	Ruan	More suitable for unbalanced scenarios, i.e., when the server set is much larger than the clients' and when the number of parties is large.
[33]	Ma	Requires two trusted cloud servers. More suitable for small datasets.

Finally, we are left with four protocols, which we will be comparing in the remainder of the thesis.

- [27] by Kissner and Song
- [28] by Mahdavi et al.
- [29] by Chandran et al.
- [26] by Bay et al.

5

Qualitative Analysis

We now analyze the protocols selected through the methodology described in Chapter 4. The goal of this qualitative comparison is to assess the viability and effectiveness of each protocol in handling sensitive data exchanges within a hospital setting. Our high-level comparison of the four T-MPSI protocols is in terms of:

- Protocol description
- Security against collusion
- Theoretical complexities
- Network topology.

A summary of the comparison is presented at the end of the chapter in Table 5.2.

Table 5.1: Symbols used throughout this chapter.

Symbol	Definition
m	Number of parties
n	Number of elements per set
T	Intersection threshold
λ	Statistical security parameter
κ	Computational security parameter

5.1. T-MPSI Protocol by Kissner and Song

Kissner and Song [27] were the first to propose a T-MPSI protocol in 2004. It is based on polynomials and additively homomorphic encryption and is secure against semi-honest adversaries.

They provide two variations of T-MPSI: perfect threshold and threshold contribution. Perfect T-MPSI conforms to our T-MPSI$_{\text{collective}}$ definition, and threshold contribution to our T-MPSI$_{\text{all}}$. In the former, the elements which appear in at least a threshold T number of the private sets are part of the threshold intersection and revealed to everyone. In the latter, parties only learn the elements of the threshold intersection if they also possess it in their private sets. In this thesis, we analyse their T-MPSI$_{\text{all}}$ protocol. Note that Kissner and Song also present a variant called *over-threshold set-intersection*, but it is not equivalent to the OT-MP-PSI protocol presented by Mahdavi et al. [28]. In Kissner and Song's version, in addition to

the parties learning which elements are part of the threshold intersection, they also learn the frequency of the intersection elements in their private inputs.

Protocol description: Each party first constructs a polynomial with roots corresponding to their set elements. Starting from the first party, each encrypts their polynomial and sends it to the next party. Each subsequent party multiplies the received polynomial by their own and forwards the result, ensuring that the final product is a polynomial that includes contributions from all parties. The last party sends the fully aggregated polynomial back to the first party, who then distributes it among a subset of parties. These selected parties then compute a derivative of the resulting polynomial and mask it to prevent direct decryption. Subsequently, every party evaluates this polynomial at points corresponding to their set elements. The evaluation is encrypted, so that the results remain confidential. The parties cooperatively decrypt these evaluations. If the decrypted value at a point is zero, it indicates that the corresponding set element meets the threshold condition of appearing in multiple sets. This method preserves the privacy of individual sets while allowing collective determination of the intersection. No additional information about non-intersecting elements is disclosed.

Security: Kissner and Song's threshold-contribution protocol is designed to be secure in the semi-honest model against collusion among up to $m-1$ dishonest parties in a network of m parties. This security level is achieved through the use of zero-knowledge proofs and secure multi-party computation techniques, which ensure that no single party or group of colluding parties can derive more information than what is revealed by the protocol output.

Communication and computational complexities: The communication complexity for the protocol by [27] is detailed in terms of the number of bits exchanged. For a threshold set-intersection protocol involving two or more honest-but-curious parties, the total communication complexity is shown as $O(m^3n)$ where m is the number of parties in the protocol and n is the size of their input sets.

The computational complexity is not given in their report.

Network Topology: The protocol assumes a fully connected network topology where each party can directly communicate with every other party. This is necessary for the distribution and collection of cryptographic proofs and for the secure aggregation of results.

5.2. T-MPSI Protocol by Mahdavi et al.

Mahdavi et al. [28] introduce two T-MPSI$_{\text{all}}$ constructions, denoted t-PSI$_0$ and t-PSI, that use techniques such as OPRF, hashing, additively homomorphic encryption and secret sharing. Since the t-PSI scheme is an improvement over the t-PSI$_0$ scheme, it will be the focus of our analysis. In their work, they denote T-MPSI as over-threshold multi-party PSI.

Protocol description: The T-MPSI protocol by [28] has two main phases: share generation and reconstruction. Additionally, parties can either have the role of keyholder, reconstructor, both or neither. The number of keyholders, k, corresponds to the collusion threshold, i.e., the maximum number of dishonest parties against which the protocol can be secure. Initially, each party engages with the keyholder(s) using an oblivious pseudorandom secret sharing protocol to generate secret shares for each of their elements, which are then stored in bins of a hash table padded to a pre-determined size and sent to the reconstructor. The reconstructor(s) form T-sized subsets from these shares across different parties' bins to recover a fixed secret, $S = 0$, confirming the elements are held by at least T parties. This result is then communicated back to each party, who can identify which of their elements are part of the threshold intersection. The keyholder is crucial in generating shares without knowing the elements, ensuring security and privacy, while the reconstructor validates the shared elements. However, the protocol still leaks the owners of the elements in the threshold.

Security: Assuming semi-honest behaviour among parties, the protocol can handle collu-

sion of up to k corrupt parties as long as k parties play the role of the keyholder. When there is a single keyholder, the T-MPSI protocol prohibits them from also having the role of reconstructor or from colluding with the reconstructor. To maintain the integrity of the protocol in setups with multiple keyholders and reconstructors, at least one keyholder should not collude with any reconstruction, and no group of $T-1$ dishonest parties should include a reconstructor.

Communication and computational complexities: Mahdavi et al. [28] describe their T-MPSI construction as running in three communication rounds, two rounds for the share generation algorithm and one more for the reconstruction of shares. The dominating step is the generation of shares, s_i^ℓ, which takes every party $O(nTk)$, where n is the maximum number of elements per set, T is the intersection threshold and k is the collusion threshold, i.e., the number of keyholders. Therefore, for m parties, the total worst case communication complexity is $O(nmTk)$.

The worst case computational complexity of the protocol is $O\big(n(m\log n/T)^{2T}\big)$ per reconstructor, where n is the maximum number of elements per set, m is the number of parties and T is the intersection threshold. Since the schemes were presented with a single reconstructor, we take that number into account for the complexity. Therefore, the computational complexity is $O\big(n(m\log n/T)^{2T}\big)$ for Mahdavi et al. Although it is exponential in the threshold T, the authors argue that their scheme still scales in practice since they have kept the constant values to a minimum.

Network Topology: The network topology exhibits characteristics of both star and potentially hybrid topologies, depending on the number of keyholders and reconstructors involved. In the share generation phase, parties send their encrypted data to the keyholder, who processes this information and returns the necessary computed results to the parties. if only one party plays the role of keyholder, then all communication is mediated through that single party, the keyholder, which acts as a central node. This scenario with one keyholder aligns with a star configuration. When multiple keyholders are involved, the communication topology tends toward a hybrid structure—between star and full mesh—because the addition of keyholders increases the need for new communication channels. Similarly, in the reconstruction phase, if there is only one reconstructor, the communication retains the star topology, with the reconstructor at the centre. However, if multiple reconstructors are used, the protocol begins to resemble a hybrid topology. While each party still communicates their data to one reconstructor, the reconstructors themselves need to communicate among each other.

5.3. T-MPSI Protocol by Chandran et al.

Chandran et al. [29] provide a cuckoo hashing and secret sharing based T-MPSI$_{\text{individual}}$ protocol, where the party leader may only learn which elements in their set are common to at least T other sets. They use the term quorum PSI instead of T-MPSI. Their paper describes two types of T-MPSI, quorum I and quorum II, which differ slightly based on the instantiated cryptographic primitive. Since the former performs better than the latter, we focus on the quorum I scheme in our analysis.

Protocol description: Each party hashes their set elements into tables using a combination of stash-less cuckoo hashing for the leader party and simpler hashing for others, managing collisions with dummy values. The parties then engage in a weak private set membership functionality, which checks for the presence of the leader's elements in other parties' tables without revealing the elements themselves. Following this, the parties engage in an equality check for each element to confirm that the outputs from the weak private set membership functionality correspond correctly to the membership status, ensuring that only the common elements across the sets are marked as such. The boolean results from the equality checks are converted into additive shares, preparing them for final aggregation. Each element's presence

across sets is then aggregated. A weak comparison protocol—a cryptographic functionality designed to perform a comparison operation among a group of parties while maintaining privacy constraints—assesses whether the number of parties holding that element meets a predefined intersection threshold, T. This comparison results in a boolean outcome for each item, indicating whether it appears frequently enough to be considered part of the quorum intersection. The results of these comparisons are then compiled by the first party, who constructs the final intersection set by identifying which elements meet the quorum requirement. This process effectively enables the parties to collectively determine the set intersection without revealing their individual inputs, ensuring both the confidentiality of private data and the accuracy of the computed intersection.

Security: The protocol is designed to be secure under the semi-honest model with an honest majority, i.e., it can withstand collusion attacks, even with the server, as long as less than half of the parties have been corrupted.

Communication and computational complexities: Chandran et al. [29] claim the communication complexity for their T-MPSI construction is $O\big(nm\lambda(\kappa + \lambda \log m)\big)$, where m is the number of parties, n is the maximum size of any set, λ is the statistical security parameter and κ is the computational security parameter. This complexity considers both the data transferred in hashing the elements and the cryptographic operations involved. They derived this worst-case complexity from the total costs of the steps of their quorum I algorithm:
$n(m-1)(\kappa\sigma + 5.8\kappa + 14\sigma + 18T'(\lceil \log m \rceil + 1) + 22\tau + 10\lceil \log m \rceil)$, where
$\sigma = \lambda + \lceil \log n \rceil + \lceil \log m \rceil + 2$
$\tau = \lambda + \lceil \log n \rceil + 3$
$T' = min(T-1, m-T)$
$1 \leq T \leq m-1$ is the intersection threshold.
However, when we compute the worst-case communication complexity from the above equation, we obtain a different result.

$$
\begin{aligned}
&n(m-1)(\kappa\sigma + 5.8\kappa + 14\sigma + 18T'(\lceil \log m \rceil + 1) + 22\tau + 10\lceil \log m \rceil) \\
&= O\big(nm(\kappa\sigma + \kappa + \sigma + T'\lceil \log m \rceil + \tau + \lceil \log m \rceil)\big) \\
&= O\big(nm(\kappa(\lambda + \lceil \log n \rceil + \lceil \log m \rceil) + \kappa + (\lambda + \lceil \log n \rceil + \lceil \log m \rceil) + T'\lceil \log m \rceil + (\lambda + \lceil \log n \rceil) + \lceil \log m \rceil)\big) \\
&= O\big(nm(\kappa(\lambda + \log n + \log m) + \kappa + \lambda + \log n + \log m + T \log m + \lambda + \log n + \log m)\big) \\
&= O\big(nm(\kappa(\lambda + \log n + \log m) + \kappa + \lambda + T \log m + \log n + \log m)\big) \\
&= O\big(nm(\kappa(\lambda + \log n + \log m) + T \log m)\big) \\
&= O\big(nm\kappa(\lambda + \log n + \log m)\big) \text{ because } \kappa >> T, \text{ so } T \log m \text{ does not affect asymptotic complexity.}
\end{aligned}
$$

For our study, we will still consider the communication cost reported by the authors in [29]. However, to make comparisons simpler, we isolate the security parameters, λ and κ, from the communication complexity resulting in a total complexity of $O(nm \log m)$.

Their computational round complexity is $10 + \lceil log\sigma \rceil + 2k'$ which is equivalent to $O\big(log(\lambda + \log n + \log m) + T\big)$. We simplify this to a total cost of $O\big(log(\log n + \log m) + T\big)$.

Network Topology: The T-MPSI protocol assumes a star network topology where the server bears the brunt of the computational workload and the rest of the parties have minimal computational demands. This design is well-suited for client-server environments, ensuring that the system is both efficient and adaptable, with the majority of clients experiencing light processing requirements.

5.4. T-MPSI Protocol by Bay et al.

Bay et al. [26] present a T-MPSI$_{\text{individual}}$ protocol which utilizes Bloom filters and threshold homomorphic public key encryption. Similarly to [29], the goal of their scheme is to return

to the server the elements that appear in their private set as well as at least a threshold T amount of other sets. Their solution is better suited for dealing with a large number of parties and fewer elements per set.

Protocol description: The T-MPSI Protocol involves multiple clients (P_1 to P_{t1}) and a server (P_t), where each client possesses a private set S_i. The server initiates the process by distributing a set of k hash functions to the clients. Each client applies these hash functions to generate BFs from their private sets, then encrypts their BFs using a shared public key pk and sends these EBFs to the server. Upon receiving all EBFs, the server's task is determining which elements from its own set S_t are common across the client sets. It does this by calculating encrypted values for each item in its set, utilizing the hash functions to locate corresponding indices in each client's EBF. The server aggregates these indices using homomorphic encryption to maintain privacy, re-randomizing and summing the encrypted outputs to yield a single encrypted count per item per client.

Next, the server employs the *secure comparison protocol* (SCP) to compare each aggregated count to the number of hash functions k to evaluate whether each item's presence in client sets reaches this threshold. The results of these comparisons are then further aggregated across all clients and compared against the intersection threshold T, to determine if the elements are common to at least T client sets. Elements meeting this threshold are decrypted collectively by the clients, and those confirmed are compiled into the final result set S_T, which represents the intersection of elements appearing in at least T client sets. This entire process ensures that any single party cannot independently deduce the presence or absence of specific elements in other parties' sets, preserving the privacy of each client's data while enabling the collective determination of set intersection.

Security: In the semi-honest model, the protocol resists to collusion by a subset of up to dishonest parties $\ell - 1$ where ℓ is the threshold of a homomorphic PKE scheme. The larger the threshold for the homomorphic PKE, the more secure the T-MPSI protocol is, but also the more complex its execution becomes. It can also handle cases where the server is corrupted.

Communication and computational complexities: The protocol's communication complexity is mainly influenced by the number of rounds, $O(m)$, and the operations within the Π_{SCP} protocol. The server's communication load is heaviest during the initial execution of Π_{SCP}, where it transmits $O(nm\ell)$ ciphertexts where n is the size of the sets, m is the number of parties and ℓ is the threshold for a homomorphic threshold PKE. Each client initially handles $O(\lambda n)$ ciphertexts. In subsequent rounds, active clients manage $O(nm)$ ciphertexts owing to the execution of the Π_{SCP} protocols. The worst case communication complexity per client per round is, therefore, $O\big(\max(\lambda, m)n\big)$ ciphertexts. To standardize the comparison of complexities, we extract the statistical security parameter, λ, from the communication complexity. Additionally, we account for the complexity across all clients, resulting in a total communication complexity of $O(m^2 n)$.

The computational demands of the protocol are mainly composed of the construction of encrypted Bloom filters by the clients, $O(\lambda m)$ operations, and the execution of the Π_{SCP} protocols nt times. Thus, both the server and each active client undergo $O(n\ell)$ homomorphic encryptions. The primary computational burden for the clients is $O\big(max(\lambda, m)\big)$ and $O(n\ell)$ for the server. Overall, the total complexity for all parties is $O(nm)$.

Network Topology: The protocol by [26] is structured in a star topology where every party communicates with a central server but not necessarily directly with every other party. This star topology is beneficial for reducing the complexity of direct peer-to-peer communication, which can be unmanageable in large networks.

5.5. Summary of Qualitative Comparison

We provide an overview of the qualitative comparison in Table 5.2. In terms of security, the protocols by Kissner and Song [27], Mahdavi et al. [28] and Bay et al. [26] perform similarly well, with a collusion threshold of up to $m-1$, where m is the number of parties. The collusion threshold for the protocol by Chandran et al. [29] is half as much. Still, all T-MPSI schemes are secure in the honest majority setting.

[27]'s protocol suffers from high communication costs, which make it impractical for real-world scenarios. [28]'s protocol also appears to scale poorly due to its computational complexity exponentially increasing with the intersection threshold T. However, Mahdavi et al. maintain that their scheme remains scalable in practical applications by minimizing constant values. The runtime complexities for [29] and [26] show better scalability, especially with multithreading.

As for network topologies, the protocols by [29] and [26] operate under a star topology, enabling practical deployment in server-client models. [28]'s protocol uses a star or hybrid topology, depending on the number of parties playing the role of keyholders and reconstructors. [27] have the least efficient topology, requiring a fully connected mesh network,

Table 5.2: Qualitative comparison of four T-MPSI protocols

Work	Collusion threshold	Total Complexities		Network topology
		Communication	Computational	
[27]	$m-1$	$O(m^3 n)$	Not mentioned	Mesh
[28]	$m-1$	$O(mnTk)$	$O\big(n(m \log n/T)^{2T}\big)$	Hybrid
[29]	$\lfloor \frac{m-1}{2} \rfloor$	$O(mn \log m)$	$O\big(log(\log n + \log m) + T\big)$	Star
[26]	$m-1$	$O(m^2 n)$	$O(m^2 n)$	Star

6

Setup for Quantitative Comparison

In this chapter, we provide the methodology for setting up the experiments to conduct a quantitative comparison of the practical performances of two T-MPSI constructions. Of the four protocols from Chapter 5, we select the two most promising based on the qualitative analysis and implement them on our computer.

6.1. Selecting Protocols for Quantitative Comparison

We must first narrow down the four protocols from chapter 5 to two that we examine in greater detail by implementing the code on our machine. To make the selection, the protocol's communication and computational complexities must be optimal. For efficient use in the medical domain, the protocol should be able to run decently fast. We also require the existence of an open-source implementation.

Kissner and Song's algorithm scales poorly, with a communication complexity that is cubic in the number of parties [27]. Furthermore, they did not provide an implementation for their T-MPSI schemes. Consequently, we do not consider them for the quantitative analysis.

Although the asymptotic complexities of Chandran et al. [29] are promising, their current implementation[1] is broken. There are open issues regarding adding subdirectories and linking libraries. Therefore, we also exclude them from further analysis.

To sum up, we decide to perform the quantitative analysis on the T-MPSI protocols by Mahdavi et al. [28] and Bay et al. [26].

6.2. Initial Configuration and Implementation

The authors of the papers conduct their practical experiments on varying hardware setups, complicating direct comparisons between studies. To address this, we aim to standardize testing by running both protocols on the same machine. We obtain the necessary code by cloning the repositories from GitHub. We conduct two rounds of runtime tests, one on the original code and another with the parallelization integrated. This section covers the general hardware setup and the arguments used in our experiments.

The implementation by Mahdavi et al. [28] is available on GitHub[2]. For share generation between participants and keyholders, the experiment utilizes virtual machine instances from Amazon's Elastic Compute Cloud, which have a single virtual CPU and are connected by a network with a maximum bandwidth of 25 Gbps. For the reconstruction phase, the setup includes a server running Ubuntu 18.04 on a 64-bit system, equipped with up to 128 cores from

[1] `https://github.com/shahakash28/PQC-mPSI`

[2] `https://github.com/cryspuwaterloo/OT-MP-PSI`

an IBM POWER8 CPU and up to 1TB of RAM. The cryptographic operations are performed with a computational security parameter of $\kappa = 2048$, adhering to NIST recommendations for security.

The implementation by Bay et al. [26] is also available on GitHub[3]. They conduct the experiments for their T-MPSI construction on a single-thread setup. The system used is a 64-bit Unix-like machine powered by an Intel Core i7-1065G7 processor, which has a clock speed of 1.30 GHz across eight cores, and is equipped with 16GB of memory. In terms of security, they opt for a key of $\kappa = 1024$ bits, a standard choice for PKE, ensuring robust data protection during their tests.

We execute our experiments of the two cryptographic protocols on a laptop configured with a 64-bit Ubuntu 22.04.4 LTS operating system. The hardware includes an AMD Ryzen 9 6900HS processor with 16 cores operating at a maximum frequency of 4.935 GHz as well as 24GB of RAM.

We define the following parameters for both sets of experiments, to make for the most consistent comparison of the protocols.

- Number of parties: $m = 5, 6, 7, 8, 9$
- Maximum set sizes: $n = 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096$
- Intersection threshold: $T = \left\lfloor \frac{m+1}{2} \right\rfloor$
- Key length: $\kappa = 1024$
- HE threshold (for the Bay et al. construction): $\ell = \left\lfloor \frac{m}{2} \right\rfloor$

To run the Mahdavi et al. implementation [28], we compile a shell file which sequentially executes the commands from Appendix B. The first line would run the t-PSI scheme (`-s 1`) on five parties (`-m 5`) who each have at most 32 elements (`-n 32`) in their set. The output would be all elements which appear in a subgroup of at least three sets (`-t 3`). The length of the key prime is 1024 bits (`-b 1024`). For every execution, the program verifies if the input datasets already exist. If not, random sets of integer elements are created to simulate the parties' data. These elements, as well as the generated shares and logs (`-l`) are stored in respective directories with the format based on the parameters: benchmark_[m][n][t]. The logs detail the runtime in milliseconds for both the share generation phase and the reconstruction phase.

We modify the logging functionality of the code in [28], so that the total execution time is measured, instead of printing separate times for the share generation per party and reconstruction. This makes for a more accurate comparison with the implementation by [26].

To ensure an equitable evaluation, we modify the original code by [26] to read the element from the elements files outputted from the code by [28]. This way, both protocols are carried out on the same input sets. The arguments passed are also the same as above. In addition to those arguments, the implementation by Bay et al. requires that we specify the threshold for the homomorphic PKE, denoted by ℓ. We set it to $\left\lfloor \frac{m}{2} \right\rfloor$, which complies with the requirements of the honest majority security model while providing a better runtime efficiency than a higher value would. Then, for every set of parameters, we execute [26] on a single thread and write the resulting intersection results as well as the total execution time in milliseconds to a file.

We repeat these experiments 10 times for every combination of parameters for both protocols. The means and standard deviations are presented in a Table E.4 to E.7 in Appendix E.

[3] `https://github.com/jellevos/threshold-multiparty-psi`

6.3. Implementation with Parallelization

Although the reconstruction step of Mahdavi et al.'s T-MPSI protocol is parallelized, the share generation step is not. The implementation of Bay et al.'s T-MPSI protocol also does not include multithreading. To simulate more realistic client-server interactions and to optimize the efficiency of the program, we integrate parallel processing of certain tasks into the original codes.

The reconstruction phase of [28]'s implementation is parallelized with OpenMP[4], an API that supports multi-platform shared-memory parallel programming in C, C++, and Fortran. Its main advantage is its simplicity and ease of use through compiler directives, library routines, and environment variables, enabling developers to parallelize code with minor modifications. For consistency, we decide to parallelize the share generation phase using OpenMP as well. The `#pragma omp for` directive from OpenMP distributes the iterations of a for loop across different threads, while the `#pragma omp critical` directive designates a block within the parallelized loop which must be executed on one thread, maintaining data integrity.

In the modified parallelized share generation process, we initiate a new thread for each client that interacts with the keyholder, enabling concurrent communication between the keyholder and all participants. This can be achieved by specifying the `#pragma omp for` directive before the for loop which manages the client interactions with the server. Therefore, the creation of client instances, their connections to the server, the loading of input elements and the share generation for the elements can be done concurrently for every party. We are also interested in enhancing the efficiency of the generating shares phase by grouping several elements into batches for each round and distributing the computations across parallel threads. To do so, the `#pragma omp for` directive is inserted before the for loop that iterates over the elements of a client, allowing multiple elements to be processed simultaneously.

To be consistent, we use OpenMP to add multithreading to Bay et al.'s T-MPSI implementation as well. Each client's data set is processed in parallel using the `#pragma omp for` directive before a for loop to create encrypted inverted Bloom filters. Then, arrays for storing client ciphertexts are initialized for each client in parallel to prepare for further cryptographic operations. This directive is applied during the two occasions where we run SCP protocols with any ℓ clients. The first time for comparing intermediate ciphertexts to the encryption of the number of hash functions k, and the second time for comparing the encryption of the number of participant sets which contain a certain element against the encryption of the intersection threshold T. Each comparison operation is independent and can be processed in parallel, significantly speeding up the SCP phase. Lastly, `#pragma omp for` is used for the collaborative decryption of ciphertexts, where each encrypted value is decrypted in parts by multiple parties. Since each decryption operation is independent, parallelizing them reduces the overall computation time.

[4] `https://www.openmp.org/`

7

Quantitative Analysis

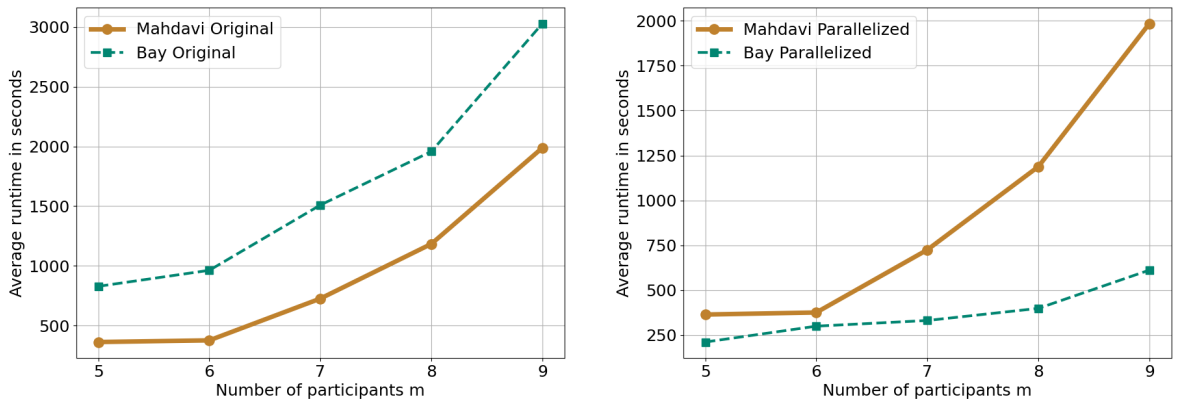
We continue our quantitative comparison and provide an in-depth analysis of the performance results for the implementations of T-MPSI by [28] and [26]. Both implementations are run on our machine with identical parameters to provide a fair comparison in terms of runtime. The goal is to determine the most apt protocol for CTI sharing between healthcare institutions.

In our experimental setup, we executed each configuration of the program parameters 10 times to obtain a diverse set of runtime data. This repetition allowed us to assess the stability of the performance across runs under identical conditions.

For each T-MPSI protocol and implementation—with or without additional parallelization—we present the mean execution times and their standard deviations in Tables E.1, E.2, E.3, E.4, E.5, E.6 and E.7 in Appendix E. While plots visually illustrate trends, tables offer more precise numerical insights on the specific execution times, which could help determine how practical running a T-MPSI solution is. Furthermore, the low standard deviations observed in the runtime experiments indicate that our results are repeatable and consistent across multiple trials.

7.1. How the Number of Parties m Affects Runtime

We now vary the number of participant sets m.



(a) Original Mahdavi et al. [28] and Bay et al. [26] implementations

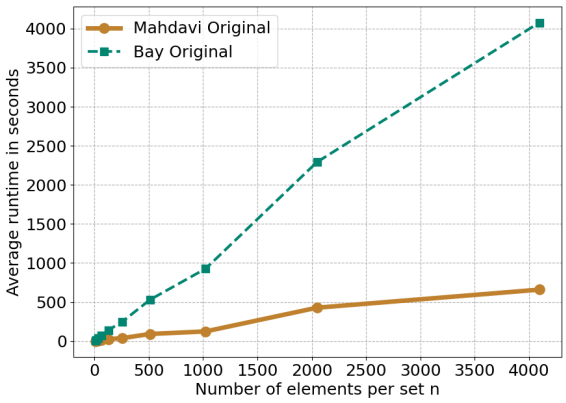
(b) Parallelized Mahdavi et al. [28] and Bay et al. [26] implementations

Figure 7.1: Average runtimes for varying number of parties m with $n = 1024$ and $T = 4$

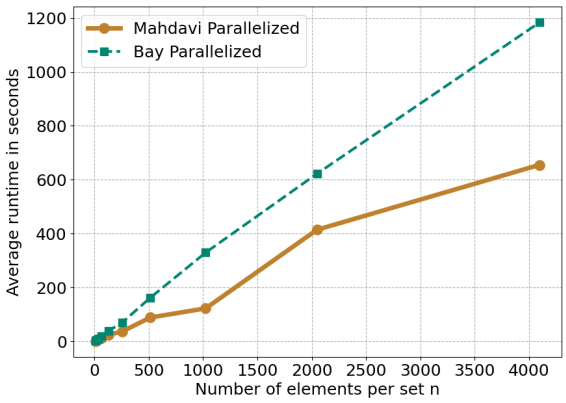
According to the total asymptotic complexities presented by both authors, $O\big(n(m \log n/T)^{2T}\big)$ for Mahdavi et al. [28] and $O(m^2 n)$ for Bay et al. [26], the original version of their T-MPSI schemes should have quadratic runtimes in m, the number of parties. This is evidenced by plot 7.1a. We observe that the original implementation of Mahdavi et al. [28] performs better than that of Bay et al. [26]. However, once parallelized, the protocol by [26] achieves quicker runtimes that appear linear in m. [28]'s runtimes remain in the thousands of seconds, while [26]'s decrease to the hundreds, even for the largest m, making it much more practical.

7.2. How the Maximum Number of Elements Per Set n Affects Runtime

In this section, we vary one of the three main parameters of a T-MPSI construction—the number of items per set n—to see how runtime is affected.

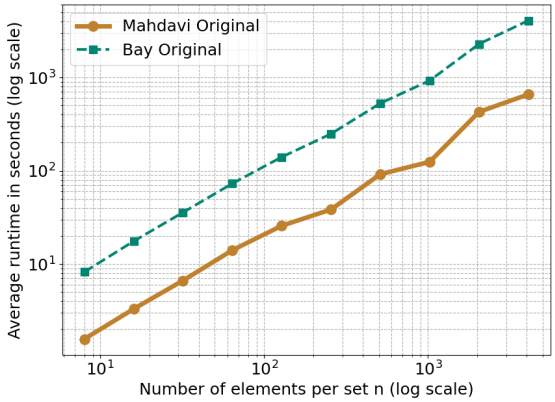


(a) Original Mahdavi et al. [28] and Bay et al. [26] implementations

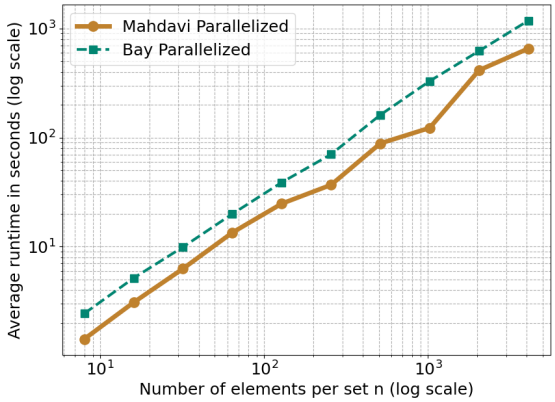


(b) Parallelized Mahdavi et al. [28] and Bay et al. [26] implementations

Figure 7.2: Average runtimes for varying number of elements n with $m = 6$ and $T = 3$



(a) Original Mahdavi et al. [28] and Bay et al. [26] implementations



(b) Parallelized Mahdavi et al. [28] and Bay et al. [26] implementations

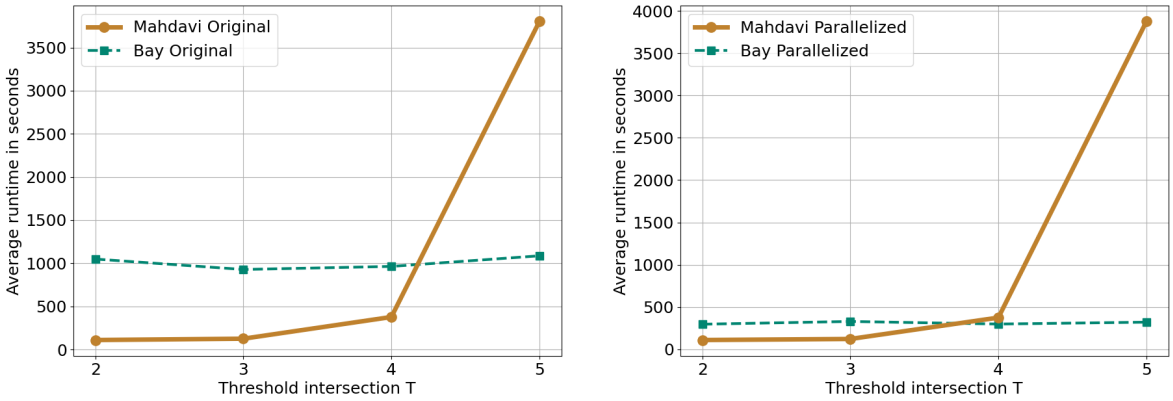
Figure 7.3: Average runtimes for varying number of elements n with $m = 6$ and $T = 3$ in logarithmic scale

Figures 7.2 and 7.3 illustrate an increase in runtime as the set size grows. For a fixed

number of parties m and a set intersection threshold T, both the runtimes for Mahdavi et al.'s [28] and Bay et al.'s [26] implementation increase linearly, albeit, [28] faster than [26]. Figure 7.3 specifically shows how [28] outperforms [26] for all values of n, given $m = 6$ and $T = 1024$.

7.3. How the Intersection Threshold T Affects Runtime

Here, we vary the third the intersection threshold T to see how runtime is affected.



(a) Original Mahdavi et al. [28] and Bay et al. [26] implementations

(b) Parallelized Mahdavi et al. [28] and Bay et al. [26] implementations

Figure 7.4: Average runtimes for varying thresholds T with $m = 6$ and $n = 1024$

For smaller threshold values, $T = 2, 3, 4$, the original implementation by Mahdavi et al. [28] outputs the intersection in less time than that by Bay et al. [26]. However, beginning from $T = 5$, [28]'s execution time increases at a faster rate, surpassing [26] in time. A similar relationship is observed between the parallelized versions of both constructions, with [28]'s performance being better up to $T = 3$, then deteriorating after $T = 4$, while [26]'s time remains consistent.

According to their paper, the protocol by [28] exhibits an asymptotic worst-case communication complexity of $O(nmTk)$ and computational complexity of $O\big(n(m \log n/T)^{2T}\big)$. Therefore, the main factor affecting the runtime is the fact that it is exponential in the intersection threshold T, while n and m are less important. This aligns with our practical experiments, as figure 7.4 shows that the execution time scales fastest with increasing T values.

In contrast, Bay et al.'s runtime complexity is not dependent on the intersection threshold, as evidenced by figure 7.4. It reveals no strong relation between the threshold value and execution times, as illustrated by the relatively constant runtimes despite variations in the threshold. This aligns with its claimed communication and computational complexities from the paper [26], which is $O(mn)$ per client.

7.4. Effects of Parallelization

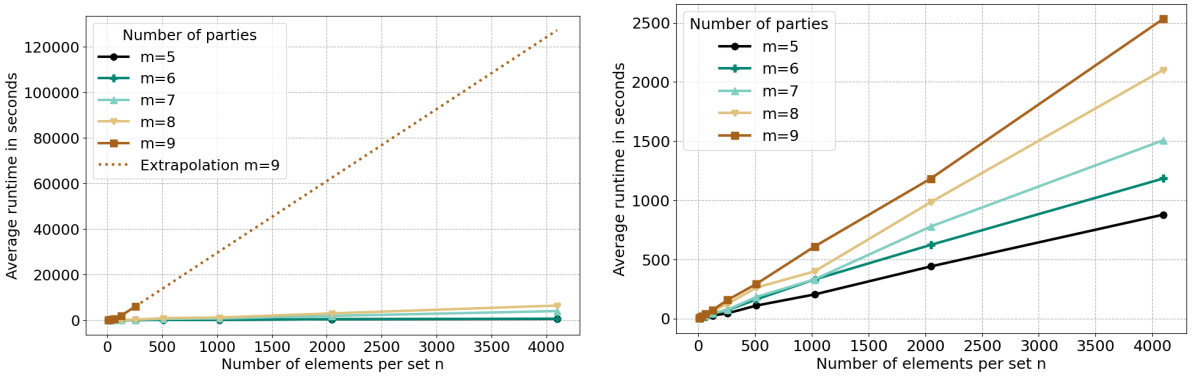
The authors note in their paper [26] that the reported runtimes reflect a worst-case scenario due to the absence of multithreading. With the introduction of parallelization, there is a significant improvement in execution times.

As shown in Table 5.2, the claimed worst case communication and computational complexities of Bay et al. are $O(mn)$ per client, therefore, $O(m^2n)$ in total if we perform the calculations sequentially instead of concurrently. This aligns with the plot of our experiments. Comparing

plot a and b of Figures 7.2, 7.1 and 7.4, we observe that the non-parallelized implementation of the scheme by Bay et al. [26] has much slower runtimes compared to the parallelized version. The reason for this is that parallelization effectively distributes the computational load across multiple processors, thus reducing overall runtime, particularly as the number of participants, m, increases. This scaling advantage becomes more pronounced with higher m values. Although we see a drastic improvement in execution time from the original to the parallelized implementation for [26], the runtime reduction for [28]'s parallelized version is minimal. This could be due to the fact that Mahdavi et al.'s had already incorporated parallelization into the reconstruction phase of their original code. Thus, the gains from adding multithreading of the share generation phase in the parallelized code are not as substantial.

7.5. Additional Practical Performance Comparison

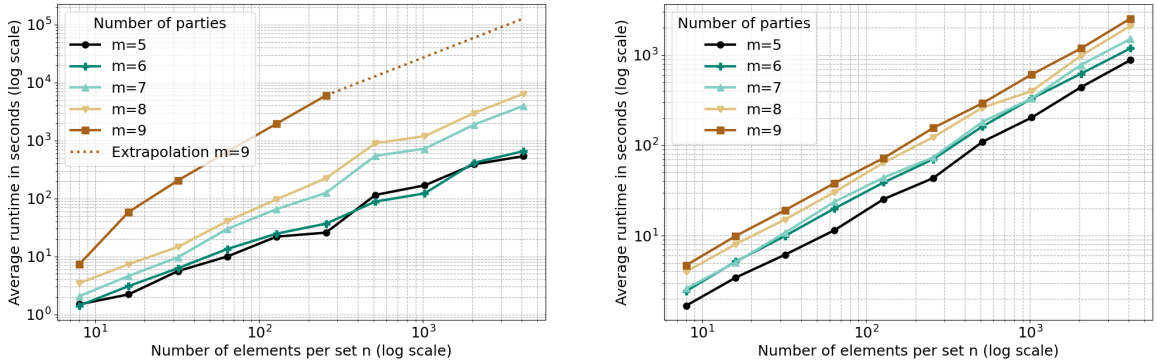
Given our goal of finding the most effective T-MPSI protocol for CTI sharing among healthcare institutions, we also evaluate the performances of [28] and [26] with intersection thresholds dependent on the number of sets m. In this case, the elements of the sets are records in CTI datasets, e.g., IP addresses, domain names, TTPs. The parties are hospital CTI staff trying to identify a common threat. We assume that a user attempting to hack into at least half of the hospitals' systems is malicious, thus we set the threshold T to $\lfloor \frac{m+1}{2} \rfloor$. Additionally, we only showcase the results for the parallelized implementation, as it represents the most efficient approach for applying the protocol in practice.



(a) Mahdavi et al. [28] parallelized implementation

(b) Bay et al. [26] parallelized implementation

Figure 7.5: Average runtimes for different number of parties m varying the number of elements per set n with threshold $T = \lfloor \frac{m+1}{2} \rfloor$



(a) Mahdavi et al. [28] parallelized implementation (log scale)

(b) Bay et al. [26] parallelized implementation (log scale)

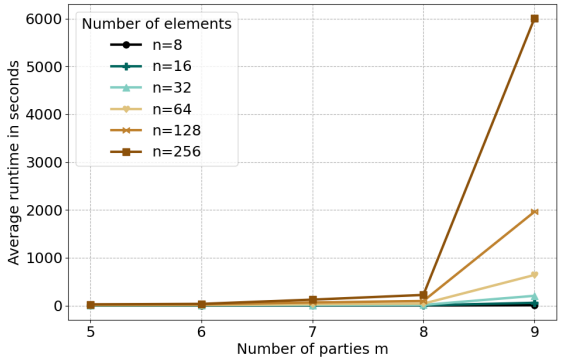
Figure 7.6: Average runtimes for different number of parties m varying the number of elements per set n with threshold $T = \lfloor \frac{m+1}{2} \rfloor$ in logarithmic scale

Figures 7.5 and 7.6 present the plots of how runtime increases as the maximum set size n increases for various m and T values. Since with nine parties, [28]'s protocol could not finish execution within a practical timeframe for larger values of n, we graphed the extrapolation of $m = 9$ for $n = 256$ to $n = 4096$ in 7.5a and 7.6a. The extrapolation should help in estimating how the protocol might behave under larger-scale conditions, so that we may still compare [28] to [26] when n is large.

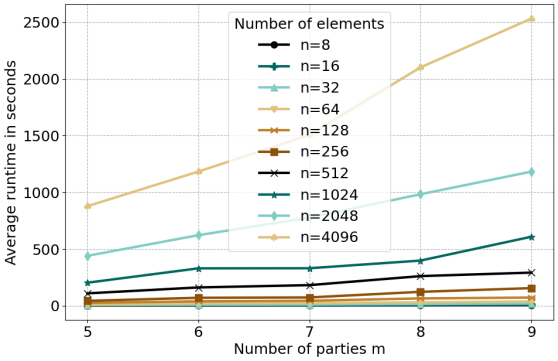
In the implementations by Mahdavi et al., the runtimes remain relatively low for fewer number of parties, $m = 5, 6, 7, 8$. Indeed, their implementation outperforms [26]'s for $m = 5, 6$. However, a notable increase in runtime is observed for $m = 9$ as the number of elements per set n increases, reflecting a significant escalation in computational load with an increase in parties. Moreover, recall that we set the intersection threshold T to be dependent on the number of parties m, and as demonstrated in Figure 7.4, [28]'s runtime scales exponentially with T. Figure 7.6 illustrate these trends better by using a logarithmic scale. We observe not only that the runtimes increase more drastically when more parties are involved, but also when the threshold increases. The runtime plots for $m = 5$ and $m = 6$, both having a threshold of $T = 3$ are relatively closer together, as are the plots for $m = 7$ and $m = 8$ which share a threshold of $T = 4$. Finally, the plot for $m = 9$ is the steepest, with a threshold of $T = 5$. Thus, Mahdavi et al.'s T-MPSI scheme is only suitable for scenarios where the number of parties and the threshold is limited to lower values.

Plots 7.5b and 7.6b indicate that Bay et al.'s protocol has better scalability as both the number of parties and the threshold increase. Although there is still an increase in runtime as the number of parties increases, the plots remain linear and predictable across different values of m.

To sum up, [28] performs better for smaller values of m and T, whereas [26] scales better with higher values of m and T.

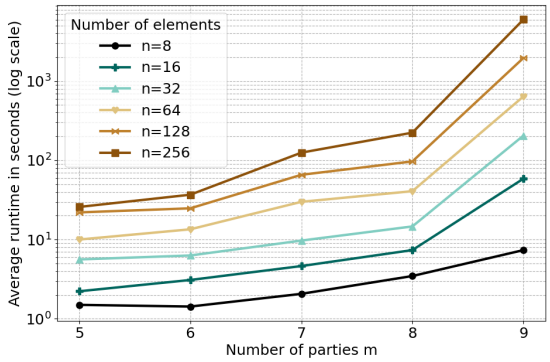


(a) Mahdavi et al. [28] parallelized implementation

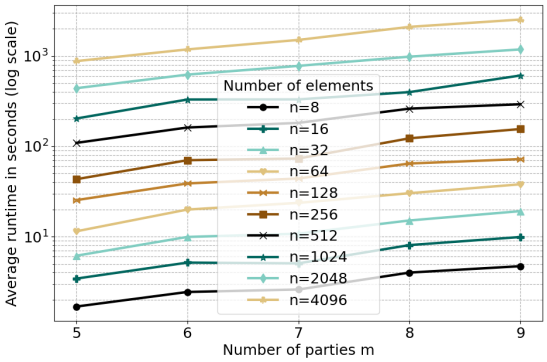


(b) Bay et al. [26] parallelized implementation

Figure 7.7: Average runtimes for different number of elements per set n varying the number of parties m with threshold $T = \lfloor \frac{m+1}{2} \rfloor$



(a) Mahdavi et al. [28] parallelized implementation (log scale)



(b) Bay et al. [26] parallelized implementation (log scale)

Figure 7.8: Average runtimes for different number of elements per set n varying the number of parties m with threshold $T = \lfloor \frac{m+1}{2} \rfloor$ in logarithmic scale

Figures 7.7 and 7.8 display the runtime performance of Mahdavi et al.'s implementation and Bay's implementation, both plotted against the number of parties m and the number of elements per set n, with threshold T being a function of m.

Similarly to plots 7.5a and 7.6a, a dramatic increase in runtime for all values of n as the number of parties m increases to 9. This is particularly pronounced for larger values of n. For $n = 256$, the runtime at $m = 9$ is around 6000 seconds, highlighting a significant performance drop as complexity increases.

Even at $n = 4096$, the highest runtime recorded at $m = 9$ is just above 2000 seconds, indicating a more robust performance under high complexity.

8 Extension

We now present an extension to transform any T-MPSI$_{\text{individual}}$ scheme [26, 29] into a T-MPSI$_{\text{collective}}$ scheme [27].

8.1. Protocol Description

To recap, in a T-MPSI$_{\text{individual}}$ type construction, one party receives the intersection of its elements and a threshold number $T-1$ of the other $m-1$ sets. In MPSI$_{\text{collective}}$, every participant receives the threshold intersection of everyone's sets, i.e., all elements which appear in at least T of the m sets are outputted to all parties even if they do not possess the element.

We see that to go from T-MPSI$_{\text{individual}}$ to T-MPSI$_{\text{collective}}$, we must first execute the T-MPSI$_{\text{individual}}$ protocol m times so that every participant has a turn to be the server to obtain their MPSI$_{\text{individual}}$ result. Then, we combine every party's T-MPSI$_{\text{individual}}$ output in a privacy preserving manner. For that, we can use *multiparty private set union* (MPSU). In MPSU, each participant holds a private set, and the objective is to find the union of all these sets without revealing any additional information about the individual sets beyond what can be inferred from the final result, i.e., the union [57]. The final result is equivalent to the T-MPSI$_{\text{collective}}$ outcome.

Note that the intermediate results of T-MPSI$_{\text{individual}}$ protocols, R_i, possibly reveal the output of the T-MPSI$_{\text{individual i}}$ in plaintext to each participant i. However, this leakage does not compromise the privacy of the overall protocol since given the end result R and the party's own private set S_i, R_i can be derived anyway.

We provide the proof in Appendix C.

8.2. Implementation

To the best of our knowledge, no prior implementation exists to convert between types of T-MPSI protocols. We present a generalized extension which can be applied to any T-MPSI$_{\text{individual}}$ scheme to transform it into a T-MPSI$_{\text{collective}}$ scheme. In this thesis, we provide an example of such a pipeline using [26] as the initial T-MPSI$_{\text{individual}}$ protocol. We begin by modifying the protocol code to concurrently find the T-MPSI$_{\text{individual}}$ for every party. If we are working with a T-MPSI$_{\text{all}}$ protocol instead of T-MPSI$_{\text{individual}}$, we do not need to add this multithreading step since the T-MPSI$_{\text{all}}$ scheme already guarantees that every participant obtains their T-MPSI$_{\text{individual}}$ set. We store the intermediate intersections in JSON files. Then, we apply the MPSU protocol by [58] on these intermediate results. The resulting union represents the T-MPSI$_{\text{collective}}$ set, See appendix D for the bash script to execute the pipeline.

9

Discussion and Future Work

In this chapter, we discuss the results and limitations of our comparative study, suggest some areas for future work, and conclude this thesis with final remarks.

9.1. Discussion

The objective of this thesis is to answer the question, *How can we efficiently compute the threshold intersection of hospitals' CTI datasets in a privacy preserving manner to identify recurring cyber threats?*

In other words, we want to determine which state-of-the-art T-MPSI protocol hospitals should employ to share their CTI data with each other. In our attempt to do so, we conduct an SLR, a qualitative analysis and a quantitative analysis. The selected protocols were evaluated based on their ability to securely and efficiently facilitate CTI sharing among hospitals. The protocols analyzed include those by Kissner and Song [27], Mahdavi et al. [28], Chandran et al. [29], and Bay et al. [26].

Our study assesses the suitability for deployment in hospital CTI sharing of four T-MPSI protocols mainly in terms of its security, as well as communication and computational complexities. All four protocols provided security guarantees against semi-honest adversaries. Although this model provides a weaker security guarantee than the malicious model, it still ensures no leakage of sensitive information, which aligns with our requirements. Malicious adversaries may be considered in the case of man-in-the-middle attacks. However, preventing these attacks is not within the scope of our research. Our focus is on developing and analyzing secure protocols rather than broader network security strategies like vulnerability management. The schemes by Kissner and Song [27], Mahdavi et al. [28] and Bay et al. [26] offer the highest level of security as they are secure against collusion among up to $m - 1$ dishonest parties out of m parties in total, whereas the scheme by Chandran et al. [29] is secure in the honest majority setting with at most $\frac{m-1}{2}$ dishonest parties. This collusion threshold is acceptable because the parties participating in the intersection are trusted entities, meaning the highest level of security is not required. The drawbacks of the Kissner and Song protocol are that it exhibits a higher asymptotic communication complexity and lacks a concrete implementation. Furthermore, [29]'s protocol yields promising communication and computational complexities. However, their implementation on GitHub is non-functional. Therefore, we proceed with the two other protocols for further comparison, [28] and [26].

We implement those two T-MPSI schemes on the same machine in order to have comparable results. To simulate real-life scenarios and improve performance, we incorporate multithreading into both schemes. We conduct our runtime tests on both the original and

parallelized implementations of the protocols, [28] and [28], to determine how their practical runtimes are impacted by the number of parties m, the maximum size of the sets n and the intersection threshold T.

We find that Mahdavi et al.'s protocol is faster than Bay et al.'s for a smaller number of parties and smaller thresholds. Therefore, we recommend employing [28] if six or fewer hospitals are interested in computing the threshold intersection of three subsets or fewer. Otherwise, we recommend the protocol proposed by Bay et al. [26], as it scales more efficiently for larger values of m and T. However, considering the scalability and efficiency requirements of the healthcare sector, neither the Bay et al. nor the Mahdavi et al. protocol is efficient in practice. CTI datasets often contain millions of entries, which translates to millions of elements per set, yet we observe runtime bottlenecks for both protocols starting at thousands of elements when there are slightly more parties. Unless all participating hospitals are equipped with powerful computers, it is difficult to employ these T-MPSI protocols in a large-scale hospital setting.

9.2. Limitations

A limitation of our comparison study is that not all T-MPSI protocols have a concrete implementation available for us to compare. Consequently, our quantitative analysis is incomplete, with only execution times for the protocols by Mahdavi et al. [28] and Bay et al. [26]. We do not have practical performance results for the protocols proposed by Kissner and Song [27] and Chandran et al. [29]. Ideally, we would have performed runtime experiments on all four schemes to provide a fair assessment of their practicability in CTI sharing.

Moreover, the actual methodology for performing the quantitative comparison also has limitations. We only use lists of integers as input sets for testing the implementations of the T-MPSI protocols. A wider range of datasets which resemble actual CTI would provide a better simulation of real-world scenarios. By benchmarking the protocols under a broader range of inputs, we can more accurately assess their efficiency and suitability for specific applications.

Lastly, we run our experiments for both protocols on a single machine to keep our testing environment as consistent as possible, but there are always slight variations in hardware, software, or other environmental factors that can affect runtime measurements. If conditions are not strictly controlled or documented, comparisons across different experiments may be unreliable.

9.3. Future Work

While this thesis has provided a comprehensive comparative analysis, several areas warrant further investigation.

Validating asymptotical complexities claimed in papers: We can conduct a more thorough investigation and actually validate the runtime complexities to avoid solely relying on the theoretical claims made by authors. This involves conducting systematic experiments to test the scalability and performance under varied conditions and configurations that extend beyond the initial setups described in the literature.

Improvements on extension for T-MPSI$_{\text{individual}}$ to T-MPSI$_{\text{collective}}$: In this thesis, we introduce a general extension applicable to any T-MPSI$_{\text{individual}}$ scheme [26, 29] to convert it into a T-MPSI$_{\text{collective}}$ scheme [27]. This extension has significant potential for further enhancement. Specifically, future work could provide an analysis of both communication and computational complexities. Additionally, evaluating the security measures against collusion among participants would be essential to ensure robustness. Conducting empirical runtime experiments would also provide insights into the practicality and efficiency of the extended protocol.

Extension for MPSI to T-MPSI: A similar direction for future work would be to develop an efficient extension for transforming an MPSI protocol into a T-MPSI protocol, as only the naive

approach exists for now. Currently, MPSI schemes can be adapted into T-MPSI schemes by computing the intersection for all subgroups of size T of the m total sets and aggregating the results. This method proves to be highly costly since the number of subgroups of size T is $\binom{m}{t}$, which indicates a significant increase in computational demand as m and t grow. Creating a more effective conversion process would significantly enhance the research utility of MPSI by facilitating its adaptation to T-MPSI applications.

Considering available network bandwidth and evaluating throughput: Finally, there is a need to expand the metrics used in the protocol's performance evaluation. Beyond average runtime, other metrics, such as network throughput, should be examined to better assess the applicability of the T-MPSI protocols to real-world scenarios. When exchanging CTI data in a privacy-preserving manner, the volume of data transferred between parties can be substantial, especially when the sets are large or the protocol requires multiple rounds of communication. For many cryptographic protocols, including T-MPSI, the network can often be a bottleneck. The available bandwidth in a region could be a critical factor in determining the feasibility of deploying a T-MPSI protocol. Even if a protocol is computationally optimized, its overall effectiveness could be compromised by insufficient bandwidth, leading to inefficient execution times.

9.4. Conclusion

A prevalent issue observed throughout the literature is the conflation of various T-MPSI protocols. Many studies do not clearly differentiate between threshold MPSI, which focuses on the presence of an element in a minimum number of sets, and cardinality MPSI, which centres on the size of the intersection set. This ambiguity complicates the selection of appropriate protocols for specific applications, such as CTI sharing in medical settings, where the distinction can significantly impact both the privacy and utility of shared data. To address this confusion and provide a clearer understanding of protocol performance, we conducted a detailed comparison study.

We conclude that [28]'s protocol performs better than [26]'s for small values of $m = 5, 6$ and $T = 2, 3$, but [26]'s protocol scales better as $m7, 8, 9$ and $T = 4, 5$ increase. Still, no T-MPSI protocol could realistically be used for threat intelligence sharing between hospitals. Both the Mahdavi et al. and Bay et al. implementations struggle with the large data volumes typical in CTI, rendering them inefficient for real-world applications.

Bibliography

[1] Claudio Ardagna et al. *ENISA Threat Landscape 2023*. European Network and Information Security Agency, 2023. DOI: 10.2824/782573.

[2] European Union Agency for Cybersecurity. *Smart hospitals – Security and resilience for smart health service and infrastructures*. European Network and Information Security Agency, 2016. DOI: 10.2824/28801.

[3] Lynne Coventry and Dawn Branley. "Cybersecurity in healthcare: A narrative review of trends, threats and ways forward". In: *Maturitas* 113 (2018), pp. 48–52. ISSN: 0378-5122. DOI: https://doi.org/10.1016/j.maturitas.2018.04.008. URL: https://www.sciencedirect.com/science/article/pii/S0378512218301658.

[4] Albert Haro Abad and Stephen Corbiaux. *ENISA Threat Landscape: Health Sector*. European Network and Information Security Agency, 2023. DOI: 10.2824/163953.

[5] John Soldatos, James Philpot, and Gabriele Giunta, eds. *Cyber-Physical Threat Intelligence for Critical Infrastructures Security: A Guide to Integrated Cyber-Physical Protection of Modern Critical Infrastructures*. Now Publishers, 2020. ISBN: 978-1-68083-686-8 978-1-68083-687-5. DOI: 10.1561/9781680836875.

[6] IBM. *What is ransomware-as-a-service (RaaS)?* URL: https://www.ibm.com/topics/ransomware-as-a-service (visited on 02/12/2024).

[7] Saman Taghavi Zargar, James Joshi, and David Tipper. "A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks". In: *IEEE Commun. Surv. Tutorials* 15.4 (2013), pp. 2046–2069. DOI: 10.1109/SURV.2013.031413.00127. URL: https://doi.org/10.1109/SURV.2013.031413.00127.

[8] Salem T. Argaw et al. "Cybersecurity of Hospitals: discussing the challenges and working towards mitigating the risks". In: *BMC Medical Informatics Decis. Mak.* 20.1 (2020), p. 146. DOI: 10.1186/S12911-020-01161-7. URL: https://doi.org/10.1186/s12911-020-01161-7.

[9] NICCS. *Cyber Threat Intelligence*. Aug. 16, 2022. URL: https://niccs.cisa.gov/education-training/catalog/sans-institute/cyber-threat-intelligence (visited on 02/13/2024).

[10] Sophos. *Cyber Threat Intelligence (CTI)*. URL: https://www.sophos.com/en-us/cybersecurity-explained/cti-cyber-threat-intelligence#:~:text=Why%20is%20CTI%20Important%3F,visibility%2C%20detection%20and%20response%20actions. (visited on 02/13/2024).

[11] IBM. *What Is Threat Intelligence?* URL: https://www.ibm.com/topics/threat-intelligence (visited on 02/12/2024).

[12] Thomas D. Wagner et al. "Cyber threat intelligence sharing: Survey and research directions". In: *Comput. Secur.* 87 (2019). DOI: 10.1016/J.COSE.2019.101589. URL: https://doi.org/10.1016/j.cose.2019.101589.

[13] European Comission. *What Is Personal Data?* URL: https://commission.europa.eu/law/law-topic/data-protection/reform/what-personal-data_en (visited on 02/15/2024).

[14] Eranga Bandara et al. “LUUNU - Blockchain, MISP, Model Cards and Federated Learning Enabled Cyber Threat Intelligence Sharing Platform”. In: *Annual Modeling and Simulation Conference, ANNSIM 2022, San Diego, CA, USA, July 18-20, 2022*. Ed. by Cristina Ruiz Martin et al. IEEE, 2022, pp. 235–245. DOI: 10.23919/ANNSIM55834.2022.9859355. URL: https://doi.org/10.23919/ANNSIM55834.2022.9859355.

[15] Maurizio Atzori. “Weak *k*-Anonymity: A Low-Distortion Model for Protecting Privacy”. In: *Information Security, 9th International Conference, ISC 2006, Samos Island, Greece, August 30 - September 2, 2006, Proceedings*. Ed. by Sokratis K. Katsikas et al. Vol. 4176. Lecture Notes in Computer Science. Springer, 2006, pp. 60–71. DOI: 10.1007/11836810_5. URL: https://doi.org/10.1007/11836810%5C_5.

[16] Muneeb Ul Hassan, Mubashir Husain Rehmani, and Jinjun Chen. “Differential Privacy Techniques for Cyber Physical Systems: A Survey”. In: *IEEE Commun. Surv. Tutorials* 22.1 (2020), pp. 746–789. DOI: 10.1109/COMST.2019.2944748. URL: https://doi.org/10.1109/COMST.2019.2944748.

[17] Silvio Micali and Phillip Rogaway. “Secure Computation (Abstract)”. In: *Advances in Cryptology - CRYPTO ’91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*. Ed. by Joan Feigenbaum. Vol. 576. Lecture Notes in Computer Science. Springer, 1991, pp. 392–404. DOI: 10.1007/3-540-46766-1_32. URL: https://doi.org/10.1007/3-540-46766-1%5C_32.

[18] Donald Beaver. “Foundations of Secure Interactive Computing”. In: *Advances in Cryptology - CRYPTO ’91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*. Ed. by Joan Feigenbaum. Vol. 576. Lecture Notes in Computer Science. Springer, 1991, pp. 377–391. DOI: 10.1007/3-540-46766-1_31. URL: https://doi.org/10.1007/3-540-46766-1%5C_31.

[19] Oded Goldreich. “Secure multi-party computation”. In: *Manuscript. Preliminary version* 78.110 (1998).

[20] Vladimir Kolesnikov et al. “Practical Multi-party Private Set Intersection from Symmetric-Key Techniques”. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*. Ed. by Bhavani Thuraisingham et al. ACM, 2017, pp. 1257–1272. DOI: 10.1145/3133956.3134065. URL: https://doi.org/10.1145/3133956.3134065.

[21] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. “Efficient Private Matching and Set Intersection”. In: *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*. Ed. by Christian Cachin and Jan Camenisch. Vol. 3027. Lecture Notes in Computer Science. Springer, 2004, pp. 1–19. DOI: 10.1007/978-3-540-24676-3_1. URL: https://doi.org/10.1007/978-3-540-24676-3%5C_1.

[22] Aner Ben-Efraim et al. “PSImple: Practical Multiparty Maliciously-Secure Private Set Intersection”. In: *ASIA CCS ’22: ACM Asia Conference on Computer and Communications Security, Nagasaki, Japan, 30 May 2022 - 3 June 2022*. Ed. by Yuji Suga et al. ACM, 2022, pp. 1098–1112. DOI: 10.1145/3488932.3523254. URL: https://doi.org/10.1145/3488932.3523254.

[23] Atsuko Miyaji and Shohei Nishida. “A Scalable Multiparty Private Set Intersection”. In: *Network and System Security - 9th International Conference, NSS 2015, New York, NY, USA, November 3-5, 2015, Proceedings*. Ed. by Meikang Qiu et al. Vol. 9408. Lecture Notes in Computer Science. Springer, 2015, pp. 376–385. DOI: 10.1007/978-3-319-25645-0_26. URL: https://doi.org/10.1007/978-3-319-25645-0%5C_26.

[24] Jelle Vos, Mauro Conti, and Zekeriya Erkin. “SoK: Collusion-resistant Multi-party Private Set Intersections in the Semi-honest Model”. In: *IACR Cryptol. ePrint Arch.* (2023). URL: https://eprint.iacr.org/2023/1777.pdf.

[25] Benny Pinkas, Thomas Schneider, and Michael Zohner. “Scalable Private Set Intersection Based on OT Extension”. In: *ACM Trans. Priv. Secur.* 21.2 (2018), 7:1–7:35. DOI: 10.1145/3154794. URL: https://doi.org/10.1145/3154794.

[26] Aslí Bay et al. “Practical Multi-Party Private Set Intersection Protocols”. In: *IEEE Trans. Inf. Forensics Secur.* 17 (2022), pp. 1–15. DOI: 10.1109/TIFS.2021.3118879. URL: https://doi.org/10.1109/TIFS.2021.3118879.

[27] Lea Kissner and Dawn Song. *Private and Threshold Set-Intersection*. 2004. URL: http://www.dtic.mil/docs/citations/ADA461119.

[28] Rasoul Akhavan Mahdavi et al. “Practical Over-Threshold Multi-Party Private Set Intersection”. In: *ACSAC ’20: Annual Computer Security Applications Conference, Virtual Event / Austin, TX, USA, 7-11 December, 2020*. ACM, 2020, pp. 772–783. DOI: 10.1145/3427228.3427267. URL: https://doi.org/10.1145/3427228.3427267.

[29] Nishanth Chandran et al. “Efficient Linear Multiparty PSI and Extensions to Circuit/Quorum PSI”. In: *CCS ’21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*. Ed. by Yongdae Kim et al. ACM, 2021, pp. 1182–1204. DOI: 10.1145/3460120.3484591. URL: https://doi.org/10.1145/3460120.3484591.

[30] Xiaopeng Yu et al. “Multiparty Threshold Private Set Intersection Protocol with Low Communication Complexity”. In: *Security and Communication Networks* 2022 (2022). Ed. by Arijit Karati, pp. 1–12. ISSN: 1939-0122, 1939-0114. DOI: 10.1155/2022/9245516. URL: https://www.hindawi.com/journals/scn/2022/9245516/ (visited on 02/14/2024).

[31] Feng-Hao Liu, En Zhang, and Leiyong Qin. “Efficient Multiparty Probabilistic Threshold Private Set Intersection”. In: *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023*. Ed. by Weizhi Meng et al. ACM, 2023, pp. 2188–2201. DOI: 10.1145/3576915.3623158. URL: https://doi.org/10.1145/3576915.3623158.

[32] Satrajit Ghosh and Mark Simkin. “The Communication Complexity of Threshold Private Set Intersection”. In: *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Vol. 11693. Lecture Notes in Computer Science. Springer, 2019, pp. 3–29. DOI: 10.1007/978-3-030-26951-7_1. URL: https://doi.org/10.1007/978-3-030-26951-7%5C_1.

[33] Liju Ma et al. “Over-threshold multi-party private set operation protocols for lightweight clients”. In: *Comput. Stand. Interfaces* 88 (2024), p. 103781. DOI: 10.1016/J.CSI.2023.103781. URL: https://doi.org/10.1016/j.csi.2023.103781.

[34] Ou Ruan et al. “A Practical Multiparty Private Set Intersection Protocol Based on Bloom Filters for Unbalanced Scenarios”. In: *Applied Sciences* 13.24 (2023), p. 13215. URL: https://doi.org/10.3390/app132413215.

[35] Yehuda Lindell and Benny Pinkas. “A Proof of Security of Yao’s Protocol for Two-Party Computation”. In: *J. Cryptol.* 22.2 (2009), pp. 161–188. DOI: 10.1007/S00145-008-9036-8. URL: https://doi.org/10.1007/s00145-008-9036-8.

[36] Burton H. Bloom. “Space/Time Trade-offs in Hash Coding with Allowable Errors”. In: *Commun. ACM* 13.7 (1970), pp. 422–426. DOI: 10.1145/362686.362692. URL: https://doi.org/10.1145/362686.362692.

[37] Prosenjit Bose et al. “On the false-positive rate of Bloom filters”. In: *Inf. Process. Lett.* 108.4 (2008), pp. 210–213. DOI: 10.1016/J.IPL.2008.05.018. URL: https://doi.org/10.1016/j.ipl.2008.05.018.

[38] Alex Davidson and Carlos Cid. “An Efficient Toolkit for Computing Private Set Operations”. In: *Information Security and Privacy - 22nd Australasian Conference, ACISP 2017, Auckland, New Zealand, July 3-5, 2017, Proceedings, Part II*. Ed. by Josef Pieprzyk and Suriadi Suriadi. Vol. 10343. Lecture Notes in Computer Science. Springer, 2017, pp. 261–278. DOI: 10.1007/978-3-319-59870-3_15. URL: https://doi.org/10.1007/978-3-319-59870-3%5C_15.

[39] Robert W. Shirey. “Internet Security Glossary, Version 2”. In: *RFC* 4949 (2007), pp. 1–365. DOI: 10.17487/RFC4949. URL: https://doi.org/10.17487/RFC4949.

[40] Rasmus Pagh and Flemming Friche Rodler. “Cuckoo Hashing”. In: *Algorithms - ESA 2001, 9th Annual European Symposium, Aarhus, Denmark, August 28-31, 2001, Proceedings*. Ed. by Friedhelm Meyer auf der Heide. Vol. 2161. Lecture Notes in Computer Science. Springer, 2001, pp. 121–133. DOI: 10.1007/3-540-44676-1_10. URL: https://doi.org/10.1007/3-540-44676-1%5C_10.

[41] Adi Shamir. “How to Share a Secret”. In: *Commun. ACM* 22.11 (1979), pp. 612–613. DOI: 10.1145/359168.359176. URL: https://doi.org/10.1145/359168.359176.

[42] Pascal Paillier. “Public-Key Cryptosystems Based on Composite Degree Residuosity Classes”. In: *Advances in Cryptology - EUROCRYPT ’99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*. Ed. by Jacques Stern. Vol. 1592. Lecture Notes in Computer Science. Springer, 1999, pp. 223–238. DOI: 10.1007/3-540-48910-X_16. URL: https://doi.org/10.1007/3-540-48910-X%5C_16.

[43] Pierre-Alain Fouque, Guillaume Poupard, and Jacques Stern. “Sharing Decryption in the Context of Voting or Lotteries”. In: *Financial Cryptography, 4th International Conference, FC 2000 Anguilla, British West Indies, February 20-24, 2000, Proceedings*. Ed. by Yair Frankel. Vol. 1962. Lecture Notes in Computer Science. Springer, 2000, pp. 90–104. DOI: 10.1007/3-540-45472-1_7. URL: https://doi.org/10.1007/3-540-45472-1%5C_7.

[44] Shimon Even, Oded Goldreich, and Abraham Lempel. “A Randomized Protocol for Signing Contracts”. In: *Commun. ACM* 28.6 (1985), pp. 637–647. DOI: 10.1145/3812.3818. URL: https://doi.org/10.1145/3812.3818.

[45] Moni Naor and Omer Reingold. “Number-theoretic Constructions of Efficient Pseudorandom Functions”. In: *38th Annual Symposium on Foundations of Computer Science, FOCS ’97, Miami Beach, Florida, USA, October 19-22, 1997*. IEEE Computer Society, 1997, pp. 458–467. DOI: 10.1109/SFCS.1997.646134. URL: https://doi.org/10.1109/SFCS.1997.646134.

[46] Michael J. Freedman et al. “Keyword Search and Oblivious Pseudorandom Functions”. In: *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*. Ed. by Joe Kilian. Vol. 3378. Lecture Notes in Computer Science. Springer, 2005, pp. 303–324. DOI: 10.1007/978-3-540-30576-7_17. URL: https://doi.org/10.1007/978-3-540-30576-7%5C_17.

[47] Sílvia Casacuberta, Julia Hesse, and Anja Lehmann. “SoK: Oblivious Pseudorandom Functions”. In: *7th IEEE European Symposium on Security and Privacy, EuroS&P 2022, Genoa, Italy, June 6-10, 2022*. IEEE, 2022, pp. 625–646. DOI: 10.1109/EUROSP53844.2022.00045. URL: https://doi.org/10.1109/EuroSP53844.2022.00045.

[48] Daniel Morales Escalera, Isaac Agudo, and Javier López. “Private set intersection: A systematic literature review”. In: *Comput. Sci. Rev.* 49 (2023), p. 100567. DOI: 10.1016/J.COSREV.2023.100567. URL: https://doi.org/10.1016/j.cosrev.2023.100567.

[49] Per A. Hallgren, Claudio Orlandi, and Andrei Sabelfeld. “PrivatePool: Privacy-Preserving Ridesharing”. In: *30th IEEE Computer Security Foundations Symposium, CSF 2017, Santa Barbara, CA, USA, August 21-25, 2017*. IEEE Computer Society, 2017, pp. 276–291. DOI: 10.1109/CSF.2017.24. URL: https://doi.org/10.1109/CSF.2017.24.

[50] Satrajit Ghosh and Mark Simkin. “The Communication Complexity of Threshold Private Set Intersection”. In: *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Vol. 11693. Lecture Notes in Computer Science. Springer, 2019, pp. 3–29. DOI: 10.1007/978-3-030-26951-7_1. URL: https://doi.org/10.1007/978-3-030-26951-7%5C_1.

[51] Saikrishna Badrinarayanan et al. “Multi-party Threshold Private Set Intersection with Sublinear Communication”. In: *Public-Key Cryptography - PKC 2021 - 24th IACR International Conference on Practice and Theory of Public Key Cryptography, Virtual Event, May 10-13, 2021, Proceedings, Part II*. Ed. by Juan A. Garay. Vol. 12711. Lecture Notes in Computer Science. Springer, 2021, pp. 349–379. DOI: 10.1007/978-3-030-75248-4_13. URL: https://doi.org/10.1007/978-3-030-75248-4%5C_13.

[52] Pedro Branco, Nico Döttling, and Sihang Pu. “Multiparty Cardinality Testing for Threshold Private Intersection”. In: *Public-Key Cryptography - PKC 2021 - 24th IACR International Conference on Practice and Theory of Public Key Cryptography, Virtual Event, May 10-13, 2021, Proceedings, Part II*. Ed. by Juan A. Garay. Vol. 12711. Lecture Notes in Computer Science. Springer, 2021, pp. 32–60. DOI: 10.1007/978-3-030-75248-4_2. URL: https://doi.org/10.1007/978-3-030-75248-4%5C_2.

[53] Tapaswini Mohanty et al. “Quantum Secure Threshold Private Set Intersection Protocol for IoT-Enabled Privacy-Preserving Ride-Sharing Application”. In: *IEEE Internet Things J.* 11.1 (2024), pp. 1761–1772. DOI: 10.1109/JIOT.2023.3291132. URL: https://doi.org/10.1109/JIOT.2023.3291132.

[54] Florian Kerschbaum, Erik-Oliver Blass, and Rasoul Akhavan Mahdavi. “Faster Secure Comparisons with Offline Phase for Efficient Private Set Intersection”. In: *30th Annual Network and Distributed System Security Symposium, NDSS 2023, San Diego, California, USA, February 27 - March 3, 2023*. The Internet Society, 2023. URL: https://www.ndss-symposium.org/ndss-paper/faster-secure-comparisons-with-offline-phase-for-efficient-private-set-intersection/.

[55] Satrajit Ghosh and Mark Simkin. “Threshold Private Set Intersection with Better Communication Complexity”. In: *Public-Key Cryptography - PKC 2023 - 26th IACR International Conference on Practice and Theory of Public-Key Cryptography, Atlanta, GA, USA, May 7-10, 2023, Proceedings, Part II*. Ed. by Alexandra Boldyreva and Vladimir Kolesnikov. Vol. 13941. Lecture Notes in Computer Science. Springer, 2023, pp. 251–272. DOI: 10.1007/978-3-031-31371-4_9. URL: https://doi.org/10.1007/978-3-031-31371-4%5C_9.

[56] Angela Carrera-Rivera et al. “How-to Conduct a Systematic Literature Review: A Quick Guide for Computer Science Research”. In: *MethodsX* 9 (2022), p. 101895. ISSN: 22150161. DOI: 10.1016/j.mex.2022.101895. URL: https://linkinghub.elsevier.com/retrieve/pii/S2215016122002746 (visited on 05/03/2024).

[57] Jiahui Gao, Son Nguyen, and Ni Trieu. “Toward A Practical Multi-party Private Set Union”. In: *IACR Cryptol. ePrint Arch.* (2023), p. 1930. URL: https://eprint.iacr.org/2023/1930.

[58] Jelle Vos, Mauro Conti, and Zekeriya Erkin. “Fast Multi-party Private Set Operations in the Star Topology from Secure ANDs and ORs”. In: *IACR Cryptol. ePrint Arch.* (2022), p. 721. URL: https://eprint.iacr.org/2022/721.

[59] Dana Dachman-Soled et al. “Secure Efficient Multiparty Computing of Multivariate Polynomials and Applications”. In: *Applied Cryptography and Network Security - 9th International Conference, ACNS 2011, Nerja, Spain, June 7-10, 2011. Proceedings*. Ed. by Javier López and Gene Tsudik. Vol. 6715. Lecture Notes in Computer Science. 2011, pp. 130–146. DOI: 10.1007/978-3-642-21554-4_8. URL: https://doi.org/10.1007/978-3-642-21554-4%5C_8.

[60] Fattaneh Bayatbabolghani and Marina Blanton. “Secure Multi-Party Computation”. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*. Ed. by David Lie et al. ACM, 2018, pp. 2157–2159. DOI: 10.1145/3243734.3264419. URL: https://doi.org/10.1145/3243734.3264419.

[61] Benny Pinkas et al. “Efficient Circuit-Based PSI via Cuckoo Hashing”. In: *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part III*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10822. Lecture Notes in Computer Science. Springer, 2018, pp. 125–157. DOI: 10.1007/978-3-319-78372-7_5. URL: https://doi.org/10.1007/978-3-319-78372-7%5C_5.

[62] En Zhang, Jian Chang, and Yu Li. “Efficient Threshold Private Set Intersection”. In: *IEEE Access* 9 (2021), pp. 6560–6570. DOI: 10.1109/ACCESS.2020.3048743. URL: https://doi.org/10.1109/ACCESS.2020.3048743.

[63] Shengnan Zhao et al. “Lightweight Threshold Private Set Intersection via Oblivious Transfer”. In: *Wireless Algorithms, Systems, and Applications - 16th International Conference, WASA 2021, Nanjing, China, June 25-27, 2021, Proceedings, Part III*. Ed. by Zhe Liu, Fan Wu, and Sajal K. Das. Vol. 12939. Lecture Notes in Computer Science. Springer, 2021, pp. 108–116. DOI: 10.1007/978-3-030-86137-7_12. URL: https://doi.org/10.1007/978-3-030-86137-7%5C_12.

[64] Dan Meng et al. “t-PSI: Efficient Multi-party Private Set Intersection with Threshold”. In: *IEEE Smartworld, Ubiquitous Intelligence & Computing, Scalable Computing & Communications, Digital Twin, Privacy Computing, Metaverse, Autonomous & Trusted Vehicles, SmartWorld/UIC/ScalCom/DigitalTwin/PriComp/Meta 2022, Haikou, China, December 15-18, 2022*. IEEE, 2022, pp. 8–15. DOI: 10.1109/SMARTWORLD-UIC-ATC-SCALCOM-DIGITALTWIN-PRICOMP-METAVERSE56740.2022.00029. URL: https://doi.org/10.1109/SmartWorld-UIC-ATC-ScalCom-DigitalTwin-PriComp-Metaverse56740.2022.00029.

A

Results of our Initial Collection of Threshold PSI Studies.

Table A.1: Overview of papers that we deem irrelevant to our comparisons, as explained in Section 4.2

Work	First author	Reason for excluding
[59]	Dachman-Soled	Not a T-MPSI protocol, just MPSI.
[60]	Bayatbabolghani	Not a T-MPSI protocol, main focus is MPC.
[61]	Pinkas	Different definition of T-MPSI.
[50]	Ghosh	Two-party protocol with a different T-MPSI definition.
[51]	Badrinarayanan	Different T-MPSI definition focusing on set differences.
[52]	Branco	Different T-MPSI definition focusing on set cardinality.
[62]	Zhang	Different T-MPSI definition focusing on intersection cardinality.
[63]	Zhao	Two-party protocol with a cardinality-based T-MPSI definition.
[64]	Meng	Not a T-MPSI protocol, utilizes Shamir's secret sharing.
[30]	Yu	Two-party protocol with a set difference-based T-MPSI definition.
[55]	Ghosh	Focuses on set differences in T-MPSI.
[31]	Liu	Emphasis on probabilistic threshold in intersection size.
[53]	Mohanty	Focuses on quantum security and threshold cardinality.

B

Bash Script for Running the Protocol by Mahdavi et al.

```
#!/bin/bash
./benchmark all -m 5 -n 8 -t 3 -b 1024 -s 1 -l &&
./benchmark all -m 5 -n 16 -t 3 -b 1024 -s 1 -l &&
./benchmark all -m 5 -n 32 -t 3 -b 1024 -s 1 -l &&
./benchmark all -m 5 -n 64 -t 3 -b 1024 -s 1 -l &&
./benchmark all -m 5 -n 128 -t 3 -b 1024 -s 1 -l &&
./benchmark all -m 5 -n 256 -t 3 -b 1024 -s 1 -l &&
./benchmark all -m 5 -n 512 -t 3 -b 1024 -s 1 -l &&
./benchmark all -m 5 -n 1024 -t 3 -b 1024 -s 1 -l &&
./benchmark all -m 5 -n 2048 -t 3 -b 1024 -s 1 -l &&
./benchmark all -m 5 -n 4096 -t 3 -b 1024 -s 1 -l &&
./benchmark all -m 6 -n 8 -t 3 -b 1024 -s 1 -l &&
./benchmark all -m 6 -n 16 -t 3 -b 1024 -s 1 -l &&
./benchmark all -m 6 -n 32 -t 3 -b 1024 -s 1 -l &&
./benchmark all -m 6 -n 64 -t 3 -b 1024 -s 1 -l &&
./benchmark all -m 6 -n 128 -t 3 -b 1024 -s 1 -l &&
./benchmark all -m 6 -n 256 -t 3 -b 1024 -s 1 -l &&
./benchmark all -m 6 -n 512 -t 3 -b 1024 -s 1 -l &&
./benchmark all -m 6 -n 1024 -t 3 -b 1024 -s 1 -l &&
./benchmark all -m 6 -n 2048 -t 3 -b 1024 -s 1 -l &&
./benchmark all -m 6 -n 4096 -t 3 -b 1024 -s 1 -l &&
./benchmark all -m 7 -n 8 -t 4 -b 1024 -s 1 -l &&
./benchmark all -m 7 -n 16 -t 4 -b 1024 -s 1 -l &&
./benchmark all -m 7 -n 32 -t 4 -b 1024 -s 1 -l &&
./benchmark all -m 7 -n 64 -t 4 -b 1024 -s 1 -l &&
./benchmark all -m 7 -n 128 -t 4 -b 1024 -s 1 -l &&
./benchmark all -m 7 -n 256 -t 4 -b 1024 -s 1 -l &&
./benchmark all -m 7 -n 512 -t 4 -b 1024 -s 1 -l &&
./benchmark all -m 7 -n 1024 -t 4 -b 1024 -s 1 -l &&
./benchmark all -m 7 -n 2048 -t 4 -b 1024 -s 1 -l &&
./benchmark all -m 7 -n 4096 -t 4 -b 1024 -s 1 -l &&
./benchmark all -m 8 -n 8 -t 4 -b 1024 -s 1 -l &&
./benchmark all -m 8 -n 16 -t 4 -b 1024 -s 1 -l &&
./benchmark all -m 8 -n 32 -t 4 -b 1024 -s 1 -l &&
./benchmark all -m 8 -n 64 -t 4 -b 1024 -s 1 -l &&
./benchmark all -m 8 -n 128 -t 4 -b 1024 -s 1 -l &&
./benchmark all -m 8 -n 256 -t 4 -b 1024 -s 1 -l &&
```

```
./benchmark all -m 8 -n 512 -t 4 -b 1024 -s 1 -l &&
./benchmark all -m 8 -n 1024 -t 4 -b 1024 -s 1 -l &&
./benchmark all -m 8 -n 2048 -t 4 -b 1024 -s 1 -l &&
./benchmark all -m 8 -n 4096 -t 4 -b 1024 -s 1 -l &&
./benchmark all -m 9 -n 8 -t 5 -b 1024 -s 1 -l &&
./benchmark all -m 9 -n 16 -t 5 -b 1024 -s 1 -l &&
./benchmark all -m 9 -n 32 -t 5 -b 1024 -s 1 -l &&
./benchmark all -m 9 -n 64 -t 5 -b 1024 -s 1 -l &&
./benchmark all -m 9 -n 128 -t 5 -b 1024 -s 1 -l &&
./benchmark all -m 9 -n 256 -t 5 -b 1024 -s 1 -l &&
./benchmark all -m 9 -n 512 -t 5 -b 1024 -s 1 -l &&
./benchmark all -m 9 -n 1024 -t 5 -b 1024 -s 1 -l &&
./benchmark all -m 9 -n 2048 -t 5 -b 1024 -s 1 -l &&
./benchmark all -m 9 -n 4096 -t 5 -b 1024 -s 1 -l &&

./benchmark all -m 6 -n 1024 -t 2 -b 1024 -s 1 -l &&
./benchmark all -m 6 -n 1024 -t 4 -b 1024 -s 1 -l &&
./benchmark all -m 6 -n 1024 -t 5 -b 1024 -s 1 -l &&

./benchmark all -m 5 -n 1024 -t 4 -b 1024 -s 1 -l &&
./benchmark all -m 6 -n 1024 -t 4 -b 1024 -s 1 -l &&
./benchmark all -m 9 -n 1024 -t 4 -b 1024 -s 1 -l
```

C

Proof that Intermediate Results Can Be Leaked

Let us consider an example with the threshold $t = 3$ and the number of parties $m = 4$.

Parties P_1, P_2, P_3, P_4 respectively each have a set S_1, S_2, S_3, S_4.

We define R as the union of intersections of all combinations of $t = 3$ sets out of the $m = 4$ sets.

$$R = (S_1 \cap S_2 \cap S_3) \cup (S_1 \cap S_2 \cap S_4) \cup (S_1 \cap S_3 \cap S_4) \cup (S_2 \cap S_3 \cap S_4)$$

We define R_1 as the intersection of R_1 with the union of intersections of all combinations of $t - 1 = 2$ sets out of the $m - 1 = 3$ other sets (so excluding R_1) as follows:

$$\begin{aligned} R_1 &= S_1 \cap [(S_2 \cap S_3) \cup (S_2 \cap S_4) \cup (S_3 \cap S_4)] \\ &= (S_1 \cap S_2 \cap S_3) \cup (S_1 \cap S_2 \cap S_4) \cup (S_1 \cap S_3 \cap S_4) \end{aligned}$$

Consider $S_1 \cap R$:

$$\begin{aligned} S_1 \cap R &= S_1 \cap (S_1 \cap S_2 \cap S_3) \cup (S_1 \cap S_2 \cap S_4) \cup (S_1 \cap S_3 \cap S_4) \cup (S_2 \cap S_3 \cap S_4) \\ &= (S_1 \cap S_1 \cap S_2 \cap S_3) \cup (S_1 \cap S_1 \cap S_2 \cap S_4) \cup (S_1 \cap S_1 \cap S_3 \cap S_4) \cup (S_1 \cap S_2 \cap S_3 \cap S_4) \\ &= (S_1 \cap S_2 \cap S_3) \cup (S_1 \cap S_2 \cap S_4) \cup (S_1 \cap S_3 \cap S_4) \cup (S_1 \cap S_2 \cap S_3 \cap S_4) \\ &= R_1 \cup (S_1 \cap S_2 \cap S_3 \cap S_4) \end{aligned}$$

Since $S_1 \cap S_2 \cap S_3 \cap S_4 \subseteq S_1 \cap S_2 \cap S_3 \cap \subseteq R_1$,

$$S_1 \cap R = R_1$$

Therefore, a participant i can generate R_i from S_i and R, which are known to them.

We can generalize this to prove for all t and m that $S_i \cap R = R_i$:

Let R represent the union of intersections of all combinations of t participant sets out of the total m sets:

$$R = \bigcup_{\substack{J \subseteq \{1,\ldots,m\} \\ |J| = t}} \bigcap_{j \in J} S_j \tag{C.1}$$

Let R_i, for a specific party i, be defined as:

$$R_i = S_i \cap \bigcup_{\substack{J \subseteq \{1,\ldots,m\} \setminus \{i\} \\ |J| = t-1}} \bigcap_{j \in J} S_j \tag{C.2}$$

We want to prove that we can obtain R_i from the intersection of S_i and R. We start by expanding R:

$$S_i \cap R = S_i \cap \left(\bigcup_{\substack{J \subseteq \{1,\ldots,m\} \\ |J| = t}} \bigcap_{j \in J} S_j \right).$$

Using the distributive property of intersection over union, this becomes

$$S_i \cap R = \bigcup_{\substack{J \subseteq \{1,\ldots,m\} \\ |J| = t}} (S_i \cap \bigcap_{j \in J} S_j).$$

If we focus on subsets J such that $i \in J$, this reduces to considering intersections of $t-1$ other sets with S_i because the presence of S_i in the intersections is guaranteed

$$S_i \cap R = \bigcup_{\substack{J \subseteq \{1,\ldots,m\} \setminus \{i\} \\ |J| = t-1}} (S_i \cap \bigcap_{j \in J \cup \{i\}} S_j).$$

Since i is added back to J to form t elements, and J initially contains $t-1$ elements not including i, the equation reduces to

$$S_i \cap R = \bigcup_{\substack{J \subseteq \{1,\ldots,m\} \setminus \{i\} \\ |J| = t-1}} (S_i \cap \bigcap_{j \in J} S_j).$$

By the distributive property of sets, we can extract the intersection of S_i out of the union

$$S_i \cap R = S_i \cap \bigcup_{\substack{J \subseteq \{1,\ldots,m\} \setminus \{i\} \\ |J| = t-1}} \bigcap_{j \in J} S_j.$$

This is exactly the definition of R_i as given by equation C.2, so we conclude that

$$S_i \cap R = R_i$$

This generalization shows that each participant i can generate R_i from S_i and R, which confirms the privacy-preserving properties of the protocol despite potential exposures of intermediate results R_i's. This ensures the protocol's confidentiality and integrity under the assumption that R and each participant's S_i are secure.

D

Bash Script for Running the Extension

```
#!/bin/bash

cd threshold-multiparty-psi

echo "Running␣T-MPSI␣individual."
mkdir build
cd build
cmake ..
make
./multiparty_psi_fresh

mv tmpsi_ind.json ../../private-logic-and-mpso

echo "Running␣MPSU."

# Compile and run the Rust program
cd ../../private-logic-and-mpso

cargo build --release
cargo run --release exact-set-union 3 10 3706452992 2

echo "Done."
```

E

Execution Times of the Implementations for the [28] and [26] T-MPSI Protocols

Table E.1: Mahdavi et al. and Bay et al. original and parallelized protocols under different numbers of elements per set n with $m = 6$ and $T = 3$ - Runtime in seconds

n	Mahdavi Original	Mahdavi Parallelized	Bay Original	Bay Parallelized
8	1.572 ± 0.239	1.423 ± 0.043	8.224 ± 0.179	2.439 ± 0.063
16	3.304 ± 0.411	3.082 ± 0.047	17.588 ± 0.445	5.157 ± 0.066
32	6.658 ± 0.825	6.284 ± 0.059	35.791 ± 0.627	9.926 ± 0.079
64	14.088 ± 1.835	13.459 ± 0.108	73.049 ± 0.726	19.947 ± 0.091
128	25.722 ± 2.816	24.775 ± 0.448	140.338 ± 1.550	38.809 ± 0.192
256	38.462 ± 4.338	36.826 ± 0.195	248.967 ± 2.922	70.189 ± 0.221
512	91.968 ± 10.202	88.463 ± 0.319	529.322 ± 9.809	161.499 ± 0.832
1024	124.711 ± 4.341	122.663 ± 0.388	927.223 ± 12.746	329.802 ± 2.298
2048	428.019 ± 36.205	414.298 ± 1.037	2288.518 ± 33.390	622.106 ± 3.950
4096	659.274 ± 7.381	655.056 ± 1.380	4080.347 ± 27.907	1183.389 ± 6.127

Table E.2: Mahdavi et al. and Bay et al. original and parallelized protocols under different numbers of parties m with $n = 1024$ and $T = 3$ - Runtime in seconds

m	Mahdavi Original	Mahdavi Parallelized	Bay Original	Bay Parallelized
5	362.3 ± 3.06	363.9 ± 4.47	828.7 ± 9.14	210.7 ± 1.65
6	376.3 ± 2.80	375.2 ± 3.49	962.8 ± 14.43	299.1 ± 2.10
7	726 ± 10.80	724.6 ± 6.17	1509 ± 30.43	330.7 ± 1.65
8	1183 ± 8.51	1188 ± 12.54	1958 ± 16.22	398.2 ± 2.47
9	1988 ± 13.09	1985 ± 6.41	3029 ± 14.49	611.5 ± 31.57

Note that certain parameters values in the T-MPSI protocols cause the protocol to run for longer than what is practical in real-world settings. Consequently, for instances where the protocol did not complete execution within a reasonable timeframe, such as in Tables E.4, E.5 and E.6, we leave the entry blank.

Table E.3: Mahdavi et al. and Bay et al. original and parallelized protocols under different intersection thresholds T with $m = 6$ and $n = 1024$ - Runtime in seconds

T	Mahdavi Original	Mahdavi Parallelized	Bay Original	Bay Parallelized
2	109.4 ± 2.22	110.3 ± 0.52	1048 ± 15.7	297.1 ± 1.42
3	124.7 ± 4.34	122.7 ± 0.39	927.2 ± 12.75	329.8 ± 2.3
4	376.4 ± 2.796	375.2 ± 3.49	963 ± 14.43	299.1 ± 2.1
5	3802 ± 43.38	3874 ± 55.46	1086 ± 23.44	321.4 ± 3.24

Table E.4: Mahdavi et al. original implementation with threshold $T = \lfloor \frac{m+1}{2} \rfloor$ - Runtime in seconds

Maximum number of elements per set n	Number of parties m				
	5	6	7	8	9
8	1.44 ± 0.03	1.57 ± 0.24	2.05 ± 0.09	3.45 ± 0.11	7.26 ± 0.07
16	2.19 ± 0.06	3.3 ± 0.41	4.65 ± 0.13	7.42 ± 0.05	58.51 ± 0.84
32	5.54 ± 0.09	6.66 ± 0.82	9.67 ± 0.18	14.66 ± 0.12	205.7 ± 2.85
64	10.02 ± 0.1	14.09 ± 1.83	29.68 ± 0.36	40.85 ± 0.23	639.1 ± 4.55
128	21.86 ± 0.15	25.72 ± 2.82	65.33 ± 0.83	96.84 ± 0.55	1962 ± 18.35
256	25.91 ± 0.2	38.46 ± 4.34	123.9 ± 1.44	223.9 ± 1.18	5988 ± 91.03
512	115.1 ± 0.4	91.97 ± 10.2	542.8 ± 6.34	903.6 ± 9.86	—
1024	169.6 ± 3.08	124.7 ± 4.34	726 ± 10.8	1184 ± 8.51	—
2048	389.1 ± 3.27	428 ± 36.2	1881 ± 25.05	2974 ± 34.77	—
4096	538.3 ± 3.65	659.3 ± 7.38	3996 ± 223	6385 ± 53.5	—

Table E.5: Mahdavi et al. parallelized implementation with threshold $T = \lfloor \frac{m+1}{2} \rfloor$ - Runtime in seconds

Maximum number of elements per set n	Number of parties m				
	5	6	7	8	9
8	1.5 ± 0.11	1.42 ± 0.04	2.06 ± 0.05	3.46 ± 0.06	7.34 ± 0.22
16	2.22 ± 0.05	3.08 ± 0.05	4.62 ± 0.07	7.35 ± 0.04	58.61 ± 0.94
32	5.6 ± 0.2	6.28 ± 0.06	9.69 ± 0.12	14.63 ± 0.14	205.2 ± 2.48
64	9.99 ± 0.1	13.46 ± 0.11	29.92 ± 0.38	40.79 ± 0.28	640.3 ± 5.83
128	22 ± 0.36	24.77 ± 0.45	65.66 ± 0.89	96.88 ± 0.67	1962 ± 24.23
256	25.79 ± 0.23	36.83 ± 0.2	125.1 ± 1.93	223.6 ± 1.39	6002 ± 35.69
512	115.3 ± 1.16	88.46 ± 0.32	546.2 ± 6.34	899.9 ± 4.26	—
1024	168.5 ± 1.05	122.7 ± 0.39	724.6 ± 6.17	1188 ± 12.54	—
2048	388.7 ± 2.76	414.3 ± 1.04	1880 ± 23.15	2980 ± 29.24	—
4096	537.7 ± 3.26	655.1 ± 1.38	3970 ± 53.67	6415 ± 40.43	—

Table E.6: Bay et al. original implementation with threshold $T = \lfloor \frac{m+1}{2} \rfloor$ - Runtime in seconds

Maximum number of elements per set n	Number of parties m				
	5	6	7	8	9
8	6.26 ± 0.15	8.22 ± 0.18	12.04 ± 0.07	19.74 ± 0.29	26.2 ± 0.24
16	11.14 ± 0.19	17.59 ± 0.45	24.16 ± 0.38	36.42 ± 0.55	60.64 ± 1.48
32	26.87 ± 2.2	35.79 ± 0.63	45.94 ± 0.77	64.31 ± 0.55	114.8 ± 2.01
64	47.78 ± 0.71	73.05 ± 0.73	115 ± 1.72	150.9 ± 2.59	206.1 ± 4.14
128	66.64 ± 15.73	140.3 ± 1.55	222.6 ± 2.6	303.4 ± 4.47	361.5 ± 10.47
256	161.5 ± 3.21	249 ± 2.92	362 ± 3.68	578.3 ± 6.18	786.8 ± 17.69
512	460.5 ± 6.17	529.3 ± 9.81	861.7 ± 50.19	1120 ± 9.44	1412 ± 15.69
1024	782.5 ± 10.16	927.2 ± 12.75	1509 ± 30.43	1958 ± 16.22	3364 ± 49.29
2048	1689 ± 29.72	2289 ± 33.39	3385 ± 18.84	4170 ± 18.02	6726 ± 44.17
4096	2804 ± 29.54	4080 ± 27.91	6856 ± 223.98	8649 ± 39.5	—

Table E.7: Bay et al. parallelized implementation with threshold $T = \lfloor \frac{m+1}{2} \rfloor$ - Runtime in seconds

Maximum number of elements per set n	Number of parties m				
	5	6	7	8	9
8	1.68 ± 0.03	2.44 ± 0.06	2.6 ± 0.08	4 ± 0.08	4.71 ± 0.2
16	3.43 ± 0.03	5.16 ± 0.07	5.06 ± 0.07	8.06 ± 0.07	9.91 ± 0.44
32	6.15 ± 0.05	9.93 ± 0.08	10.77 ± 0.12	15.1 ± 0.11	19.15 ± 1.16
64	11.46 ± 0.13	19.95 ± 0.09	23.65 ± 0.17	30.25 ± 0.21	37.98 ± 2.41
128	25.38 ± 0.25	38.81 ± 0.19	44.02 ± 0.21	64.5 ± 0.33	72.23 ± 3.52
256	43.24 ± 0.28	70.19 ± 0.22	73.27 ± 0.39	122.7 ± 0.52	155.4 ± 7.86
512	109.2 ± 0.45	161.5 ± 0.83	181.7 ± 1.04	261.4 ± 1.24	292.3 ± 12.16
1024	203.2 ± 1.14	329.8 ± 2.3	330.7 ± 1.65	398.2 ± 2.47	608.6 ± 19.32
2048	440.1 ± 1.96	622.1 ± 3.95	777.6 ± 4.5	983.2 ± 2.93	1183 ± 56.68
4096	877.8 ± 7.17	1183 ± 6.13	1507 ± 4.94	2102 ± 10.42	2533 ± 89.67