Private Set Intersection: A Systematic Literature Review

Daniel Morales[b,*], Isaac Agudo[b], Javier Lopez[b]

[a]*Network, Information and Computer Security (NICS) Lab, University of Malaga, Spain*

Graphical Abstract

Private Set Intersection: A Systematic Literature Review

Daniel Morales, Isaac Agudo, Javier Lopez

[*] Corresponding author

Email addresses: damesca@uma.es (Daniel Morales), isaac@uma.es (Isaac Agudo), javierlopez@uma.es (Javier Lopez)

Highlights

Private Set Intersection: A Systematic Literature Review

Daniel Morales, Isaac Agudo, Javier Lopez

- We review Private Set Intersection, a very wide problem that can be solved with many different techniques.
- We identify Homomorphic Encryption, Oblivious Transfer, and Oblivious Pseudo-Random Functions as the main building blocks to solve Private Set Intersection.
- We survey the different solutions for Private Set Intersection from a practical point of view, with comparisons between complexity, running times, and usability.
- We identify open problems whose resolution may derive to more efficient, practical, and accepted Private Set Intersection protocols.

Private Set Intersection: A Systematic Literature Review

Daniel Morales[b,*], Isaac Agudo[b], Javier Lopez[b]

[b] *Network, Information and Computer Security (NICS) Lab, University of Malaga, Spain*

Abstract

Secure Multi-party Computation (SMPC) is a family of protocols which allow some parties to compute a function on their private inputs, obtaining the output at the end and nothing more. In this work, we focus on a particular SMPC problem named Private Set Intersection (PSI). The challenge in PSI is how two or more parties can compute the intersection of their private input sets, while the elements that are not in the intersection remain private. This problem has attracted the attention of many researchers because of its wide variety of applications, contributing to the proliferation of many different approaches. Despite that, current PSI protocols still require heavy cryptographic assumptions that may be unrealistic in some scenarios. In this paper, we perform a Systematic Literature Review of PSI solutions, with the objective of analyzing the main scenarios where PSI has been studied and giving the reader a general taxonomy of the problem together with a general understanding of the most common tools used to solve it. We also analyze the performance using different metrics, trying to determine if PSI is mature enough to be used in realistic scenarios, identifying the pros and cons of each protocol and the remaining open problems.

Keywords: Private Set Intersection, Secure Multiparty Computation, Privacy, Security

1. Introduction

Secure Multiparty Computation (SMPC) is a broad area of research in cryptography which has attracted the attention of many researchers. From the very first protocols proposed 40 years ago, this field is now evolving into a more practical one, with performance rates closer to some realistic use cases and a better acceptance from both the research and enterprise communities. In general, SMPC is a family of protocols which allows a set of distrustful parties

[*] Corresponding author

Email addresses: damesca@uma.es (Daniel Morales), isaac@uma.es (Isaac Agudo), javierlopez@uma.es (Javier Lopez)

$P = \{P_1, P_2, ..., P_N\}$ to perform a joint computation on their private inputs. The main properties that a SMPC protocol has to ensure are the inputs privacy for the whole computation and the correctness of the protocol, i.e., a subset of colluding parties cannot force the honest parties to output an incorrect result.

The first SMPC protocols [1, 2] focused on computing functions in a general manner, describing the functions as Boolean circuits and performing cryptographic operations on those circuits. However, some specific problems in the area have presented particular characteristics that have allowed researchers to solve them using different techniques. One of the main problems of this subset is Private Set Intersection (PSI). This protocol allows two or more parties to introduce their private sets as inputs and compute the intersection, but nothing else out of it can be inferred. PSI has attracted the attention of many researchers because of its wide usability on interesting use cases, which belong to different areas like medical analysis, anomaly detection, social networks, etc. Because of the huge interest it generates, this paper revolves around PSI, and tries to give a broad description, focusing on its practical characteristics. Like general SMPC, PSI has evolved from the first time it was introduced [3]. Performance has always been a central focus point, because SMPC in general and PSI in particular are known to be costly protocols. In spite of their irrefutable progress in terms of performance, these protocols are still heavy and have to be carefully instantiated, having in mind platform specific characteristics, like computation limits, bandwidth, or network delay. For that reason, this work tries to perform an empirical analysis on PSI and discuss if this protocol is ready from a practical point of view.

1.1. Contributions

We summarize in this section the main contributions of this work. First of all, we perform an analysis on PSI using a methodology known as Systematic Literature Review, which offers some sequenced guidelines that allow researchers to answer some specific questions. Also, we contribute with a survey which is self-explanatory, where the different building blocks used to achieve PSI are described in detail. Finally, PSI is reviewed from a practical point of view, analyzing specific costs and implications related to applications, leading to some open challenges discussion.

1.2. Organization

We will now briefly describe the organization of the paper. Section 2 introduces the methodology applied to perform the review and explains how papers are classified and data are extracted. Then, Section 3 analyzes different scenarios where PSI is applied and typical applications that are instantiated upon them, identifying some limitations presented by the actual technology. Section 4 introduces the basic technical concepts that are needed to understand the different PSI approaches, including basic building blocks and performance-improving techniques. Section 5 discusses the different solutions proposed by the reference literature to achieve different variants of PSI. Performance characteristics are the main focus point, which leads to a discussion in Section 6, where the predominant limitations are identified, together with some proposed techniques to increase the performing levels. The discussion is only made from the point of view of semi-honest security, however, Section 7 adds some additional details on malicious security. Finally, some conclusions and open challenges are remarked in Section 8.

2. Systematic Literature Review methodology

The objective of a Systematic Literature Review (SLR) is to analyze a topic and try to answer research questions, through a specific methodology that makes the process valid, reliable, and repeatable. There are different ways to develop reviews that depend on what the objective of the researchers is, e.g., a Systematic Literature Mapping, whose purpose is to present a wide amount of papers to get introduced to the research area. The SLR goes further, as it does not just give a list of interesting papers in the studied field but also analyzes them to answer specific questions.

In [4] the authors categorize the literature review approaches into 5 different categories: Describe, Test, Extend, Critique, and Hybrid Reviews. This work is settled between the two first, Describe and Test, because it tries to give the reader a list with some of the best papers in the scoped area and also answer some questions based on the data extracted from these papers.

A SLR is composed of 3 main phases: (1) planning the review, (2) conducting the review, and (3) reporting the review. Phase 1 is very important, as it makes possible the desired properties of validity, reliability, and repeatability. In this phase, the researchers formulate the problem and develop the sequences that they will follow to obtain the desired information. Phase 2 represents the wider one, where all the planned phases are carried out. The process of conducting the review is usually composed of several iterative phases where the reviewers drop papers from an initial list. The process starts just checking the title, then the abstract, and finally the whole body of the text. Each phase leaves the researcher with a smaller amount of papers until a final reference list is obtained, from where data will be extracted and analyzed. Finally, Phase 3 is the one where researchers summarize their findings and explain them, together with their corresponding answers to the initial questions. The following subsections describe each phase as well as the instantiation of the specific methodology for this paper.

2.1. Planning the review

To plan the review, the first point is to set the objective. This paper provides an analysis of PSI from a practical point of view. The conclusions will aim at resolving which approaches are realistic to develop an implementation which fulfills conditions in terms of efficiency, latency, time, memory occupancy, etc. In addition, an overview of typical applications and scenarios is given, which completes the analysis with the requirements for the PSI protocols.

To achieve the objectives, some sub-phases are defined and performed in a sequential manner. First of all, four research questions are specified, which are partially related between them:

- **(RQ1)** What are the most typical building blocks used in PSI protocols?
- **(RQ2)** What are the existing practical implementations of PSI?
- **(RQ3)** What are the existing use cases for PSI?
- **(RQ4)** What are the major challenges to achieve efficient practical implementations of PSI?

The main idea behind RQ1 is to identify the main building blocks that make it possible to achieve PSI protocols in a secure manner. This aspect is key to accurate classification and also allows the identification of performance limitations, which are usually highly dependent on the specific components used. With respect to RQ2, its main purpose is to gather the main practical implementations that exist on the PSI area. In general for cryptography and specifically for SMPC protocols, the step from a theoretical design to a practical implementation can be large sometimes, so identifying realistic implementations can be useful to target the most successful lines of research within PSI area. The purpose of RQ3 is somehow close to the one for RQ2, but from the use case point of view. At the end, the most widespread use cases can establish specific design requirements that constraints how PSI protocols are researched and implemented. Finally, RQ4 gathers the specific aspects covered in the previous research questions and establishes the focus on the challenges that make it difficult to achieve realistic and practical implementations of PSI. We aim at identifying the key lines of research where more effort has to be put in order to increase the widespread of PSI.

The research questions can be split into key terms, and using synonyms a search string can be developed that will cover the objectives and will be used as input on different digital libraries. The final search string is presented in Table 1.

Table 1: Search string

((“private” OR “secure”) AND “set” AND (“intersection” OR “computation” OR “operation”)) AND (((“protocol” OR “algorithm” OR “building block”) AND ((“implementation” OR “framework” OR “development”) OR (“efficient” OR “performance”))) OR “use case” OR “practical”)

The next step is to define the sources where the search string will be executed. The most relevant digital libraries in the area have been included: ACM Digital Library, IEEE Digital Library, ISI Web of Science, Science Direct, Scopus, and Springer Link. Finally, some exclusion criteria must be defined to have rules that allow selecting if a paper is suitable for this review or not (Table 2). The exclusion criteria aims at disregarding papers that do not contribute with novelty on the PSI area. Papers that do not present the full text available or are not written in English are directly excluded from the study. The same happens with works not published in conferences, journals or books. In addition, papers that briefly mention PSI, but are too generic or not focused on PSI are also excluded, because they do not contribute to analyze this protocol in depth. Also, to be more precise, we do not consider papers that do not offer a novel PSI protocol or a novel implementation using a PSI protocol. Finally, after a suitable list is accepted, duplicated papers are excluded.

Table 2: Exclusion criteria

Exclusion criteria
Full text not available
Not published in English
Not a conference/journal/book paper
Too generic
Not PSI-focused
Not a novel PSI protocol or PSI implementation
Duplicated papers

There are two additional planning steps that will be explained in Section 2.2 together with some findings: the design of a Quality Assessment Checklist and a Data Extraction Form.

2.2. Conducting the review

The first step of this phase is to search the literature using the search string presented in Section 2.1. This string may need to be adapted to the different Digital Libraries formats, but the core idea remains. Initially, a set of 610 papers is obtained from all the sources together. This set is reviewed up to three times from different points of view: the first time by title and keywords, the second one assessing the abstract, and the third one including Introduction and Conclusions. During the whole process, a snowballing method is carried out to add relevant papers that were not included in the query search. This process is called a *backward search* and selects interesting papers that have been cited by the reviewed articles.

At the end of this process, 215 papers are selected as potentially useful papers for the review. Figure 1 shows the number of papers initially obtained and then accepted by each source.

The next step is to perform a Quality Assessment Checklist. This step defines a list of features that a paper has to fulfill to be finally included. Each feature of the list adds a point to the score of the paper. Those that do not reach a minimum threshold will be excluded from the final reference list. In this work, we have defined a list with 5 features, where each of them can be

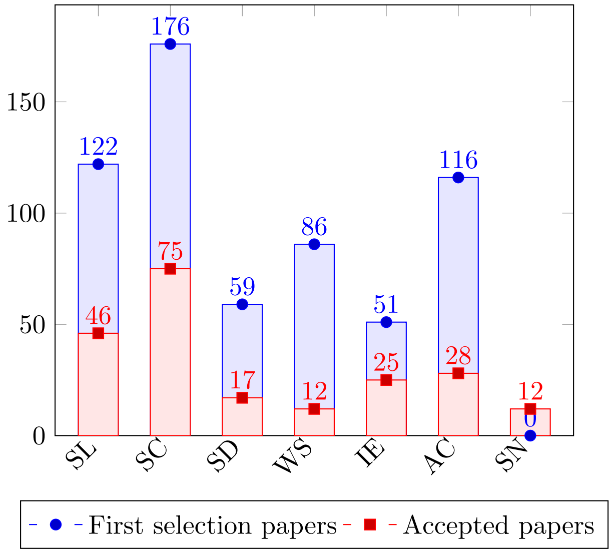


Figure 1: First selection and accepted papers. The digital libraries are the following: (SL) Springer Link, (SC) Scopus, (SD) Science Direct, (WS) ISI Web of Science, (IE) IEEE Digital Library, (AC) ACM Digital Library, (SN) Snowballing.

evaluated with Yes (1.0), Partially (0.5), or No (0.0). The maximum grade is 5 and the minimum required to be included in the final list is 2.5.

- Does the paper propose a new theoretical PSI protocol?
- Does the paper explain the building blocks used in the proposed solution?
- Does the paper propose a PSI practical implementation?
- Does the paper mention some use cases for the PSI paradigm or resolve one?
- Does the paper mention terms like efficiency, asymptotic complexity, or related considerations?

The idea of this checklist is to reward both the theoretical novelty and practical characteristics of the papers. The results obtained are specified in Figure 2. Significantly enough, most papers obtain a medium score. That is because they focus only on theoretical aspects or present only an implementation of one protocol, without focusing on PSI aspects.

Finally, after performing the Quality Assessment Checklist, we can analyze the number of papers obtained from each year (Figure 3). The trend is clearly upwards, as shown by the regression line, emphasizing that PSI has been acquiring more relevance through the years. Because the number of papers which have passed the Quality Assessment Checklist is huge (177), we have only considered papers from 2017 to 2022 (113). The reasons are: (1) to make the reference

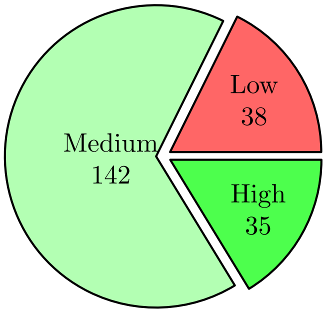


Figure 2: Quality Assessment List results. The score ranges are: Low [0-2.5), Medium [2.5-4), and High [4-5]. The papers scored as Low are not included in the final list.

list easier to handle and (2) to disregard older protocols, which usually perform slower than newer approaches. The final reference list of papers is shown in Table 3.

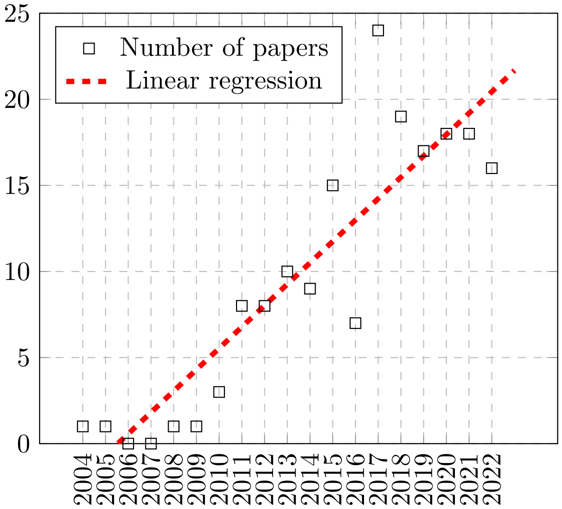


Figure 3: Number of papers selected per year and regression line

The last step of the second phase is to extract the relevant data desired to perform the review. This step is the most time-consuming and is performed iteratively. It is necessary to read with detail each paper and fulfill a form with relevant data like type of PSI, resource complexity, running times, etc.

2.3. Reporting the review

In this section, some statistics about findings discovered after classifying the reference papers will be reported.

The first interesting thing to analyze is the type of each paper, understanding *type* as the predominance of theoretical or practical characteristics. By "theoretical" we mean papers which focus on the PSI protocol itself, improving security or asymptotic costs. On the other hand, "practical" are those which apply PSI protocols on real scenarios, taking into account other aspects related to implementation, scalability, or usability. Of all the 113 papers, only 17 can be classified as practical, while the remaining 96 are purely theoretical (with the exception of 9 of them that mention a use case where their proposed protocol may fit). These findings allow us to conclude that the trend for PSI research has been mainly theoretical, while considerations related to practical aspects and infrastructure integration are not so common.

Another key aspect which is interesting to understand before any comparative is that PSI has different variants, i.e., different ways to define the inputs, the outputs, and how the parties interact between them, leading to specific considerations when designing and implementing protocols. While the basic and traditional primitive is just PSI, where two parties (or just one of them) get the intersection of their input sets, other protocols may compute a function on the intersection without revealing the intersection result. Others may need previous authorization for the elements in the set in order to be considered for the computation or may need a certain threshold for intersection cardinality to be reached in order to finally output the results. Although there are many variants, some of them have been identified and appear to be relevant:

- PSI cardinality (PSI-CA), where the output is not the intersection but the cardinality of the intersection. In this section we include its generalization circuit-PSI (C-PSI), where a generic function other than the cardinality can be computed on the intersection. There are 22 papers which belong to this category: 14 for PSI-CA and 8 for C-PSI.

- Multi-party PSI (MP-PSI), where there are more than two parties that want to compute the intersection of their input sets. There are 16 papers related to MP-PSI in our reference list.

- Outsourced or verifiable delegated PSI (O-PSI/VD-PSI), where the computation of the intersection is outsourced to an external server or cloud provider that is not supposed to learn anything about the private elements

Table 3: Reference list of papers

Year	**Papers**
2017	[5–28]
2018	[29–48]
2019	[49–66]
2020	[67–85]
2021	[86–103]
2022	[104–119]

or the result. There are 20 papers belonging to this group in our reference list.

3. Scenarios and applications

PSI may be classified based on which party obtains the intersection at the end of the computation with security warranties. This is traditionally named as *one-way* PSI, if only one of the parties obtains the intersection, or *mutual* PSI, if every party obtains it. This classification is very interesting from the security point of view, but there is another somewhat more interesting from a practical point of view, based on the parties set sizes. This classification distinguishes between *balanced* PSI, if set sizes are approximately similar, and *unbalanced* PSI, if there is a significant difference. These two scenarios have settled two focus points when designing PSI protocols, because special considerations can be made to improve performance when targeting one or the other. In fact, this classification even sets limits on the election of the underlying building blocks. We will now discuss very generally some aspects of these two scenarios, relating them to the different applications that have been proposed along the evolution of PSI and some technological considerations.

First, when considering the balanced scenario, one can imagine two o more entities that want to achieve the intersection of their databases or some statistics from it. By "entities" we refer to enterprises, researchers, medical or financial institutions, etc. While these examples are not strictly subject to the balanced scenario, it is a setting that appears frequently when, e.g., two financial entities want to compute common customers or patterns for their businesses. Some applications that have been proposed are secure computations on genomics [23, 35] or on medical data [15, 56], data mining on private data [65, 80], measuring ad conversion rates [26, 47, 76, 82], security incident information sharing [47], collaborative botnet detection [120], aviation safety [121], satellite collisions matching [122], private network interference discovery [115] etc. Looking for common characteristics, some assumptions can be ventured for this scenario. First, it is very likely that databases hold many items, because we are envisaging parties that work with a lot of information (customers, statistics, etc.). Secondly, we can assume a certain degree of flexibility when acquiring resources, i.e., computation capabilities like computers, servers, or infrastructure. Finally, this scenario can tolerate some amount of delay, e.g., a computation on medical private data can be set up to take a whole day, and this may be acceptable if it is previously considered by design, and the computation equipment is correctly prepared to handle it. One can think of two application instances in cloud environments where databases pour their data to perform a private data mining computation. Initially, the entities involved may consider the cost of the computation and determine if the investment benefits them.

The second scenario, the unbalanced one, can be easily compared with a classical approach that has been existing on the Internet from its origins, the client-server model. The server allows different clients accessing to different resources and is always assumed to have much higher computational capabilities than its clients. Turning this into PSI, we can assume that the client's set size will be much smaller than the server's. In fact, it may have just one item. This specific problem is known as Private Membership Test, which may be seen as a PSI subcase. Some applications that have been proposed to this scenario are malware detection by signature [16], checking of compromised credentials [16, 61], private database querying [29], biometric authentication [62], private contact discovery [47, 55], shoppers' personal preference matching [45], etc. The assumptions made on the balanced scenario can also be discussed here. First, the client's database size is much smaller than the server's. Secondly, the client's resources are limited compared with the server's, because in these applications, the client is supposed to carry a smartphone or a personal computer. In fact, the device could be even an IoT device, where computation, communication, and memory capabilities are very constrained. Finally, the service may not tolerate long delays. If it is a user handling the computation, e.g., with a smartphone, she will not want to wait a lot; otherwise, the quality of experience will be perceived as a bad one. A specific variation that has gained a great importance the last years is reversed unbalanced PSI, where client set size is larger than the server's, e.g., in COVID contact tracing [88, 100, 104, 113].

There are other applications where some devices (maybe limited in terms of resources) may want to perform PSI, but they depend on a central server to perform or coordinate the computation, e.g., private ridesharing [27, 48, 62], online matchmaking [32], social network discovery [30], online dating [62], etc.

The previous discussion has been made with a lot of generalizations, but when implementing a PSI protocol, there are many considerations to have in mind, e.g., the network environment or the computational limitations, and this is one of the main problems that limit PSI acceptance. Some protocols are efficient on communication but heavy on computation and viceversa. If one relies, e.g., on a cloud to avoid computational constraints, maybe the latency can be very high, and communications are limited, so the election of the protocol implementation has to be made according to this.

4. Notation and Preliminaries

This section introduces some basic aspects (schemes, data structures, etc.) which are required to fully understand how PSI is solved with the most extended techniques. Because of the large set of symbols used through the rest of the paper, it is highly recommended for the reader to check Table A.13.

4.1. Set Representation

Set Representation is a fundamental task that is typically made at the beginning of the PSI protocol. It basically sets a starting point for the tools that will make it possible to compute the intersection in a privacy-preserving way. It may also involve data compression, which is a desired property to reduce communication costs.

4.1.1. Polynomial Representation

Given a set $S = \{s_1, ..., s_n\}$, it can be represented as $P(x) = \prod_{i=1}^{n}(x - s_i)$, where each element s_i is a root of the polynomial. When two sets are represented as $P_A(x)$ and $P_B(x)$, the computation of $gcd(P_A, P_B)$ is the representation of $S_A \cap S_B$. For random polynomials R_A and R_B, $R_A P_A + R_B P_B = \mu \cdot gcd(P_A, P_B)$, where μ is a random polynomial. If the roots of the random polynomial can be deleted, the intersection can be computed.

When sets are encoded as polynomials, they are typically represented in point-value form. That means that a set of pairs $\{(x_1, y_1), ..., (x_d, y_d)\}$ are used as the polynomial representation, where $d \geq n$. If x is fixed, $y = (y_1, ..., y_d)$ is sufficient to represent $P(x)$.

4.1.2. Vector Representation

Given a set $S \subset \mathbb{U}$, where $\mathbb{U}$ is the universe set, one can define a vector of length $|\mathbb{U}|$. If the set has ordering property, each bit of the vector can be viewed as the representation of each $x \in \mathbb{U}$, where 0 means that $x \notin S$, and 1 means that $x \in S$. However, this representation technique has a straightforward limitation, which is that the vector length growths directly with $|\mathbb{U}|$.

4.1.3. Probabilistic Filter Representation

A probabilistic data structure allows the insertion of data and a check function. The typical constructions are Bloom Filter (BF) [123] and Cuckoo Filter (CF) [124], which are structures that allow a rate of false positives but not of false negatives.

A BF insertion maps an element x to k places into a bit array, setting those bits to 1. The positions are computed by a set of hash functions $\{h_1, ..., h_k\}$. To check if y is inserted inside the filter, the values $h_k(y)$ are computed, and if those vector bits are all set to 1, that means y is in the filter with high probability.

A CF insertion maps an element x to one from two possible buckets into a hash table, where there are B buckets and each can have more than one entry. To determine the two possible buckets, it uses Cuckoo Hashing, with the buckets being $b_1 = hash(x)$ and $b_2 = b_1 \oplus hash(fingerprint(x))$. Items can be relocated from one bucket to the other to make the table more efficient. Also, the value stored is the fingerprint of x, so the items can be deleted.

Both BF and CF are very common and work well along different schemes, e.g., given two BF containing two sets BF_X and BF_Y, $BF_{X \cap Y}$ represents the intersection in the plain model where $BF_{X \cap Y}[i] = BF_X[i] \wedge BF_Y[i]$. It is straightforward to combine this technique with homomorphic encryption (Section 4.2.2) in order to achieve the required privacy.

Table 4: BF and CF space costs extracted from [79].

Data Structure	**Bits/item**	
	Formula	$\alpha = 95.5\%$, $\epsilon = 10^{-3}$
BF	$1.44\ log_2(1/\epsilon)$	14.3
CF	$\lceil log_2(1/\epsilon) + 3 \rceil / \alpha$	13.5

Table 4 shows the formulas to compute the space costs of BF and CF with optimal parameters, assuming the load factor α is very high and the false error rate ϵ is relatively low.

4.2. Privacy Preserving Building Blocks

There is another thing that must be considered when classifying PSI papers: the basic building blocks that constitute the protocols, allowing privacy-preserving computations. The way that the protocols work varies considerably depending on the building blocks selected, and performance aspects as computation or communication complexity can also be modified. We have identified some building blocks which are broadly adopted, so they are introduced in this section.

4.2.1. Oblivious Transfer and OT-Extension

Oblivious Transfer (OT) was firstly introduced in [125, 126], where a sender owns two messages x_0 and x_1, and a receiver has a choice bit c. Then, the sender ignorantly sends x_c, and the receiver knows nothing about x_{1-c}. This protocol receives the name of "1-out-of-2 OT" or $\binom{2}{1}$-OT and was later generalized to $\binom{n}{1}$-OT, where the sender owns n messages, and $\binom{n}{k}$-OT, where the receiver selects k of them [127, 128]. OT typically involves some kind of public key cryptography to work, and this makes it a bit heavy for wide computations (Protocol 1 shows a simple implementation from [129]). It could be approximated as one exponentiation per OT instance; however, [130] shows that one exponentiation can be consumed to achieve some OT, using a trade-off which increases the communication cost.

Protocol 1 Simplest OT protocol

Inputs: the sender inputs two messages (M_0, M_1), and the receiver inputs a choice bit c.
Outputs: the sender outputs nothing, and the receiver outputs M_c.

Protocol:

1. The sender randomly samples $a \leftarrow \mathbb{Z}_p$ and sends $A = g^a$ to the receiver.
2. The receiver randomly samples $b \leftarrow \mathbb{Z}_p$ and computes:
   - $B = g^b$, if $c = 0$
   - $B = Ag^b$, if $c = 1$,

   then sends B to the sender and computes a symmetric key $K_R = H(A^b)$, where H is a hash function.
3. The sender computes the two possible symmetric keys:
   - $K_0 = H(B^a)$
   - $K_1 = H((B/A)^a)$,

   then sends the encryption of the two messages to the receiver:
   - $e_0 \leftarrow E_{K_0}(M_0)$
   - $e_1 \leftarrow E_{K_1}(M_1)$
4. Finally, the receiver can decrypt the selected message $M_c = D_{K_R}(e_c)$.

Nevertheless, a better performance can be achieved thanks to OT extension (OTe) [131], especially in terms of computational cost. First of all, a common notation is specified, where the execution of m OT with l-bit items is written as OT_l^m. An OTe scheme allows to compute OT_l^m from OT_κ^κ, where $m = \kappa^c$, κ is the OTe security parameter, and $c > 1$ is a fixed constant, plus some cheaper computations, i.e., calls to Pseudo-Random Generator (PRG) and Hash Functions. Table 5 shows some specific costs for two well-known protocols. Typically, $\kappa = 128$, which may be approximated as the number of exponentiations performed. The security parameter of [132] is twice as large as that of [131]; however, it can be amortized when performing OT with short secrets (small l), because it achieves $\binom{n}{1}$-OT directly (with $n \leq \kappa$), instead of $\binom{2}{1}$-OT.

Table 5: Computation and communication costs for two semi-honest OT-Extension protocols, where $PRG(m)$ generates m-bit items, and RO are calls to a Random Oracle (which is typically instantiated as a hash function).

Protocol	**Costs**		**Sec. Param**
[131] IKNP $\binom{2}{1}$-OT_l^m	Comp.	$OT_\kappa^\kappa + 2\kappa PRG(m) + 2mRO$	κ
	Comm.	$2m\kappa + 2ml$	
[132] KK $\binom{n}{1}$-OT_l^m	Comp.	$OT_{\kappa'}^{\kappa'} + 2\kappa' PRG(m) + mnRO$	$\kappa' \approx 2\kappa$
	Comm.	$2m\kappa' + nml$	

4.2.2. Homomorphic Encryption

A Homomorphic Encryption (HE) scheme is a tuple of algorithms {Key Generation, Encryption, Evaluation, Decryption}, where the Evaluation algorithm allows to evaluate a function f on ciphertexts as if they were plaintexts. This can be achieved thanks to the evaluation of simple operations, like addition or multiplication, on ciphertexts which are encrypted with the same key. More generally, $E_k(m_0) \star E_k(m_1) = E_k(m_0 \diamond m_1)$, where the first operation is performed on the ciphertext space and the second one on the plaintext space.

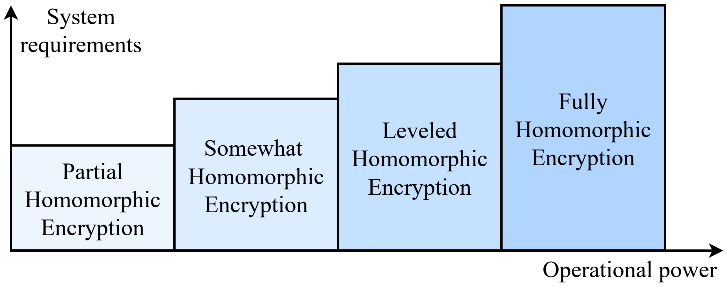

System requirements
Partial Homomorphic Encryption
Somewhat Homomorphic Encryption
Leveled Homomorphic Encryption
Fully Homomorphic Encryption
Operational power

Figure 4: HE schemes comparing the operational power of each scheme against the system requirements to execute it.

There exist different types of HE schemes [133] which offer different properties. Figure 4 shows a high-level diagram with the relations between them. First of all, Partial HE schemes only implement one operation, so the allowed computations are constrained to that, but they are the most efficient. These schemes are typically obtained from well-known public key schemes based on Factorization Problem or Discrete Logarithm Problem, e.g., RSA [134] and ElGamal [135] are Multiplicative HE, and Paillier [136] is Additive HE. A generalization about the allowed type of operations leads to Somewhat HE [137, 138], where both additions and multiplications are supported. These schemes are constrained by design, where the depth of the computable circuits is limited to some threshold. However, this can be solved with an external parameter d, which modifies the scheme on its initialization to allow a different circuit depth. The schemes which have this property are called Leveled HE [139] and represent a generalization of Somewhat HE. Finally, the most general category is Fully HE [140], which allows the different operations and has no restrictions on the circuit depth. This is achieved thanks to a computational expensive process called bootstrapping, which resets the noise added to the ciphertext after performing operations. Although Fully HE is known to be very costly, it has evolved a lot from its conceiving, leading to new constructions that may be practical for some realistic scenarios [141].

4.2.3. Oblivious Polynomial Evaluation

Oblivious Polynomial Evaluation (OPE) [142] involves two parties: a sender, whose input is a polynomial P, and a receiver, whose input is a value x. At the end of the protocol, the receiver learns $P(x)$ and the sender learns nothing.

This protocol can be instantiated with different primitives, like OT, Pseudo-Random Function (PRF), Secret Sharing..., but the most common is HE. The basic idea for PSI using OPE is that a user (let us call her Alice) represents her set as roots of a polynomial (Section 4.1.1) and builds $P(x)$. Then, the other user (let us call him Bob) obliviously evaluates his own set elements within the polynomial, e.g., using the homomorphic encrypted coefficients of the polynomial (sent by Alice).

4.2.4. Oblivious Pseudo-Random Function

A PRF is a deterministic function which receives a key K and an input x and which output is indistinguishable from a truly random function of the input x. In an Oblivious Pseudo-Random Function (OPRF) [143], the key K belongs to Alice and data x belong to Bob. Bob only obtains $F_K(x)$ and knows nothing about K, and Alice knows nothing about x. OPRF is typically instantiated using OT.

The basic idea for PSI using OPRF is that Alice gets its $PRF_K(A)$ using the key K of Bob and sends the result to Bob. Bob locally computes $PRF_K(B)$ and determines which elements are equal to those received from Alice.

4.2.5. Oblivious Key Value Store

An Oblivious Key Value Store (OKVS), formally introduced in [85, 95], is a generalization of data structures that allow mapping a set of keys k_i to a set of values v_i, e.g., a polynomial or a PRF. Informally, an OKVS is composed by two algorithms: (1) *Encode*, that takes the set of pairs (k_i, v_i) as input and outputs a data structure S, and (2) *Decode*, that evaluates a value k on S obtaining v as output. It is important to note that when S is evaluated on k_i (used to build S), it outputs the corresponding v_i.

4.2.6. Commutative Encryption

A Commutative Encryption (CE) algorithm is one where $E_{K_1}(E_{K_2}(m)) = E_{K_2}(E_{K_1}(m))$. It was originally proposed to solve the *Mental Poker* problem [144]. The basic idea for PSI using this primitive is that Alice can obtain $E_{BobPK}(A)$, being A her data set. First Alice sends $E_{AlicePK}(A)$ to Bob. Then Bob sends $E_{BobPK}(E_{AlicePK}(A)) = E_{AlicePK}(E_{BobPK}(A))$ to Alice. Then Alice decrypts it to obtain $E_{BobPK}(A)$. Bob sends Alice its own encrypted data set $E_{BobPK}(B)$, and Alice only needs to compare values to achieve the intersection.

4.2.7. Pairings

Let $\mathbb{G}_1, \mathbb{G}_2$ be two additive groups, and let $\mathbb{G}_T$ be a multiplicative group, all with prime order p. A Bilinear Pairing [145] is defined as a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ and satisfies some properties:

1. Bilinearity: For all $A, B \in \mathbb{G}_1$ and $C, D \in \mathbb{G}_2$, $e(A+B, C) = e(A, C)e(B, C)$, and $e(A, C + D) = e(A, C)e(A, D)$.
2. Non-degeneracy: There exist P, Q such that $e(P, Q) \neq 1$.

3. Computability: e can be efficiently computed.

Thanks to the bilinearity property, the Discrete Logarithm Problem in $\mathbb{G}_1$ can be efficiently reduced to $\mathbb{G}_T$. The security on those schemes is typically based on the bilinear Diffie-Hellman problem (BDHP), which is: given P, aP, bP, cP, compute $e(P, P)^{abc}$, and the implementations work on elliptic curves.

Pairings have been used for interesting problems, e.g., the one-round 3-party Diffie-Hellman problem [146] or constructions for Identity-based Encryption [147].

4.2.8. Generic Circuits

Generic Circuits are the most extended constructions to achieve SMPC protocols on generic functionalities and can be computed from different building blocks, e.g., garbling schemes or secret sharing.

A garbling scheme is a tuple of algorithms $G = (Gb, En, De, Ev, ev)$.

- $ev(f, \cdot) : \{0,1\}^n \rightarrow \{0,1\}^m$ evaluates the original function f with some input of length n and gives an output of length m.
- $(F, e, d) \leftarrow Gb(1^k, f)$ is the garbling algorithm, which uses a security parameter 1^k and the original function f to output a garbled function F, an encoding function e, and a decoding function d.
- $X = En(e, x)$ is the encoding algorithm, which inputs the encoding function e and the original input x and outputs the garbled input X.
- $Y = Ev(F, X)$ is the evaluation algorithm, which inputs the garbled function F and the garbled input X and outputs the garbled output Y.
- $y = De(d, Y)$ is the decoding algorithm, which inputs the decoding function d and the garbled output Y and outputs the final output y.

There are two main properties which must be satisfied for such a scheme. First, *privacy*, which allows an evaluator to obtain Y from X without being able to derive x. Second, *correctness*, where $De(d, Ev(F, En(e, x))) = ev(f, x)$, i.e., the garbling scheme must output the same value as directly evaluating the function without privacy. Garbling schemes are typically instantiated with Yao's Garbled Circuit protocol [148], where the original function is represented as a Boolean circuit in $\mathbb{F}_2$ before being garbled and evaluated.

Another way is to use secret sharing-based schemes [149], which allow to operate on distributed secrets. These schemes represent the original function as an arithmetic circuit, typically in $\mathbb{F}_p$. A secret-sharing scheme is usually instantiated as a (t, n)-threshold scheme, where the secret value is split among n participants and no $t - 1$ set of shares allows to reconstruct the secret.

While generic circuits allow to compute any desired function, they tend to be slower than protocol-specific constructions. However, they add the property of flexibility and offer many capabilities from a single tool, so selecting a generic or a specific construction is sometimes a discussion topic.

4.3. Hashing Techniques

Hashing has become a fundamental technique to achieve efficient PSI protocols. Its main purpose is to reduce the number of comparisons that have to be made between the set elements to achieve the intersection. In the plaintext model, without hashing and assuming two sets with n elements each, the cost of computing the intersection is $O(n^2)$ comparisons.

4.3.1. Hashing to bins

Hashing to bins employs a hash function which maps the set items from one domain to the other, $H : S \to M$, where $|M| < |S|$. Each hash address points to one bin with an associated maximum capacity $|B| \geq 1$, because two or more items may be mapped to the same bin. If every party uses the same hash function for the mapping, the same items will be mapped to the same bin, so comparisons can be made bin-to-bin. Thanks to this technique, instead of performing $O(n^2)$ secure comparisons to achieve PSI, the parties can reduce it to $O(B|B|^2)$, where $|B|$ is a pre-defined constant. According to [60], if $B = O(n)$, then with high probability $|B| = logB/loglogB$, and it would require $O(logB/loglogB)^2$ secure comparisons. If we recursively hash each bin into sub-bins, we can obtain a protocol with $O(BlogB)$ secure comparisons.

There exists a variant, named *Dual Hashing*, which employs two hash functions, H_1 and H_2. When one item needs to be inserted, it is mapped both to $H_1(x)$ and $H_2(x)$. Then, the item is inserted into the bin where there are less items already inserted. When using this technique, $|B| \geq 1$.

4.3.2. Cuckoo Hashing

Cuckoo Hashing employs two different hash functions, H_1 and H_2, associated with two tables, T_1 and T_2. To insert an element, it is first mapped to $H_1(x)$ and placed in that address of T_1 if it is free. Otherwise, it is also placed in that address, but the previous item is remapped to T_2 using $H_2(y)$. Using this technique, each element has two possible positions. If the number of replacements for one insertion exceeds a threshold, the item is inserted into the stash, which is a free address area. When using Cuckoo Hashing, $|B| = 1$.

4.3.3. Permutation-Based Hashing

This technique, also named Phasing [150], tries to reduce the bit length of the items that are stored into the bins. It first defines an item as $x = x_L || x_R$, where $|x_L| = logB$. Also, a random function f is defined with range $[0, B-1]$. The hash function H maps an item x to the bin with address $x_L \oplus f(x_R)$, and the value stored in that address is only x_R. As the stored value has a reduced bit length of only $|x| - logB$, there are no collisions.

4.4. Adversary Models

Adversary Models are crucial for PSI and, in general, SMPC protocols. They specify the adversary capabilities against which the protocol remains secure and have to be well specified in the designing phase. Traditionally, two main adversary models have been considered in the literature:

- **Passive or Semi-Honest[1] Adversary**, where the corrupted parties do not deviate from the protocol but may try to infer additional information from the data they obtain.
- **Active or Malicious Adversary**, where the corrupted parties may deviate from the protocol to get advantage.

In addition to that, the *Mixed Adversary*, which can corrupt some parties passively and others actively, must be considered. Achieving malicious security is not an easy task and has traditionally been put aside in favor of semi-honest security. In fact, malicious adversaries may harm the protocol in many ways, and not all possibilities are always covered, e.g., deviation allows the corrupted party to abort the protocol, aspect which is usually ignored. Other adversary models have been proposed to set up intermediate scenarios between the semi-honest and the malicious, e.g., the *Covert Adversary* [151].

5. PSI Taxonomy

PSI has many variants, which derive from a common root but offer different properties. The classical definition of PSI allows two parties to compute the intersection on their sets. A straight variant extends the problem to the multi-party setting, called MP-PSI, where the communication topology shows up as a key aspect. Other variants limit some aspects, like Threshold PSI (T-PSI), where the parties get the intersection only if it reaches a certain threshold, or PSI Cardinality (PSI-CA), where the size of the intersection is the only output. On Size-Hiding-PSI (SH-PSI), the cardinality of the sets is hidden, and on Authorized-PSI (A-PSI), there has to be a previous authorization phase (TTP signature) for the elements of the set. Finally, Outsourced-PSI (O-PSI) or Verifiable-Delegated-PSI (VD-PSI) break with the classical P2P topology and let the parties to outsource their encrypted sets to an external party (server or cloud) which is able to compute the intersection.

This section briefly describes different techniques to solve PSI problems, proposing a taxonomy for PSI. This taxonomy also considers the different cryptography building blocks used in existing protocols. Figure 5 shows how the different building blocks are related. With respect to basic primitives, public key is the most extended one. It is difficult to find PSI protocols that only require a generic public key scheme to work, i.e., traditional asymmetric encryption and digital signature. On the contrary, they usually require some extra capabilities. Some protocols rely on the existence of pairings and others on properties such as commutative encryption, but HE is undoubtedly the most typical building block. The use of other primitives is minimal; this is why we have included them in an “other” group that comprises symmetric cryptography techniques, non-standard primitives, and quantum-based protocols. Finally, there are some

[1] Sometimes named Honest-but-Curious.

papers based on more complex protocols like generic circuits (garbled circuits and secret sharing), OT, OPRF, and OKVS, which are the most extended building blocks after HE, and achieve the most efficient protocols.

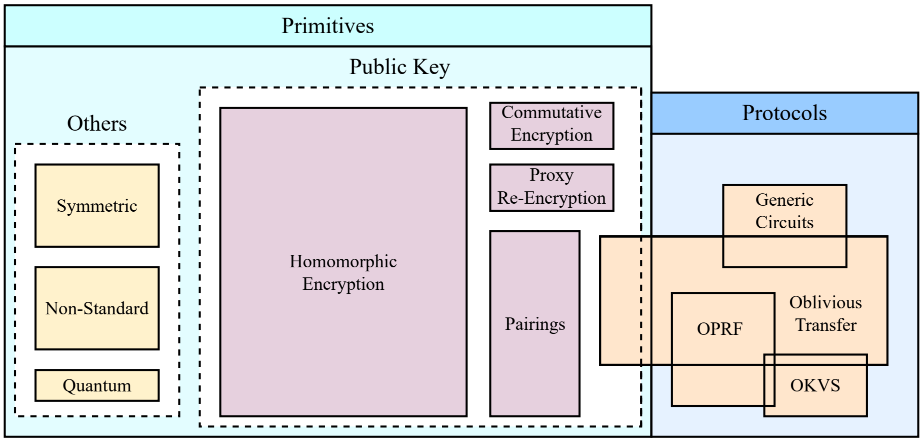


Figure 5: Building blocks classification. Each block size is proportional to the number of papers that belong to that category.

5.1. General PSI

General PSI may be described as the following:

1. **Input:**
   P_1 inputs $S_1 = \{a_1, a_2, ..., a_{n_1}\}$
   P_2 inputs $S_2 = \{b_1, b_2, ..., b_{n_2}\}$
2. **Output:**
   P_1 and/or P_2 output $S_1 \cap S_2$

Homomorphic Encryption: It constitutes one of the main building blocks used for PSI. This approach comes from the traditional OPE-based protocols, where sets are represented as polynomials whose coefficients are encrypted with HE. This allows the parties to evaluate their points homomorphically on the polynomials [20, 80]. The OPE-based protocol in [20] is a PSI with Projection and Histogram (PSI-P-H), where the client learns the projection (or second attribute in a table) of the items in the intersection and their histogram, but authors move to an OPRF to improve efficiency for some variants (Existential PSI [PSI-X], which indicates if the intersection is empty or not, and PSI with Data Transfer [PSI-DT]). As discussed in [28], OPE-based PSI protocols are computationally more expensive than those based on other primitives, e.g. [152]. Some protocols are based on Bloom filter to efficiently represent sets [7, 30]. After inserting the set elements in a Bloom filter, each bit is encrypted separately with an Additive HE scheme, so the computation cost will be $O(m_{BF})$. On the other hand, [79] applies cuckoo filter because it takes less bits to encode each item. This protocol also proposes a different representation after cuckoo filter encapsulation using matrixes and a generalized N-dimensional approach. Another way to achieve PSI is by encoding the elements into a bit vector [56], as described in Section 4.1.2. This protocol also applies Additive HE, and the computational cost grows in a similar way that in Bloom filter-based protocols but is $O(|\mathbb{U}|)$. For its part, [13] proposes a protocol for the malicious setting, achieving mutual PSI based on distributed ElGamal Encryption (i.e., a key is split between parties thanks to homomorphic properties), Verifiable Encryption, Semi-trusted Third Party, and Zero Knowledge Proofs (ZKP).

There are some protocols based on HE schemes with stronger properties [10, 42]. The first one, which is based on leveled HE, targets the unbalanced scenario and applies some techniques to improve the performance, like partitioning, batching, or windowing, together with preprocessing. It is focused on reducing communication costs and achieves $O(c\ log\ s)$. For its part, [42] is based on [10] and introduces polynomial representation and OPRF in the preprocessing phase. Also, the protocol is built for the malicious model, with added support for arbitrary length items, and achieves labeled PSI. On the other hand, [62] analyze the Threshold-PSI setting with the focus on communication complexity. It sets some bounds, which are $O(t)$ for fully HE and $O(t^2)$ for additive HE. However, [94] improves these results and obtains $\tilde{O}(t)$ for additive HE. For its part, [100] proposes a Conditional-PSI protocol, that outputs the intersection of the elements that fulfills a predicate. Finally, [117] is based on [10] and proposes a V-PSI protocol, thanks to an inner product computation on encrypted data.

Table 6: Rough computation costs for HE-based protocols on client side. C_{len} represents the bit length of a ciphertext.

Method	**Reference**	**Client Computation**	**Client Communication**
OPE	[20][80]	$n(Enc + Dec)$	$n\ C_{len}$
Bloom filter	[7][30]	$m_{BF}(Enc + Dec)$	$m_{BF}\ C_{len}$
Bit vector	[56]	$\lvert\mathbb{U}\rvert(Enc + Dec)$	$\lvert\mathbb{U}\rvert\ C_{len}$

Table 6 summarizes the costs on the client side for some of the previous protocols. It is clear that the size of the encoding data structure directly limits performance, in relation to the number of encryptions that the client must generate. An observation is that public key operations are the main bottleneck for the client device, so very big sets may be a problem. According to [153], a Bloom filter achieves its optimal performance when $k = (m/n)ln2$ and $m \geq nlog_2(e)log_2(1/\epsilon)$, so the number of bits (and the number of encryptions) will be higher than the number of elements it contains, with an incremental factor typically greater than 10. The same applies to bit vector representation, where it can be assumed that $|\mathbb{U}|$ will typically be much greater than n.

Oblivious Transfer: It is another fundamental building block for PSI. There are two classic approaches: Garbled Bloom Filter (GBF) and OPRF, but PSI is also achievable directly from OT. In the GBF approach [11, 63, 91], the client builds a Bloom filter and the server a GBF. Each bit i of the Bloom filter is used as a selection bit for a $\binom{2}{1}$-OT, where the server sends $GBF[i]$ if $BF[i] = 1$ and a random string otherwise. Both papers work in the malicious setting thanks to probabilistic verification techniques (Section 7). While [11] employs the cut-and-choose technique, [63] proposes a Traceable-PSI with bit commitment and proof of ownership. The computation cost for both papers (excluding precomputation and verification techniques) is similar to [153], which is $m_{BF}\binom{2}{1}$-OT for λ-bit strings. With respect to [91], it achieves T-PSI from GBF and threshold secret sharing for the semi-honest model.

The second approach is OPRF [12, 16, 47, 51, 60, 103, 118]. The work in [12] extends the paradigm in [152] (typically called PSZ) to the malicious setting. Both use an oblivious encoding technique, which is similar to an OPRF for a Set Inclusion protocol. Basically, P_1 learns a random mapping F, and P_2 learns $F(y)$. Then, P_1 sends $F(X)$ to P_2, where element inclusion can be checked. PSI is achieved by executing the Set Inclusion protocol n times and the malicious security thanks to dual execution. There are other protocols based on the PSZ paradigm, like [23, 27]. The first one achieves an approximation of the edit distance based on shingles and permutation-based hashing, while the second one achieves Threshold-PSI. Both [16] and [55] modify the OPRF-based NR-PSI protocol [154]. The modifications establish a precomputation phase (where OT are executed and later consumed) and send the PRF values using filters. Both protocols need to precompute lc_{max} OT, lc multiplications and c exponentiations on the client side. Specifically, [16] uses OT and Bloom filter, while [55] uses random OT and cuckoo filter. The protocol in [47] executes hashing-to-bins, instantiated with simple hashing on P_1 and cuckoo hashing on P_2, and then evaluates an OPRF instantiated with random OT. Other protocols based on [47] propose balance hashing [51] or cuckoo hashing [35] instead of simple hashing. For its part, [60] introduces some improvements that are focused on reducing communication. The proposed scheme instantiates a one-time OPRF using random OT, specifically the BaRK-OPRF scheme [155]. Another OPRF-based PSI is proposed in [118], with a structure-aware OPRF using function secret sharing in the semi-honest model. Other OT-based papers are [18, 75, 83]. The first one builds a $n \cdot n$ hamming distance matrix, where each cell with value 0 means that the big string is in the joint set; [75] employs an $\binom{n}{k}$-OT scheme to achieve PSI with equality tests; and [83] proposes a Java-based PSI framework to execute OT-based protocols.

A recent line of research is OKVS-based PSI, introduced in [85]. It achieves malicious PSI introducing a new linear solver named PaXoS (a binary OKVS) based on a Garbled Cuckoo Filter and combining it with OT. This protocol is almost as efficient as the semi-honest one of [152]. Following this line of research, [103] proposes an OPRF-based PSI, where the OPRF is built from random Vector Oblivious Linear Evaluation (VOLE) and PaXoS, improving the communication cost. On the other hand, [95] achieves a compromise between computation and communication, with a new OKVS based on 3-Hash Garbled Cuckoo Table. Finally, [105], which is based on [103], obtains the most efficient protocol to the date. More precisely, this paper proposes a more efficient OKVS instead the one from [85], and uses subfield-VOLE.

Generic Public Key: The protocol in [19] achieves Size Hiding-PSI based on smooth projective hash functions, which are very close to public key [156], while [46] studies the PSI problem in the multiple interaction environment and tries to mitigate the cross-interaction attack. Finally, [29] proposes an Authorized-PSI based on Schnorr signature, where the client items are signed by a certification authority.

Commutative Encryption: [5] achieves malicious PSI through commutative encryption and hash-based commitments. The computation of the intersection is practically the same as presented in Section 4.2.6. The only difference is the addition of test numbers and commitments which lead to verification phases achieving the desired security against malicious adversaries.

Pairings: An Authorized-PSI protocol is also proposed in [73] but built with Identity-based Encryption using pairings. Following this line, the protocol in [43] is based on Bilinear Accumulators to achieve a Reactive-PSI, where the output of the PSI protocol depends on previous instances. Pairing-based protocols tend to be costly due to the pairing operations. Table 7 shows the computation costs for some public key-based protocols.

Table 7: Computation costs for public key-based PSI protocols. **T:** set cardinality of test elements.

Protocol	**Computation Cost**	
	Asymptotic Cost	**Specific Cost**
CE [5]	$O(n_1 + n_2)\ Enc$	$2(n_1 + n_2 + 2T)\ Enc$
HE [13]	$O(c + s)\ Exp$	$(39c + 72s + 27)\ Exp + (ec + 4s)\ Inv+$ $+(c + 3s + 12)\ Hash$
PK [19]	Client: $O(c)\ Exp$ Server: $O(cs)\ Exp$	-
PK [46]	$O(c + s)\ Exp$	$(3c + 2s)\ Exp$
PK [29]	$O(c + s)\ Exp + O(c)\ Mult$	-
Pair [73]	$O(n)$	$2\ Mult + (3n + 1)\ Pair + (2n + 1)\ Exp$ $+(n + 1)\ Hash$
Pair [43]	$O(c^2/logc)$	$c\ Pair + O(c^2/logc)\ Exp$

Others: There are less common techniques employed to achieve PSI. The approach presented in [59] is based on the naive hashing technique and also proposes a protocol to check if the intersection is empty or not using blind exponentiation. The latter is also the foundation for [69], which is based on [157]. This protocol works with multi-sets and per-set tags and is applied on a system which allows journalists to share documents in a privacy-preserving way. On the other hand, [17] achieves PSI using matrix algebra in the mixed model (malicious client). This protocol is applied in [45] to the Food Adequacy Check use case. The method in [71] follows the oblivious polynomial evaluation approach, but instead of using HE, it blinds the polynomials with PRF outputs $z_i = PRF(key, i)$, represented in point-value form as $(x_i, P(x_i) + z_i)$, and [44] proposes a PSI protocol based on algebraic PRF, which may serve as a building block for both OPE and OPRF approaches. The work in [92] proposes a laconic (2-round) PSI protocol with O(s) communication. On the other hand, [108] proposes PSI based on functional encryption, for the reverse unbalanced setting, where the client's set cardinality is much larger than the server's. Finally, the PSI in [111] is based on greatest common divisor and dummy sets.

Applications: Some works implement PSI to achieve interesting applications, e.g., [31], which is based on Bloom filter like [158] to build a secure linkability protocol on Vehicular Ad-Hoc Networks, while [49] uses PSI to accomplish a proximity-based secure pairing using WiFi signals, where the authentication is correct if $|I|$ reaches a predefined threshold. Also, the users can derive a pairing key with the intersection material. The authors in [61] analyze protocols for checking compromised credentials where the protocol Google Password Checkup (GPC) is based on PSI. Finally, [48] proposes a privacy preserving ride-sharing system, which is based on [159], and the protocol in [92] can be used for self-detecting encryption.

5.2. PSI-CA and C-PSI

PSI-CA is a specific PSI protocol where the result of the computation is not the intersection set elements but their cardinality. More precisely, PSI-CA may be analyzed as a specific instantiation of C-PSI, where a generic function is computed on the intersection elements, and the result of that function is the only result learned by the parties. PSI-CA or other functionalities of C-PSI cannot be directly instantiated from general PSI when using specific purpose primitives. Instead of that, the protocol has to go through a different path from the beginning. A general description of C-PSI is formalized below:

1. **Input:**
   P_1 inputs $S_1 = \{a_{1,1}, a_{1,2}, ..., a_{1,n_1}\}$
   P_2 inputs $S_2 = \{a_{2,1}, a_{2,2}, ..., a_{2,n_2}\}$
2. **Output:**
   P_1 and/or P_2 output $f(S_1 \cap S_2)$ and nothing else.

Homomorphic Encryption: Like in general PSI, HE remains as a widely used building block. For instance, [21] proposes a mutual PSI-CA protocol for malicious parties where both parties get the output with fairness with the help of a semi-honest third party. It is based on Bloom filter, multiplicative HE, verifiable encryption, and ZKP. Like other PSI protocols, the set elements are encoded into a Bloom filter, and it is encrypted bit by bit. Each ciphertext is sent to the other party together with proofs of knowledge. This protocol heavily relies on public key cryptography, so it takes $22(\frac{kn_1}{ln\ 2}) + 83n_2 + 62\ Exp$

as computation cost. In [34], a verifiable-PSI-CA protocol is proposed with input certification for the Ad Conversion Rate use case, specifically applied on V2X-assisted Proximity Marketing. This protocol is based on multiplicative HE, ZKP, and bilinear pairings and is secure under the malicious model. The computation cost is $(23c + 13s + 14)$ Exp in G and 2 pairings on the client side. The works in [26, 76] achieve the variant PSI-Sum-CA, where one data set has an integer value associated with each element. At the end of the computation, the parties only learn the intersection cardinality and the sum of the associated integer values. The protocol in [26] is based on Diffie-Hellman-style double masking, and [76] adds two variants based on random OT and encrypted Bloom filter. However, the three protocols use additive HE as the base of their functionalities. The authors in [76] claim that the Diffie-Hellman-style protocol is the most efficient in both communication and computation. This protocol can handle data sets of size 2^{13} with a running time faster than 10 seconds. On the other hand, [32] achieves some Threshold-PSI protocols where the base is an encrypted PSI-CA protocol with output $Enc_{PK_1}(|C \cap S|)$ and the public key scheme is homomorphic. The protocol is based on Bloom filter and encrypted polynomials. The computation cost is $(k+2)n_1$ Enc for P_1 and $(log_2 e)kn_2$ Enc and n_1 Dec for P_2. With respect to [70], its protocols are based on leveled HE and compute differentially private PSI-CA. Finally, [119] proposes some protocols that allow one party learning a statistic on PSI (e.g., the mean of the intersection), while the other party learns PSI-CA.

Oblivious Transfer: In [8], an OT-based protocol for PSI-CA is proposed which use Flajolet-Martin sketches as a data structure encoder. A Flajolet-Martin sketch is a probabilistic counter of the number of distinct elements in a multiset. The authors propose a data oblivious algorithm to combine the sketches and extract the estimator and then give two protocols: (1) to achieve the intermediate result z, which is the index of the first 0 bit in the union sketch $F_{S_1 \cup S_2}$, and (2) to estimate the cardinality of the intersection. Both results are given to the parties in the form of secret shares.

Generic Public Key: The protocol in [77] is based on the Learning With Errors (LWE) problem to achieve post-quantum secure SH-A-PSI-CA. LWE is based on some parameters: the dimension $n \in \mathbb{N}$, the prime modulus q, the vector length $l = O(n\ log\ q)$, and a probability distribution $X(n)$ over $\mathbb{Z}_q$. Next, some values are specified: $\mathbf{a} \leftarrow_R \mathbb{Z}_q^n$, $\mathbf{s} \in \mathbb{Z}_q^n$, and $e \leftarrow X(n)$. The Decisional LWE problem sets that samples in the form $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e)$ are indistinguishable from uniformly random samples $(\mathbf{a}, b) \leftarrow_R \mathbb{Z}_q^n \times \mathbb{Z}_q$. This problem allows instantiating post-quantum public key cryptosystems [160]. It is combined with Bloom filter to achieve PSI-CA, where the client encapsulates data into the Bloom filter and encrypts it bit by bit before transmitting to the server. This protocol's computation cost is $ns+nl$ $Mult$ on the client side and $s(nl+l+1)+s(n+1)+ln+lt$ $Mult$ on the server side.

Commutative Encryption: The authors in [82] achieve PSI-CA in the unbalanced scenario, thanks to Bloom filter and commutative encryption. The receiver gets a shuffled copy of its own set encrypted in the form $Enc_{PK_S}(Enc_{PK_R}(Y))$, where it can be decrypted with the receiver's private key and get $Enc_{PK_S}(Y)$. Then, the sender just sends $Enc_{PK_S}(X)$ to the receiver, who can compare the number of coincidences. In this protocol, Bloom filters are used to increase the efficiency, and the execution takes $(n_1 + n_2)$ *Enc* on the sender side and n_2 *Enc* on the receiver side (with n_2 as the receiver's set cardinality), disregarding Bloom filter encapsulation costs. For its part, [109] proposes a PSI-CA based on commutative encryption, hash-prefix filer and cuckoo filter, for the reverse unbalanced setting.

C-PSI: Some papers [33, 36, 52, 99] propose C-PSI protocols using OT, Oblivious Programmable Pseudo Random Function (OPPRF), and generic circuits. These protocols are heavily influenced by the PSZ approach [150]. While [33] is based on OT, the output of PSI is given in an "encrypted" form where the result can be consumed by a garbled circuit, secret sharing, or HE schemes. They use an undirected graph construction to encode the set elements where it can be obliviously navigated, which allows basically a Private Set Membership [2] (PSM) instantiation. Papers [36] and [52] propose protocols which allows the computation of *symmetric functions* on the intersection result. Both perform a hashing step and a comparison step. The authors in [36] propose a novel hashing technique called 2D cuckoo hashing, while [52] implements the traditional PSZ hashing technique (P_1 uses cuckoo hashing, and P_2 uses simple hashing). Also, while [36] directly compares each bin using a generic Boolean circuit which can be extended to compute C-PSI, [52] differs in the usage of an OPPRF functionality to compare the bins, and the output results are later consumed by a generic Boolean circuit which computes C-PSI. On the other hand, [99] (which is based on [52]), uses an Oblivious Switching Network and Private Equality Tests, and achieves better communication (see Table 8). However, this protocol leaks the cardinality of the intersection. Finally, [87] proposes a mixed-protocol framework for generic 2PC that outperforms the PSI in [52] thanks to a new efficient equality testing protocol.

Conversely, [54] proposes some protocols for PSI-CA and C-PSI for a 3-party setting where the third party acts as a helper with no input but learns the results of the intersection cardinality. The protocols in [64] also compute PSI-CA and C-PSI, the second one applied both on the intersection payload and on the intersection indices. The PSI-CA is based on polynomial interpolation, and the C-PSI protocols rely on arithmetic circuits (with secret sharing). While the circuit-based protocol is linear in computation, the polynomial-based and hybrid ones are $O(n\ log^2\ n)$. However, the circuit-based protocol consumes more resources on communication and number of rounds. The work presented in [74] builds PSI with bi-oblivious data transfer using a PSM subprotocol based on a

[2] A PSM protocol is a specific case of PSI where the client only inputs one item and checks whether the server set contains it or not.

Table 8: Communication in MB for circuit-based f(PSI) on n elements of l bit length. **Arb.:** Arbitrary Bit Length; **SCS:** Sort, Compare, and Shuffle (PSI protocol fully built with generic circuits).

Protocol	l	**n**		
		$\mathbf{2^{12}}$	$\mathbf{2^{16}}$	$\mathbf{2^{20}}$
SCS [161]	32	104	2174	42976
	Arb.	205	4826	106144
[36]	32	51	612	6582
	Arb.	115	1751	25532
[52]	Both	9	149	2540
[99]	60	2.93	55.49	1030

Bloom filter-based batch one-time OPPRF protocol and hashing techniques. In this protocol, P_2 (with input set Y) outputs $f_i(b_i)$, where $b_i = 1$ if $y_i \in X$ or 0 otherwise, and $f_i(\cdot)$ is defined by P_1.

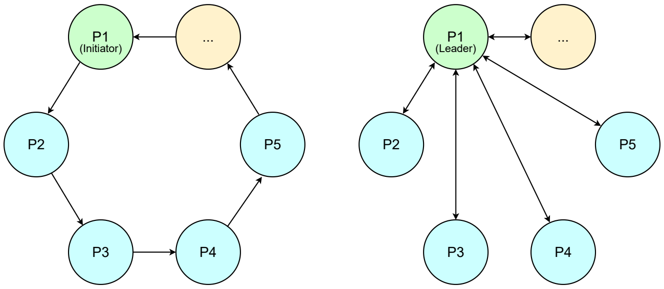
Applications: There exist some works apply C-PSI to different scenarios. First, [101] applies C-PSI to the use case of secure featurization, specifically applied to secure phishing detection. On the other hand, [104] applies PSI-CA to COVID contact tracing. The work in [65] applies O-PSI-CA on vertically partitioned data for data mining purposes. Finally, [84] proposes an additive HE-based protocol to compute PSI-emptiness for the Secure Inter-domain Forwarding Loop Test problem.

5.3. MP-PSI

Multi-party PSI is the natural extension of PSI to more than two parties. Although it has not originally received as much attention as the two-party case, it has become a trend the last years thanks to the performance improvements of 2-party PSI. The main limitation in an MP-environment tends to be the communication cost and the number of rounds the protocol takes. Increasing the number of parties in PSI, and more general in SMPC, entails an increment of resource consumption and may lead a protocol to a prohibitive cost that eliminates its feasibility for realistic scenarios. The next description summarizes the MP-PSI behavior:

1. **Input:**
   P_i inputs $S_i = \{a_{i,1}, a_{i,2}, ..., a_{i,n_1}\}$,
   where $i \in 1, ..., N$ and $N > 2$.
2. **Output:**
   One or all the parties get $\cap_i S_i$ and nothing else.

A key aspect in MP-PSI is how parties interact between each other. While generic SMPC typically assumes a fully-meshed network with equally amount of work for each party, for MP-PSI there are two specific topologies widely used (shown in Figure 6): (1) the sequential and balanced topology, where parties can be logically organized as a ring or a chain, and each of them performs approximately the same amount of computation; and (2) the star network topology, where one party assumes the role of leader, and every other party communicates through the leader, which typically carries a heavier amount of work. These specific topologies have contributed to achieve complexities that grow linearly with N, as shown in Figure 7 for some of the most efficient protocols.



(a) Ring network topology (b) Star network topology

Figure 6: Different topologies for MP-PSI

Of the selected papers, [78, 93] propose MP-PSO protocols based on vector set encoding and partial HE. However, [93] solutions assumes the external decider setting, i.e., the result is given to a party that does not have an input set. Like other vector-based protocols, each bit of the vector represents an element of the universe set $\mathbb{U}$. Also, each bit needs to be encrypted using HE, so it is desirable for $|\mathbb{U}|$ not to be very large. Both protocols work in a chain setup where P_i makes some process on data received by P_{i-1}. At the end, data published by P_N can be decrypted using threshold cryptosystem to recover the desired function (union, intersection...). While [78] requires $O(N\mathbb{U})$ encryptions and decryptions, [93] improves to $O(\mathbb{U})$. Somehow similar is [96], but instead of encrypting full-universe vectors, it encrypts BF. Another difference is that it works in the star network topology.

Another approach is to extend a 2-party PSI to a MP-PSI using the star network topology in an offline phase, where each pair (P_1, P_i) performs an independent 2-party protocol. Different works [38, 54, 90, 116] achieve this through executions of OT (offline phase) and GBF (online phase). The cost of these protocols is typically $O(N)$ times the cost of the base protocol. While [38] and [90] (based on [162]) achieve semi-honest security, [54] (based on [11]) and [116] (based on [12]) achieve malicious security thanks to cut-and-choose techniques. Partially related is [106], which proposes an MPC-PSI-Emptiness protocol using GBF but HE instead of OT. This protocol is applied to vertically partitioned data and achieves $O(Nn)$ computation and communication in the semi-honest model.

Another common approach for MP-PSI is the zero sharing technique [14, 58, 89, 102, 107], where the parties share some data that sum up to zero if one particular item is in the intersection. To achieve that, [14] performs a first phase for a conditional zero-sharing, where P_i obtains for x_j a share of zero ${s_j}^i$. In a second phase the parties perform a conditional reconstruction thanks to

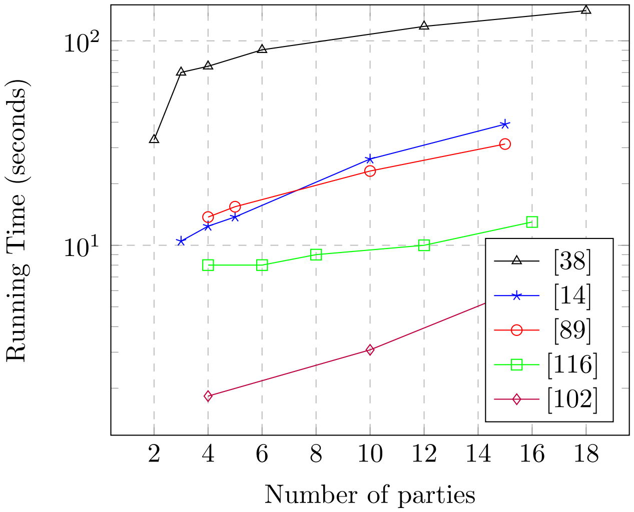


Figure 7: Running time of different MP-PSI protocols as a function of the number of parties involved N. We have restricted the figure to set size $n = 2^{16}$, threshold corruption $t = 1$, and network WAN setting, as representative values. Note that higher values for t lead to slower protocols.

OPPRF evaluated on x_j and programmed to output the shares ${s_j}^i$. If every party programmed the OPPRF with x_j, the sum outputs 0, i.e., x_j is in the intersection. The work in [89] improves [14] in terms of communication, and proposes variants for circuit-PSI and quorum-PSI. On the other hand, [107] also improves [14], but specifically in the small set setting. The work in [102] follows the same technique but is based on the efficient OKVS from [85]. Finally, [58] applies the zero sharing technique on vertically partitioned data, in order to achieve a random forest between a client and some servers.

There exist other papers that achieve MP-PSI from not so widespread techniques, e.g., [9] proposes an MP-PSI-CA based on commutative encryption and random permutations, in the star network topology. On the other hand, [22] presents two protocols, one secure against semi-honest adversaries and the other maliciously secure, both in the star network topology, based on threshold additive HE and balance hashing. Finally, [94] extends a 2P HE-based T-PSI with $O(t)$ communication to the MP setting, with $O(Nt)$ communication.

5.4. O-PSI

O-PSI is an interesting variant which has a main focus point: delegate the PSI computation to an outsourced party, in a privacy preserving manner. This type of PSI seems to be promising, because clients do not need to handle heavy computations on their devices and PSI can be offered as a service in a cloud environment. However, security and communication aspects are still weak points for this variant. A general description of the O-PSI functionality is given below:

1. **Outsource data sets:** each client C_i sends $E(S_i)$ to the cloud.

2. **Authorize the cloud:** clients send specific data $Auth_i$ to the cloud which allows it to compute the intersection on authorized data sets.
3. **Compute the set intersection:** cloud computes $f(E(S_i), Auth_i, E(S_j), Auth_j) = E(S_i \cap S_j)$.
4. **Decrypt and verify the intersection:** clients computes $S_i \cap S_j = Dec(Enc(S_i \cap S_j))$ and verify it.

Homomorphic Encryption: It is, once again, the main building block. In [53], two different constructions are proposed. One of them, named O-PSI, was previously presented in [163]. This protocol has been taken as a reference for many subsequent protocols. It is essentially based on polynomial representation and additive HE. Each client represents its set in point-value form, blinds the points using a PRF and then outsources them to the cloud. The clients communicate to compute some delegation data and send them to the cloud, which can perform the intersection thanks to additive HE. Finally, the cloud sends a vector to client B. The vector is then decrypted, the blinding factors are deleted, and the resultant polynomial is interpolated so the client can obtain the intersection from it. It takes $(2c+1)Exp$ for the authorizer and $2(2c+1)Exp + Fact$, where $Fact$ is $O(c^2)$, for the receiver. Following this direction, [65] achieves OPE-based O-PSI-CA. The scheme proposed in [40] is based on the O-PSI from [53] and it is very similar but uses multiplicative HE instantiated with RSA. It takes $(2n+1)(Enc + Mult)$ for client A and $(2n+1)(Enc + Dec + Mult)$ for client B. The main difference from [53] is that client B encrypts the random values and directly sends them to the cloud rather than to client A. Also, client A saves $(2n+1)Mult$ operations.

The works in [15, 30, 110, 114] achieve O-PSI using Bloom filter encapsulation and bit-by-bit encryption with additive HE. While [30] works in the 2-party setting, [15, 114] achieves an O-MP-PSI scheme and [110] achieves O-MP-PSI-CA for independent set sizes, where the computational cost for each party is independent of the number of parties. In [15], all the parties have to perform a threshold decryption process and then check if each item from their sets is included in the resultant Bloom filter which contains the intersection. Computation complexity for each client is $O(c_i)$, but communication is $O(c+N)$ because of the threshold decryption phase. Conversely, [30] works in the client-server scenario and proposes to outsource the encryption and decryption processes to the cloud. However, this entails a higher communication cost, more network delay, and even security issues. The work in [66] also employs an *encryption of vector's bits* fashion, but using full universe vector encoding, which is not desirable for large universe sets.

On the other hand, [25] targets the problem of Verifiable-Delegated-PSI. Based on the O-PSI from [53], it considers a cloud which can be malicious. This protocol proposes a verification mechanism that allows a client to verify the correctness of the result without having access to his own outsourced data set. To achieve this, a secret value β is agreed for both clients and then sent encoded to the cloud. The latter has to include this value into the intersection in a way that, when the client unblinds the result, β has to be correct, otherwise the client gets a random set, and it is supposed the cloud misbehaved. This protocol takes $(6n + 9)Exp$ and $(4n + 6)Exp$ on clients A and B respectively, but verification is only $O(|I|)$.

For its part, [41] allows outsourcing and verification too. The authors analyze the need for secure channels between clients in [25]. Therefore, they propose a protocol that does not require secure channels between clients, who do not even need to negotiate with each other in advance. Also, clients' costs are balanced with respect to other protocols. It takes $(4n+6)Enc$ and $(2n+3)(Enc+Dec)$ on both clients respectively. On the other hand, [67] is also based on polynomial representation, additive HE, and PRF. It is extensible to the Multi-party setting and can handle normal channels. Encrypted data outsourced to the cloud can be used directly by clients without additional computations. The authors call this solution a *natural secure data storage*. This protocol takes $(2c+1)Mult+Dec+2(2c+1)Exp$ on client A and $(2c+1)Enc+(2c+1)Dec$ on client B. Finally, [97] can be viewed as HE, but it is built upon secret sharing for a multi-owner database model. Despite the computation cost is lower, it needs a setup with some non-colluding servers.

Generic Public Key: Both [6] and [39] are the same protocol, based on public key encryption, where the first one encrypts using ElGamal and the second one RSA. Basically, clients upload their data sets in an encrypted form, where each sent element is $(Enc(d_i), g^{d_i})$, and the clients have to authorize the cloud to compute the intersection. Basically, the cloud can compare exponential obfuscated elements and then send to each client the encrypted intersection and a witness to verify it. The computation of the intersection takes $(n_1 + n_2)Exp$.

Table 9: Computation costs of some algorithms in [68]. $|L_t|$ is the cardinality of the leaf node set of the access tree; n_1 and n_2 are the set cardinalities for the Data Owner and Data User respectively.

Algorithm	**Cost**
Blind (Owner)	$(1+2|L_T|)\ Exp_{G_1} + (n_1+1)\ Exp_{G_2} + n_1\ Hash + |L_T|\ Op_{G_1}$
Token Generation (User)	$5\ Exp_{G_1} + Exp_{G_2} + |Att|\ Op_{G_1}$
PSI (User)	$n_2\ (Exp_{G_2} + Hash) + Op_{G_2} + Intersection(n_1, n_2)$

Pairings: The work in [81] is also verifiable but based on pairings. The clients outsource their encrypted data sets with an associated tag. When the cloud is authorized to compute the intersection between two data sets, the users generate re-encryption keys, which are used to transform similar data encrypted with different keys into common intermediate data. The cloud can then compute the intersection and send it to the clients together with verification witnesses. However, the whole process is resource intensive: both the encryption and set intersection processes need $O(n)(Exp + Pair + Mult)$. For a set of around 2^{12} items, the encryption algorithm takes more than 20,000 seconds and the set intersection process more than 15,000 seconds, but these results are prohibitive for most applications. The authors in [68] propose an Attribute-Based PSI protocol with fine-grained access control, also based on pairings, public key, and access tree. The protocol allows some Data Owners to outsource their sets in an encrypted form to a Cloud, thanks to an access tree. Then, registered Data Users who obtain delegated keys from a CA can perform a PSI computation against the Cloud. Basically, they send some tokens to the Cloud, and the Cloud computes some tokens for the users, who can locally compute the intersection. Table 9 shows the users computation cost for some steps, which may be critical, as they are supposed to set a low threshold due to resource constraints. Finally, [98] proposes an A-O-PSI protocol.

Others: Finally, there are other approaches based on symmetric key techniques, like [24], that presents a two-server aided PSI protocol which combines multiple keys with symmetric key proxy re-encryption, together with a reputation system to prevent colussion. Roughly speaking, a proxy re-encryption scheme allows Alice to give Bob decryption rights on her ciphertext, thanks to a proxy that re-encrypts the ciphertext but cannot recover the original plaintext. In [57] Bloom filter, deterministic encryption, and Pseudo-Random Permutation (PRP) are employed. Basically, the clients encode their sets into a Bloom filter, which is then encrypted bit-by-bit, and the corresponding ciphertexts are permuted according to a common PRP for both clients. The server tests which ciphertexts are the same on both parties and sends their indices back to the clients so they can use them to get the Bloom filter intersection. This protocol is semi-honest, but a malicious server variant is proposed using disjoint dummy sets which are included into the intersection. For its part, [72] proposes a modification to the EO-PSI scheme from [53], which is secure against passive attacks, while it does not need any secure channels. This protocol presents more balanced computation costs for the clients. Finally, [112] proposes a protocol for O-MP-PSI where the clients' sets are updatable. This protocol is based on PRF, hash tables and BF.

6. Performance: a Semi-Honest perspective

As we have seen in the previous sections, PSI has been approached from different perspectives, using different primitives. That diversity makes it difficult to embed a given protocol directly in real applications. There are variants designed for very specific scenarios, while others are designed from a general perspective. Selecting the right protocol for a given scenario and setting is a complex challenge for developers trying to put PSI intro practice.

When selecting a PSI variant, there are different parameters that have to be taken into account. First, the restrictions on the resources available in the devices and the infrastructure involved, including computation, communication and memory. Secondly, the ability of the protocol to be optimized, including aspects such as pre-processing or parallelization. Finally, the way the parties interact in the protocol and the application specific requirements. Different trade-offs and running time constrains may be considered depending on the use case. In this section, some of these aspects are covered, from a semi-honest PSI point of view.

6.1. Default performance

Through the whole paper, performance has been a central point of analysis. The default performance can be analyzed directly from the costs, which depend on the building blocks and the protocol design itself. In general, computation and communication costs are well analyzed on most of the papers, but the situation is not the same for memory, which is typically not well addressed.

6.1.1. Computation

PSI protocols tend to be heavy on computation, and that is because they are mainly based on public key schemes.

Fully HE schemes are well known as heavy on computation, because of plaintext encoding, encryption, and circuit evaluation. However, this technology is growing, and specific implementations can achieve better results; e.g., using leveled HE [10, 42] avoids the need for bootstrapping. However, the cost is still significant, and many applications combine it with high-performance setups [164], like multi-threaded CPUs or GPUs, to achieve better running times. Some works have proposed solutions for lightweight scenarios, but they do not achieve optimal results yet [165], especially for applications like PSI, where the number of encryptions may be noteworthy.

On the other hand, partial HE schemes achieve better performance, closer to classical public key schemes, but at the expense of the number of operations supported and the communication level (section 6.1.2). However, public key cryptography schemes are still heavy for constrained scenarios if the number of operations is large; e.g., [166] achieves competitive running times at the expense of key lengths.

According to [167, 168], the computation cost of one pairing can be established in between 4 and 8 exponentiations in G_1. However, pairing-based PSI protocols are typically complex constructions, with additional functionalities compared to general PSI, e.g., Attribute-based PSI (AB-PSI) [68], Authorized-PSI [73], or Tag-based VD-PSI [81]. That means that the computational cost may be increased, but with the profit of useful functionalities.

With respect to commutative encryption schemes, a similar limitation appears, where constructions like SRA [144] and Pohlig-Hellman [169] are based on integer factorization and discrete logarithm problem. Exponentiations are the main bottleneck, and they may be unacceptable for lightweight devices if the number to perform is large.

On the other hand, there are constructions much more efficient on computation, mainly based on symmetric key cryptography. Despite base OT is built with public key cryptography, thanks to OTe, as described in Section 4.2.1, many OT can be achieved mainly with symmetric key techniques.

Also, OPRF , OKVS, and VOLE constructions meet these requirements (e.g., [85] is built on OTe). In fact, they are known as the most efficient PSI protocols to the date [60, 85, 103, 105, 170]. They work well for the unbalanced setting, as they only need $O(c)$ OPRF invocations (or OKVS encodings), but at the expense of $O(s)$ communication. Typical constructions are instantiated with a previous hashing phase (cuckoo and simple hash) on both parties and later B independent OPRF executions. According to [170], it is said that the roughly cost to compute n OPRFs is the cost to compute $3.5n$ $\binom{2}{1}$-OT (with OT extension). On the other hand, the OPPRF variant is typically used in f(PSI) protocols, where some computations are executed on the intersection. These protocols share the same principle, but they may present some additional phases with other techniques, like additive HE [76] or general circuit computation [52], to achieve the additional functionalities.

Finally, there are some protocols directly based on symmetric ciphers, but they are very uncommon (because of the difficulty to achieve strong and secure functionalities). The same applies to constructions directly based on algebra techniques. These protocols typically rely on very specific designs and they are not normally flexible to fit on different applications, as it may happen with other building blocks which have been well studied and improved.

6.1.2. Communication

Communication presents a reverse scenario with respect to computation, and for that reason trade-offs are always needed when selecting the correct protocol for a specific application.

In general, public key-based protocols do not consume a lot of resources (typically sending a number of ciphertexts proportional to the set sizes), and communication rounds tend to be few. However, it depends on the security design, because large ciphertexts (e.g., one that uses a 4,096-bit modulus with 2^{20} elements would cost 512 MiB [3].), may be unacceptable depending on the setting. Fully HE encryption-based works typically focus on reducing communication [10, 42, 62], as it is their strong characteristic with respect to other building blocks. A theoretical analysis for communication bounds on Threshold-PSI is performed in [62] and it establishes the communication bounds as $O(t)$ for fully HE, $O(t^2)$ for partial HE, and $O(t^3)$ for garbled circuit approaches. These results highlight the need for a trade-off, where authors argue for partial HE being a better option than fully HE, because it does not consume such high computation resources even if it is a bit more costly in terms of communication.

On the other hand, protocols built on fast primitives like OT, OPRF, and OKVS are known to be more costly in terms of communication, e.g., with the Hash + OT setting for PSI, which is linear with the size of the larger set. That can be proven with the OPRF-based protocols [60, 103], which effectively sends $O(s)$ PRF values. However, improved extraction techniques [103] allow to reduce the sender's communication by half.

Table 10 presents some interesting values to compare communication costs of OTe protocols and OPRF-based PSI protocols. In the first place, with re-

[3] MiB and MB are both understood as 1,024 KB

Table 10: Communication costs for cheap computation protocols in MiB. For the OPRF costs, the values are set as 23kb/OPRF when instantiated with LowMC, $k = 2$ hash functions and $PRF_{val} = 128$ bits. The formula to compute the OPRF-based costs from [60]

is $c\,OPRF + ks\,PRF_{val}$, where $c = s$ in the balanced case.

Protocol	Param.	Set Size						
		$\mathbf{2^{10}}$	$\mathbf{2^{12}}$	$\mathbf{2^{14}}$	$\mathbf{2^{16}}$	$\mathbf{2^{18}}$	$\mathbf{2^{20}}$	$\mathbf{2^{24}}$
IKNP $\binom{2}{1}$-OT_{32}	$\kappa = 128$	0.039	0.156	0.625	2.5	10	40	-
KK $\binom{2}{1}$-OT_{32}	$\kappa = 256$	0.07	0.281	1.125	4.5	18	72	-
KK $\binom{100}{1}$-OT_{32}	$\kappa = 256$	0.453	1.813	7.25	29	116	464	-
GBF	$\lambda = 128$	0.0625	0.25	1	4	16	64	-
PSI (OPRF) [60] (balanced)	-	2.91	11.63	46.5	186	744	2976	-
PSI (OPRF) [60] (unbalanced)	$c = 2^{10}$	2.91	3	3.38	4.88	10.88	34.88	-
PSI (OPRF) [103]	$c = 2^{10}$	2.86	2.9	3.1	3.84	6.84	18.84	258.84
PSI (OPRF) [105]	$c = 2^{10}$	0.4	0.44	0.62	1.37	4.37	16.37	256.37

spect to OTe, we can observe that IKNP outperforms KK for standard length items (32 bits). It takes 40 MiB for 2^{20} items, which may be acceptable for some applications. On the other hand, garbled Bloom filter can be seen as $\binom{2}{1}$-$OT_{128}^{m_{BF}}$, where $\lambda = 128$ is the GBF security parameter (the length of the items it contains). For that reason, it is a bit more costly than the typical OT instantiation. For the OPRF setting, a rough calculation has been made, based on the communication scenario of [60], where, assuming a 100% of occupation on the client cuckoo filter, the protocol takes c times the communication cost of an OPRF and ks times the length of a PRF value. It is clear that the unbalanced scenario totally outperforms the balanced one, as the number of OPRF instantiations is directly related with the client set size. It is also noteworthy that the OKVS paradigm [103, 105] has allowed the most compact protocols in terms of communication.

Finally, generic circuit PSI has the problem of sending the whole circuit from one party to the other, as it happens, e.g., when using Yao's Garbled Circuit approaches. As it was shown in Table 8 (Section 5.2), new techniques which include hashing and OPRF outperforms General Circuit fully-based protocols in terms of communication.

6.1.3. Memory

With respect to memory, it is not typically a problem itself except for data structures that must be full loaded on memory. On public key schemes, processing and checking the ciphertexts can be made independently so the memory consumption may be well handled. On OT schemes, being interactive schemes, the exchanged data must be typically loaded on memory; e.g., IKNP works with $m \cdot \kappa$ matrixes, which would take 16 MiB for an $OT^{2^{20}}$. However, matrixes can be read line by line and loading from persistent memory, at the expense of running times. OPRF-based protocols are cheaper, as 4.19 MiB of client storage on memory are needed when instantiated with [55]. On the other hand, data structures which perform fast arbitrary checks, as it happens with Bloom and cuckoo filters, must be full loaded on memory; e.g., when modeled with error $\epsilon = 10^{-5}$ and load factor $\alpha = 95.5\%$, a Bloom filter takes 2.989 MiB and a cuckoo filter 2.617 MiB. While those values may be acceptable for general use devices, they may be a problem for many IoT devices if they handle the computation themselves. With respect to generic circuits PSI, memory is highly dependent on the circuit design. If many layers of gates cannot be split as separated processes, they all must be loaded on memory, which may grow to unmanageable values.

6.2. Performance improvements

As it has been analyzed, PSI protocols are heavy by definition, especially when the set sizes are large. For that reason, several performance improvement techniques have been proposed on many papers.

6.2.1. Parallelization

Parallelization is a very important technique that can be applied when the computations to perform are independent. Parallelization can be achieved from different perspectives, and many protocols benefit from it. A straightforward case is to run the protocol over a pool of threads. As previously mentioned, public key schemes typically perform independent encryptions over the data sets, which can be split through the threads [86]. However, the most promising parallelization technique for those settings seems to be the use of Single Instruction Multiple Data (SIMD) [171, 172], where one instruction can be executed on multiple data at the same time. Fully HE-based protocols are, in fact, the ones which most rely on those techniques [10, 42] to reduce the heavy computational cost they present. Other schemes, like OTe or OPRF, can also be executed as independent threads, while dynamic allocation structures like cuckoo filter do not allow it. In the case of generic circuit designs, the circuit evaluation phase does not fit well with parallelization because it tend to be very sequential. The main problematic issue of parallelization techniques is that they are platform-dependent and typically associated with devices that are not very constrained on resources.

6.2.2. Precomputation

On the other hand, precomputation is one of the most extended techniques on the SMPC paradigm, as it allows the parties to compute resources in advance of the inputs, considering only the online phase as the important running time to analyze. Precomputation is well integrated into OT-based protocols, as the base OT generates correlated data which are computed previously to the input data. However, public key schemes do not fit well with this solution, because the encryptions tend to be directly performed on the input data. With respect to OPRF schemes, it depends on how they are instantiated, while the general case, as previously analyzed, is to build them using OT. For generic circuits approaches, precomputation is straightforward and one of their strongest performance improvement techniques. While Boolean circuit-based schemes are based on OT, Arithmetic circuit-based schemes are based on Secret Sharing, which are also precomputable thanks to Beaver Multiplication Triples [173]. However, precomputation has a caveat, namely it still needs to be computed. For periodic or automatic executions, which may be well coordinated, it has sense, but if the protocol relies on user interaction, the precomputation process still needs to wait for some kind of input to be executed, translating into waiting times for the user. Another interesting but not much covered aspect (because of the security losses) is parameter reuse. If that characteristic is allowed in a PSI protocol, it may lead to a combination with precomputation, which would translate into faster protocols.

6.2.3. Outsourcing

Finally, outsourcing is another technique whose main purpose is to reduce the cost of the operations performed on the client side. Those protocols, as analyzed in Section 5.4, outsource the data in an encrypted way to a high-performance server or cloud, where heavy computations are carried on to compute the intersection. This approach is very straightforward with HE-based protocols, and it has been applied too on generic public key and pairing-based protocols; however, this is not the case with OT-based protocols, where the computation parties interact directly between them. Outsourcing presents several problems: (1) data sets need to be moved from the local setting to an external setting, which may be very slow if the available bandwidth is not high enough or the data set is large; (2) they are typically related to verification processes, which may be costly, slowing down the computation; (3) the designs to the date do not significantly reduce the computation costs on the client side, as can be observed in Table 11. While the whole process is costly for the client, O-PSI may be thought as split phases. If parameter generation and setup can be made in advance, then the clients may be able to encrypt their data sets when they are updated to save that computation to a future outsourcing and intersection computation. Also, how to dynamically adapt the encrypted set when new items are included into the client set without re-encrypting and outsourcing every item is an interesting open problem.

6.2.4. Multi-party setting

In SMPC protocols, the number of parties (N) which are involved into the protocol is a key parameter. When N is increased, the protocol running time tend to increase exponentially. This is overall a communication problem, because the number of interactions between parties grows very high, especially if the topology is a full mesh. For that reason, the topology is a key aspect that has influence on performance. In section 5.3, it has been shown for MP-PSI that two main topologies have been proposed: the star network and the ring network (Figure 6). Clearly, these two approaches lead to less communication rounds thanks to the topologies themselves, however with some trade-off. The star network topology has the problem of establishing a leader who will consume much more resources than the others, typically $O(Nn)$ for the leader and $O(n)$ for the clients. For that reason, that topology may only have sense for some specific

Table 11: O-PSI Client-side Computation Costs. The presented results show all the costs which concern the clients added together.

ID	**BB**	**Client**	**Computation Cost**
[67]	HE	A	$(2c+1)Mult + Dec+$ $2(2c+1)Exp$
		B	$(2c+1)Enc + (2c+1)Dec$
[53] O-PSI	HE	Auth	$(2c+1)Exp$
		Recv	$(2c+1)Enc+$ $(2c+1)Dec + Fact$
[72]	Symm	A	$Bnd\ Add + Bn(d+1)Mult$
		B	$Bn\ Add + Bn\ Mult+$ $B(Int + Fact)$
[81] TVDPSI	Pair.	A	$(2n+5)\ Exp+$ $(2(n+\|I\|)+3)\ Pair+$ $(n+\|I\|)\ Mult$
[39]	PK	A	$(4n+2\|I\|+3)\ Exp$
[98] A-O-PSI	Pair.	A	$8c\ Exp + 4c\ Pair+$ $O(1)$ or $O(c)\ Enc$

scenarios. On the other hand, the ring network may not be always possible, due to protocol design or building blocks characteristics, in addition to being easily susceptible to DoS attacks (corrupting any party breaks the full execution). In general, achieving fast protocols on MP-PSI with high density of parties is a difficult task, but we identify a trend in mapping 2P-PSI to MP-PSI. It is clear that novel and more efficient 2P-PSI have contributed to improve running times in MP-PSI, as shown in Figure 7.

6.3. Running time and usability

An interesting thing to analyze is running time, because it gives direct information about the protocol performance in a specific (typically realistic) setting. Table 12 shows some numbers for different protocols and settings. At first sight, the difference between Local Area Network (LAN) and Wide Area Network (WAN) settings may be noticed. WAN instantiations (typically with 100 Mbps of bandwidth and 100 ms of Round Trip Time [RTT]) are clearly slower than LAN instantiations, still being much more common for most of the applications described on Section 3. Some protocols on Table 12 do not even consider any network constrain, so the specified times are expected to be slower than they are.

First, there exist a clear distinction between HE-based protocols and those from OPRF, OKVS approaches. For the first category, in order to achieve fast running times (slower than 10 seconds), they need to compute on small set sizes, with the exception of [10], which is surprisingly fast even when it is based on fully HE, works on WAN setting, and employs just one thread. Other protocols [7, 54] seem to be the most problematic protocols, maybe even for balanced applications where running time is not a direct constrain (not only for the waiting time, but also for the necessity of re-execution in case a problem occurs when the protocol is on computation). The second category [85, 105, 155]

Table 12: Running Time (in seconds) for different protocols and set sizes. If asymmetric set size is not specified, the protocol works in the balanced setting. Simulation equipment tends to be general purpose processors and main memory > 100 GB, except [53, 68, 74, 76, 83], where it is < 100 GB. **NNt:** no network delay; **NS:** not specified; ***x*Th:** number of threads; ***x*P:** number of parties; **(*):** only phase for computing the intersection.

ID	**Type**	**Setting**	**Asymm.**	**Set Size**							
			Set Size	$\mathbf{2^8}$	$\mathbf{2^{10}}$	$\mathbf{2^{12}}$	$\mathbf{2^{14}}$	$\mathbf{2^{16}}$	$\mathbf{2^{18}}$	$\mathbf{2^{20}}$	$\mathbf{2^{24}}$
[155]	(PSI)	LAN	-	0.19	-	0.21	-	0.39	-	3.78	58.57
	OT	WAN	-	0.56	-	0.59	-	1.26	-	7.46	106.83
[83]	PRTY20 (PSI)	LAN	-	0.54	-	1.22	-	11.40	-	215.00	-
	Malicious, OT	WAN	-	1.84	-	2.36	-	13.00	-	204.00	-
	CM20 (PSI)	LAN	-	0.81	-	1.13	-	8.08	-	180.90	-
	OPRF	WAN	-	2.12	-	2.11	-	9.49	-	186.00	-
[51]	(PSI)	LAN	2^{12}	-	-	-	-	0.80	-	2.96	33.94
	ROT, Hash	WAN		-	-	-	-	2.28	-	5.43	53.33
[7]	(PSI) HE, BF	NS	-	11.76	44.68	175.79	702.39	2834.68	11327.78	-	-
[10]	(PSI) FHE	1Th WAN	2^{12}	-	-	-	-	2.1	-	9.3	106.2
		64Th WAN		-	-	-	-	1	-	3.1	15.30
[85]	(PSI)	LAN	-	-	-	0.12	-	0.25	-	5.6	-
	Malicious, OKVS	WAN	-	-	-	0.64	-	1.8	-	18.62	-
[105]	(PSI) Malicious, OKVS	LAN	-	-	-	-	-	0.062	-	0.439	8.055
[70]	(DiPSI-CA) LHE	NS	2^{12}	-	-	89.50	-	530.70	-	11751.90	-
[76]	(PSI-Sum-CA) DH+Paill.	NNt	-	-	-	3.17	-	48.93	-	776.46	-
[74]	(C-PSI)	LAN	-	1.41	2.60	6.80	22.49	85.13	-	-	-
	OPPRF, BF	WAN	-	5.03	16.81	62.58	245.28	975.38	-	-	-
[99]	(C-PSI)	LAN	-	-	-	1.58	-	6.36	-	84.41	-
	OT, OPRF	WAN	-	-	-	7.11	-	24.84	-	282.89	-
[54]	(MP-PSI) GBF	4P	-	0.8	-	2.95	-	40.91	-	513.02	-
[89]	(MP-PSI)	(4P,1t)	-	-	-	1.9	-	7	-	69.6	-
	OPPRF	(10P,4t)	-	-	-	3	-	10.4	-	153.9	-
[102]	(MP-PSI)	(4P,1t)	-	-	-	0.36	-	1.83	-	18.48	-
	OKVS	(10P,4t)	-	-	-	1.78	-	8.42	-	124.26	-
[68]	(O-PSI)* Pair.	NS	2^{10}	-	10.28	-	-	-	-	20.72	-
[53]	(O-PSI) Symm	NS	-	-	6.20	25.20	101.20	415.80	1719.10	6864.20	-

achieves much faster protocols with acceptable running times even for large set sizes, making it possible to embed them in real applications.

With respect to LAN settings, they may have sense in intra-network services, when PSI can be executed to perform some kind of authentication [49] or between devices that are inside the same corporation. However, there are few applications where this has any sense. The normal instantiation is made between organizations which must communicate over the Internet, and the same is true for client-server applications. Only edge computing techniques may be useful to move the computation closer to the user, but that needs the service providers to deploy such resources on different edge servers instantiations.

Finally, there are applications where an O-PSI setting may have more sense, e.g., when PSI is coordinated by the application but performed by the users, like Private Ridesharing, Online Matchmaking, or Social Network Discovery. For these settings, users may upload their data sets in an encrypted way to an application instance and compute the intersection on demand. However, O-PSI computations are still immature, with slow performance. On Table 12, [68] is shown as faster than [53], but the data reflected here only specify the intersection computation phase (ignoring encryption and other costly primitives). On the other hand, [53] specifies the whole process but is very slow even when it is based on symmetric techniques.

7. Overhead for the Malicious Adversary model

This section will discuss the different techniques which are applied to achieve security against malicious adversaries for PSI protocols and their influence on performance. As previously stated on section 4.4, semi-honest is usually much better covered than malicious behavior; however, on the reference list there is a group of papers which target the last model. Specifically, there are 13 for the malicious model and 10 for the mixed model.

7.1. Zero Knowledge Proofs

A Zero Knowledge Proof is a protocol where a prover wants to convince the verifier of the knowledge of some secret value, but without revealing it. The verifier has to be able to check if the proof of knowledge is true. A general construction for a Proof of Knowledge (PoK) is denoted by

$$\pi = PoK\{(\alpha_1, ..., \alpha_l) | \bigwedge_{i=1}^{M} X_i = f_i(\alpha_1, ..., \alpha_l)\}, \quad (1)$$

being α_j the values to be proven and X_i the commitments which allow the verifier to check the proofs. ZKP allows a party to prove the execution of one step using some private parameter; e.g., that some data have been correctly encrypted using the corresponding private key. For that reason, they allow parties to protect against deviations, because they can abort if the proof is not correct. This building block has been the most extended to achieve security against malicious adversaries, especially in the form of Σ-protocols [174] and logarithm-based proofs [175]. From the reference list, [13, 21] apply them in the mixed model, while [22, 34, 44] target the malicious model.

Nevertheless, the disadvantage is that they are costly in terms of computation and communication, whose costs are increased linearly. The work in [13] claims that the number of exponentiations needed for a ZKP is $Exp = M + 2\sum_{i=1}^{M} Exp_{X_i}$, where Exp_{X_i} is the number of exponentiations to compute X_i, and the number of group elements to send is $GE = M + l + 1$.

7.2. Commitment Scheme

A Commitment Scheme [176, 177] is a protocol involving two parties, where one of them commits to a private value (committing phase) which is only revealed to the other party at the end (revealing phase). This protocol allows binding one party to a value and using it through the protocol; otherwise, the revealing phase will not carry out correctly, and the other party will detect the cheating.

Some papers include commitment schemes (all for the malicious setting), typically together with other techniques: [22] with polynomial checks and ZKP, [5] with random test number to allow verification, and [64] with the help of an untrusted third party. On the other hand, [54, 63] include bit commitment with cut-and-choose techniques, which will be covered in the next subsection. Finally, [36] performs dual execution to achieve malicious security, together with commitments. This protocol works similarly to the PSZ setting, with items mapped to hash tables and OPRF execution. The problem with dual execution is that the OPRF is executed in both directions for each phase, so both communication and running time are increased.

7.3. Probabilistic Verification Techniques

A Probabilistic Verification protocol allows a party to challenge the other party to reveal some of its private values, typically randomly selected. If those values fulfill some desired conditions, the rest of the private values are also supposed to fulfill them with high probability. One classical technique to achieve probabilistic verification is cut-and-choose [178, 179]. Those schemes are related to OT, where the receiver commits to some selection values and has to reveal some of them. Cut-and-choose is achieved through the following reduction[4]: $F_{ROT}^{\kappa,l} \leq F_{COT_e}^{\kappa,l'}$, where $l' = l + (\kappa + \lambda)$. From the reduction, one can observe that the number of OTs needed is increased, but unlike older protocols, which need $S = \{2, 3, 4\}$ OT per original OT and sacrifice $S-1$ OT on revealing phase, new protocols achieve better performance. That is thanks to the cheap cost of OTe and new designs for probabilistic verification: no more than 5% of passive IKNP from [131] in [55], and +1-10% of OTe in [11]).

The works in [11, 54, 63] achieve malicious security for the garbled Bloom filter approach, where cut-and-choose is employed to limit the number of 1s the receiver use in the Bloom filter [5]. On the other hand, [55] achieves security against malicious clients in the mixed model, where server is semi-honest because it may learn information about the intersection if it is computed in plain, or may influence the correctness of the computation using wrong labels or circuit description.

Finally, one interesting thing is that [54, 63] use bit commitment, but [11, 55] do not. In fact, OT has a committing property itself, by which if the sent values are randomly generated, the receiver just needs to reveal the received value as a commitment for the specific bit.

7.4. Other techniques

Some protocols achieve Malicious security thanks to Semantic Security or IND-CPA schemes [181]. Specifically, [25, 41, 79] achieve security in the mixed

[4] COT_e stands for Committed OTe.

[5] This achieves security, e.g., against Full Universe Attack [180].

model against malicious cloud or server, unlike other protocols, where the trend is to protect only against malicious clients. On the other hand, [30] works in the malicious model. If the public key scheme is semantically secure, security comes with no additional cost but only for privacy aspects. If the wish is to obtain cheating detection or data verification, other techniques must be used.

With respect to [42], it achieves security for the malicious model, but only privacy against a malicious sender. To protect against a malicious receiver, an OPRF pre-processing is employed, and for the sender, a labeled PSI protocol is used, which is based on hashing. Like previously discussed, when discussing semantically secure protocols, to avoid deviation, using ZKP would be required.

Other protocols achieve privacy against malicious adversaries, like [69], which uses hashing and exponentiations, or [58], which is based on secret sharing. The latest, however, assumes a trusted commodity server. The works in [17, 45] achieve privacy too, thanks to randomized matrixes. To allow verification, [57] adds to the computation 3 dummy sets, which are extracted after the intersection. However, it is necessary to agree previously on the dummy sets, that also add more data to the computation, lowering the protocol performance. Finally, [61] argues malicious security, but from application specific issues, while the sublayer PSI protocol is semi-honest.

Finally, constructions based on OKVS [85, 103, 105] achieve malicious security because their proposed structures do not leak information as standard semi-honest Cuckoo-based techniques do. More precisely, achieving malicious security in these protocols only involves changing a set of parameters from the base protocol, achieving overheads not so critical with respect to the semi-honest setting.

8. Conclusions

We have carried out a Systematic Literature Review on Private Set Intersection that shows the ascending popularity of PSI in the number of published papers, as well as the many different perspectives from where it is addressed, identifying protocols, building blocks, and the most typical applications. The analysis of the 76 papers from the reference list compares not only the different security models and primitives but primarily the performance of each protocol, serving as a good reference when trying to decide which protocol to use for each particular scenario.

The goal of this paper was to give some answers to the initial questions proposed in Section 2.1, that deal with the use of different building blocks, relative performance of different approaches, different use cases, and challenges to improve efficiency.

Regarding building blocks, as it is shown in Figure 5, HE, OT, and OPRF remain as the main building blocks upon which PSI is built. While HE is identified as an easy construction to build PSI on, OT and OPRF-based PSI presents the most competitive performance. Other building blocks, like generic Public Key, Commutative Encryption, or Pairings, present some limitations regarding protocol capacity or a slower performance. We also remark a notable trend in Quantum PSI, where more papers have been proposed the last years [182–189] than in previous years [37, 50].

In general, PSI has been covered from a theoretical point of view, like most of the papers from the reference list. Researchers tend to analyze how PSI could be improved to be more efficient or secure, but there are only a few papers which analyze PSI when it is embedded into practical applications or how to develop efficient and versatile libraries.

PSI has been widely studied because of the wide set of use cases where these protocols can be applied. Section 3 gives a description of the different use cases which have been proposed, together with some classifications to specific scenarios and their characteristics. The discussion has been based on the balanced and unbalanced approaches, relating them to specific applications.

There are still some remaining challenges if the PSI paradigm is to be improved. First of all, working on the performance is essential. Even when some protocols achieve very competitive running times, they are typically based on some assumptions and trade-offs. Some approaches (e.g., HE) assume high computational power, while others (e.g., OPRF) assume high bandwidth. The asymptotic levels are well placed, as nearly every protocol is sublinear. However, improving techniques like hashing structures or getting a faster OT may lead to better running times. In this line, OKVS-based protocols have boosted performance with respect other techniques, so we identify them as the new standard building block for PSI. From the trade-offs which have to be made, another aspect arises, namely how to classify or unify the different approaches. Generic circuit techniques solve this problem and allow PSI to be easily embedded into other SMPC protocols, although at the expense of performance. In addition to that, dynamically switching from the OT to the HE setting may not be possible. In light of the above, developing libraries and leading new constructions to more adaptive scenarios may be an interesting open problem. For now, some works are introducing hybrid compilers that optimize circuit computations performance, selecting different building blocks for each sub-circuit. Other properties like multi-interaction or parameter reusability may help to improve performance too. Another constrain is the number of parties, as MP-PSI with high N typically means unmanageable running times. For that reason, improving the multi-party case is also identified as an open problem. As MP-PSI are obtaining more attention through the years, most designs are asymmetric, assuming one party with a higher number of resources available. Finally, with respect to security models, most of the protocols are only secure against semi-honest adversaries. For the malicious setting, different techniques have been analyzed, and some of them (e.g., malicious OT) do not add very high overloads. The main problem is how to achieve efficient proofs of correct behavior, allowing more secure protocols.

Acknowledgements

This work has been partially supported by the projects: BIGPrivDATA (UMA20-FEDERJA-082) from the FEDER Andalucía 2014-2020 Program and SecTwin 5.0 funded by the Ministry of Science and Innovation, and the European Union (Next Generation EU) (TED2021-129830B-I00). The first author has been funded by the Spanish Ministry of Education under the National F.P.U. Program (FPU19/01118). Funding for open access charge: Universidad de Málaga / CBUA.

References

[1] A. C. Yao, Protocols for secure computations, in: 23rd Annual Symposium on Foundations of Computer Science (sfcs 1982), 1982, pp. 160–164, iSSN: 0272-5428. doi:10.1109/SFCS.1982.38.

[2] O. Goldreich, S. Micali, A. Wigderson, How to play ANY mental game, in: Proceedings of the nineteenth annual ACM symposium on Theory of computing, STOC '87, Association for Computing Machinery, New York, NY, USA, 1987, pp. 218–229. doi:10.1145/28395.28420.

[3] M. J. Freedman, K. Nissim, B. Pinkas, Efficient Private Matching and Set Intersection, in: C. Cachin, J. L. Camenisch (Eds.), Advances in Cryptology - EUROCRYPT 2004, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2004, pp. 1–19. doi:10.1007/978-3-540-24676-3_1.

[4] Y. Xiao, M. Watson, Guidance on Conducting a Systematic Literature Review, Journal of Planning Education and Research 39 (1) (2019) 93–112, publisher: SAGE Publications Inc. doi:10.1177/0739456X17723971.

[5] X. Cao, H. Li, L. Dang, Y. Lin, A two-party privacy preserving set intersection protocol against malicious users in cloud computing, Computer Standards & Interfaces 54 (2017) 41–45. doi:10.1016/j.csi.2016.08.004.

[6] M. M. Oliaiy, M. H. Ameri, J. Mohajeri, M. R. Aref, A Verifiable Delegated Set Intersection without pairing, in: 2017 Iranian Conference on Electrical Engineering (ICEE), 2017, pp. 2047–2051. doi:10.1109/IranianCEE.2017.7985395.

[7] A. Davidson, C. Cid, An efficient toolkit for computing private set operations, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 10343 LNCS (2017) 261–278. doi:10.1007/978-3-319-59870-3_15.

[8] C. Dong, G. Loukides, Approximating Private Set Union/Intersection Cardinality With Logarithmic Complexity, IEEE Transactions on Information Forensics and Security 12 (11) (2017) 2792–2806, conference Name: IEEE Transactions on Information Forensics and Security. doi:10.1109/TIFS.2017.2721360.

[9] S. Zander, L. L. H. Andrew, G. Armitage, Collaborative and privacy-preserving estimation of IP address space utilisation, Computer Networks 119 (2017) 56–70. doi:10.1016/j.comnet.2017.03.010.

[10] H. Chen, K. Laine, P. Rindal, Fast Private Set Intersection from Homomorphic Encryption, in: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS '17, Association for Computing Machinery, New York, NY, USA, 2017, pp. 1243–1255. doi:10.1145/3133956.3134061.

[11] P. Rindal, M. Rosulek, Improved private set intersection against malicious adversaries, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 10210 LNCS (2017) 235–259. doi:10.1007/978-3-319-56620-7_9.

[12] P. Rindal, M. Rosulek, Malicious-Secure Private Set Intersection via Dual Execution, in: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS '17, Association for Computing Machinery, New York, NY, USA, 2017, pp. 1229–1242. doi:10.1145/3133956.3134044.

[13] S. Debnath, R. Dutta, New realizations of efficient and secure private set intersection protocols preserving fairness, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 10157 LNCS (2017) 254–284. doi:10.1007/978-3-319-53177-9_14.

[14] V. Kolesnikov, N. Matania, B. Pinkas, M. Rosulek, N. Trieu, 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, Association for Computing Machinery, New York, NY, USA, 2017, pp. 1257–1272. doi:10.1145/3133956.3134065.

[15] A. Miyaji, K. Nakasho, S. Nishida, Privacy-Preserving Integration of Medical Data, Journal of Medical Systems 41 (3) (2017) 37. doi:10.1007/s10916-016-0657-4.

[16] A. Kiss, J. Liu, T. Schneider, N. Asokan, B. Pinkas, Private set intersection for unequal set sizes with mobile applications, Proceedings on Privacy Enhancing Technologies 2017 (4) (2017) 177–197. doi:10.1515/popets-2017-0044.

[17] Z. Gheid, Y. Challal, Private and efficient set intersection protocol for big data analytics, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 10393 LNCS (2017) 149–164. doi:10.1007/978-3-319-65482-9_10.

[18] A. Rasheed, A. Kenneth, R. Mahapatra, D. Puthal, Private matching and set intersection computation in multi-agent and industrial control systems, in: Proceedings of the 12th Annual Conference on Cyber and Information Security Research, CISRC '17, Association for Computing Machinery, New York, NY, USA, 2017, pp. 1–6. doi:10.1145/3064814.3064817.

[19] P. D'Arco, M. González Vasco, A. Pérez Del Pozo, C. Soriente, R. Steinwandt, Private set intersection: New generic constructions and feasibility results, Advances in Mathematics of Communications 11 (3) (2017) 481–502. doi:10.3934/amc.2017040.

[20] X. Carpent, S. Faber, T. Sander, G. Tsudik, Private Set Projections &amp; Variants, in: Proceedings of the 2017 on Workshop on Privacy in the Electronic Society, WPES '17, Association for Computing Machinery, New York, NY, USA, 2017, pp. 87–98. doi:10.1145/3139550.3139554.

[21] S. Debnath, R. Dutta, Provably secure fair mutual private set intersection cardinality utilizing bloom filter, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 10143 LNCS (2017) 505–525. doi:10.1007/978-3-319-54705-3_31.

[22] C. Hazay, M. Venkitasubramaniam, Scalable Multi-party Private Set-Intersection, in: S. Fehr (Ed.), Public-Key Cryptography – PKC 2017, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2017, pp. 175–203. doi:10.1007/978-3-662-54365-8_8.

[23] M. Aziz, D. Alhadidi, N. Mohammed, Secure approximation of edit distance on genomic data, BMC Medical Genomics 10 (2017). doi:10.1186/s12920-017-0279-9.

[24] E. Zhang, F. Li, B. Niu, Y. Wang, Server-aided private set intersection based on reputation, Information Sciences 387 (2017) 180–194. doi:10.1016/j.ins.2016.09.056.

[25] A. Abadi, S. Terzis, C. Dong, VD-PSI: Verifiable delegated private set intersection on outsourced private datasets, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 9603 LNCS (2017) 149–168. doi:10.1007/978-3-662-54970-4_9.

[26] M. Ion, B. Kreuter, E. Nergiz, S. Patel, S. Saxena, K. Seth, D. Shanahan, M. Yung, Private Intersection-Sum Protocol with Applications to Attributing Aggregate Ad Conversions, Tech. Rep. 738 (2017).

[27] P. Hallgren, C. Orlandi, A. Sabelfeld, PrivatePool: Privacy-Preserving Ridesharing, in: 2017 IEEE 30th Computer Security Foundations Symposium (CSF), 2017, pp. 276–291, iSSN: 2374-8303. doi:10.1109/CSF.2017.24.

[28] M. Kim, B. Z. Kim, An experimental study of encrypted polynomial arithmetics for private set operations, Journal of Communications and Networks 19 (5) (2017) 431–441. doi:10.1109/JCN.2017.000075.

[29] Y. Wen, Z. Gong, Z. Huang, W. Qiu, A new efficient authorized private set intersection protocol from Schnorr signature and its applications, Cluster Computing 21 (1) (2018) 287–297. doi:10.1007/s10586-017-0940-2.

[30] X. Wang, F. Xhafa, X. Luo, S. Zhang, Y. Ding, A privacy-preserving fuzzy interest matching protocol for friends finding in social networks, Soft Computing 22 (8) (2018) 2517–2526. doi:10.1007/s00500-017-2506-x.

[31] T. N. D. Pham, C. K. Yeo, Adaptive trust and privacy management framework for vehicular networks, Vehicular Communications 13 (2018) 1–12. doi:10.1016/j.vehcom.2018.04.006.

[32] Y. Zhao, S. S. Chow, Can You Find The One for Me?, in: Proceedings of the 2018 Workshop on Privacy in the Electronic Society, WPES'18, Association for Computing Machinery, New York, NY, USA, 2018, pp. 54–65. doi:10.1145/3267323.3268965.

[33] M. Ciampi, C. Orlandi, Combining private set-intersection with secure two-party computation, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 11035 LNCS (2018) 464–482. doi:10.1007/978-3-319-98113-0_25.

[34] D. Liu, J. Ni, H. Li, X. Lin, X. Shen, Efficient and Privacy-Preserving Ad Conversion for V2X-Assisted Proximity Marketing, in: 2018 IEEE 15th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), 2018, pp. 10–18, iSSN: 2155-6814. doi:10.1109/MASS.2018.00014.

[35] L. Shen, X. Chen, D. Wang, B. Fang, Y. Dong, Efficient and Private Set Intersection of Human Genomes, in: 2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), 2018, pp. 761–764. doi:10.1109/BIBM.2018.8621291.

[36] B. Pinkas, T. Schneider, C. Weinert, U. Wieder, Efficient circuit-based psi via cuckoo hashing, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 10822 LNCS (2018) 125–157. doi:10.1007/978-3-319-78372-7_5.

[37] R. Shi, Efficient Quantum Protocol for Private Set Intersection Cardinality, IEEE Access 6 (2018) 73102–73109, conference Name: IEEE Access. doi:10.1109/ACCESS.2018.2872741.

[38] R. Inbar, E. Omri, B. Pinkas, Efficient scalable multiparty private set-intersection via garbled bloom filters, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture

Notes in Bioinformatics) 11035 LNCS (2018) 235–252. doi:10.1007/978-3-319-98113-0_13.

[39] Q. Wang, F.-c. Zhou, T.-m. Ma, Z.-f. Xu, Faster fog-aided private set intersection with integrity preserving, Frontiers of Information Technology & Electronic Engineering 19 (12) (2018) 1558–1568. doi:10.1631/FITEE.1800518.

[40] X. Yang, X. Luo, X. A. Wang, S. Zhang, Improved outsourced private set intersection protocol based on polynomial interpolation, Concurrency and Computation: Practice and Experience 30 (1) (2018) e4329, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.4329. doi:https://doi.org/10.1002/cpe.4329.

[41] S. Terada, K. Yoneyama, Improved Verifiable Delegated Private Set Intersection, in: 2018 International Symposium on Information Theory and Its Applications (ISITA), 2018, pp. 520–524. doi:10.23919/ISITA.2018.8664310.

[42] H. Chen, Z. Huang, K. Laine, P. Rindal, Labeled PSI from Fully Homomorphic Encryption with Malicious Security, in: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS '18, Association for Computing Machinery, New York, NY, USA, 2018, pp. 1223–1237. doi:10.1145/3243734.3243836.

[43] A. Cerulli, E. De Cristofaro, C. Soriente, Nothing refreshes like a RePSI: Reactive private set intersection, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 10892 LNCS (2018) 280–300. doi:10.1007/978-3-319-93387-0_15.

[44] C. Hazay, Oblivious Polynomial Evaluation and Secure Set-Intersection from Algebraic PRFs, Journal of Cryptology 31 (2) (2018) 537–586. doi:10.1007/s00145-017-9263-y.

[45] Z. Gheid, Y. Challal, L. Chen, Private and efficient set intersection protocol for RFID-based food adequacy check, in: 2018 IEEE Wireless Communications and Networking Conference (WCNC), 2018, pp. 1–6, iSSN: 1558-2612. doi:10.1109/WCNC.2018.8377207.

[46] S. Chatterjee, C. Kamath, V. Kumar, Private set-intersection with common set-up, Advances in Mathematics of Communications 12 (1) (2018) 17–47. doi:10.3934/amc.2018002.

[47] B. Pinkas, T. Schneider, M. Zohner, Scalable private set intersection based on ot extension, ACM Transactions on Privacy and Security 21 (2) (2018). doi:10.1145/3154794.

[48] U. M. Aïvodji, K. Huguenin, M.-J. Huguet, M.-O. Killijian, SRide: A Privacy-Preserving Ridesharing System, in: Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks, ACM, Stockholm Sweden, 2018, pp. 40–50. doi:10.1145/3212480.3212483.

[49] W. Cui, C. Du, J. Chen, PSP: proximity-based secure pairing of mobile devices using WiFi signals, Wireless Networks 25 (2) (2019) 733–751. doi:10.1007/s11276-017-1588-9.

[50] R. Shi, M. Zhang, A Feasible Quantum Protocol for Private Set Intersection Cardinality, IEEE Access 7 (2019) 72105–72112, conference Name: IEEE Access. doi:10.1109/ACCESS.2019.2919119.

[51] L. Shen, X. Chen, J. Shi, B. Fang, A More Efficient Private Set Intersection Protocol Based on Random OT and Balance Hash, in: ICC 2019 - 2019 IEEE International Conference on Communications (ICC), 2019, pp. 1–7, iSSN: 1938-1883. doi:10.1109/ICC.2019.8761417.

[52] B. Pinkas, T. Schneider, O. Tkachenko, A. Yanai, Efficient circuit-based PSI with linear communication, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 11478 LNCS (2019) 122–153. doi:10.1007/978-3-030-17659-4_5.

[53] A. Abadi, S. Terzis, R. Metere, C. Dong, Efficient Delegated Private Set Intersection on Outsourced Private Datasets, IEEE Transactions on Dependable and Secure Computing 16 (4) (2019) 608–624, conference Name: IEEE Transactions on Dependable and Secure Computing. doi:10.1109/TDSC.2017.2708710.

[54] E. Zhang, F.-H. Liu, Q. Lai, G. Jin, Y. Li, Efficient Multi-Party Private Set Intersection Against Malicious Adversaries, in: Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop, CCSW'19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 93–104. doi:10.1145/3338466.3358927.

[55] D. Kales, C. Rechberger, T. Schneider, M. Senker, C. Weinert, Mobile Private Contact Discovery at Scale, 2019, pp. 1447–1464.

[56] O. Ruan, Z. Wang, J. Mi, M. Zhang, New Approach to Set Representation and Practical Private Set-Intersection Protocols, IEEE Access 7 (2019) 64897–64906, conference Name: IEEE Access. doi:10.1109/ACCESS.2019.2917057.

[57] S. Qiu, Z. Dai, D. Zha, Z. Zhang, Y. Liu, PPSI: Practical Private Set Intersection Over Large-Scale Datasets, in: 2019 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud

Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), 2019, pp. 1249–1254. doi:10.1109/SmartWorld-UIC-ATC-SCALCOM-IOP-SCI.2019.00232.

[58] D. Boer, Z. Ahmadi, S. Kramer, Privacy Preserving Client/Vertical-Servers Classification, in: C. Alzate, A. Monreale, L. Bioglio, V. Bitetta, I. Bordino, G. Caldarelli, A. Ferretti, R. Guidotti, F. Gullo, S. Pascolutti, R. G. Pensa, C. Robardet, T. Squartini (Eds.), ECML PKDD 2018 Workshops, Lecture Notes in Computer Science, Springer International Publishing, Cham, 2019, pp. 125–140. doi:10.1007/978-3-030-13463-1_10.

[59] S. Ramezanian, T. Meskanen, V. Niemi, Privacy preserving 2-party queries on bipartite graphs with private set intersection, in: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, SAC '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 1867–1870. doi:10.1145/3297280.3297610.

[60] B. Hemenway Falk, D. Noble, R. Ostrovsky, Private Set Intersection with Linear Communication from General Assumptions, in: Proceedings of the 18th ACM Workshop on Privacy in the Electronic Society, WPES'19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 14–25. doi:10.1145/3338498.3358645.

[61] L. Li, B. Pal, J. Ali, N. Sullivan, R. Chatterjee, T. Ristenpart, Protocols for Checking Compromised Credentials, in: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, ACM, London United Kingdom, 2019, pp. 1387–1403. doi:10.1145/3319535.3354229.

[62] S. Ghosh, M. Simkin, The Communication Complexity of Threshold Private Set Intersection, in: A. Boldyreva, D. Micciancio (Eds.), Advances in Cryptology – CRYPTO 2019, Lecture Notes in Computer Science, Springer International Publishing, Cham, 2019, pp. 3–29. doi:10.1007/978-3-030-26951-7_1.

[63] T. Jiang, X. Yuan, Traceable Private Set Intersection in Cloud Computing, in: 2019 IEEE Conference on Dependable and Secure Computing (DSC), 2019, pp. 1–7. doi:10.1109/DSC47296.2019.8937666.

[64] P. H. Le, S. Ranellucci, S. D. Gordon, Two-party Private Set Intersection with an Untrusted Third Party, in: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 2403–2420. doi:10.1145/3319535.3345661.

[65] Y. Li, Z. L. Jiang, L. Yao, X. Wang, S. M. Yiu, Z. Huang, Outsourced privacy-preserving c4.5 decision tree algorithm over horizontally and vertically partitioned dataset among multiple parties, Cluster Computing 22 (1) (2019) 1581–1593. doi:10.1007/s10586-017-1019-9.

[66] L. Chen, Z. Li, Z. Chen, Y. Liu, Two anti-quantum attack protocols for secure multiparty computation, in: H. Zhang, B. Zhao, F. Yan (Eds.), Trusted Computing and Information Security, Springer Singapore, Singapore, 2019, pp. 338–359.

[67] O. Ruan, X. Huang, H. Mao, An Efficient Private Set Intersection Protocol for the Cloud Computing Environments, in: 2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS), 2020, pp. 254–259. doi:10.1109/BigDataSecurity-HPSC-IDS49724.2020.00053.

[68] M. Ali, J. Mohajeri, M.-R. Sadeghi, X. Liu, Attribute-based fine-grained access control for outscored private set intersection computation, Information Sciences 536 (2020) 222–243. doi:10.1016/j.ins.2020.05.041.

[69] K. EdalatNejad, W. Lueks, J. Martin, S. Ledésert, A. L'Hôte, B. Thomas, L. Girod, C. Troncoso, Datasharenetwork a decentralized privacy-preserving search engine for investigative journalists, 2020, pp. 1911–1927.

[70] B. Kacsmar, B. Khurram, N. Lukas, A. Norton, M. Shafieinejad, Z. Shang, Y. Baseri, M. Sepehri, S. Oya, F. Kerschbaum, Differentially Private Two-Party Set Operations, in: 2020 IEEE European Symposium on Security and Privacy (EuroS P), 2020, pp. 390–404. doi:10.1109/EuroSP48549.2020.00032.

[71] O. Ruan, H. Mao, Efficient Private Set Intersection Using Point-Value Polynomial Representation, iSSN: 1939-0114 Pages: e8890677 Publisher: Hindawi Volume: 2020 (Sep. 2020). doi:https://doi.org/10.1155/2020/8890677.

[72] A. Kavousi, J. Mohajeri, M. Salmasizadeh, Improved Secure Efficient Delegated Private Set Intersection, in: 2020 28th Iranian Conference on Electrical Engineering (ICEE), 2020, pp. 1–6, iSSN: 2642-9527. doi:10.1109/ICEE50131.2020.9260663.

[73] Y. Wen, F. Zhang, H. Wang, Y. Miao, Z. Gong, Intersection-policy private mutual authentication from authorized private set intersection, Science China Information Sciences 63 (2) (2020) 122101. doi:10.1007/s11432-019-9907-x.

[74] F. Karakoç, A. Küpçü, Linear Complexity Private Set Intersection for Secure Two-Party Protocols, in: S. Krenn, H. Shulman, S. Vaudenay (Eds.), Cryptology and Network Security, Lecture Notes in Computer Science, Springer International Publishing, Cham, 2020, pp. 409–429. doi:10.1007/978-3-030-65411-5_20.

[75] X. Wang, X. Kuang, J. Li, J. Li, X. Chen, Z. Liu, Oblivious Transfer for Privacy-Preserving in VANET's Feature Matching, IEEE Transactions on Intelligent Transportation Systems (2020) 1–8Conference

Name: IEEE Transactions on Intelligent Transportation Systems. doi:10.1109/TITS.2020.2973738.

[76] M. Ion, B. Kreuter, A. E. Nergiz, S. Patel, S. Saxena, K. Seth, M. Raykova, D. Shanahan, M. Yung, On Deploying Secure Computing: Private Intersection-Sum-with-Cardinality, in: 2020 IEEE European Symposium on Security and Privacy (EuroS P), 2020, pp. 370–389. doi:10.1109/EuroSP48549.2020.00031.

[77] S. K. Debnath, P. Stănică, T. Choudhury, N. Kundu, Post-quantum protocol for computing set intersection cardinality with linear complexity, IET Information Security 14 (6) (2020) 661–669, conference Name: IET Information Security. doi:10.1049/iet-ifs.2019.0315.

[78] W. Wang, S. Li, J. Dou, R. Du, Privacy-preserving mixed set operations, Information Sciences 525 (2020) 67–81. doi:10.1016/j.ins.2020.03.049.

[79] S. Ramezanian, T. Meskanen, M. Naderpour, V. Junnila, V. Niemi, Private membership test protocol with low communication complexity, Digital Communications and Networks 6 (3) (2020) 321–332. doi:10.1016/j.dcan.2019.05.002.

[80] K. Nomura, Y. Shiraishi, M. Mohri, M. Morii, Secure Association Rule Mining on Vertically Partitioned Data Using Private-Set Intersection, IEEE Access 8 (2020) 144458–144467. doi:10.1109/ACCESS.2020.3014330.

[81] Q. Wang, F. Zhou, J. Xu, S. Peng, Tag-based Verifiable Delegated Set Intersection over Outsourced Private Datasets, IEEE Transactions on Cloud Computing (2020) 1–1Conference Name: IEEE Transactions on Cloud Computing. doi:10.1109/TCC.2020.2968320.

[82] S. Lv, J. Ye, S. Yin, X. Cheng, C. Feng, X. Liu, R. Li, Z. Li, Z. Liu, L. Zhou, Unbalanced private set intersection cardinality protocol with low communication cost, Future Generation Computer Systems 102 (2020) 1054–1061. doi:10.1016/j.future.2019.09.022.

[83] Z. Liang, W. Liu, F. Zhang, B. Zhang, J. Liu, L. Zhang, K. Ren, A Framework of Private Set Intersection Protocols., Tech. Rep. 1541 (2020).

[84] Y. Zhang, B. Zhu, Y. Fang, S. Guo, A. Zhang, S. Zhong, Secure inter-domain forwarding loop test in software defined networks, IEEE Transactions on Dependable and Secure Computing 17 (1) (2020) 162–178. doi:10.1109/TDSC.2017.2731773.

[85] B. Pinkas, M. Rosulek, N. Trieu, A. Yanai, Psi from paxos: Fast, malicious private set intersection, in: A. Canteaut, Y. Ishai (Eds.), Advances in Cryptology – EUROCRYPT 2020, Springer International Publishing, Cham, 2020, pp. 739–767.

[86] S. Mishima, K. Nakasho, Y. Takano, A. Miyaji, A practical parallel computation in a scalable multiparty private set intersection, in: 2021 Ninth International Symposium on Computing and Networking Workshops (CANDARW), 2021, pp. 332–338. doi:10.1109/CANDARW53999.2021.00063.

[87] A. Patra, T. Schneider, A. Suresh, H. Yalame, ABY2.0: Improved Mixed-Protocol secure Two-Party computation, in: 30th USENIX Security Symposium (USENIX Security 21), USENIX Association, 2021, pp. 2165–2182.

[88] L. Reichert, M. Pazelt, B. Scheuermann, Circuit-based psi for covid-19 risk scoring, in: 2021 IEEE International Performance, Computing, and Communications Conference (IPCCC), IEEE Computer Society, Los Alamitos, CA, USA, 2021, pp. 1–8. doi:10.1109/IPCCC51483.2021.9679360.

[89] N. Chandran, N. Dasgupta, D. Gupta, S. L. B. Obbattu, S. Sekar, A. Shah, Efficient linear multiparty psi and extensions to circuit/quorum psi, in: Proceedings 2021 acm sigsac conference on computer communications security, Association for Computing Machinery, New York, NY, USA, 2021, p. 1182–1204. doi:10.1145/3460120.3484591.

[90] A. Kavousi, J. Mohajeri, M. Salmasizadeh, Efficient scalable multi-party private set intersection using oblivious prf, in: R. Roman, J. Zhou (Eds.), Security and Trust Management, Springer International Publishing, Cham, 2021, pp. 81–99.

[91] E. Zhang, J. Chang, Y. Li, Efficient threshold private set intersection, IEEE Access 9 (2021) 6560–6570. doi:10.1109/ACCESS.2020.3048743.

[92] N. Alamati, P. Branco, N. Döttling, S. Garg, M. Hajiabadi, S. Pu, Laconic private set intersection and applications, in: K. Nissim, B. Waters (Eds.), Theory of Cryptography, Springer International Publishing, Cham, 2021, pp. 94–125.

[93] S. Ramezanian, T. Meskanen, V. Niemi, Multi-party private set operations with an external decider, in: K. Barker, K. Ghazinour (Eds.), Data and Applications Security and Privacy XXXV, Springer International Publishing, Cham, 2021, pp. 117–135.

[94] S. Badrinarayanan, P. Miao, S. Raghuraman, P. Rindal, Multi-party threshold private set intersection with sublinear communication, in: J. A. Garay (Ed.), Public-Key Cryptography – PKC 2021, Springer International Publishing, Cham, 2021, pp. 349–379.

[95] G. Garimella, B. Pinkas, M. Rosulek, N. Trieu, A. Yanai, Oblivious key-value stores and amplification for private set intersection, in: T. Malkin, C. Peikert (Eds.), Advances in Cryptology – CRYPTO 2021, Springer International Publishing, Cham, 2021, pp. 395–425.

[96] S. K. Debnath, T. Choudhury, N. Kundu, K. Dey, Post-quantum secure multi-party private set-intersection in star network topology, Journal of Information Security and Applications 58 (2021) 102731. doi:https://doi.org/10.1016/j.jisa.2020.102731.

[97] Y. Li, D. Ghosh, P. Gupta, S. Mehrotra, N. Panwar, S. Sharma, Prism: Private verifiable set computation over multi-owner outsourced databases, in: Proceedings of the 2021 International Conference on Management of Data, SIGMOD '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 1116–1128. doi:10.1145/3448016.3452839.

[98] Y. Wang, Q. Huang, H. Li, M. Xiao, S. Ma, W. Susilo, Private set intersection with authorization over outsourced encrypted datasets, IEEE Transactions on Information Forensics and Security 16 (2021) 4050–4062. doi:10.1109/TIFS.2021.3101059.

[99] G. Garimella, P. Mohassel, M. Rosulek, S. Sadeghian, J. Singh, Private set operations from oblivious switching, in: J. A. Garay (Ed.), Public-Key Cryptography – PKC 2021, Springer International Publishing, Cham, 2021, pp. 591–617.

[100] J. Takeshita, R. Karl, A. Mohammed, A. Striegel, T. Jung, Provably secure contact tracing with conditional private set intersection, in: J. Garcia-Alfaro, S. Li, R. Poovendran, H. Debar, M. Yung (Eds.), Security and Privacy in Communication Networks, Springer International Publishing, Cham, 2021, pp. 352–373.

[101] A. Shah, N. Chandran, M. Dema, D. Gupta, A. Gururajan, H. Yu, Secure featurization and applications to secure phishing detection, in: Proceedings of the 2021 on Cloud Computing Security Workshop, CCSW '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 83–95. doi:10.1145/3474123.3486759.

[102] O. Nevo, N. Trieu, A. Yanai, Simple, fast malicious multiparty private set intersection, in: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, CCS '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 1151–1165. doi:10.1145/3460120.3484772.

[103] P. Rindal, P. Schoppmann, Vole-psi: Fast oprf and circuit-psi from vector-ole, in: A. Canteaut, F.-X. Standaert (Eds.), Advances in Cryptology – EUROCRYPT 2021, Springer International Publishing, Cham, 2021, pp. 901–930.

[104] D. Wang, X. Chen, L. Zhang, Y. Fang, C. Huang, A blockchain-based human-to-infrastructure contact tracing approach for covid-19, IEEE Internet of Things Journal 9 (14) (2022) 12836–12847. doi:10.1109/JIOT.2021.3138971.

[105] S. Raghuraman, P. Rindal, Blazing fast psi from improved okvs and subfield vole, in: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 2505–2517. doi:10.1145/3548606.3560658.

[106] Y. Zhang, Y. Shi, Z. Zhou, C. Xue, Y. Xu, K. Xu, J. Du, Efficient and secure skyline queries over vertical data federation, IEEE Transactions on Knowledge and Data Engineering (2022) 1–12doi:10.1109/TKDE.2022.3222415.

[107] L. Wei, J. Liu, L. Zhang, W. Zhang, Efficient and collusion resistant multiparty private set intersection protocols for large participants and small sets setting, in: X. Chen, J. Shen, W. Susilo (Eds.), Cyberspace Safety and Security, Springer International Publishing, Cham, 2022, pp. 118–132.

[108] L. Xiong, Z. L. Jiang, Y. Huang, J. Wang, J. Xiao, W. Zhang, X. Wang, Efficient private set intersection based on functional encryption, in: 2022 4th International Conference on Data Intelligence and Security (ICDIS), 2022, pp. 9–15. doi:10.1109/ICDIS55630.2022.00009.

[109] H. Li, Y. Gao, Efficient private set intersection cardinality protocol in the reverse unbalanced setting, in: W. Susilo, X. Chen, F. Guo, Y. Zhang, R. Intan (Eds.), Information Security, Springer International Publishing, Cham, 2022, pp. 20–39.

[110] A. Adavoudi Jolfaei, H. Mala, M. Zarezadeh, Eo-psi-ca: Efficient outsourced private set intersection cardinality, Journal of Information Security and Applications 65 (2022) 102996. doi:https://doi.org/10.1016/j.jisa.2021.102996.

[111] M. Wu, T. H. Yuen, Gcd-filter: Private set intersection without encryption, in: L. Wang, M. Segal, J. Chen, T. Qiu (Eds.), Wireless Algorithms, Systems, and Applications, Springer Nature Switzerland, Cham, 2022, pp. 429–440.

[112] A. Abadi, C. Dong, S. J. Murdoch, S. Terzis, Multi-party updatable delegated private set intersection, in: I. Eyal, J. Garay (Eds.), Financial Cryptography and Data Security, Springer International Publishing, Cham, 2022, pp. 100–119.

[113] F. Kato, Y. Cao, M. Yoshikawa, Pct-tee: Trajectory-based private contact tracing system with trusted execution environment, ACM Trans. Spatial Algorithms Syst. 8 (2) (dec 2022). doi:10.1145/3490491.

[114] A. Bay, Z. Erkin, J.-H. Hoepman, S. Samardjiska, J. Vos, Practical multiparty private set intersection protocols, IEEE Transactions on Information Forensics and Security 17 (2022) 1–15. doi:10.1109/TIFS.2021.3118879.

[115] D. R. George, S. Sciancalepore, Prm - private interference discovery for ieee 802.15. 4 networks, in: 2022 IEEE Conference on Communications and Network Security (CNS), 2022, pp. 136–144. doi:10.1109/CNS56114.2022.9947236.

[116] A. Ben-Efraim, O. Nissenbaum, E. Omri, A. Paskin-Cherniavsky, Psimple: Practical multiparty maliciously-secure private set intersection, in: Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security, ASIA CCS '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 1098–1112. doi:10.1145/3488932.3523254.

[117] Y. Jiang, J. Wei, J. Pan, Publicly verifiable private set intersection from homomorphic encryption, in: X. Chen, X. Huang, M. Kutyłowski (Eds.), Security and Privacy in Social Networks and Big Data, Springer Nature Singapore, Singapore, 2022, pp. 117–137.

[118] G. Garimella, M. Rosulek, J. Singh, Structure-aware private set intersection, with applications to fuzzy matching, in: Y. Dodis, T. Shrimpton (Eds.), Advances in Cryptology – CRYPTO 2022, Springer Nature Switzerland, Cham, 2022, pp. 323–352.

[119] J. H. M. Ying, S. Cao, G. S. Poh, J. Xu, H. W. Lim, Psi-stats: Private set intersection protocols supporting secure statistical functions, in: G. Ateniese, D. Venturi (Eds.), Applied Cryptography and Network Security, Springer International Publishing, Cham, 2022, pp. 585–604.

[120] S. Nagaraja, P. Mittal, C.-Y. Hong, M. Caesar, N. Borisov, BotGrep: Finding P2P Bots with Structured Graph Analysis 16.

[121] E. D. Cristofaro, J. Kim, G. Tsudik, Linear-Complexity Private Set Intersection Protocols Secure in Malicious Model, Tech. Rep. 469 (2010).

[122] L. Kamm, J. Willemson, Secure Floating-Point Arithmetic and Private Satellite Collision Analysis, Tech. Rep. 850 (2013).

[123] B. H. Bloom, Space/time trade-offs in hash coding with allowable errors, Communications of the ACM 13 (7) (1970) 422–426. doi:10.1145/362686.362692.

[124] B. Fan, D. G. Andersen, M. Kaminsky, M. D. Mitzenmacher, Cuckoo Filter: Practically Better Than Bloom, in: Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies, ACM, Sydney Australia, 2014, pp. 75–88. doi:10.1145/2674005.2674994.

[125] M. O. Rabin, How To Exchange Secrets with Oblivious Transfer, Tech. Rep. 187 (2005).

[126] S. Even, O. Goldreich, A. Lempel, A Randomized Protocol for Signing Contracts, in: D. Chaum, R. L. Rivest, A. T. Sherman (Eds.), Advances in Cryptology, Springer US, Boston, MA, 1983, pp. 205–210. doi:10.1007/978-1-4757-0602-4_19.

[127] G. Brassard, C. Crepeau, J.-M. Robert, All-or-Nothing Disclosure of Secrets, in: A. M. Odlyzko (Ed.), Advances in Cryptology — CRYPTO' 86, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 1987, pp. 234–238. doi:10.1007/3-540-47721-7_17.

[128] M. Naor, B. Pinkas, Computationally Secure Oblivious Transfer, Journal of Cryptology 18 (1) (2005) 1–35. doi:10.1007/s00145-004-0102-6.

[129] T. Chou, C. Orlandi, The Simplest Protocol for Oblivious Transfer, Tech. Rep. 267 (2015).

[130] M. Naor, B. Pinkas, Efficient oblivious transfer protocols, in: Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms, SODA '01, Society for Industrial and Applied Mathematics, USA, 2001, pp. 448–457.

[131] Y. Ishai, J. Kilian, K. Nissim, E. Petrank, Extending Oblivious Transfers Efficiently, in: D. Boneh (Ed.), Advances in Cryptology - CRYPTO 2003, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2003, pp. 145–161. doi:10.1007/978-3-540-45146-4_9.

[132] V. Kolesnikov, R. Kumaresan, Improved OT Extension for Transferring Short Secrets, in: R. Canetti, J. A. Garay (Eds.), Advances in Cryptology – CRYPTO 2013, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2013, pp. 54–70. doi:10.1007/978-3-642-40084-1_4.

[133] F. Armknecht, C. Boyd, C. Carr, K. Gjøsteen, A. Jäschke, C. A. Reuter, M. Strand, A Guide to Fully Homomorphic Encryption, Tech. Rep. 1192 (2015).

[134] R. L. Rivest, A. Shamir, L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, Communications of the ACM 21 (2) (1978) 120–126. doi:10.1145/359340.359342.

[135] T. Elgamal, A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms, IEEE TRANSACTIONS ON INFORMATION THEORY (4) (1985) 4.

[136] P. Paillier, Public-Key Cryptosystems Based on Composite Degree Residuosity Classes, in: J. Stern (Ed.), Advances in Cryptology — EUROCRYPT '99, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 1999, pp. 223–238. doi:10.1007/3-540-48910-X_16.

[137] C. Gentry, A fully homomorphic encryption scheme, Ph.D. thesis, Stanford University, `crypto.stanford.edu/craig` (2009).

[138] Z. Brakerski, Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP, Tech. Rep. 078 (2012).

[139] Z. Brakerski, C. Gentry, V. Vaikuntanathan, Fully Homomorphic Encryption without Bootstrapping, Tech. Rep. 277 (2011).

[140] C. Gentry, Fully homomorphic encryption using ideal lattices, in: Proceedings of the forty-first annual ACM symposium on Theory of computing, STOC '09, Association for Computing Machinery, New York, NY, USA, 2009, pp. 169–178. doi:10.1145/1536414.1536440.

[141] I. Chillotti, M. Joye, D. Ligier, J.-B. Orfila, S. Tap, CONCRETE: Concrete Operates oN Ciphertexts Rapidly by Extending TfhE (2020) 6.

[142] M. Naor, B. Pinkas, Oblivious transfer and polynomial evaluation, in: Proceedings of the thirty-first annual ACM symposium on Theory of Computing, STOC '99, Association for Computing Machinery, New York, NY, USA, 1999, pp. 245–254. doi:10.1145/301250.301312.

[143] M. J. Freedman, Y. Ishai, B. Pinkas, O. Reingold, Keyword Search and Oblivious Pseudorandom Functions, in: J. Kilian (Ed.), Theory of Cryptography, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2005, pp. 303–324. doi:10.1007/978-3-540-30576-7_17.

[144] A. Shamir, R. L. Rivest, L. M. Adleman, Mental Poker, in: D. A. Klarner (Ed.), The Mathematical Gardner, Springer US, Boston, MA, 1981, pp. 37–43. doi:10.1007/978-1-4684-6686-7_5.

[145] H.-G. Rück, On the discrete logarithm in the divisor class group of curves, Mathematics of Computation 68 (226) (1999) 805–807. doi:10.1090/S0025-5718-99-01043-1.

[146] A. Joux, A One Round Protocol for Tripartite Diffie–Hellman, Journal of Cryptology 17 (4) (2004) 263–276. doi:10.1007/s00145-004-0312-y.

[147] D. Boneh, M. Franklin, Identity-Based Encryption from the Weil Pairing (2001) 17.

[148] A. C.-C. Yao, How to generate and exchange secrets, in: 27th Annual Symposium on Foundations of Computer Science (sfcs 1986), 1986, pp. 162–167, iSSN: 0272-5428. doi:10.1109/SFCS.1986.25.

[149] M. Keller, V. Pastro, D. Rotaru, Overdrive: Making SPDZ Great Again, in: J. B. Nielsen, V. Rijmen (Eds.), Advances in Cryptology – EUROCRYPT 2018, Lecture Notes in Computer Science, Springer International Publishing, Cham, 2018, pp. 158–189. doi:10.1007/978-3-319-78372-7_6.

[150] B. Pinkas, T. Schneider, G. Segev, M. Zohner, Phasing: Private Set Intersection Using Permutation-based Hashing, 2015, pp. 515–530.

[151] Y. Aumann, Y. Lindell, Security Against Covert Adversaries: Efficient Protocols for Realistic Adversaries, Tech. Rep. 060 (2007).

[152] B. Pinkas, T. Schneider, M. Zohner, Faster Private Set Intersection based on OT Extension, Tech. Rep. 447 (2014).

[153] C. Dong, L. Chen, Z. Wen, When Private Set Intersection Meets Big Data: An Efficient and Scalable Protocol, Tech. Rep. 515 (2013).

[154] C. Hazay, Y. Lindell, Efficient Protocols for Set Intersection and Pattern Matching with Security Against Malicious and Covert Adversaries, in: R. Canetti (Ed.), Theory of Cryptography, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2008, pp. 155–175. doi:10.1007/978-3-540-78524-8_10.

[155] V. Kolesnikov, R. Kumaresan, M. Rosulek, N. Trieu, Efficient Batched Oblivious PRF with Applications to Private Set Intersection, Tech. Rep. 799 (2016).

[156] R. Cramer, V. Shoup, Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption (2002) 20.

[157] E. De Cristofaro, P. Gasti, G. Tsudik, Fast and Private Computation of Cardinality of Set Intersection and Union, in: J. Pieprzyk, A.-R. Sadeghi, M. Manulis (Eds.), Cryptology and Network Security, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2012, pp. 218–231. doi:10.1007/978-3-642-35404-5_17.

[158] M. Lentz, V. Erdélyi, P. Aditya, E. Shi, P. Druschel, B. Bhattacharjee, {SDDR}: Light-Weight, Secure Mobile Encounters, 2014, pp. 925–940.

[159] U. M. Aïvodji, S. Gambs, M.-J. Huguet, M.-O. Killijian, Meeting points in ridesharing: A privacy-preserving approach, Transportation Research Part C: Emerging Technologies 72 (2016) 239–253. doi:10.1016/j.trc.2016.09.017.

[160] M. I. Mihailescu, S. L. Nita, Ring-Learning with Errors Cryptography, in: Pro Cryptography and Cryptanalysis, Apress, Berkeley, CA, 2021, pp. 343–357. doi:10.1007/978-1-4842-6367-9_14.

[161] Y. Huang, D. Evans, J. Katz, Private Set Intersection: Are Garbled Circuits Better than Custom Protocols? 15.

[162] M. Chase, P. Miao, Private Set Intersection in the Internet Setting from Lightweight Oblivious PRF, in: D. Micciancio, T. Ristenpart (Eds.), Advances in Cryptology – CRYPTO 2020, Vol. 12172, Springer International Publishing, Cham, 2020, pp. 34–63, series Title: Lecture Notes in Computer Science. doi:10.1007/978-3-030-56877-1_2.

[163] A. Abadi, S. Terzis, C. Dong, O-PSI: Delegated Private Set Intersection on Outsourced Datasets, in: H. Federrath, D. Gollmann (Eds.), ICT Systems Security and Privacy Protection, IFIP Advances in Information and Communication Technology, Springer International Publishing, Cham, 2015, pp. 3–17. doi:10.1007/978-3-319-18467-8_1.

[164] A. A. Badawi, Y. Polyakov, K. M. M. Aung, B. Veeravalli, K. Rohloff, Implementation and Performance Evaluation of RNS Variants of the BFV Homomorphic Encryption Scheme, Tech. Rep. 589 (2018).

[165] A. Caudhari, R. Bansode, Securing IoT Devices Generated Data Using Homomorphic Encryption, in: V. E. Balas, V. B. Semwal, A. Khandare, M. Patil (Eds.), Intelligent Computing and Networking, Lecture Notes in Networks and Systems, Springer, Singapore, 2021, pp. 219–226. doi:10.1007/978-981-15-7421-4_20.

[166] W. Ren, X. Tong, J. Du, N. Wang, S. C. Li, G. Min, Z. Zhao, A. K. Bashir, Privacy-preserving using homomorphic encryption in Mobile IoT systems, Computer Communications 165 (2021) 105–111. doi:10.1016/j.comcom.2020.10.022.

[167] F. Benhamouda, G. Couteau, D. Pointcheval, H. Wee, Implicit Zero-Knowledge Arguments and Applications to the Malicious Setting, Tech. Rep. 246 (2015).

[168] Y. Rouselakis, B. Waters, Efficient Statically-Secure Large-Universe Multi-Authority Attribute-Based Encryption, Tech. Rep. 016 (2015).

[169] S. Pohlig, M. Hellman, An improved algorithm for computing logarithms overGF(p)and its cryptographic significance (Corresp.), IEEE Transactions on Information Theory 24 (1) (1978) 106–110, conference Name: IEEE Transactions on Information Theory. doi:10.1109/TIT.1978.1055817.

[170] V. Kolesnikov, R. Kumaresan, M. Rosulek, N. Trieu, Efficient Batched Oblivious PRF with Applications to Private Set Intersection, Tech. Rep. 799 (2016).

[171] N. P. Smart, F. Vercauteren, Fully Homomorphic SIMD Operations, Tech. Rep. 133 (2011).

[172] P. Martins, L. Sousa, On the Evaluation of Multi-core Systems with SIMD Engines for Public-Key Cryptography, in: 2014 International Symposium on Computer Architecture and High Performance Computing Workshop, 2014, pp. 48–53. doi:10.1109/SBAC-PADW.2014.10.

[173] D. Beaver, Efficient Multiparty Protocols Using Circuit Randomization, in: J. Feigenbaum (Ed.), Advances in Cryptology — CRYPTO '91, Vol. 576, Springer Berlin Heidelberg, Berlin, Heidelberg, 1992, pp. 420–432,

series Title: Lecture Notes in Computer Science. doi:10.1007/3-540-46766-1_34.

[174] C. Hazay, Y. Lindell, Sigma Protocols and Efficient Zero-Knowledge, in: Efficient Secure Two-Party Protocols: Techniques and Constructions, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 147–175. doi:10.1007/978-3-642-14303-8_6.

[175] S. Bayer, Practical Zero-Knowledge Protocols Based on the Discrete Logarithm Assumption 212.

[176] T. P. Pedersen, Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing, in: J. Feigenbaum (Ed.), Advances in Cryptology — CRYPTO '91, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 1992, pp. 129–140. doi:10.1007/3-540-46766-1_9.

[177] M. Naor, H. Road, S.-J. Ca, Bit Commitment Using Pseudo-Randomness 10.

[178] G. Asharov, Y. Lindell, T. Schneider, M. Zohner, More Efficient Oblivious Transfer Extensions with Security for Malicious Adversaries, Tech. Rep. 061 (2015).

[179] M. Keller, E. Orsini, P. Scholl, Actively Secure OT Extension with Optimal Overhead, Tech. Rep. 546 (2015).

[180] C. Dong, L. Chen, Z. Wen, When Private Set Intersection Meets Big Data: An Efficient and Scalable Protocol, Tech. Rep. 515 (2013).

[181] Oded Goldreich, Foundations of Cryptography, Vol. 1, Cambridge University Press, 2007.

[182] W. Liu, H.-W. Yin, A novel quantum protocol for private set intersection, International Journal of Theoretical Physics 60 (6) (2021) 2074–2083. doi:10.1007/s10773-021-04824-x.

[183] S. K. Debnath, K. Dey, N. Kundu, T. Choudhury, Feasible private set intersection in quantum domain, Quantum Information Processing 20 (1) (2021) 41. doi:10.1007/s11128-021-02987-4.

[184] B. Liu, O. Ruan, R. Shi, M. Zhang, Quantum private set intersection cardinality based on bloom filter, Scientific Reports 11 (1) (2021) 17332. doi:10.1038/s41598-021-96770-1.

[185] Y. Wang, P. Hu, Q. Xu, Quantum protocols for private set intersection cardinality and union cardinality based on entanglement swapping, International Journal of Theoretical Physics 60 (9) (2021) 3514–3528. doi:10.1007/s10773-021-04925-7.

[186] R.-H. Shi, Quantum bloom filter and its applications, IEEE Transactions on Quantum Engineering 2 (2021) 1–11.

[187] W.-J. Liu, W.-B. Li, H.-B. Wang, An improved quantum private set intersection protocol based on hadamard gates, International Journal of Theoretical Physics 61 (3) (2022) 53. doi:10.1007/s10773-022-05048-3.

[188] R.-H. Shi, Y.-F. Li, Quantum private set intersection cardinality protocol with application to privacy-preserving condition query, IEEE Transactions on Circuits and Systems I: Regular Papers 69 (6) (2022) 2399–2411. doi:10.1109/TCSI.2022.3152591.

[189] S. K. Debnath, V. Srivastava, T. Mohanty, N. Kundu, K. Sakurai, Quantum secure privacy preserving technique to obtain the intersection of two datasets for contact tracing, Journal of Information Security and Applications 66 (2022) 103127. doi:https://doi.org/10.1016/j.jisa.2022.103127.

Appendix A. Symbols

Table A.13: List of symbols used in the paper

Symbol	**Description**
n	Symmetric Set Size
c	Client Set Size
s	Server Set Size
n_1, n_2	Asymmetric Set Sizes
N	Number of parties
B	Number of bins
$\|B\|$	Bins capacity
S_S	Stash Size
κ	Computational Security Parameter
λ	Statistical Security Parameter
l	Length of elements
k	Number of hash functions on BF
m	Bloom Filter length
α	Load factor in a filter
ϵ	False error rate
n_{sub}	Number of elements in set subgroups
t	Threshold
$\|\mathbb{U}\|$	Universe Set Cardinality
$\|I\|$	Intersection Cardinality
C_{len}	Ciphertext length
$\leftarrow_R$	Randomly sampled
Enc	Computation cost of one encryption
Dec	Computation cost of one decryption
Exp	Computation cost of one exponentiation
$Mult$	Computation cost of one multiplication
Add	Computation cost of one addition
Inv	Computation cost of one inversion
$Hash$	Computation cost of one hash function
$Pair$	Computation cost of one pairing
$Fact$	Computation cost of one factorization
Op_G	Computation cost of one group operation