Private Matching of IoC and Network Data

by

Rutger van der Beek

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday September 2, 2025 at 14:00.

Student number:	6075002
Project duration:	November 18, 2024 – September 2, 2025
Thesis committee:	Dr. Z. Erkin, TU Delft, supervisor
	Dr. J. Urbano, TU Delft

TU Delft

Preface

After working at the SOC of Fox-IT and thoroughly enjoying the Privacy Enhancing Technologies course during the Master's degree, this subject came up as a fine combination of both. While remaining focused on a single subject for 9 months was challenging, I have enjoyed these last months of my time in Delft.

First, I want to thank my thesis supervisor, Dr. Erkin, for his advice throughout the thesis. With his help, I was able to sufficiently confine the goal of this research so that it was possible to actually finish in time. I want to thank the rest of the research group as well for the interesting conversations about many different research topics. Learning about completely different areas of research or about something that only differs marginally from my own research every now and then was a welcome change of pace.

Additionally, I want to thank my friends and family for their support. Ranging from encouraging me to finally stop reading papers and start with the rest of the research, to proofreading this thesis.

Rutger van der Beek
Delft, August 2025

Abstract

Computer networks are an integral part of our society and functioning without them is difficult, as computers rely on this connection for their data or shared computing power. While this connectivity is often beneficial, it has downsides as well. A malicious actor can try to break into a network remotely, which organisations try to prevent by monitoring their networks in order to detect such an attacker. Monitoring can be done, for example, by searching for Indicators of Compromise (IoC) within the network traffic. These IoC can take the form of a single attribute, such as an IP address, or a combination of multiple attributes, such as an IP address, a protocol and a domain name. If during this search IoC match with the network data, a malicious actor might be present in the network. An in-house solution is not feasible for all organisations as this would result in more financial overhead, thus a managed Security Operations Center (mSOC) can be contracted. Often, such an mSOC has access to all network data in order to match their IoC. However, this may be undesirable for organisations that want to keep their sensitive network data as private as possible. Therefore, sharing relevant data only when a match has been found is preferable. Additionally, an mSOC has reasons to want to keep their IoC private as well, as they invest resources into gathering these IoC and sharing them would pose a risk to their business model. In this work we aim to match IoC, consisting of a variable number of attributes, with network data in order to retrieve data associated with matches while preserving the confidentiality of the unmatched data of both the mSOC and the organisation. While there exist privacy-preserving solutions that can aid in parts of this problem, no solution yet exists, to the best of our knowledge, that efficiently solves the problem entirely with our constraints on confidentiality.

To this end, we propose two privacy-preserving protocols that enable exact matching of variable multi-attribute IoC and network data. For both protocols, we analyse the theoretical complexity and test proofs of concept in order to highlight their strengths and weaknesses.

Contents

1 Introduction

If you want to view a store's inventory, you visit their website. If you want to apply for a job, you send the organisation an email. If you want to play a game, you download the software. What these services have in common is that they use the Internet, a computer network, to enable communication and connections between different users of this network. The Internet is an overarching computer network that connects many smaller networks, such as that of an organisation. The mentioned services allow an external user to interact with the network of an organisation. It is of vital importance that only certain parts of these internal networks can be accessed from the Internet in order to protect sensitive information. While in the past much of an organisation's infrastructure would be on-premise, meaning it is exclusively accessible using the organisation's internal network, nowadays the cloud is often used as an alternative [19]. As a consequence, organisations are more dependent on the Internet and their connection to it, as much of their data storage and computation is done in the cloud. These dependencies cause the Internet to be essential to the proper functioning of such organisations, as well as a potential risk.

While a connected world enables access to worldwide information for consumers and organisations, it can be used by malicious actors as well. This connectivity enables these actors to infiltrate the network of an organisation, to which they should not have access, without physical access to the premises. A malicious actor could even be on the other side of the planet, which impedes investigations by national law enforcement agencies [17]. A common approach for infiltration is to try to abuse a vulnerability, such as a lack of input sanitisation, which can be abused by giving malicious input, in the internet-facing infrastructure of a potential victim [35]. These approaches to break into a network often have certain indicators that can be used to detect attempts, successful or not. As an example, if the user input for the username field of a login page contains an SQL query, this indicates an infiltration attempt.

Using the access gained via a network, an actor has a plethora of options [48], such as: exfiltrate sensitive data from internal systems; disrupt the network by shutting down computers or encrypting its contents; remain hidden in order to spy or disrupt at a later moment. The impact of such attacks can be disastrous. For example, in 2015, Ukraine's power grid was hacked, which caused power outages for multiple hours [1]. The impact of this cyber attack was felt locally. Nevertheless, it is not unimaginable that such an attack may have global consequences. A more recent accident that illustrates this is the faulty update of Crowdstrike in the summer of 2024, which paralysed digital systems used for businesses around the globe. Planes were grounded and healthcare information was inaccessible as millions of computers were affected [20]. While this incident was not caused by a malicious actor, such an actor could have a similar impact given the right circumstances.

1.1. Security Operations Center

Fortunately, organisations can take measures to become more resilient to cyber attacks. They can, as an example, monitor their network traffic by duplicating it and sending it to a Security Operations Center (SOC) that attempts to find malicious activity in the data. This is illustrated in Figure 1.1, where

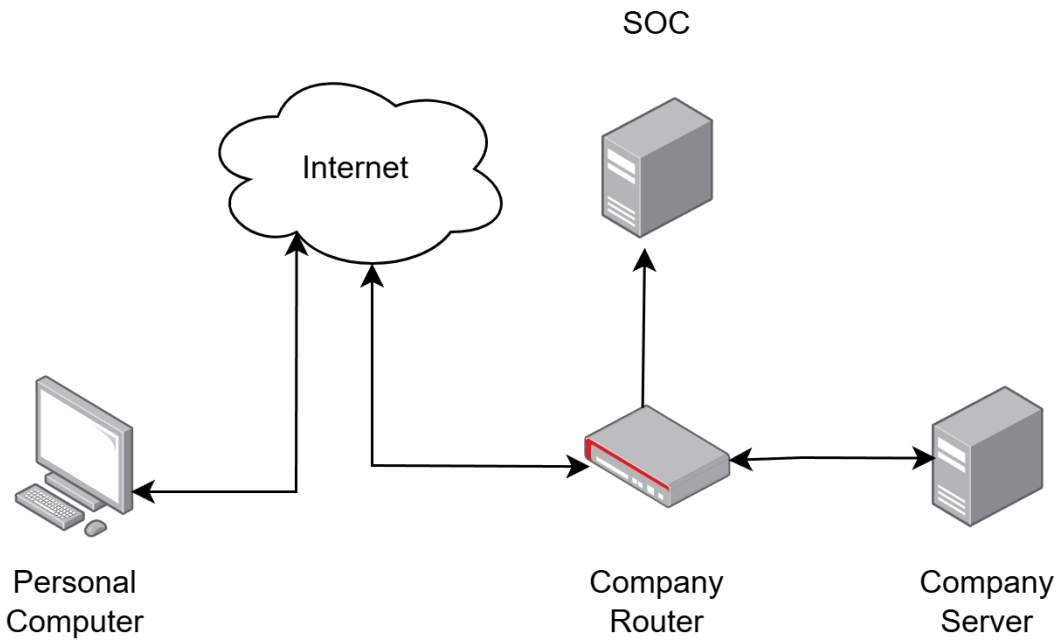

Figure 1.1: Example of how a SOC would fit in a network

all network packets that arrive at the Company Router are copied and sent to the SOC as well as the original recipient. This process does not disrupt the flow of the network packets. The tasks of a SOC relevant to network monitoring can be categorised as detection of malicious activity and threat hunting. Detection is determining whether a certain part of the data is possibly malicious using Indicators of Compromise (IoC). If an IoC matches with network data, this data is presented to the SOC in an alert. Then, the SOC verifies whether the detected data is indeed malicious or not, after which appropriate measures are taken. Threat hunting is digging through network data to determine what can function as IoC. These IoC are subsequently used for better detection and thus need to be of high quality.

IoC can be composed of anything present in a network packet that can differentiate between a malicious actor and a benign actor. In general, IoC are composed of one or more attributes, where an IoC matches if all of its attributes match. For example, a group of hackers may repeatedly attempt to access a network in the same manner by connecting with a specific port using a specific TCP/IP protocol. The two attributes of this IoC are the port and the protocol. While reusing a method may seem careless, one should realise that malicious actors have invested in such methods. As the costs of reusing a method are insignificant compared to the costs of devising a new method, these actors likely reuse methods in order to increase their profit. If such a method is rarely used by benign actors, it can be used to search for these malicious actors without causing regular false positive alerts at the SOC. A low false positive rate is important as a large number of false positives would ask more time of the SOC analysts and would affect their vigilance, called alert fatigue [40]. Searching for IoC in network data can be assisted with Intrusion Detection Systems, such as Snort [43] or Suricata [44]. These systems attempt to inspect all packets in real-time. One major performance improvement is obtained by designating certain attributes of the IoC as “important”. This designation is referred to as fast patterns. In this case “important” means that these attributes most uniquely identify the malicious activity that the IoC tries to detect. Due to these considerations, carefully constructing IoC is time-consuming and costly.

SOCs can be divided into two categories: managed SOCs (mSOC), an external party that monitors the infrastructure of many organisations as a service, and in-house SOCs, which only monitor the infrastructure of the organisation they are a part of. Privacy and confidentiality issues surrounding the detection of malicious activity in network data are predominantly relevant for mSOCs, as with an in-house SOC the data does not leave the organisation. Contracting an mSOC is interesting for organisations for various reasons, such as better scalability and lower costs [34, 47].

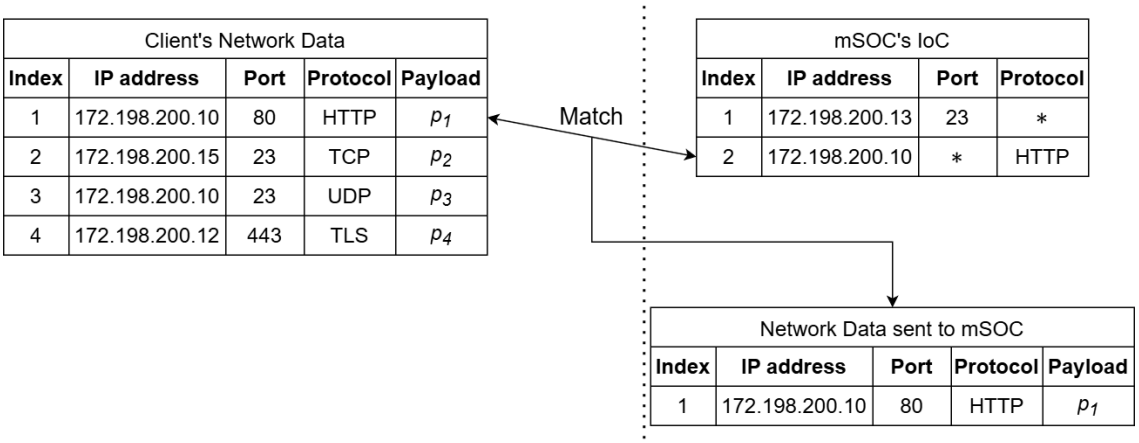

Figure 1.2: Example of how IoC are matched with network data

1.2. Privacy Preserving Matching of IoC

In most setups, an mSOC has access to all network data of an organisation that employs the services of the mSOC. This access is required as the IoC need to be compared to all network packets in order to find matches. While in normal circumstances this data is primarily accessed by an Intrusion Detection System, in principle all data is accessible to the mSOC and its technical employees. In practice, this accessibility is useful, for instance, for searching for context that is not directly linked to an alert but still relevant for an indication of the threat or for threat hunting. However, downsides to this access also exists. For example, a SOC analyst can see all network traffic of every employee of the organisation. While this data is often encrypted, URLs are not (entirely) encrypted, still giving the SOC analyst insight into the browsing behaviour of an employee. If you always start the day at work by browsing the news and checking your mail, the mSOC can observe this. In theory, the SOC can access metadata about everything anyone does within the network of an organisation. In cases where the network data is not encrypted, the mSOC can access even more private data.

Not all organisations may be concerned if a trusted mSOC accesses their data. The risk of a malicious actor on the network can outweigh the privacy loss of the employees, especially if the mSOC is bound by contract to only investigate data when relevant. However, organisations may have reasons to be more cautious about sharing access to their network data. For example, the organisation may have sensitive data, such as trade secrets, or they do not wish to trade privacy for security. Ideally, the organisation would share only the minimum amount of data needed to determine whether a malicious actor is in the network or not. In other words, they would only share data when it matches with an IoC, while all unmatched data remains private. This would result in a situation where the mSOC *cannot* observe the employee browsing the news, but *can* investigate the suspicious connection with a phishing website. On the other hand, an mSOC has reason to keep their IoC private. Due to the investments in these IoC, they are a significant part of the business model of the mSOC. While sharing all IoC with organisations may preserve more of the privacy of the organisation, this is a risk for the mSOC. Ideally, the mSOC would only share IoC when an organisation's network data matches, keeping the unmatched IoC private.

Figure 1.2 illustrates a simplified version of our problem. The organisation has network data with an associated payload, which is the context the SOC would need in order to analyse the network data matched on an IoC. The SOC has IoC where an attribute, a column in the figure, can contain "$*$" as an indication of a wildcard. The IoC does not take this attribute into consideration. In other words, per IoC the attributes that are considered for a match can differ. Any matches found are then sent to the SOC. In this case, only one entry in the organisation's network data matches on all attributes of an IoC and is sent to the SOC for investigation. Neither the SOC nor the organisation should learn any other information about the other party's data.

State-of-the-art research that provides solutions to similar problems is found in multiple fields within privacy enhancing technologies. Set intersection with privacy guarantees, known as Private Set In-

tersection (PSI), is a specialised form of Multiparty Computation that has been studied extensively in the last decade [33]. General PSI allows the participating parties to determine the intersection of their sets without learning any other information of another party outside of this intersection [30, 38]. In addition, different variants of PSI have been constructed. For example, Labeled PSI [12, 15] allows for the private transfer of data associated with the intersection and Threshold PSI [23] allows for finding an intersection with the additional constraint that the size of the intersection should exceed the threshold. Besides PSI, various solutions to similar problems have been proposed. For instance, Private Database Queries [7, 45] can be seen as a special case of PSI where one of the two parties has a set with only a single entry. Lastly, specifically on the problem of finding privacy preserving protocols for an Intrusion Detection System, Encrypted Packet Inspection has been proposed [9, 16, 42]. Encrypted Packet Inspection protocols offer an alternative for an Intrusion Detection System that can be used in real-time, while giving better privacy guarantees than a plaintext Intrusion Detection System.

1.3. Research Questions

In this work, we aim to match IoC, consisting of a variable number of attributes, with network data in order to retrieve data associated with these matches while preserving the privacy of the unmatched data of both the mSOC and the organisation. Important in this setting is that if X and Y denote the organisation's and the mSOC's dataset respectively, then $|X| \gg |Y|$. State-of-the-art methods for retrieving associated data [11, 15] do not efficiently support matching on multiple attributes. The state-of-the-art method for matching on combinations of attributes [25] requires the structure of the combinations to be known to both parties and scales linearly with the number of combinations, which is equal to scaling exponentially with the number of attributes if all combinations are allowed. However, none of the current solutions solve our problem entirely and no complete solution yet exists, to the best of our knowledge, that efficiently solves our problem, considering the privacy constraints described above. Therefore, we present the following research questions:

1. How can a variable number of attributes per IoC efficiently be matched with network data, preserving privacy when only a proper subset of attributes match?
2. How can network data, associated with matched IoC, be privately and efficiently transferred to a SOC?

While we focus on the problem in a SOC setting, we note that this problem also applies to other settings, such as querying a database where unique identifiers are not present or not known by the querent. A querent would then be able to use multiple identifiers, such as the combination of a name and an address, to retrieve relevant entries of the database.

1.4. Contributions

To address the gap as defined by our research questions, we propose two different protocols for a challenge that has not yet been extensively researched. We summarise our contributions as follows:

1. We combine different techniques and protocols in order to propose two protocols that provide a solution to our challenge. These protocols each use a different approach to the problem, which results in varying outcomes in terms of, e.g., efficiency.
2. We analyse our proposed protocols on both the complexity and actual runtimes using proofs of concept, which we have made publicly available. We highlight the strengths and weaknesses of our protocols.
3. Using our results, we provide a baseline for future research on this challenge. For future research, we identify possible directions of such research.

1.5. Outline

This thesis is structured as follows. Chapter 2 introduces relevant primitives and other preliminaries. Chapter 3 provides an overview of the existing work related to our research questions. In Chapter 4, we specify two protocols to answer our research questions. Chapter 5 provides the analysis of the two proposed protocols. Lastly, in Chapter 6, we discuss our findings and identify possible future research directions.

2

Preliminaries

In this Chapter, we provide the context necessary to understand current state-of-the-art works and the improvements we propose. We elaborate on possible methods for representing a set, cryptographic primitives used in PSI and the security models used in research to show under which assumptions a protocol is secure.

2.1. Set Representations

Set representations are used to make queries more efficient. A straightforward approach is the bitset, where each element is represented by a bit. Namely, if an element is in the set, its bit is set to 1 and vice versa. While this results in a bitstring with no loss of information, it scales directly with the size of the Universe of the set. A consequence is that even a small set is encoded in a disproportionately large bitstring, given that the Universe $\mathbb{U}$ is significantly larger than the size of the set. For example, a set of 100 IPv4 addresses would result in a bitstring with a length of 2^{32} as $\mathbb{U}$ of IPv4 addresses is 2^{32}. In this section, we review existing set representations used for, among other applications, PSI. However, while some set representation methods may seem to add some kind of privacy due to hashing the elements, this is not the case, since anyone with knowledge of the Universe can query all possible elements to reconstruct the original set with high probability. In the context of PSI, these methods should never be used without cryptographic primitives to ensure the privacy of the sets.

2.1.1. Bloom Filter

A Bloom Filter is one such set representation. The Bloom Filter was created by Burton Bloom [6] as a probabilistic data structure designed for fast set membership queries and low space requirements. The basic idea is to map each element with k different uniform hash functions to a series of m binary bins, setting the value of the mapped bin to 1. Then, the Bloom Filter can be queried to determine whether the queried element is in the set or not. If all k bins of the queried element are 1, this element is in the original set with high probability. As the hash functions are deterministic and since a bin flipped to 1 can never become 0 again, false negatives are impossible. However, removing an item from the Bloom Filter is impossible and false positives are possible. The False Positive Rate (FPR) is adjustable by choosing k and m. More specific, the FPR $error_N$, where N is the set size, is as follows:

$$error_N = (1 - exp(\frac{-kN}{m}))^k. \tag{2.1}$$

As this formula suggests, using a Bloom Filter can be counterproductive if the desired FPR can only be achieved by choosing m such that $m \geq U$ as using a bitset would result in a smaller and more accurate set representation.

Bloom Filters are sometimes used in Private Set Intersection protocols to obtain a concise representation of a set. Given two or more Bloom Filters, the intersection can be obtained by computing the bitwise AND of the two Bloom Filters, if the size m is equal for both Bloom Filters. However, the Bloom Filter itself does not provide privacy, as anyone with knowledge of the hash functions and the Universe

of the elements can reconstruct the original set. Therefore, in PSI, Bloom Filters are combined with Homomorphic Encryption in order to compute the bitwise AND of the bins without sharing an unencrypted Bloom Filter.

2.1.2. Cuckoo Hashing

Cuckoo Hashing is another method of representing a set using hashing. First proposed by Pagh and Rodler [36], this method uses two hash functions (H_1 and H_2) to hash elements to bins in the table. When an element x is inserted into the table, Cuckoo Hashing first attempts to insert x at $H_1(x)$. If this bin is already occupied, the element at $H_1(x)$ is displaced and x is inserted. The old element y of $H_1(x)$ is now inserted into bin $H_2(y)$. If this operation displaces an element, this process of displacing elements is repeated until a condition is met. Such a condition could be that all displaced elements are inserted into another bin, a predetermined amount of tries is reached or a loop is detected. If the Cuckoo Hashing table does not have enough bins, it is possible that an element cannot be inserted anywhere. This problem can be solved by re-choosing hash functions until all elements fit or by increasing the number of bins. Querying the Cuckoo Hashing table is straightforward. For querying element a, the querent only has to check bins $H_1(a)$ and $H_2(a)$. As a bin houses the actual element, it is possible to remove elements from a Cuckoo Hashing table.

Different implementations may slightly alter the Cuckoo Hashing method to better accommodate specific scenarios. For example, a stash can be added, which holds displaced elements that could not be reinserted. More hash functions can be added to give elements more possible bins and thus a lower chance of being unable to insert the elements.

2.1.3. Polynomial Roots

The final set representation method we will highlight is by using polynomial roots. This method differs from the previous two methods in that it does not use bins to assign elements to. Instead, it calculates a polynomial of which the roots are the elements. Using V as the set and $v \in V$ as elements in the set, the polynomial can be determined with the following formula:

$$F(x) = \prod_{v \in V} (x - v). \tag{2.2}$$

To find the intersection of two polynomials $F(x)$ and $G(x)$, one can simply compute $F(x) - G(x)$ and find the roots of the resulting polynomial. These roots are the matching elements with high probability. By encrypting the coefficients of polynomials F and G using homomorphic encryption, this method can be used for PSI.

A variant or generalisation of this set representation method is interpolating a polynomial over multiple points. Whereas the polynomial roots method encodes a set by interpolating a polynomial over the points $(v, 0) \quad \forall v \in V$, a general approach can encode key-value pairs into a polynomial where the points for the interpolation are $(a, v) \quad \forall (a, v) \in (A, V)$ where A is a set of keys or attributes paired with the set of values V. When subtracting two of these polynomials, the resulting polynomial has roots for all keys in the intersection. That is to say, for key-value pairs that match, the polynomial evaluates to $F(a) - G(a) = v - v = 0$.

2.2. Primitives

PSI protocols are often built on either Oblivious Transfer (OT) or Homomorphic Encryption (HE). In this section, we give an explanation of these cryptographic primitives and of primitives derived from these primitives.

2.2.1. Oblivious Transfer

The basic objective of OT is that a receiver can obtain one message out of two or more messages from a sender without sharing which message the receiver requests nor which other messages the sender holds. For example, 1-out-of-2 OT works as follows. Given Alice's messages $\{m_0, m_1\}$ and Bob's choice bit b, Bob should learn only m_b and nothing about m_{1-b}, and Alice should learn nothing about which message Bob received. An extension on OT, called OT extension (OTe) [4, 26], can be used to transfer many messages obliviously without executing as many instances of OT. In PSI, OT

can be used to facilitate the private computation of the intersection, making sure the elements not in the intersection remain hidden to the receiver, while the sender does not learn which elements are in the intersection.

2.2.2. Oblivious Pseudorandom Functions

Freedman et al. [21] introduce Oblivious Pseudorandom Functions (OPRFs) for the challenge of Keyword Search, where an OPRF can be used to make sure Bob does not learn any of Alice's unmatched records by making these random from Bob's perspective. An OPRF works as follows. Alice and Bob agree on an OPRF F. Alice samples a secret key k. Now for all of Bob's elements, Alice and Bob engage in the OPRF protocol, where Bob learns $F_k(x) \quad \forall x \in X$ and Alice does not learn Bob's x's. Bob does not learn k nor $F_k(y) \quad \forall y \notin X$. To implement such a function, OT is often used, but other methods, such as using elliptic curves, are possible as well.

2.2.3. Homomorphic Encryption

Homomorphic Encryption is a form of encryption for which certain computations on ciphertexts result in a valid ciphertext. The resulting ciphertext is the encrypted result of the same or a different operation on the original plaintexts. Formally, HE can be defined as follows. Given plaintexts m_1 and m_2 and an operation $+$ in the plaintext space, there exists an operation $*$ in the ciphertext space for which the following holds:

$$\mathsf{Dec}(\mathsf{Enc}(m_1) * \mathsf{Enc}(m_2)) = m_1 + m_2. \tag{2.3}$$

The number of different operations that can be performed on the ciphertexts determines the type of HE. Variants of Homomorphic Encryption are Partially Homomorphic Encryption, Somewhat Homomorphic Encryption, Levelled Homomorphic Encryption and Fully Homomorphic Encryption. Partially HE allows for operations on ciphertexts that correspond to either addition or multiplication in the plaintext space; Somewhat HE can evaluate some circuits, consisting of both additions and multiplications, in the ciphertext space. However, Somewhat HE does not have to be able to evaluate any specific circuits. Levelled HE takes another input which determines the maximum depth of the circuits the scheme can evaluate. Levelled HE can evaluate all circuits with a depth of at most this maximum. The main difference between Levelled HE and Somewhat HE is that Somewhat HE can handle circuits with increased depth if the parameters are chosen accordingly, usually increasing the ciphertext size. Levelled HE, on the other hand, takes a separate parameter for the depth, which does not influence the ciphertext size [3]. For both Somewhat and Levelled HE, each operation introduces noise. Once the noise becomes too large, the ciphertext cannot be correctly decrypted. This noise budget is what determines the maximum depth of the circuits. Fully HE is the most powerful variant as it allows for all circuits of operations of any type on the ciphertexts. However, Fully HE currently requires expensive bootstrapping after a certain number of operations to reduce the noise introduced by each operation.

3

Related Work

In this Chapter, we look at existing methods that attempt to tackle problems similar to ours, i.e. privately matching network data with IoC. For this, we primarily look at Private Set Intersection. However, other approaches will be considered as well.

3.1. Encrypted Packet Inspection

While the primary focus for finding a solution to our problem will be in the area of Private Set Intersection, Encrypted Packet Inspection attains similar results and is thus of some interest as well. Encrypted Packet Inspection was introduced by Sherry et al. with BlindBox [42], a Deep Packet Inspection system that attempts to tackle the problem that firewalls as middlebox often face nowadays: how to inspect network packets that are encrypted without compromising the privacy of the user. For this, BlindBox [42] leverages Searchable Encryption, which allows the firewall to search for IoC in network packets and to recover the plain text, using a trapdoor in the encryption, only if a match is found. Searching for IoC is achieved as follows: per packet, the client, i.e. the party whose network data is monitored by BlindBox, sends a tokenised version of the packet, where the tokens have a fixed length; if a token matches one or more IoC, the packet can be decrypted. In order to match on IoC of different lengths, tokens can be combined to form the IoC, as long as the length of the IoC is not smaller than the length of the tokens. For example, matching on the word “hacker” with a token size of 3 means the following tokens need to match: “hac”, “ack”, “cke” and “ker”. However, this tokenisation may leak information about the substrings if these substrings occur in other strings, e.g. “ack” the word “acknowledge”.

Over the years, different papers have proposed new methods that add some aspect of Deep Packet Inspection to Encrypted Packet Inspection or improve either the privacy or efficiency of Encrypted Packet Inspection. Canard et al. [9] improve on BlindBox by reducing the performance costs and by limiting the information available to the firewall, specifically the third-party sourced patterns it is trying to match. Desmoulins et al. [16] introduced flexible pattern sizes and the possibility to match regular expressions, which negatively affects the performance. In this method, a client can issue the trapdoors used for matching, giving them more control over what IoC are allowed to match on the network data at the cost of the effectiveness of the firewall.

Overall, Encrypted Packet Inspection is promising for applications such as firewalls. However, the existing Encrypted Packet Inspection protocols require network packets to be encrypted with trapdoors, which means the encryption for all network traffic is altered. This might be acceptable for uses such as firewalls as these are already in the middle of the network traffic. A SOC, however, generally only duplicates the network traffic to their servers without noticeable interference to the traffic. Therefore, it is a significant hurdle for companies that want to employ a SOC without altering a lot of their infrastructure.

3.2. Private Database Queries

A field of research more adjacent to PSI is Private Database Queries (PDQ). Simplified, PDQ is PSI where one party has a set with a single entry with one important difference: PDQ generally returns all

occurrences of the query, whereas PSI generally only returns whether there is a match or not. Returning all occurrences is desirable for our problem as IoC can occur in more than one network conversation. Therefore, a SOC would want to investigate, and thus receive, all matched network conversations. While these protocols often do not translate directly or efficiently to a setting where both sets are larger than one, they give insight into novel approaches of specific types of PDQ that may be applied to PSI as well.

Boneh et al. [7] propose a protocol for database queries with more than one attribute, specifically conjunctive queries, where records match only if all attributes match. The authors accomplish this functionality within reasonable time frames by using a polynomial representation of the query and dataset, where the resulting polynomial has the matching indexes as its roots. For example, if attribute *A* occurs in the database entries with index *{1, 4, 5}*, *B* in *{2, 3, 4}* and *C* in *{1, 4, 6}*, $A \wedge B \wedge C$ results in index *4*. Boneh et al. [7] mention that with these indexes the associated database entries can be fetched using Private Information Retrieval.

Tan et al. [45] propose a protocol which allows for the use of order comparisons. They achieve this by first encoding set elements to a vector of field elements followed by bitwise (order) comparisons of the query with the field elements in the vector of field elements. In short, this means all entries of the database are individually compared to the query. While private order comparisons are desirable for the SOC setting, e.g. for matching on conversations consisting of more than 500 MB to detect data exfiltration, the needed time and space complexity in order to apply this to the SOC setting is too high. Based on the results in [45] it would take multiple days for a set Y of 100 elements and set X of $1.000.000$ elements.

3.3. Private Set Intersection

PSI is a field of Privacy Enhancing Technologies where the objective is to privately compare two or more sets and receive the intersection without learning anything about the elements outside of the intersection. In the last decades, this field has improved significantly which has been accompanied by a plethora of papers. Therefore, we will select papers for this related work section based on their importance and applicability to our problem. Generally, PSI refers to the two different sets as the sender and the receiver, where the receiver is the party that learns the intersection.

Pinkas et al. [37] propose in their paper a novel protocol and optimisations of existing protocols using OTe. The novel protocol proposed uses OTe, specifically 1-out-of-n OTe, to execute a type of PDQ, namely private equality tests, for each element in the set of the receiver. In order to reduce the amount of private equality tests, the authors use Cuckoo Hashing, which significantly reduces the communication costs. In practice, the sender and receiver perform multiple OTs so that the receiver learns a randomised representation of an element in their own set using the bits of this element as choice bits similar to an OPRF. Subsequently, the sender sends the randomised representation of their element to the receiver. If the receiver determines that these two values are the same, the element is in the intersection with a high probability.

Kolesnikov et al. [29] propose a novel type of OPRF to use for PSI protocols. This OPRF, named a Batched, related-key OPRF or BaRK-OPRF due to a large number of OPRF instances with related keys, removes the dependence of the PSI protocol on the bit-length of set elements. This dependence is removed by replacing the error-correcting code of [37] with a pseudorandom code and constructing the OPRF such that each instance scales with the length of the pseudorandom code instead of the bit-length of the element. A pseudorandom code differs from a hash function as for the former it should be hard to find near collisions. Using the independence of the bit-length of the elements, Kolesnikov et al. improve protocols relying on OT, such as that of Pinkas et al. [37]. Specifically, they substitute the OT instances with instances of their BaRK-OPRF.

Chen et al. [11] propose a PSI protocol based on HE. The authors aim to construct a protocol that allows receivers with small sets and little computing power to efficiently perform PSI with a server with a significantly larger set. They achieve a communication complexity linear in the set size of the smaller set. By utilising Levelled HE, the authors reduce the computational cost.

In [38] Pinkas et al. build upon the PSI protocol based on OTe in [37] with added efficiency for the

unbalanced setting and scalability in set size. Using a smaller error correcting code, their protocol retains its dependence on the bit-length of the set elements. However, the total amount of bits of communication per OT is generally lower than that in [29] namely $\sigma - log_2 n$ where σ is the bit-length of an element and n is the set size. Additionally, the authors propose improvements to circuit-based PSI by leveraging OT and permutation-based hashing. Circuit-based PSI is a more general-purpose type of PSI allowing the parties to compute a function over the intersection.

In [39], Pinkas et al. introduce a novel way to hash set elements with probe-and-XOR of strings (PaXoS), which is generalised as an Oblivious Key-Value Store (OKVS) in later work. PaXoS leverages Cuckoo Hashing where binary strings can be stored and retrieved by XORing items stored in the Cuckoo Hashing table. Whereas Cuckoo Hashing stores a value to the location of one of its hashes, PaXoS stores a value at all locations associated with the value by the hash functions such that XORing these values results in the original value. The authors propose a PSI protocol based on PaXoS and OT. This protocol uses the XOR homomorphic properties of PaXoS to enable both parties to compute the same decoded value if the element they decode exists in both sets. Then, the sender hashes these decoded values for their elements and sends them to the receiver who checks for equalities.

Rughuraman and Rindal [41] improve on the OKVS data structure derived from PaXoS achieving low communication complexity. Since its introduction in [39], this data structure has become a new prevalent building block for PSI [33]. The main improvements to OKVS by the authors are transforming a matrix based on the keys to reduced row-echelon form and clustering non-zero values in a row. This results in efficient encoding and decoding operations for the OKVS. Using subfield Vector Oblivious Linear Evaluation PSI is constructed from the OKVS.

3.3.1. Labelled PSI

As general PSI has become more popular among research and practice, more subcategories have been coined in order to tackle more specific problems. One of these new subcategories is Labelled PSI (LPSI), where the receiver also receives labels associated with the set elements, instead of just the intersection. This new category was introduced by Chen et al. [12], who based it on Private Information Retrieval by keywords (PIR) introduced by Chor et al. [13]. In short, PIR can be seen as a specialised form of PDQ, analogous to LPSI as a form of specialised PSI.

Chen et al. [12] propose two initial approaches for LPSI for the unbalanced PSI setting using Fully HE, which are secure against a malicious sender. They first propose an approach compatible with the protocol proposed in [11], which uses polynomial interpolation in order to get the label if there is a match and a random element in the finite field $\mathbb{F}$ otherwise. The second proposed protocol improves the online computation of the former by using an OPRF pre-processing phase. First, the labels are encrypted using the OPRF values associated with the set elements. Subsequently, these encrypted labels can be sent to the receiver. If an element of the receiver is in the intersection, they can decrypt the associated label. This approach relies on the one-to-one relation of the elements and the labels. For multiple attributes, each combination would result in a unique encryption of the label in order to preserve confidentiality for partial matches.

Cong et al. [15] improve on the PSI protocol of [12]. By always using OPRF values instead of the elements themselves, instead of as an optional optimisation as in [12], Cong et al. remove the need for the costly extension field arithmetic. This improvement, however, has the consequence that the "labelled" part of [12] is not usable for this protocol. Cong et al. cope with this by extending the OPRF output to use one part to represent the element and one part to function as a key for a symmetrically encrypted label.

3.3.2. Fuzzy PSI

Another upcoming subcategory in PSI is Fuzzy PSI. For Fuzzy PSI the goal is to find matches that are not necessarily equal but can be "close" as well. Defining closeness and minimising the amount of extra set elements needed to cover these "close" elements is the main focus of these PSI protocols.

Chakraborti et al. [10] introduce a method that attempts to minimise the amount of elements added to the set for Fuzzy PSI with Hamming distance or Minkowski distance of first order. The Hamming distance is often used to determine the distance between two strings of equal length by counting the

number of positions at which the bits or characters are different. The Minkowski distance of first order is often used to determine the distance between two integers and is calculated by computing $|a - b|$ for integers a and b. Fuzzy PSI with Hamming distances can be useful for applications such as private biometrics, e.g. facial recognition, and Fuzzy PSI with Minkowski distance for applications such as verifying whether an IP address is in a range known to be malicious. Especially the latter is useful for the problem in this thesis. To attain this more efficient set expansion, Chakraborti et al. [10] propose to only add the prefixes of the elements that are needed to cover the range to set A. For example, for the range $(a - d, a + d)$ where $a = 7$ and $d = 2$ the inefficient solution would result in the (binary) set $0101, 0110, 0111, 1000, 1001$, whereas the solution presented in [10] would result in the (binary) set $0101, 011*, 100*$ where $*$ is a wildcard. The downside is that set B requires a small expansion as well, as it should cover the possible prefixes, which depends on the distance d. However, these expansions are significantly less than the expansion linear with distance d in the inefficient solution.

Garimella et al. [22] attempt to tackle this expansion of the sets, by exploiting a publicly known structure in the input sets. In short, if A is the original set and A^* is the augmented set with all elements close to $a \quad \forall a \in A$, then the objective is to make a PSI protocol that scales with the set size of A instead of the set size of A^*. The proposed protocol utilises functional secret sharing and aims to find the intersection of set A and set B where $(a, b) \in A \times B$ are within the distance threshold δ of each other. This type of fuzzy matching is accomplished by using an OKVS that stores functional secret sharing shares linked to a grid cell encompassing the desired range. However, this protocol assumes that these cells are disjoint, which is an assumption that may not hold in practice.

Uzun et al. [46] propose a Fuzzy Labelled PSI protocol in order to privately compare biometric data for, for example, facial recognition. They base the fuzzy part of their protocol on t-out-of-T matching, which is also known as Threshold PSI, where t is the threshold. Specifically, their protocol generates subsamples of the biometric query, which returns a secret share of the label if it matches with a subsample of the dataset of the server. The secret shares can then be combined to reconstruct the label if the querent obtains at least t shares. The threshold t is set by the server.

3.3.3. Private Matching for Compute

Private Matching for Compute is a subcategory of Private Set Operations, which is the overarching category for subcategories such as Private Set Intersection and Private Set Union. However, it introduces a solution that does not aim to solve a more specific problem necessarily. Rather, it attempts to give more flexibility to Private Set Operations as the output of Private Matching for Compute can be used for other types of Multiparty Computation without compromising the confidentiality of this initial output.

Buddhavarapu et al. [8] contribute to this subcategory with specific interest for Private Set Unions. However, the protocol presented is compatible with PSI as well. The authors aim to get Private Matching for Compute functionalities for sets with multiple attributes and non-unique attributes. For this they introduce Universal Identifiers that aim to reduce many-to-many connections to one-to-one connections. Additionally, the parties learn a mapping in order to get their respective set element with a given universal identifier. These Universal Identifiers can then be used as the input for further private computations.

Han et al. [25] propose a protocol with a similar objective to that of [8]. However, Han et al. [25] introduce an ordered-threshold-one matcher, which matches two elements if they have at least one matching feature. Features are valid combinations of attributes that identify an element. Which combinations are valid is determined by the participating parties beforehand. In practice, this means that the set is expanded, limited by the amount of valid combinations. The intersection is then calculated by using circuit-based PSI, which outputs the intersection in the form of secret shares.

4

Multi-Attribute Private Set Intersection

In this Chapter, we present our two protocols for Multi-Attribute Private Set Intersection. These protocols each have their own approach of tackling this novel challenge using different existing ideas and protocols, combined with ideas on how to enable and optimise this kind of set intersection. The first protocol focuses on inverted indexes in order to obtain a list of the matching indexes, which can then be used to retrieve the associated data. The second protocol transforms the set of the server into a new set with all possible combinations of attributes allowing for the usage of LPSI to get the matching records and the associated data. For both protocols, the intersection contains the decryption key of the associated data of matching records. In the remainder of this thesis, we will refer to the SOC as the Receiver and to the organisation as the Sender. The objective of the protocols is to obliviously transfer the matching associated data from the Sender to the Receiver. Both protocols are secure as long as both parties do not deviate from the protocol, i.e. in the Semi-Honest security model.

4.1. Inverted Index

The first protocol is a modified version of a protocol proposed by Boneh et al. in the fifth chapter of their paper [7] to which we will refer as *2PPDQ* (Two Party Private Database Queries). They describe *2PPDQ* as a two-party version of their main protocol. However, as *2PPDQ* was not the focus of their paper, it was not subjected to experiments and, to the best of our knowledge, was not implemented. Table 4.1 contains symbols and definitions we use for the remainder of this thesis. In Table 4.2, we showcase an improved version of *2PPDQ*. We will refer to this improved version as Inverted Index.

4.1.1. 2PPDQ

First, we give an overview of *2PPDQ* [7]. Boneh et al. propose two variants of their protocol, where the main difference is that the second variant, the fully-private variant, provides privacy for the attributes used and the first, the semi-private variant, does not. Privacy for the attributes means that the Sender does not learn which subset of the possible attributes A' are in the query. For example, if $A' = \{\text{sIP}, \text{dIP}, \text{Protocol}\}$ and an IoC consists of only an sIP, the Sender knows that the IoC only contains an sIP for the semi-private variant. Using the fully-private variant, the Sender does not learn which attributes in A' are used in the IoC.

Semi-Private 2PPDQ

The semi-private variant is defined as follows, where mentions of *steps* refer to the steps in Figure 4.2. The Sender first transforms each record in their dataset into a polynomial $D(j)$ where $D(a) = v$ (step 3). Then, the Sender computes the polynomials $\lambda(i)$ and combines both sets of polynomials such that: $D(i, j) = \sum_{r \in [|X|]} \lambda_r(i) \cdot D_r(j)$ (step 4). Now, the Receiver transforms their query into $Q(j)$ where $Q(a) = v, \forall (a, v) \in y$ and sends Enc$(Q(j))$ to the Sender (steps 9, 10, 11). The Receiver sends the attributes in the query $a' = \{a_1, ..., a_{|y|}\}$ to the Sender. The Sender computes the intersection

Table 4.1: Symbols and definitions

Symbol	Definition
	Both Protocols
X	The set of the Sender
Y	The set of the Receiver
$\|X\|$	The size of set X
R	Results of the protocol
k	Encryption key of a symmetric scheme
x	A single record in X
y	A single record in Y
(a, v)	Attribute-value pairs in x or y, the values are the entries of the sets
A	The attributes in X (i.e. the column headers)
A'	The attributes used in the intersection
$\mathbb{F}$	The plaintext space, a finite field
	Inverse Index
r	Record numbers $r \in [1, \|X\|]$
β	Cluster size
B	Number of Clusters
b	Cluster number $b \in [1, B]$
sk	Secret key used in either the OPRF or the HE scheme
pk	Public key used in the HE scheme
λ_r	Lagrange polynomial where $\lambda_r(i)$ evaluates to 1 only when $i = r$
i	Polynomial variable in the range $[1, \|X\|]$
j	Polynomial variable with values $a \in A'$
$D_r(j)$	Record polynomial for record r
$D_b(i, j)$	Database polynomial for cluster b
$\mathbf{v}$	Vector of attributes
ρ	Random scalar
T	The order of the bivariate database polynomial for each cluster
U	The largest order of the univariate record polynomials per cluster

Table 4.2: Inverted Index Protocol, improvements compared to Boneh et al. [7] in red

Step	Receiver $(Y,\ sk_{HE})$	Shared $(A' \subset A,\ \beta,\ pk_{HE})$	Sender $(X,\ sk_{OPRF})$
		Communication	
1			$B = \left\lceil \frac{\lvert X \rvert}{\beta} \right\rceil$
2			$v' = OPRF_{sk_{OPRF}}(v), \forall v \in X$
3			$D_r(j) = (D_r(a) = v'), \forall (a, v') \in x$
4			$D_b(i,j) = \sum_{r \in b} \lambda_r(i) \cdot D_r(j)$
5		$\xleftarrow{v'=OPRF_{sk_{OPRF}}(v), \forall v \in Y}$	
6		$\xleftarrow{(T, U, \lvert X \rvert)}$	
7	$\mathbf{v}_b = \sum_{a \in A'} \rho_a \cdot (1, a, a^{2 \bmod U}, ..., a^{T \bmod U})$		
8	$\mathbf{v}'_b = \mathsf{Enc}_{pk_{HE}}(\mathbf{v}_b)$		
9	$Q_y(j) = (Q_y(a) = v'), \forall (a, v') \in y$		
10	$Q'_y = \mathsf{Enc}_{pk_{HE}}(Q_y(j))$		
11		$\xrightarrow{(Q'_y, \mathbf{v}'_b)}$	
12			$R_{y_b}(i,j) = D_b(i,j) - Q'_y(j)$
13			$R_{y_b}(i) = \rho \cdot R_{y_b}(i,j) \cdot \mathbf{v}'_b$
14		$\xleftarrow{(R_{y_b}(i))}$	
15	$R'_{y_b}(i) = \mathsf{Dec}_{sk_{HE}}(R_{y_b}(i))$		
16	$Intersection_y \leftarrow \mathsf{Eval}(R'_{y_b}(i))$		

polynomial $A(i,j) = D(i,j) - Q(j)$ (step 12). If an (a,v)-pair in record r is equal to a pair in the query, $A(r,a)$ will result in zero.

Subsequently, the Sender computes $A_t(i) = A(i, a'_t), \forall t \in [|a'|]$ (step 14). Lastly, the Sender uses a technique by Kissner and Song [28] to blind all polynomials A_t by computing the cross product of the resulting polynomials and a random polynomial $S(i)$ of "appropriate degrees": $R(i) = \sum_{t \in [|a'|]} S_t(i) A_t(i)$. The resulting polynomial $R(i)$ is sent to the Receiver, who can decrypt it and find the roots in order to learn which indexes match the query and nothing else (steps 14, 15, 16).

Fully-Private 2PPDQ

The fully-private variant slightly differs from the semi-private variant. The Sender first transforms their dataset into $D(i,j)$ in the same manner (steps 3, 4). The Receiver transforms their query into $Q(j)$ (steps 9, 10). However, the Receiver then computes a vector $\mathbf{v}$ as an encryption of the attributes a' as follows: $\mathbf{v} = \sum_{t \in [|a'|]} \rho_t \cdot (1, a_t, a_t^2, ..., a_t^{|A'|-1})$ where ρ is a random scalar in $\mathbb{F}$ (steps 7, 8). The Receiver sends both $Q(j)$ and $\mathbf{v}$ to the Sender (step 11). Now, the Sender computes $A_r(j) = D(r,j) - Q(j)$ for each record (step 12). Then, the Sender computes the inner product of each polynomial $A_r(j)$ with the attribute vector $\mathbf{v}$ in order to evaluate the polynomial (step 13). By summing all of the results of the inner product, the Sender obtains the encrypted result of $\sum_{t \in [|a'|]} A_r(a_t)$ and sends it to the Receiver. The Receiver can decrypt the $|X|$ results and learns the indexes of the intersection by verifying which results decrypt to zero. All results for indexes not in the intersection will decrypt to randomness.

4.1.2. OPRF

For the semi-private variant of *2PPDQ* [7], which leaks the attributes of the query, Boneh et al. use a technique by Kissner and Song [28] to compute the resulting polynomial $R(j)$ by computing the cross product of the polynomial $A(j)$ and a random polynomial $S(j)$. This blinding ensures the resulting polynomial does not reveal any information other than the intersection. A side effect of using this method of blinding the result is that each polynomial coefficient has to be encrypted individually as opposed to batching multiple coefficients into a single ciphertext. The ciphertexts encrypting single coefficients are needed because calculating the cross product is not feasible with a batched ciphertext as this would result in the same number of ciphertexts as without batching, namely $|X| \cdot U$ ciphertexts.

For the fully-private variant, Boneh et al. propose an approach which evaluates all $D_r(j)$ individually before sending the results to the Receiver, resulting in $|X|$ ciphertexts.

The first improvement we propose is using an OPRF to replace the entries of both sets with a pseudo-random value in $\mathbb{F}$ that only the Sender can calculate, similar to the approach by Chen et al. [12]. Using an OPRF is similar to hashing, which Boneh et al. [7] propose to use for input values not in $\mathbb{F}$. Only if the Universe of the input values is in $\mathbb{F}$, *2PPDQ* can omit hashing the input values, thus avoiding the possibility of a collision between two input values. As seen in Table 4.2, adding an OPRF results in extra communication as the Receiver and the Sender interactively calculate the new entry values for the Receiver (step 5). The Sender can independently, thus offline, calculate the OPRF values of their own entry values (step 2). The Receiver does not know the OPRF values of values that are not in their set Y. The usage of an OPRF to randomise the entries together with the randomisation introduced by the other steps of the fully-private *2PPDQ* protocol causes the blinding of the result polynomial with the technique by Kissner and Song [28] to be redundant. The OPRF ensures the entries not in the intersection are random for the Receiver, who then cannot learn anything other than the intersection. By removing the need to calculate the cross product of polynomials, it is now possible to use batching to encrypt multiple coefficients into a single ciphertext. This significantly reduces the communication costs as fewer ciphertexts need to be sent. To give an example, using the HE scheme BFV [18] the number of coefficients in a single ciphertext can easily be 8192 or larger. As a result, the number of required ciphertexts is equal to the number of coefficients divided by the number of slots in the ciphertext. The computation costs are lowered as well, since additions and multiplications on the polynomials can be done at once instead of per coefficient.

4.1.3. Clustering

Another bottleneck of *2PPDQ* is that for large sets finding the interpolating polynomials λ_r of the record numbers is inefficient. Using Lagrange interpolation to find the coefficients of λ_r, together with an optimisation based on the fact that the points on which λ_r is based are all but one equal to $(i, 0)$, results in a complexity of $O(|X|^2)$ per polynomial, thus $O(|X|^3)$ for the entire dataset.

An intuitive improvement would be to divide the dataset in multiple clusters of a certain size β (step 1). This division reduces the number of polynomials and the order of these polynomials to β and $\beta - 1$ respectively as opposed to $|X|$ and $|X| - 1$. We can reuse the same polynomials for each cluster as long as the Receiver knows the order in which the resulting polynomials are sent. Then, the Receiver can add a multiple of β to the root of the resulting polynomial in order to get the correct index. A side effect is that each cluster has to be encrypted individually, which can be mitigated by choosing the parameters of the HE scheme such that the cluster size times the number of used attributes is approximately equal to, but not larger than, the batch size γ, i.e. $\beta \cdot |A'| \leq \gamma$. As long as this boundary is upheld, there exists a complexity trade-off in the size of β. For a larger β, we need fewer ciphertexts, but larger polynomials. Larger polynomials lead to more computation for both the evaluation of these polynomials as well as the interpolation of these polynomials, where the latter is part of the offline preprocessing phase. Fewer ciphertexts result in less online computation, for both the evaluation and the decryption of ciphertexts, and less communication. While the larger polynomials scale linearly with β, the ciphertexts scale incrementally as it is $\frac{|X|}{\beta}$ rounded up.

Lastly, using clusters and an OPRF affects the vector **v**. Where in *2PPDQ* the Receiver could simply compute each required power of $a \in a'$ and let the Sender reuse the encrypted powers, we cannot do the same due to the usage of batching. Instead, if the Sender sends for each cluster both the order T of the entire cluster polynomial and the highest order U of j, the Receiver can compute a single vector with a repeating pattern of the powers (step 7).

4.1.4. Associated Data

Akin to *2PPDQ*, our improved protocol solely provides the matching indexes as output. Our challenge, however, is to retrieve the associated data as well. Boneh et al. [7] propose that PIR can be used for this purpose. While PIR would provide an answer, it generally does not guarantee the confidentiality of the Sender's dataset nor does it allow for more than one query at a time. As LPSI can be seen as a form of multi-query PIR, while also providing confidentiality for the Sender's dataset, we opt to use LPSI instead. A drawback of current LPSI is that the protocols do not scale well with the label size.

Table 4.3: Inverted Index combined with LPSI

Step	Receiver (Y)	Shared	Sender (X)
		Communication	
1		$\xleftarrow{\bar{Y}=II(X,Y)}$	
2			$\bar{X} = \{r, k_r\}$
3			$\hat{X} = \mathsf{Enc}(X)$
4		$\xleftarrow{R=LPSI(\bar{X},\bar{Y})}$	
5		$\xleftarrow{\hat{X}}$	
6	$\mathsf{Dec}(\hat{X}_r, k_r) \quad \forall r \in R$		

In order to limit the label size, the Receiver constructs a new set consisting of the output of the Inverted Index protocol, the matched indexes. The Sender encrypts all the rows of their dataset with unique keys using a symmetric encryption scheme, resulting in $|X|$ encryption keys and encryptions. We call the encrypted dataset $\hat{X}$. Then, the Sender constructs a new set consisting of the indexes of X, each with a label containing the encryption key associated with the index. These labels are constant in size. Using these new sets the Sender and the Receiver engage in any LPSI protocol in which the Sender outputs nothing and the Receiver outputs the matching elements and their associated label. The matching elements are equal to the matching indexes. The Sender sends $\hat{X}$ to the Receiver in parallel to the LPSI protocol. Now the Receiver can decrypt only the rows of $\hat{X}$ where the index is in the multi-attribute intersection $X \cap Y$. As the Receiver knows which key encrypts which index, they do not have to attempt to decrypt each ciphertext.

4.1.5. Inverted Index Protocol

Table 4.3 defines the complete protocol, which we refer to as IIPSI (Inverted Index PSI). First, the Sender and the Receiver engage in the Inverse Index protocol of which the resulting matched indexes are the new set of the Receiver. The Sender then computes a new set $\bar{X}$ consisting of the indexes of the original set X as well as an encryption of X where each record is encrypted using a unique key, i.e. $\hat{X}$. These keys are the labels of the new set $\bar{X}$. Subsequently, the Sender and the Receiver engage in an LPSI protocol using their new sets as input. Finally, the Receiver can decrypt all records in $\hat{X}$ that match with the IoC using the keys obtained in the LPSI protocol.

For Inverted Index false positives are possible. After all, if a polynomial evaluates to randomness for a record that does not match, this randomness can be zero, thus indicating a match. However, false negatives are not possible as the resulting polynomials cannot evaluate to randomness for a matching record, but will always evaluate to zero. Likewise, LPSI generally allows for false positives, but no false negatives. False positives leak information to the SOC, but do not cause extra work as the SOC can match the IoC in plaintext to the output of the protocol in order to make sure only actual matches are analysed by the SOC employees.

4.2. Attribute Combination

As a second protocol, we propose to use a combination of LPSI and a modified version of the method used by Han et al. [25] to encode a set of quasi-identifiers, attributes in our case, into features. Since the method by Han et al. transforms the original multi-column set into a, albeit larger, single-column set, we can use this to enable typical LPSI to handle the multi-column sets in our challenge. For our second research question of retrieving the associated data, we use an altered version of our solution presented in Section 4.1.4. We will refer to the entire protocol as Attribute Combination PSI (ACPSI).

Han et al. [25] propose a method for encoding a multi-attribute set into a set of features. These features are combinations of the different attributes. Han et al. define a set of valid combinations that can identify a record and compute these features by concatenating the attributes in a valid combination. Now, two parties, each with a dataset, can use this transformation to obtain these features. Using the features the parties can perform, for example, a “join” operation on the two datasets, where certain entries that

Table 4.4: Attribute Combination for the Sender (AC_S)

Index	sIP	dIP	Prot
1	180	210	TCP
2	170	220	HTTP
3	180	220	TCP

→

Combination	Key
180	d
180|210	a
180|TCP	d
180|210|TCP	a
210	a
210|TCP	a
TCP	d
170	b
170|220	b
170|HTTP	b
170|220|HTTP	b
220	e
220|HTTP	b
HTTP	b
180|220	e
180|220|TCP	c
220|TCP	c

Table 4.5: Attribute Combination for the Receiver (AC_R)

Index	sIP	dIP	Prot
1	180		TCP
2		240	
3	170	220	HTTP

→

Combination
180||TCP
|240|
170|220|HTTP

match can be joined.

4.2.1. Transformation of Sets

First, we have to transform our sets into a form suitable for LPSI. While Han et al. [25] use the same transformation for both sets, we cannot do the same as our objective is to match an entire IoC and not a subset of an IoC. Additionally, the Sender cannot simply use a subset of the possible combinations as this would require the Receiver to send which combinations are possible, thus leaking some information about the IoC. For this reason, we propose two similar approaches to transform the data, which together accomplish what we wish to achieve. The Sender and the Receiver can still decide to only use a subset of the possible attributes in X if the Receiver deems it an acceptable risk or if omitting some attributes does not give the Sender more information. Choosing a subset of the attributes can be beneficial as the Sender then computes fewer combinations.

Transforming the Sender

Table 4.4 illustrates our approach for transforming the multi-column set of the Sender into a single-column set with label. In the example, we have an original set with a size of 3 and 3 attributes. The augmented set contains the combinations of the attributes of the original set, where "|" denotes concatenation, and an encryption key as label. Concretely, the Sender's set is transformed from

$$X = \{v_0, ..., v_{|x|}\} \quad \forall x \in X \tag{4.1}$$

into

$$X' = \{\{v_0\}, \{v_0|v_1\}, ..., \{v_0|v_1|...|v_{|x|}\}\} \quad \forall x \in X, \tag{4.2}$$

where duplicate combinations are omitted. The encryption keys are chosen such that each key encrypts one or multiple rows of the original set. If a combination occurs in more than one row, the associated key will decrypt all those rows. In this example, key a encrypts index 1, key b index 2, key c index 3, key d indexes 1 and 3 and key e indexes 2 and 3. By using the same key for combinations that are associated with the same records, we can reduce the number of data we encrypt, and thus the size of

Table 4.6: Attribute Combination Protocol

Step	Receiver (Y)	Shared $(A' \subset A)$	Sender (X)
		Communication	
1	$Y' = AC_R(Y, A')$		$X' = AC_S(X, A')$
2			$\hat{X} = \mathsf{Enc}(X)$
3		$\xleftarrow{\hat{X}}$	
4		$\xleftarrow{R=LPSI(X',Y')}$	
5	$\mathsf{Dec}(\hat{X}, r) \quad \forall r \in R$		

the encrypted database that has to be sent to the Receiver, compared to using a unique key for each combination.

Transforming the Receiver

Table 4.5 shows our approach for transforming the multi-column set of the Receiver into a single-column set. We are not interested in all possible combinations of the Receiver's set as each row represents an IoC of which individual attributes are of no interest. Therefore, we transform the Receiver's set

$$Y = \{v_0, ..., v_{|y|}\} \quad \forall y \in Y \tag{4.3}$$

into

$$Y' = \{v_0|...|v_{|y|}\} \quad \forall y \in Y. \tag{4.4}$$

Note that y can contain empty values as long as at least one v_i is not empty for $i \in |y|$.

4.2.2. Attribute Combination Protocol

Using these methods of transforming our datasets, we define the resulting protocol as illustrated in Table 4.6. First, both sets are transformed into a single-column form. This transformation incidentally makes the protocol more flexible for the Sender. If any unwanted combination would exist, the Sender can simply remove it from the augmented set, making sure the Receiver cannot match on the unwanted combination. Unwanted combinations may occur, for example, when the combination is too general and thus not likely a valid IoC, e.g., "Port==53". Subsequently, the set of the Sender is encrypted using the encryptions keys generated during the execution of AC_S as seen in Table 4.4. This encrypted set is then sent to the Receiver. Steps 2 and 3 are not dependent on any subsequent steps. Therefore, it can be done in parallel to the next steps. Next, the LPSI protocol is executed with X' and Y' as input. The result is sent to the Receiver only. Lastly, the Receiver attempts to decrypt every record using the keys they received. Only the records that match at least one of the Receiver's IoC can be successfully decrypted.

5 Analysis

In this Chapter, we will analyse the protocols described in Chapter 4. Firstly, we will give an analysis of the theoretical complexity of the proposed protocols and compare them to existing work when possible. Secondly, we will give an analysis based on implementations of the protocols where we analyse the computation in runtime and the communication in MB. We did not implement an LPSI protocol ourselves and will thus use a state-of-the-art protocol by Cong et al. [15] and their implementation *APSI* [14]. For both the complexity and the results in our implementation, we use *APSI* as the LPSI mentioned in our protocol description. We stress that our protocol works with any LPSI protocol, as long as the LPSI protocol allows for label sizes equal to the size of the symmetric encryption keys, and is not dependent on *APSI* specifically.

5.1. Complexity Analysis

We evaluate the communication complexity and computation complexity for both protocols. Table 5.1 denotes the extra symbols used in this analysis in addition to the symbols in 4.1.

5.1.1. Communication Complexity

For IIPSI, we first consider only the Inverted Index part and *2PPDQ*. Subsequently, we provide the communication complexity analysis of the entire Inverted Index PSI protocol when combined with *APSI*. Lastly, we illustrate how Attribute Combination affects the communication complexity of *APSI*.

Inverted Index

The communication complexity of the Inverted Index protocol consists of two significant parts: the ciphertexts and the OPRF. The parameters shared with the Receiver by the Sender, $(T, U, |X|)$, scale with B.

The number of times the OPRF is executed and thus the communication complexity of the OPRF phase depends on the size of the Receiver's set Y and on the number of attributes per IoC in Y. The number of attributes can differ per IoC. Therefore, we can only give an upper bound for the communication complexity. The computation complexity for the OPRF is the same as its communication complexity

Table 5.1: Extra symbols and definitions for the complexity analysis

Symbol	Definition
S	Number of slots in the Somewhat HE scheme, the polynomial modulus degree
B	$\frac{\lvert X\rvert}{\beta}$ rounded up, number of clusters
C	$\frac{\lvert A'\rvert}{S}$ rounded up, number of ciphertexts per query
D	$\frac{\beta \cdot \lvert A'\rvert}{S}$ rounded up, number of ciphertexts per cluster

$O(|Y| \cdot |A'|)$. The remainder of the communication complexity depends on the number of ciphertexts sent to the Receiver or to the Sender. For each query, the Sender receives the number of ciphertexts necessary to encrypt the query, equal to C. This results in a communication complexity of $O(|Y| \cdot C)$. Each query consists of a single ciphertext, assuming $|A'|$ is not larger than S. Lastly, the Sender sends back the resulting polynomials. The number of ciphertexts sent back is bounded by $B \cdot D$. For each query, the Sender sends B ciphertexts, assuming that $\beta \cdot |A'|$ is not larger than S. This results in a communication complexity of $O(|Y| \cdot B \cdot D)$.

The final communication complexity of the Inverted Index protocol is the summation of the three above:

$$O\big(|Y| \cdot |A'| + |Y| \cdot C + |Y| \cdot B \cdot D\big) = O\Big(|Y| \cdot (|A'| + C + B \cdot D)\Big). \tag{5.1}$$

With the above assumptions on the parameter choice, C and D are 1 and the complexity is:

$$O\big(|Y| \cdot (|A'| + B)\big) \rightarrow O(|X|). \tag{5.2}$$

Generally, the most significant variable is B since it scales linearly with the Sender's set size $|X|$ and $|X| \gg |Y| \gg |A'|$.

2PPDQ

For *2PPDQ*, we consider only the fully-private version for the complexity analysis. The only communication for this protocol is sending the query ciphertexts to the Sender and the result ciphertexts to the Receiver. However, remember that *2PPDQ* encrypts each coefficient individually. For a single query, *2PPDQ* sends at most $|A'|$ ciphertexts, thus $O(|Y| \cdot |A'|)$. For the results, the fully-private version sends one ciphertext per record in X, thus $O(|Y| \cdot |X|)$. The final communication complexity for *2PPDQ* is:

$$O\big(|Y| \cdot (|A'| + |X|)\big) \rightarrow O(|X|), \tag{5.3}$$

as again, $|X|$ is the most significant variable.

IIPSI

From the paper by Cong et al. [15] we learn that the communication complexity of *APSI* is $O((\log\log|X|)^2)$. In addition, we send an encrypted X to the Receiver which has a complexity of $O(|X|)$. Therefore, the combined communication complexity of IIPSI is:

$$O\Big(|Y| \cdot (|A'| + C + B \cdot D) + (\log\log|X|)^2 + |X|\Big) \rightarrow O(|X| + |X|). \tag{5.4}$$

The encrypted dataset is the most significant part of the communication complexity. However, as this dataset can be sent to the Receiver independently of the rest of the protocol, we do not remove the communication complexity of the protocol itself. In this case, both the protocol and the sending of the encrypted dataset have a complexity of $O(|X|)$.

Attribute Combination

The communication complexity of our Attribute Combination protocol is somewhat complex. While the main components are the communication complexity of *APSI* and the encrypted dataset X, for both X needs to be replaced by X', where X' denotes the transformed set X using the Attribute Combination protocol. We note that $|X'|$ is bounded by the equation $|X| \cdot (2^{|A'|} - 1)$, where the second term is the number of possible combinations, given that the number of attributes is equal to $|A'|$. However, the real number of combinations is largely dependent on how many unique values in each column of the Sender's dataset exist. If the values are mostly unique, $|X'|$ will approach $|X| \cdot (2^{|A'|} - 1)$, but if many values per column are equal or from a relatively small subset $|X'|$ will be closer to $|X|$. In general, the following holds: $|X| \leq |X'| \leq |X| \cdot (2^{|A'|} - 1)$.

The size of the encrypted dataset, however, is based on the number of unique keys and how many records each key encrypts, which is dependent on the number of unique combinations of records associated with the attribute combinations. Recall that in Table 4.4 five unique keys were needed for $|X'| = 17$ and these keys encrypted a total of seven records. Contrary to the size of $|X'|$, if the values of the columns are largely unique, the size of the encrypted dataset is lower. At the same time, if the

values of the records are largely equal to other records, the size of the encrypted dataset is lower as well. To get the theoretical maximum size of the encrypted dataset, X' should contain a large number of combinations that are as much as possible associated with a unique set of records. Intuitively, the upper bound of the encrypted dataset would consist of $\binom{|X|}{1}$ keys encrypting 1 record, $\binom{|X|}{2}$ keys encrypting 2 records, up until $\binom{|X|}{|X|} = 1$ key encrypting $|X|$ records, summed to $\sum_{i \in |X|}(\binom{|X|}{i} \cdot i) = |X| \cdot 2^{|X|-1}$ encrypted records. However, all combinations of records can only exist with a large enough $|A'|$. Each record can only be a part of $2^{|A'|} - 1$ combinations of records. If each record is in the maximum number of unique combinations of records, each record can be encrypted at maximum $2^{|A'|} - 1$ times. The upper bound of the size of the encrypted dataset is thus at most $|X| \cdot (2^{|A'|} - 1)$. Therefore, the following holds with E as the number of encrypted records in the encrypted dataset: $|X| \leq E \leq (|X| \cdot (2^{|A'|} - 1))$.

Where the original communication complexity of *APSI* is $O((\log\log|X|)^2)$, our protocol alters this to:

$$O\Big((\log\log(|X| \cdot (2^{|A'|} - 1)))^2 + |X| \cdot (2^{|A'|} - 1)\Big) \rightarrow O((\log\log(|X|))^2 + |X|). \tag{5.5}$$

The encrypted dataset, as in IIPSI, can be sent to the Receiver independently of the rest of the protocol and is thus again noted separately in the complexity.

5.1.2. Computation Complexity

For the computation complexity we follow the same structure as for the communication complexity. First, we give the complexity of the Inverted Index part of IIPSI. Then the complexity of *2PPDQ*. Followed by the complexity of the complete IIPSI protocol. Lastly, the computation complexity of the Attribute Combination protocol. We will give the complexity based on how many ciphertext-ciphertext multiplications are computed, but we will mention other significant computational bottlenecks as well.

Inverted Index

The computation complexity of the Inverted Index protocol is, similar to its communication complexity, dependent on the number of clusters B and the number of ciphertexts per cluster D. Per query, the Sender has to compute $D \cdot B$ ciphertext-ciphertext multiplications. Therefore, the total computation complexity is:

$$O\big(|Y| \cdot D \cdot B\big) \rightarrow O(|X|). \tag{5.6}$$

Two other significant parts of the computation are computing the interpolation polynomials over the record numbers with a complexity of $O(\beta^3)$ and computing the final result by evaluating the resulting polynomial. Using Horner's method, evaluating a polynomial has a complexity of $O(n^2)$ where n is the order of the polynomial.

2PPDQ

For *2PPDQ* the computation complexity is similar to that of the Inverted Index protocol, but requires more ciphertext-ciphertext multiplications. As the coefficients are encrypted individually, each query requires $|X| \cdot |A'|$ multiplications for the computation of the record polynomial multiplied by the attribute vector. Resulting in a total computation complexity of:

$$O\big(|Y| \cdot |X| \cdot |A'|\big) \rightarrow O(|X|). \tag{5.7}$$

IIPSI

From the paper by Cong et al. [15] we learn that the computation complexity of *APSI* is $O(\sqrt{|X|})$. Therefore, the total computation complexity of our IIPSI protocol is:

$$O\Big(|Y| \cdot D \cdot B + \sqrt{|X|}\Big) \rightarrow O\Big(|X| + \sqrt{|X|}\Big). \tag{5.8}$$

Attribute Combination

The computation complexity for the Attribute Combination protocol is once more largely identical to *APSI*. As our set size expands from $|X|$ to $|X| \cdot (2^{|A'|} - 1)$ resulting in a complexity of:

$$O\Big(\sqrt{|X| \cdot (2^{|A'|} - 1)}\Big). \tag{5.9}$$

Table 5.2: Complexity Comparisons; $|X|$ is the Sender's set size, $Z = 2^{|A'|} - 1$ is the number of combinations of attributes

Protocol	Communication	Computation
II	$O(\|X\|)$	$O(\|X\|)$
2PPDQ	$O(\|X\|)$	$O(\|X\|)$
IIPSI	$O(\|X\| + \|X\|)$	$O\left(\|X\| + \sqrt{\|X\|}\right)$
ACPSI	$O((\log\log(\|X\| \cdot Z))^2 + \|X\| \cdot Z)$	$O\left(\sqrt{\|X\| \cdot Z}\right)$

Another significant part of the computation are the decryption attempts using the results of the intersection. Unlike IIPSI, in the Attribute Combination protocol the Receiver does not know which records are encrypted using the keys in the intersection. Therefore, the Receiver has to attempt to decrypt all encrypted records, leading to a complexity of $O\big(|X| \cdot (2^{|A'|} - 1) \cdot I\big)$ decryptions, where I is the size of the intersection.

5.1.3. Comparison

Table 5.2 contains an overview of the complexities constructed in this section. Most variables can be left out in the complexity as, in general, we assume $|X| \gg |Y|$ and $|X| \gg |A'|$ to be true. To reiterate, in our use case realistic values for $|X|$, $|Y|$ and $|A'|$ are 10^6, 10^2 and 10 respectively.

Our Inverted Index compared to *2PPDQ* shows that both scale linearly with the large set size $|X|$ for both communication and computation. However, Inverted Index has a better performance in both communication and computation due to the lower scalars for $|X|$.

When comparing the communication complexity of IIPSI to ACPSI, we note that a significant part of both complexities is similar, because the same LPSI protocol and a similar approach to sending associated data are used. Given a large enough $|A'|$, ACPSI will have a higher communication complexity than IIPSI. However, if we are interested in answering solely our first research question, thus without the associated data, ACPSI will have a lower communication complexity as the double logarithmic term does not reach the size of IIPSI's $|Y| \cdot (|A'| + B \cdot D) \rightarrow O(|X|)$ for a large enough $|X|$.

The computation complexity of IIPSI is linear in $|X|$. However, while ACPSI is sublinear in $|X|$, it is exponential in $|A'|$. Therefore, for a larger number of attributes, IIPSI will have to compute relatively fewer ciphertext-ciphertext multiplications.

5.2. Implementation

We chose to implement a proof of concept of both the protocols in C++ and using Microsoft SEAL [32] for the homomorphic encryption. As the implementation of *APSI* by Cong et al. [15] uses C++ and SEAL as well, we argue the results are more suitable to compare the runtime of the protocols than if we used another programming language or HE library. BFV [18], which SEAL offers as one of its HE schemes, is used as it allows for some multiplications and is suitable for integers. For implementing the OPRF, we took inspiration from the OPRF used in *APSI*, which uses an elliptic curve. The Sender has a secret key k, the Receiver an item y. First, the Receiver hashes their item to an elliptic curve point Q and multiplies it with a random scalar r to get the point $T = rQ$, which is sent to the Sender. Then, the Sender multiplies T with the key k to obtain $U = kT$ and sends the point back to the Receiver, who then computes $r^{-1}U = r^{-1}krQ$ to obtain kQ. As elliptic curve, we use ristretto255 [24], which provides Curve25519 [5] as a prime order group instead of a curve with co-factor 8. We use the Ristretto variant of Curve25519 as implemented by libsodium [27], a library for cryptographic operations in C. Our implementation can be found on GitHub (`https://github.com/Rutger30/MAPSI`). This repository contains the code we wrote ourselves, a readme with instructions on how to install the dependencies and bash scripts to run the tests.

Our tests are run on a Virtual Machine using VMWare, which has 26GB of RAM allocated. The actual hardware consists of an Intel Core i7-11700KF CPU @ 3.6GHz and 32GB of RAM. All tests are run

Table 5.3: Parameters used for the experiments

Protocol	Inverted Index	LPSI [15]
Plaintext Modulus	2^{30}	2^{22}
Coefficient Modulus	174	218
Poly Modulus Degree	8192	8192
Cluster Size	1000	-

using a single core, except for the *APSI* parts, which use two cores as the Sender and the Receiver are executed separately. While this may speed up the execution of *APSI* marginally, the most significant part of the runtime depends only on the Sender, thus on a single core. All tests are run five times and the figures we use show the mean of these five runs.

We tested both protocols on a subset of the NetFlow dataset NetFlow v3 by Luay et al. [31]. Using a real NetFlow dataset rather than randomly generated data leads to results more representative for real-world scenarios. NetFlow is a method used to collect and analyse network data, which is used by SOCs as well to match IoC on. However, as mentioned in the complexity analysis, the actual computation and communication costs of the protocols depend to a great extent on how unique the values of the attributes are. As random data usually contains more unique values, the costs may vary significantly. Similarly, when tested on existing datasets, differences in costs can be explained by the number of unique values in the datasets.

Table 5.3 contains the parameters we used in our experiments. We use AES-256 as the symmetric encryption scheme for encrypting the associated data, which means our labels for the LPSI are 32 bytes. For Inverted Index, we use a cluster size of $\beta = 1000$ unless specified otherwise. This β results in $\frac{|X|}{1000}$ clusters and ciphertexts for the Sender's set. The used coefficient modulus 174 of the BFV scheme results in a security level of at least 128 bits [2]. For both *APSI* executions in our protocols, we use the parameter set *16M-4096-32* [14] made by Cong et al. [15]. For this parameter set, the Sender set size should be 16 million elements or lower and the Receiver set size 4096 elements or lower. The label size can be 32 bytes or lower. For IIPSI, the Receiver set size can be larger than 4096 based on the number of matches. In this case, the Receiver can make multiple queries of size 4096 or lower to query the Sender's dataset, which only increases the relatively short online time of *APSI* and not the costly offline time.

The polynomial modulus degree used is the same as we use in our Inverted Index protocol, namely 8192. This number is equal to the number of plaintexts that can be batched into one ciphertext. Higher polynomial modulus degrees result in more costly homomorphic multiplications. The plaintext modulus differs: Inverted Index uses 2^{30} while *APSI* uses 2^{22}. We opted for a larger plaintext modulus to reduce the false positive rate of IIPSI. However, as the noise budget decreases and the noise generation increases for a larger plaintext modulus, the plaintext modulus cannot be too large. For Inverted Index, each ciphertext only has one homomorphic multiplication, thus we can use the larger plaintext modulus without introducing too much noise.

While the parameter choice can limit the False Positive Rate (FPR), both methods can yield false positives. Inverted Index has a FPR of $\frac{1}{|\mathbb{F}|}$ per evaluation of the resulting polynomial as provided by Boneh et al. [7]. This FPR is the probability the resulting polynomial erroneously evaluates to zero. With our parameters, this results in a FPR of 2^{-30} per record. Under the assumption that evaluations of the resulting polynomial are uniformly random, the FPR per IoC would be $1 - (1 - 2^{-30})^{|X|}$. *APSI* does not give a general FPR, but with the used parameters it is $2^{-53.26}$ per element in the Receivers set, as found in the log files of the protocol. Analogous to Inverted Index, *APSI* does not produce any false negatives.

For 100 IoC and 1 million network data entries, this results in an FPR for IIPSI of $1-(1-x)^{100} = 8.9 \cdot 10^{-2}$, where $x = 1 - (1 - 2^{-30})^{10^6}$, for the Inverted Index part and subsequently an FPR of $1 - (1 - 2^{-53.26})^{|R|}$ for the *APSI* part, where $|R|$ is the size of the initial intersection or in other words, the number of matched records. The latter is bounded by $|R| \leq |X|$ and is thus at most $1 - (1 - 2^{-53.26})^{10^6} = 9.3 \cdot 10^{-11}$.

For 100 IoC and 1 million network data entries, the FPR for ACPSI is $1-(1-2^{-53.26})^{100} = 9.3 \cdot 10^{-15}$. Therefore, for our used parameters, the FPR for the protocols differ greatly with a factor of 10^9. This difference means that IIPSI will have significantly more false positives than ACPSI on average.

All Figures in the following subsections are based on the numerical data in Tables A.1 and A.2 in Appendix A. While the figures give accurate impressions of the results, the numerical data is valuable for exact comparisons.

5.2.1. Computation Comparison

We divide the computation into two categories: offline and online, where the former describes the preprocessing time and the latter describes the time starting at the first interaction per part of the protocol. In IIPSI the preprocessing time of the *APSI* part is counted as offline, even though it takes place after the first interaction in the Inverse Index protocol, as the parties will not interact for a relatively long time during this second preprocessing phase. Recall that the protocols are run with a single thread. Therefore, the following results give an indication of the efficiency of the protocols in practice, but the exact runtimes may differ.

In Figures 5.1 to 5.3 we compare the runtime for both protocols with three, six and eight attributes. The bars represent, from left to right, IIPSI offline, ACPSI offline, IIPSI online, ACPSI online, IIPSI total time and ACPSI total time. We use a log scale for the time to make sure all bars are meaningful and visible. In order to compare the three figures with each other, we configured the plots so that the y-axis has the same range for all three figures.

First, we consider Figure 5.1 comparing the runtimes with three attributes. For IIPSI we observe that the online time scales less favourably with the Sender set size than the offline time, although both scale with the Sender set size in some capacity. The growing offline time for larger Sender set sizes can be explained by the extra records that need to be interpolated. These extra records, and thus clusters and ciphertexts, explain the growing online time as well. The ciphertext-ciphertext multiplications grow linearly with the Sender set size and the Receiver has to evaluate more polynomials in order to get the indexes of the intersection.

For ACPSI, we notice that the online time scales significantly less with the Sender set size compared to the offline time. The explosive growth in the offline time can be explained by the augmented set size of the Sender which grows with a scalar of, in this case, at most $2^3 - 1 = 7$. However, the larger the original set size, the smaller the increase of the offline time relatively. We notice that this trend correlates with the augmented set size of the Sender, which becomes approximately four times the original set size for $|X| = 5$k in the Figure and decreases to approximately two times the original set size for $|X| = 1000$k as seen in Figure 5.5 and Table A.2. We speculate that the scalar decreases for larger sets as the attributes contain fewer unique values since network data from a single source will hold similar data over time. As an example, if one reloads a webpage, the network traffic will be similar to the initial request for that webpage. Comparing IIPSI to ACPSI with 3 attributes, we notice that while the online time of ACPSI is better in all cases, the total time of IIPSI and ACPSI approaches each other for larger set sizes. For the two largest set sizes we tested, the total time taken by IIPSI is lower than that of ACPSI.

In Figures 5.2 and 5.3, comparing the runtimes with six and eight attributes, we first notice that in both Figures IIPSI does not differ significantly from IIPSI in Figure 5.1. Therefore, we conclude that the same analysis as for with three attributes holds and that IIPSI is not significantly dependent on the number of attributes. The conclusion that the computation costs of IIPSI are not significantly dependent on the number of attributes is supported by Figure 5.4, which shows only small differences in computation time for the different number of attributes. For ACPSI, however, we observe that the offline time grows even more explosively, which is in line with our expectations as the Sender's set size grows with a larger scalar: at maximum $2^6 - 1 = 63$ for six attributes and at maximum $2^8 - 1 = 255$ for eight attributes. The online time shows the same trend as with three attributes, namely that if the augmented set size grows, the online time grows with a fraction. Results for set size $250 \cdot 10^3$ and upwards are missing for six attributes and set size $50 \cdot 10^3$ and upwards are missing for eight attributes as we did not have enough RAM in our test setup. However, the augmented set sizes can be seen in Figure 5.5.

When comparing IIPSI and ACPSI across all numbers of attributes, we observe that for the total time

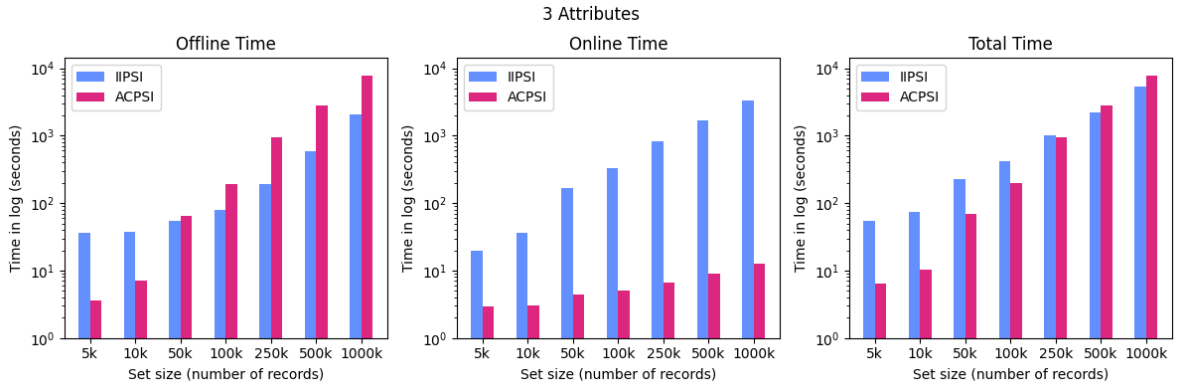

Figure 5.1: Computation results for both protocols with 3 attributes, time in seconds (in log scale) and Sender set size $|X|$, Receiver set size $|Y|$ is 94

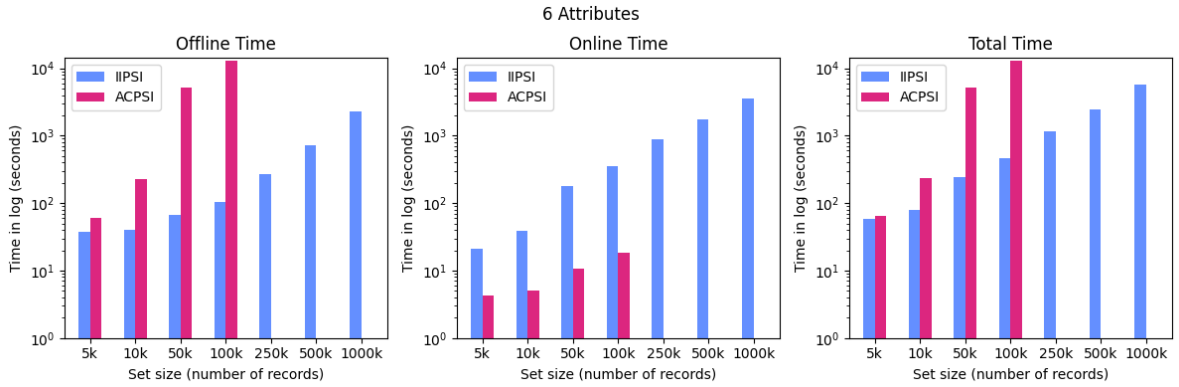

Figure 5.2: Computation results for both protocols with 6 attributes, time in seconds (in log scale) and Sender set size $|X|$, Receiver set size $|Y|$ is 100

ACPSI is more efficient when the number of attributes is low or when the original set size is small. For larger sets and more attributes, IIPSI is more efficient as it does not scale significantly with the number of attributes. However, the online time for ACPSI is lower across all tests, which can be a major factor in deciding whether a protocol is desirable or not as both parties need to be active and online for the duration of the online time.

5.2.2. Communication Comparison

To test the communication, we kept track of the amount of MB sent to the Sender or to the Receiver in order to get results independent of the network setting, like WAN or LAN. The communication time depends on the number of MBs and the throughput of the network. In Figures 5.6 to 5.8 we compare the communication for both protocols with three, six and eight attributes. The graphs represent, from left to right, the communication for sending the encrypted dataset, the communication for the protocols and the total communication. The total communication is equal to the encrypted dataset and protocol communication bars combined, but we chose to plot the bars separately as the encrypted dataset is the most significant part of the communication for nearly all Sender set sizes. Separating the costs more clearly conveys the added costs for answering our second research question, sending the associated data. We use a log scale for the time to make sure all bars are meaningful and visible. In order to compare the three figures with each other, we configured the plots so that the y-axis has the same range for all three figures.

First, we observe that the encrypted dataset of ACPSI is larger than that of IIPSI for all Sender set sizes, which is supported by the complexity mentioned in Table 5.2. This table shows that the encrypted dataset of IIPSI is bounded by $|X|$ and that of ACPSI by $|X| \cdot (2^{|A'|} - 1)$. As the lower bound of both is $|X|$, IIPSI will never have a larger encrypted dataset. The rest of the communication costs follow a

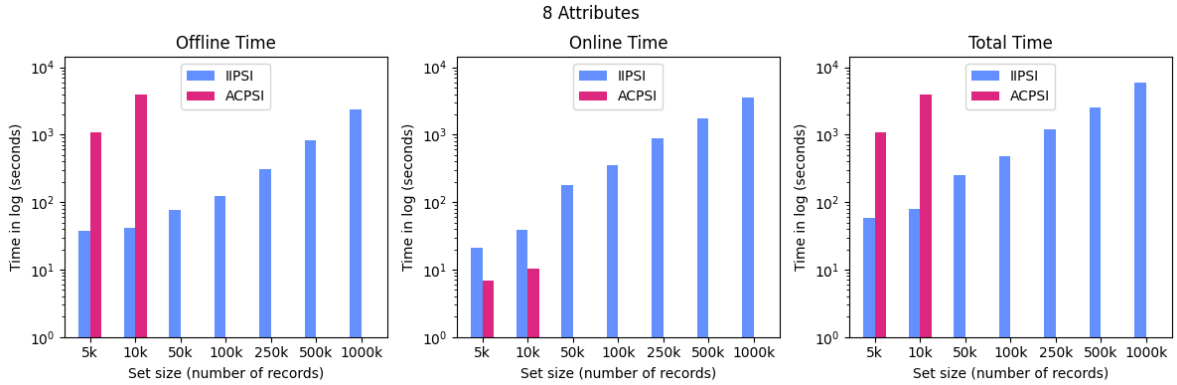

Figure 5.3: Computation results for both protocols with 8 attributes, time in seconds (in log scale) and Sender set size $|X|$, Receiver set size $|Y|$ is 100

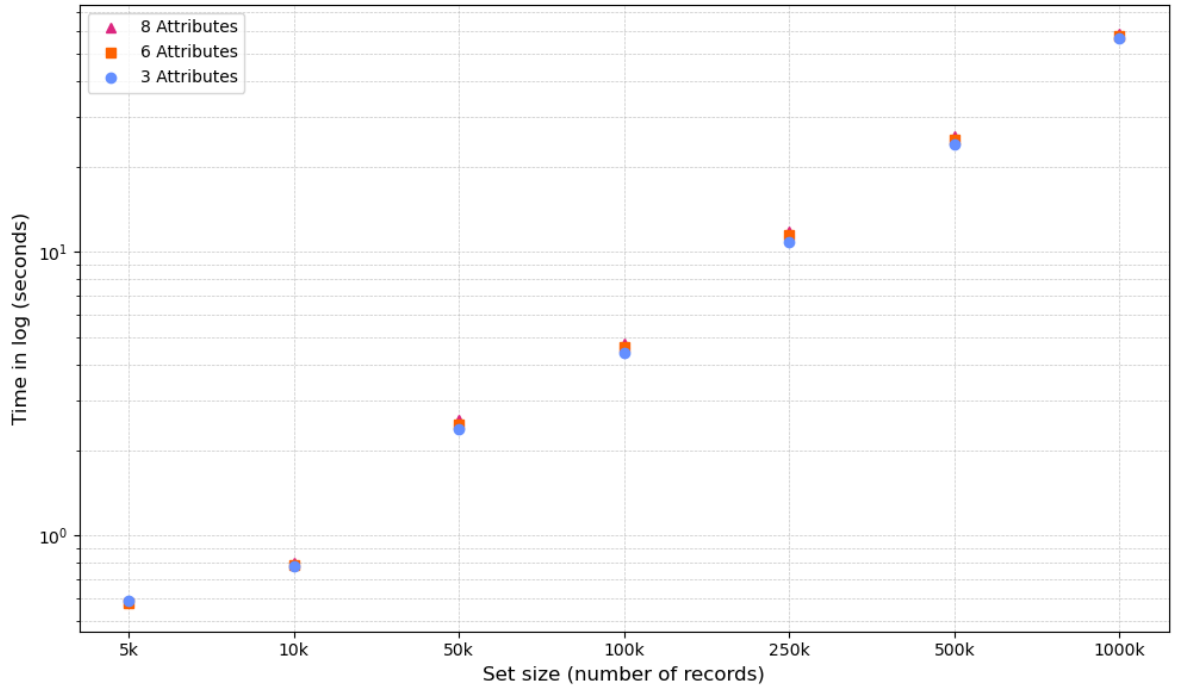

Figure 5.4: Computation results per query (in log scale) for each of the tested Sender set sizes for IIPSI

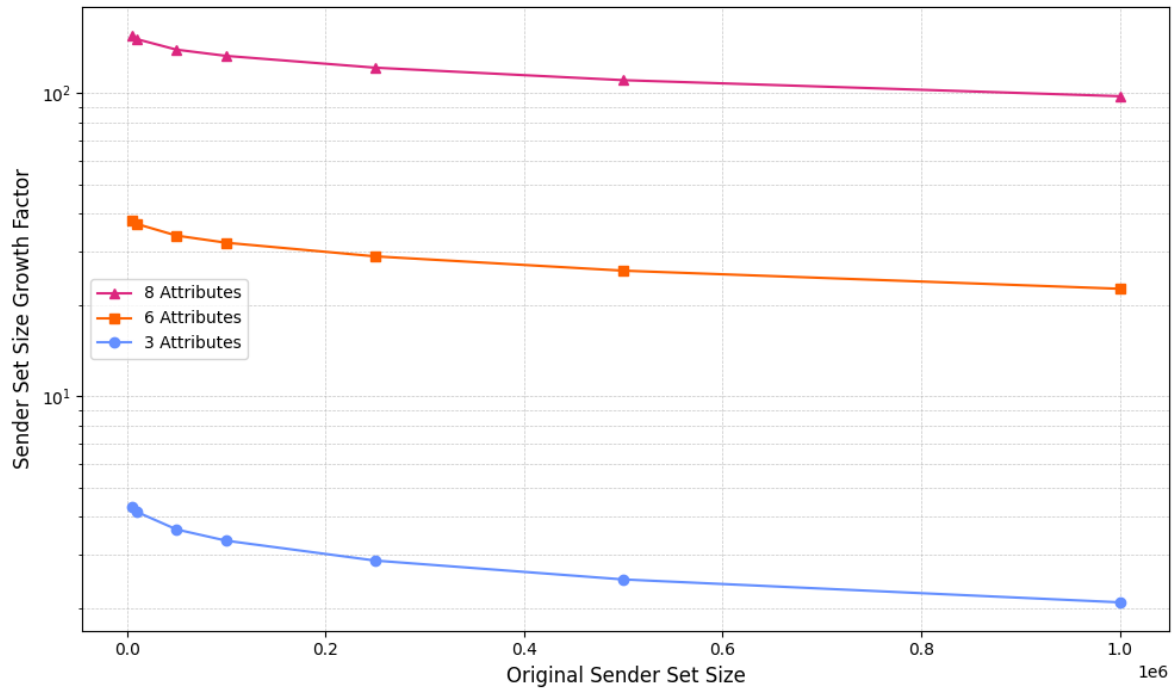

Figure 5.5: The factors with which the Augmented Sender set sizes grow (in log scale) compared to the original Sender set sizes for ACPSI

different pattern. In Figure 5.6, we notice that the costs for ACPSI with three attributes are lower for all set sizes of $50k$ or larger compared to IIPSI with three attributes. In Figures 5.7 and 5.8, we see that for both six and eight attributes follow the same trend. For the smaller Sender set sizes ($< 50k$) the costs of IIPSI are lower than that of ACPSI. If we compare these findings to the complexities in Table 5.2, this trend can be explained by the double logarithmic term $(\log\log|X|)^2$. For a small $|X|$, this term differs relatively more from the term $(\log\log(|X| \cdot (2^{|A'|} - 1)))^2$ at which point IIPSI's other term $|Y| \cdot (|A'| + B \cdot D)$ results in a smaller increase of communication costs than the difference of the two double logarithmic terms. For the total communication costs, we observe that the costs of ACPSI are always higher than the costs of IIPSI, which can be another factor, besides the runtime, in deciding what approach is the best for a specific use case. Lastly, similar to the computation costs, we observe that IIPSI does not differ significantly across the different numbers of attributes, which reflects the communication complexity that states that the dependence on the number of attributes is relatively small given $|X| \gg |A'|$. The conclusion that the communication costs of IIPSI are not significantly dependent on the number of attributes is supported by Table A.1, which shows only small differences in communication costs for the different number of attributes.

5.2.3. Cluster Size

Finally, we compare different cluster sizes β for the Inverted Index protocol. Intuitively, one would assume that a larger cluster size would result in more offline computation costs, less online computation costs and less communication costs. After all, the costs for interpolating the cluster indexes scale with the cluster size cubed β^3 and the online computation and communication costs primarily scale with the number of ciphertexts, which decreases as long as $|A'| \cdot \beta \leq S$ holds. However, in Figure 5.9 we observe that while the cluster interpolation and the communication costs follow our intuition, the online costs decrease from cluster size 250 to 750, but increase for the larger cluster sizes. While the ciphertext-ciphertext multiplications decrease for larger cluster sizes, thus leading to less computation costs for the Sender, the Receiver has to evaluate larger resulting polynomials in order to get the final results, which ultimately leads to an increase in the online computation time, despite the lower number of ciphertexts.

Lastly, for the cluster interpolation in the left graph of Figure 5.9 we notice that it does not follow the

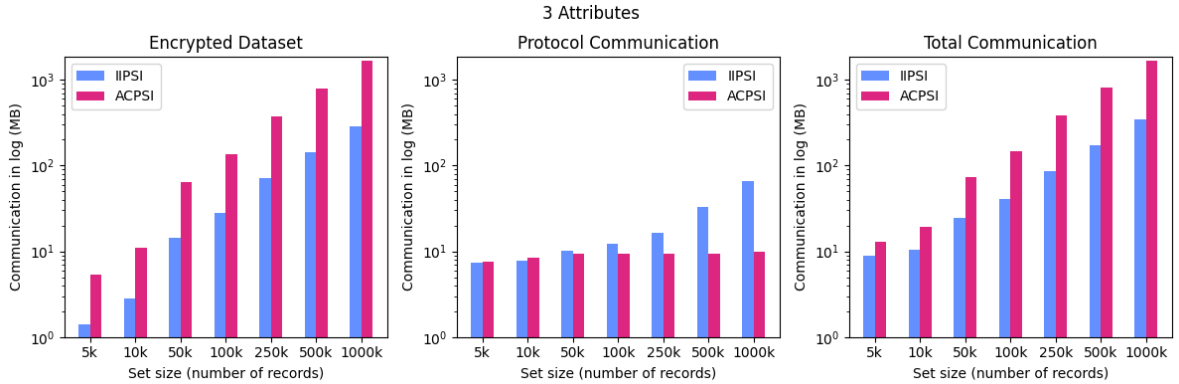

Figure 5.6: Communication results for both protocols with 3 attributes, communication in MB (in log scale) and Sender set size $|X|$ on the x-axis, Receiver set size $|Y|$ is 94

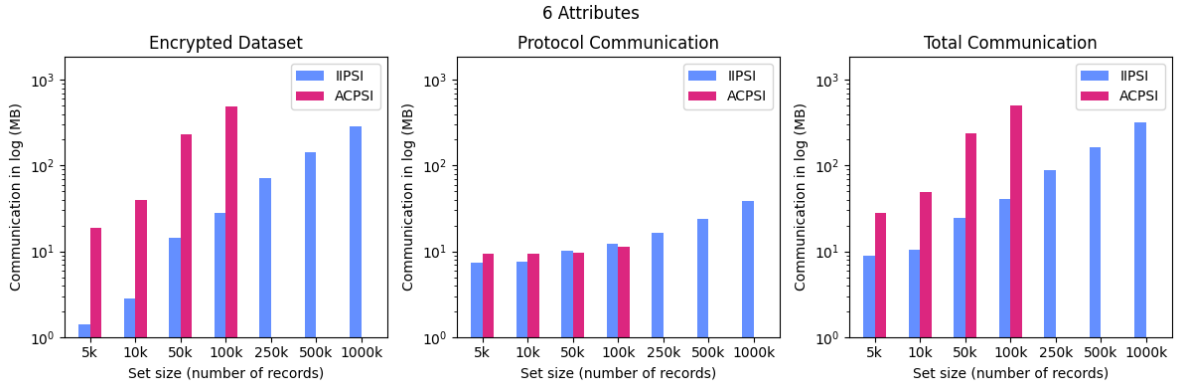

Figure 5.7: Communication results for both protocols with 6 attributes, communication in MB (in log scale) and Sender set size $|X|$ on the x-axis, Receiver set size $|Y|$ is 100

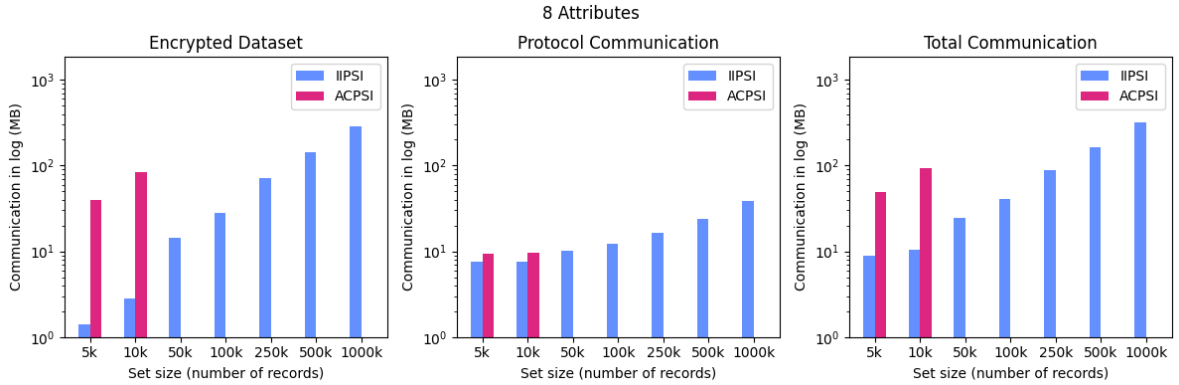

Figure 5.8: Communication results for both protocols with 8 attributes, communication in MB (in log scale) and Sender set size $|X| onthe x - axis$, Receiver set size $|Y|$ is 100

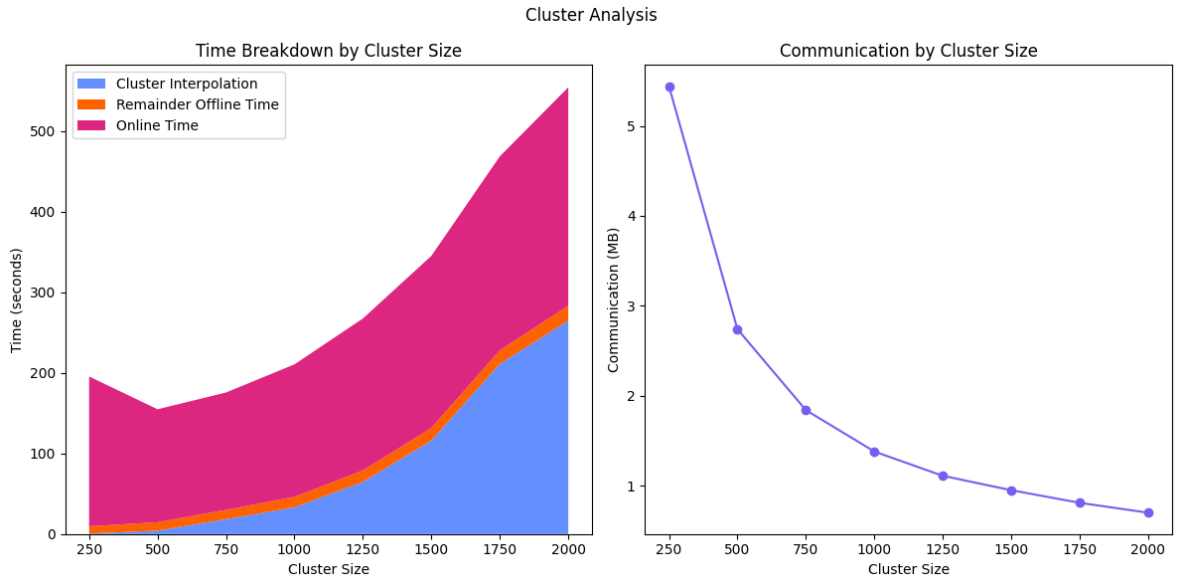

Figure 5.9: Results of Inverted Index for different cluster sizes with Sender set size 50k, Receiver set size 94 and 3 attributes

expected curve of β^3. This may be explained by the fact that the polynomials obtained by interpolating the cluster indexes are reused for each cluster. However, if β is not a divisor of $|X|$, another polynomial with an order equal to $|X| \mod \beta$ has to be calculated, leading to higher computation costs for cluster sizes 750, 1500 and 1750.

5.2.4. Correctness

In our experiments, a single false positive occurred, in one of the runs of IIPSI with set size $1M$ and 8 attributes. As the FPR of IIPSI is low and that of ACPSI negligible, these results are not unexpected. However, during initial experiments to determine realistic parameters, we noticed that when multiple false positives occur for IIPSI, these false positives were sometimes related to each other. For example, if IIPSI resulted in 10 false positives, 8 out of 10 false positives had the same source port number and destination IP address. Both attributes were in the set of attributes used for the intersection. However, these combinations of port numbers and destination IP addresses did not exist in the set of IoC. We speculate that these false positives may occur when OPRF values within an attribute collide. If such a collision causes all of the attributes of an IoC to match on the respective attributes in some records, all records with these attributes will result in a match. For IoC with more attributes, more collisions are needed to produce such a false positive. As a network dataset may contain many records with partially matching attributes, these false positives can result in the leakage of many unmatched records. How many unmatched records may be leaked this way depends largely on the number of attributes in the IoC, where an IoC with only one attribute may result in the most false positives, and on how many subsets of attributes are shared among the records. Even one collision in the values of an attribute can result in multiple false positives if the records already partially matched with the affected IoC.

These additional false positives are introduced by the usage of an OPRF to randomise the values. *2PPDQ* by Boneh et al [7] does not suffer these collision based false positives *if* the values are elements of the finite field $\mathbb{F}$. For values not within the range of the finite field $\mathbb{F}$, Boneh et al. propose to hash the values to $\mathbb{F}$, thereby introducing the collision-based false positives for *2PPDQ* as well.

While the probability for a collision of the OPRF output for a given IoC value is $\frac{1}{|\mathbb{F}|}$, given a uniform OPRF, the actual FPR caused by these collisions is difficult to determine. Depending on the network data and the IoC, a single collision could cause many false positives and many collisions could cause none. For example, if the OPRF value of a single-attribute IoC collides with an attribute that occurs x times in the network data, the single collision will cause x false positives. However, for a collided OPRF value of a multi-attribute IoC, where the other attributes of the IoC do not fully match or collide with any entries in the network data, the collision causes no false positives. The probability for at least

one collision can be calculated using the birthday problem, which results in:

$$P(\text{Collision}) = 1 - \frac{|\mathbb{F}|!}{(|\mathbb{F}| - \delta)!|\mathbb{F}|^{\delta}}, \tag{5.10}$$

where δ is the number of unique values in a certain attribute. We observe that the probability of at least one collision grows with an increasing number of unique values. On the other hand, the maximum number of false positives caused by a single collision decreases with an increasing number of unique values as each unique value occurs less often, given a constant set size. Therefore, while these collisions may occur relatively frequently, the impact on the FPR of IIPSI is marginal, yet significant. Estimating a general FPR is non-trivial as it largely depends on the network data and IoC. However, as in our tests only 1 false positive has occurred, we conclude that IIPSI is usable in realistic scenarios without incurring a large number of false positives.

6

Discussion and Future Work

In our current geopolitical situation, security in the cyber domain becomes increasingly important. SOCs play an important role for this type of security. However, in the current form the privacy and confidentiality of network data of organisations is not guaranteed. In this thesis, we propose a novel step towards more privacy and confidentiality in network monitoring by assuring that, with high probability, only network data with indicators of maliciousness are shared with the SOC. In this Chapter, we will discuss the results and limitations of our proposed protocols and whether our research questions are answered. Subsequently, we propose possible future directions for research related to our challenges.

6.1. Discussion

First, we restate our research questions:

1. How can a variable number of attributes per IoC efficiently be matched with network data, preserving privacy when only a proper subset of attributes match?
2. How can network data, associated with matched IoC, be privately and efficiently transferred to a SOC?

Both our proposed protocols provide an answer to the first research question, with the main difference being the efficiency and FPR. As analysed in Chapter 5, IIPSI has relatively high online computation costs and relatively low communication costs. The main benefit of IIPSI is that the costs do not scale significantly when the number of attributes is increased. On the other hand, ACPSI retains significantly lower online computation costs. If the FPR should be minimised, ACPSI is better suited. IIPSI grants the Receiver/SOC full freedom over their choice of IoC, while with ACPSI, the Sender/organisation can choose to make the protocol ignore some possible IoC by omitting combinations. Assuming the encryption scheme chosen to encrypt the dataset is secure, the protocols preserve the privacy of all unmatched network data, except for any false positives. We conclude that, depending on the above considerations, both protocols can be an efficient solution to our first research question.

For our second research question, both protocols use a similar approach. While both protocols provide an answer to the second research question, the actual result for ACPSI is significantly less efficient than that for IIPSI due to the larger encrypted dataset for the former. Additionally, both protocols send all data, relying on symmetric encryption for the privacy and confidentiality of the unmatched data.

In general, our protocols provide mSOCs and organisation a privacy preserving alternative to network monitoring. While our protocols are significantly slower than the real-time network monitoring offered by software such as Snort [43], they can provide a solution to organisations that do not wish to share all network data, but only the matched network data, if it is acceptable the IoC are matched within, for example, an hour.

6.2. Limitations

A limitation of our analysis is that all tests are run just once, causing the results to possibly vary more than they should. Additionally, our hardware was not sufficiently powerful, resulting in missing data and possibly slower runtimes if the RAM was effectively full and resorted to extensive paging. This issue is more likely to affect ACPSI due to the explosive set growth, which is a limitation of ACPSI itself. Furthermore, we did not implement parallelisation into our protocols, which means the results, while accurate, are not per se representative for real world applications where the protocols would be run on multiple threads.

Limitations on our protocols are as follows. For IIPSI, the organisation has no guarantees that the IoC used by the SOC are sensible, that is to say that the IoC are not fabricated in such a way that they will likely match on as many records as possible. While the impact can be somewhat mitigated by limiting the number of queries done on the dataset, the organisation still has to trust the SOC not to abuse the confidentiality of their IoC. As mentioned, this limitation is somewhat mitigated by ACPSI as the organisation can leave out combinations that are non-sensible according to the organisation, although this mitigation may impede the effectiveness of the SOC. Another limitation of IIPSI is that the possible OPRF collisions for the attribute values may have a large impact on the number of false positives.

Our protocols allow for solely exact matching. However, IoC often contain other types of matching as well. For example, matching on a group of similar domain names may use a regular expression and matching on the number of outgoing bytes requires order comparisons ($<, \leq, >, \geq$). Additionally, our protocols send all network data instead of only the associated data, resulting in more communication costs. Lastly, where plaintext Intrusion Detection Systems can be used in real-time, our protocols are not suited for such an approach. However, if the network data can be processed by our protocols every few hours and this delay in detection is not critical to the security of the organisation's network, our protocols may be a useful private alternative.

Lastly, the lion's share of computation is done at the Sender, which is the organisation, while the mSOC offers the network monitoring as a service. This distribution of the computation might limit the usability of our protocols, as an organisation is likely not keen on sacrificing significant computational power for a service they buy. An mSOC could attempt to limit the impact of this distribution of computation by providing hardware at the premises of the organisation. However, this solution increases the costs for mSOC. Therefore, if this distribution of computation is an issue in a certain scenario, it impacts the usability of our proposed protocols.

6.3. Future Work

Future work can be divided into two options, improving the computation costs, communication costs or FPR of our protocols or improving the functionalities in regards to the IoC matching. Obvious improvements to our protocols are solutions to the weaknesses of both protocols. For IIPSI, a direction could be to reduce the FPR or decrease the online computation costs. For ACPSI, on the other hand, reducing the explosive set expansion due to the many combinations would reduce the overall costs of the protocol. For both protocols, sending the entire encrypted dataset is a significant cost. Thus, future work could focus on a new method in which only the matched records are sent, instead of the keys for the matched records and the entire encrypted dataset.

As for the functionalities of our protocols, one could attempt to combine Fuzzy PSI [10, 22, 46] with our protocols in order to be able to match on IP ranges or simple regular expressions. Especially for matching on IP ranges, the fixed structure of these ranges could be exploited. As each IP range notes the number of relevant bits out of the total 32 bits of an IPv4 address, these relevant bits can be used as a prefix as proposed by Chakraborti et al. [10]. However, this approach would further increase the Sender's set size.

6.4. Conclusion

Privacy and data confidentiality are growing concerns for individuals and organisations alike. Security is often used to diminish the importance of privacy and data confidentiality. For example, outsourcing network monitoring to managed SOCs is a common business decision. We propose solutions to this

supposed contradiction in the setting of managed SOCs. We note that our solutions can be applied in other settings as well. For example, if the Sender has a set containing personal data of customers and the Receiver wants to learn which customers of the Sender live in a certain country or have the combination of a certain age and name [7, 25]. This research is a step towards a safer and more secure society with a significantly lower impact on the privacy of individuals and organisations alike.

References

[1] Cybersecurity & Infrastructure Security Agency. *Cyber-Attack Against Ukrainian Critical Infrastructure*. URL: https://www.cisa.gov/news-events/ics-alerts/ir-alert-h-16-056-01 (visited on 03/13/2025).

[2] Martin Albrecht et al. *Homomorphic Encryption Security Standard*. Tech. rep. Toronto, Canada: HomomorphicEncryption.org, Nov. 2018.

[3] Frederik Armknecht et al. *A Guide to Fully Homomorphic Encryption*. Cryptology ePrint Archive, Paper 2015/1192. 2015. URL: https://eprint.iacr.org/2015/1192.

[4] Gilad Asharov et al. "More Efficient Oblivious Transfer Extensions". en. In: *Journal of Cryptology* 30.3 (July 2017), pp. 805–858. ISSN: 1432-1378. DOI: 10.1007/s00145-016-9236-6. URL: https://doi.org/10.1007/s00145-016-9236-6.

[5] Daniel J Bernstein. "Curve25519: new Diffie-Hellman speed records". In: *Public Key Cryptography-PKC 2006: 9th International Conference on Theory and Practice in Public-Key Cryptography, New York, NY, USA, April 24-26, 2006. Proceedings 9*. Springer. 2006, pp. 207–228.

[6] Burton H. Bloom. "Space/time trade-offs in hash coding with allowable errors". In: *Commun. ACM* 13.7 (July 1970), pp. 422–426. ISSN: 0001-0782. DOI: 10.1145/362686.362692. URL: https://doi.org/10.1145/362686.362692.

[7] Dan Boneh et al. *Private Database Queries Using Somewhat Homomorphic Encryption*. Publication info: Published elsewhere. Full version of ACNS 2013 paper. 2013. URL: https://eprint.iacr.org/2013/422.

[8] Prasad Buddhavarapu et al. *Multi-key Private Matching for Compute*. Publication info: Preprint. MINOR revision. 2021. URL: https://eprint.iacr.org/2021/770.

[9] Sébastien Canard et al. "BlindIDS: Market-Compliant and Privacy-Friendly Intrusion Detection System over Encrypted Traffic". In: *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*. ASIA CCS '17. New York, NY, USA: Association for Computing Machinery, Apr. 2017, pp. 561–574. ISBN: 978-1-4503-4944-4. DOI: 10.1145/3052973.3053013. URL: https://doi.org/10.1145/3052973.3053013.

[10] Anrin Chakraborti, Giulia Fanti, and Michael K Reiter. "{Distance-Aware} Private Set Intersection". In: *32nd USENIX Security Symposium (USENIX Security 23)*. 2023, pp. 319–336.

[11] Hao Chen, Kim Laine, and Peter Rindal. "Fast Private Set Intersection from Homomorphic Encryption". en. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. Dallas Texas USA: ACM, Oct. 2017, pp. 1243–1255. ISBN: 978-1-4503-4946-8. DOI: 10.1145/3133956.3134061. URL: https://dl.acm.org/doi/10.1145/3133956.3134061.

[12] Hao Chen et al. "Labeled PSI from Fully Homomorphic Encryption with Malicious Security". en. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. Toronto Canada: ACM, Oct. 2018, pp. 1223–1237. ISBN: 978-1-4503-5693-0. DOI: 10.1145/3243734.3243836. URL: https://dl.acm.org/doi/10.1145/3243734.3243836.

[13] Benny Chor, Niv Gilboa, and Moni Naor. *Private Information Retrieval by Keywords*. Cryptology ePrint Archive, Paper 1998/003. 1998. URL: https://eprint.iacr.org/1998/003.

[14] Kelong Cong et al. *APSI*. URL: https://github.com/microsoft/APSI (visited on 06/15/2025).

[15] Kelong Cong et al. "Labeled PSI from Homomorphic Encryption with Reduced Computation and Communication". en. In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. Virtual Event Republic of Korea: ACM, Nov. 2021, pp. 1135–1150. ISBN: 978-1-4503-8454-4. DOI: 10.1145/3460120.3484760. URL: https://dl.acm.org/doi/10.1145/3460120.3484760.

[16] Nicolas Desmoulins et al. *Pattern Matching on Encrypted Streams*. Publication info: A major revision of an IACR publication in ASIACRYPT 2018. 2017. URL: https://eprint.iacr.org/2017/148.

[17] Eurojust and Europol. *Common Challenges in Cybercrime*. 2024. URL: https://www.europol.europa.eu/publications-events/publications/common-challenges-in-cybercrime (visited on 03/13/2025).

[18] Junfeng Fan and Frederik Vercauteren. "Somewhat practical fully homomorphic encryption". In: *Cryptology ePrint Archive* (2012). URL: https://eprint.iacr.org/2012/144.

[19] Tobias Fiebig et al. "Heads in the Clouds? Measuring Universities' Migration to Public Clouds: Implications for Privacy & Academic Freedom". In: *Proceedings on privacy enhancing technologies symposium*. Vol. 2023. Issue: 2. 2023, pp. 117–150. URL: https://doi.org/10.56553/popets-2023-0044.

[20] G. Fraser. *CrowdStrike: What was the impact of the global IT outage*. URL: https://www.bbc.com/news/articles/cr54m92ermgo (visited on 03/13/2025).

[21] Michael J. Freedman et al. "Keyword Search and Oblivious Pseudorandom Functions". In: *Lecture Notes in Computer Science*. ISSN: 0302-9743, 1611-3349. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 303–324. ISBN: 978-3-540-24573-5 978-3-540-30576-7. DOI: 10.1007/978-3-540-30576-7_17. URL: http://link.springer.com/10.1007/978-3-540-30576-7_17.

[22] Gayathri Garimella, Mike Rosulek, and Jaspal Singh. *Structure-Aware Private Set Intersection, With Applications to Fuzzy Matching*. Publication info: A minor revision of an IACR publication in CRYPTO 2022. 2022. URL: https://eprint.iacr.org/2022/1011.

[23] Chelsea Guan, Zekeriya Erkin, and Gosia Migut. "A Comparative Study of Threshold Multiparty Private Set Intersection Protocols". MA thesis. TU Delft, 2024.

[24] Mike Hamburg et al. *Why Ristretto?* URL: https://ristretto.group/why_ristretto.html (visited on 06/24/2025).

[25] Kyoohyung Han, Seongkwang Kim, and Yongha Son. "Private Computation on Common Fuzzy Records". In: *Proceedings on Privacy Enhancing Technologies* (2025). URL: https://doi.org/10.56553/popets-2025-0031.

[26] Yuval Ishai et al. "Extending Oblivious Transfers Efficiently". In: *Lecture Notes in Computer Science*. ISSN: 0302-9743, 1611-3349. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 145–161. ISBN: 978-3-540-40674-7 978-3-540-45146-4. DOI: 10.1007/978-3-540-45146-4_9. URL: http://link.springer.com/10.1007/978-3-540-45146-4_9.

[27] jedisct1. *libsodium*. URL: https://github.com/jedisct1/libsodium (visited on 06/14/2025).

[28] Lea Kissner and Dawn Song. "Privacy-Preserving Set Operations". In: *Lecture Notes in Computer Science*. ISSN: 0302-9743, 1611-3349. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 241–257. ISBN: 978-3-540-28114-6 978-3-540-31870-5. DOI: 10.1007/11535218_15. URL: http://link.springer.com/10.1007/11535218_15.

[29] Vladimir Kolesnikov et al. "Efficient Batched Oblivious PRF with Applications to Private Set Intersection". en. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. Vienna Austria: ACM, Oct. 2016, pp. 818–829. ISBN: 978-1-4503-4139-4. DOI: 10.1145/2976749.2978381. URL: https://dl.acm.org/doi/10.1145/2976749.2978381.

[30] 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 '17. New York, NY, USA: Association for Computing Machinery, Oct. 2017, pp. 1257–1272. ISBN: 978-1-4503-4946-8. DOI: 10.1145/3133956.3134065. URL: https://dl.acm.org/doi/10.1145/3133956.3134065.

[31] Majed Luay et al. *Temporal Analysis of NetFlow Datasets for Network Intrusion Detection Systems*. 2025. arXiv: 2503.04404 [cs.LG]. URL: https://arxiv.org/abs/2503.04404.

[32] Microsoft. *Microsoft SEAL*. URL: https://github.com/microsoft/SEAL/ (visited on 06/02/2025).

[33] Daniel Morales, Isaac Agudo, and Javier Lopez. “Private set intersection: A systematic literature review”. In: *Computer Science Review* 49 (Aug. 2023), p. 100567. ISSN: 1574-0137. DOI: 10.1016/j.cosrev.2023.100567. URL: https://www.sciencedirect.com/science/article/pii/S1574013723000345.

[34] Cyber Security News. *Building a SOC: Should You Go In-House or Outsource?* 2024. URL: https://www.linkedin.com/pulse/building-soc-should-you-go-in-house-outsource-cybersecurity-news-xealc (visited on 03/13/2025).

[35] OWASP. *OWASP Top Ten*. URL: https://owasp.org/www-project-top-ten/ (visited on 03/13/2025).

[36] Rasmus Pagh and Flemming Friche Rodler. “Cuckoo hashing”. In: *Journal of Algorithms* 51.2 (May 2004), pp. 122–144. ISSN: 0196-6774. DOI: 10.1016/j.jalgor.2003.12.002. URL: https://www.sciencedirect.com/science/article/pii/S0196677403001925.

[37] Benny Pinkas, Thomas Schneider, and Michael Zohner. *Faster Private Set Intersection based on OT Extension*. Publication info: Published elsewhere. Major revision. USENIX Security Symposium 2014. 2014. URL: https://eprint.iacr.org/2014/447.

[38] Benny Pinkas, Thomas Schneider, and Michael Zohner. “Scalable Private Set Intersection Based on OT Extension”. en. In: *ACM Transactions on Privacy and Security* 21.2 (May 2018), pp. 1–35. ISSN: 2471-2566, 2471-2574. DOI: 10.1145/3154794. URL: https://dl.acm.org/doi/10.1145/3154794.

[39] Benny Pinkas et al. *PSI from PaXoS: Fast, Malicious Private Set Intersection*. Publication info: Preprint. MINOR revision. 2020. URL: https://eprint.iacr.org/2020/193.

[40] Proofpoint. *What Is Alert Fatigue?* 2024. URL: https://www.proofpoint.com/us/threat-reference/alert-fatigue (visited on 03/13/2025).

[41] Srinivasan Raghuraman and Peter Rindal. “Blazing Fast PSI from Improved OKVS and Subfield VOLE”. In: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. Los Angeles CA USA: ACM, Nov. 2022, pp. 2505–2517. DOI: 10.1145/3548606.3560658. URL: https://dl.acm.org/doi/10.1145/3548606.3560658.

[42] Justine Sherry et al. “BlindBox: Deep Packet Inspection over Encrypted Traffic”. en. In: *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. London United Kingdom: ACM, Aug. 2015, pp. 213–226. ISBN: 978-1-4503-3542-3. DOI: 10.1145/2785956.2787502. URL: https://dl.acm.org/doi/10.1145/2785956.2787502.

[43] Snort. URL: https://www.snort.org/ (visited on 03/25/2025).

[44] Suricata. URL: https://suricata.io/ (visited on 03/25/2025).

[45] Benjamin Hong Meng Tan et al. “Efficient Private Comparison Queries Over Encrypted Databases Using Fully Homomorphic Encryption With Finite Fields”. In: *IEEE Transactions on Dependable and Secure Computing* 18.6 (Nov. 2021). Conference Name: IEEE Transactions on Dependable and Secure Computing, pp. 2861–2874. ISSN: 1941-0018. DOI: 10.1109/TDSC.2020.2967740. URL: https://ieeexplore.ieee.org/document/8962262/?arnumber=8962262.

[46] Erkam Uzun et al. “Fuzzy labeled private set intersection with applications to private {Real-Time} biometric search”. In: *30th USENIX Security Symposium (USENIX Security 21)*. 2021, pp. 911–928. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/uzun.

[47] H. Wiederhoeft. *SOC Strategy: When to Keep It In-house & When to Outsource*. 2024. URL: https://www.alertlogic.com/blog/in-house-or-outsourced-what-a-security-operations-center-means-to-your-organization-d54/ (visited on 03/13/2025).

[48] Tarun Yadav and Arvind Mallari Rao. “Technical Aspects of Cyber Kill Chain”. en. In: *Security in Computing and Communications*. Ed. by Jemal H. Abawajy et al. Cham: Springer International Publishing, 2015, pp. 438–452. ISBN: 978-3-319-22915-7. DOI: 10.1007/978-3-319-22915-7_40.

A

Execution Times and Communication Costs

Table A.1: Results for IIPSI, where $|X|$ is the set size of the Sender and $|Y|$ the set size of the Receiver, Comm stands for communication costs. The table shows the means and standard deviations of the computation times and communications, where each mean is based on 5 runs of the protocol.

Attr.	$\|Y\|$	$\|X\|$	Computation Time (s)			Total Comm (MB)
			Offline	Online	Total	
3	94	5k	35.81 ± 0.58	19.61 ± 0.11	55.42 ± 0.62	8.86 ± 0.21
	94	10k	37.13 ± 0.34	36.12 ± 0.16	73.25 ± 0.49	10.52 ± 0.26
	94	50k	53.84 ± 0.37	169.41 ± 2.49	223.25 ± 2.26	24.36 ± 0.33
	94	100k	78.12 ± 1.31	334.9 ± 3.2	413.03 ± 3.47	40.48 ± 0.0
	94	250k	194.5 ± 5.32	826.88 ± 0.57	1021.38 ± 5.4	87.25 ± 0.0
	94	500k	586.35 ± 26.47	1657.81 ± 5.97	2244.15 ± 30.29	174.48 ± 0.0
	94	1000k	2034.07 ± 28.82	3318.8 ± 7.06	5352.87 ± 25.93	349.33 ± 0.84
6	100	5k	36.91 ± 0.23	20.86 ± 0.16	57.76 ± 0.39	8.77 ± 0.0
	100	10k	40.12 ± 0.3	38.5 ± 0.29	78.63 ± 0.58	10.44 ± 0.21
	100	50k	67.4 ± 0.29	179.13 ± 0.32	246.53 ± 0.35	24.31 ± 0.39
	100	100k	104.95 ± 1.05	356.26 ± 0.62	461.22 ± 1.26	40.58 ± 0.0
	100	250k	265.97 ± 2.52	883.46 ± 3.05	1149.43 ± 2.16	87.51 ± 0.0
	100	500k	717.3 ± 11.62	1763.4 ± 3.12	2480.7 ± 12.17	165.71 ± 0.0
	100	1000k	2269.75 ± 116.45	3519.36 ± 9.21	5789.11 ± 114.72	322.31 ± 0.42
8	100	5k	37.72 ± 0.66	20.82 ± 0.36	58.54 ± 1.03	8.98 ± 0.42
	100	10k	41.67 ± 0.53	38.58 ± 0.89	80.25 ± 1.41	10.34 ± 0.0
	100	50k	76.72 ± 1.98	178.35 ± 0.85	255.07 ± 2.63	24.42 ± 0.47
	100	100k	122.91 ± 1.09	352.93 ± 0.64	475.84 ± 1.59	40.58 ± 0.0
	100	250k	308.4 ± 7.97	877.43 ± 3.92	1185.83 ± 11.82	87.51 ± 0.0
	100	500k	811.42 ± 18.48	1751.69 ± 7.81	2563.11 ± 12.16	165.71 ± 0.0
	100	1000k	2404.63 ± 58.75	3500.29 ± 12.4	5904.92 ± 53.53	322.31 ± 0.26

Table A.2: Results for ACPSI, where $|X|$ is the set size of the Sender and $|Y|$ the set size of the Receiver, Comm stands for communication costs. The table shows the means and standard deviations of the computation times and communications, where each mean is based on 5 runs of the protocol.

Attr.	$\|Y\|$	$\|X\|$		Computation Time (s)			Total Comm (MB)
		Original	Augment	Offline	Online	Total	
3	94	5k	21.6k	3.57 ± 0.57	2.89 ± 0.24	6.46 ± 0.79	12.77 ± 0.42
	94	10k	41.53k	7.19 ± 0.34	3.06 ± 0.21	10.25 ± 0.42	19.4 ± 0.39
	94	50k	181.69k	65.2 ± 3.02	4.38 ± 0.29	69.58 ± 3.06	73.01 ± 0.0
	94	100k	333.59k	194.65 ± 4.92	5.12 ± 0.42	199.77 ± 4.77	145.82 ± 0.0
	94	250k	716.99k	936.58 ± 55.1	6.63 ± 0.43	943.21 ± 55.33	382.72 ± 0.21
	94	500k	1242.21k	2786.7 ± 147.6	8.91 ± 0.38	2795.61 ± 147.82	804.62 ± 0.21
	94	1000k	2087.58k	7634.52 ± 722.06	12.54 ± 0.42	7647.05 ± 722.24	1693.53 ± 0.51
6	100	5k	191.1k	60.3 ± 4.02	4.28 ± 0.19	64.58 ± 4.19	28.01 ± 0.0
	100	10k	370.74k	229.21 ± 5.93	5.12 ± 0.24	234.32 ± 6.1	48.76 ± 0.0
	100	50k	1696.21k	5123.67 ± 334.03	10.76 ± 0.33	5134.43 ± 334.13	238.74 ± 0.42
	100	100k	3213.53k	13007.83 ± 818.53	18.48 ± 0.97	13026.31 ± 818.63	505.88 ± 0.0
	100	250k	7243.18k	-	-	-	-
	100	500k	12992.91k	-	-	-	-
	100	1000k	22643.83k	-	-	-	-
8	100	5k	778.01k	1076.76 ± 43.37	6.81 ± 0.35	1083.57 ± 43.64	48.96 ± 0.0
	100	10k	1510.81k	3874.03 ± 330.52	10.25 ± 0.5	3884.28 ± 330.52	93.39 ± 0.39
	100	50k	6973.13k	-	-	-	-
	100	100k	13301.95k	-	-	-	-
	100	250k	30403.48k	-	-	-	-
	100	500k	55299.55k	-	-	-	-
	100	1000k	97888.74k	-	-	-	-