SoK: Outsourced Private Set Intersection

Sophie Hawkes[1][0009-0005-6063-3939]
and Christian Weinert[1][0000-0003-4906-6871]

Department of Information Security
Royal Holloway, University of London, Egham, UK
sophie.hawkes.2022@live.rhul.ac.uk, christian.weinert@rhul.ac.uk

Abstract. Private set intersection (PSI) protocols are an essential privacy-enhancing technology for many real-world use cases, ranging from mobile contact discovery to fraud detection. However, PSI executed directly between input parties can result in unreasonable performance overhead. This motivates the study of outsourced PSI, where clients delegate the heavy PSI operations to an untrusted (cloud) server. In this SoK, we introduce a framework of 12 distinct properties that characterize outsourced PSI protocols based on security, functionality, and efficiency. By analyzing 20 protocols through this framework, we provide a valuable resource and an interactive tool for researchers and practitioners to select the most suitable protocols for their specific requirements. Finally, we discuss research gaps between trends in regular PSI and the current state of outsourced PSI, identifying promising avenues for future work.

Keywords: Private Set Intersection · Cloud Computing · Outsourcing · MPC.

1 Introduction

As modern life becomes increasingly digitized, the amount of data generated, collected, and stored is growing exponentially. This motivates the need for efficient and secure methods of processing and analyzing data to provide important insights. Many organizations are now realizing the potential benefits of sharing data with each other, for example, detecting financial fraud across multiple banks (e.g. [2]). However, strict privacy regulations and concerns about losing business advantage often prevent them from doing this. To address this challenge, the field of *secure multi-party computation (MPC)* has gained momentum, with the aim of allowing the joint computation on private data without revealing any sensitive information.

In particular, *private set intersection (PSI)* is an important cryptographic protocol that enables two (or more) parties, each with a private database, to compute the intersection of their sets without revealing any information about the non-matching items. PSI has many interesting real-world applications, such as privacy-preserving matchmaking [63] and integration of medical data [40]. Without such cryptographic techniques, parties wishing to compute on their

joint data often create a data-sharing agreement (DSA) governing data usage to ensure privacy, security, and legal compliance. In practice, facilitating DSAs can be highly complex and time consuming. Even with a DSA in place, there is always the risk of malicious behaviour, collusion with another party, or a data breach. However, the main barrier preventing the use of PSI is often the high computation and communication costs involved, limiting its practical scalability.

As a general solution for scaling resource-intensive workloads in a cost-effective way, *cloud computing* has become widespread for the delegation of data storage and processing. Thus, organizations with limited resources can potentially outsource also tasks such as PSI to third-party cloud service providers (CSPs). Although outsourcing provides clear benefits in terms of efficiency and scalability, it raises significant concerns about the security and privacy of sensitive data. So, one may ask, is it possible to have the best of both worlds?

Consequently, researchers have proposed novel protocols for a new kind of PSI, known as *outsourced PSI*, in which clients outsource the heavy PSI computations to a third-party server or CSP. Other terms frequently used include *delegated*, *server-aided*, or *cloud-assisted PSI*, making this area of research particularly incomprehensible. Some outsourced PSI protocols completely outsource even the storage of datasets securely to a server (removing the need for a local copy), making them increasingly feasible even for memory-restricted clients.

Contributions. In this SoK, we present a comprehensive overview of the existing literature on outsourced PSI, the first to focus solely on the detailed exposition and discussion of such protocols. Papers were selected based on a focused search using the criteria ("outsourced" OR "server-aided" OR "cloud-assisted") AND "private set intersection", followed by a snowballing approach, then screening out works beyond the scope of this paper (e.g., PSI cardinality). Beginning with Kerschbaum in 2012 [30, 31], numerous different outsourced PSI protocols have been designed, based on a range of underlying cryptographic building blocks, and with a variety of additional properties. With so many subtle variations in design and diverse real-world applications, it is difficult to determine what schemes are best suited to which use cases, hence the need for an in-depth analysis of the key considerations developers/researchers should consider when selecting an outsourced PSI protocol.

Motivated by this, we review 20 outsourced PSI protocols and develop a framework to characterize them according to key properties that define their functionality and security. We identify 12 distinct characteristics, such as whether the server learns the cardinality of the intersection, the verifiability of the server's computation, and the security model. We provide this framework as an interactive selection tool to researchers or practitioners wishing to implement a solution based on outsourced PSI, by provoking reflection on the requirements for their specific use case, and hence forming a criteria for which protocol they should select. Furthermore, we analyze the asymptotic complexity for both communication and computation costs, thereby providing insights into how the protocols scale, e.g., with increasing database size and number of participants. In addition, we synthesize existing benchmarks where available, giving an understanding of

which schemes are best suited to practical deployment. These findings also form the basis for a performance estimator included in our interactive tool to further narrow down the protocol selection.

Finally, we discuss the insights gathered from our findings and identify research gaps to inspire future research directions in this area. With particular reference to current advances in traditional PSI such as unbalanced and fuzzy PSI, we look ahead to how this progress could translate to outsourced settings and the impact these new properties could have on real-world applications.

2 Related Work

2.1 Private Set Intersection

Private set intersection (PSI) enables two or more parties, each with a private dataset, to jointly compute the intersection of their datasets without revealing any information about non-matching elements. The first PSI protocols were introduced by Meadows [39], based on Diffie-Hellman key exchange, and by Freedman et al. [19], based on homomorphic encryption and oblivious transfer (OT). Significant breakthroughs for efficiency include the use of OT extension [7] and batching for oblivious pseudo-random function (OPRF) techniques [32], as well as improvements from oblivious key-value store (OKVS) and vector oblivious linear evaluation (VOLE) approaches [46, 47].

Of the two existing surveys on PSI, Vos et al. [55] focus on collusion-resistant multi-party PSI in the semi-honest model and explicitly exclude server-assisted protocols. Morales et al. [42] do discuss outsourced PSI, but consider a smaller subset of protocols within the much broader scope of their paper, leaving an opportunity for a detailed, comparative analysis of the specific properties and nuances of these protocols and the outsourced setting.

Outsourced PSI-CA. Instead of learning the elements in the intersection of sets, sometimes it suffices to output only the size of the intersection, known as *PSI cardinality* or PSI-CA. Whilst, as we discuss in § 4, unintentionally leaking the intersection cardinality during standard PSI is not desirable, learning just the intersection cardinality is preferable in many real-world applications. For example, in secure mobile contact tracing for monitoring the spread of a disease [16] by comparing devices in close proximity, it is only necessary to output the number of matches to positive records, not the records themselves. Similar to regular PSI, there have been numerous works in the past decade on outsourced or cloud-assisted PSI-CA, such as [14, 26, 52, 54]. These protocols are considered out of scope for this work, as it is not meaningful to directly compare PSI and PSI-CA protocols due to their inherent difference in functionality [14].

Third-Party PSI. Outsourced PSI is known by many names, such as cloud-assisted, server-assisted, and delegated PSI. However, the third-party PSI proposed by Yeo et al. [59] is in a markedly different setting. In these protocols, an inputless third-party receives the intersection result, whilst the two sending parties learn nothing about the other's set, differently to outsourced PSI in which the client(s) learn the result and the third party (server) learns nothing.

Arbiter PSI. The requirement that either both (or all) parties that contribute their sets for the PSI computation receive the result, or else none do, is known as *fairness*. One popular method for achieving this is to use a third-party called an *arbiter*, which may be trusted (e.g., Asokan et al. [8]) or semi-trusted (e.g., Dong et al. [15]), but either way merely steps in to ensure fairness to honest parties after a malicious abort, rather than aiding computation.

2.2 Outsourced Secure Computation

Whilst our work focuses solely on outsourcing PSI, we briefly discuss outsourcing both more general computations and other specific private set operations.

Outsourced MPC. PSI sits within the broader category of secure multi-party computation (MPC), in which arbitrary functions can be computed over the parties' private data, rather than specifically the intersection. In the outsourced MPC scenario [27, 37, 38], typically clients send secret shares of their private inputs to outsourcing servers, which run a two- or multi-party protocol and return shares of the output. Parties then jointly reconstruct the result from these secret shares. It is possible to compute the intersection with this outsourced MPC approach, however, not as efficiently as with a protocol optimized for PSI. Yet the main advantage of general outsourced MPC protocols is their flexibility, enabling the secure computation of arbitrary functions. Recent work by Liu et al. [36] provides a server-aided two-party protocol with malicious security, for heterogeneous networks of smart devices with limited and/or unequal computational resources.

Outsourced Private Database Operations. Beyond the intersection of sets, there exists research into privacy-preserving variants of other set operations in the outsourced setting. Two such operations are outsourced/delegated private set union (PSU) [13, 33, 51], and database join operations on, e.g., user IDs without revealing additional data [12, 41]. More complex SQL-like queries or aggregates – both summary functions (e.g., sum, count, average) and exemplary functions (e.g., min/max, median) – may also be privately computed over outsourced datasets [33, 41]. Asharov et al. [6] design a secure statistical analysis platform with several variants of join and group-by operations.

3 Preliminaries

3.1 Definitions and Notation

In the two-party case, we define two clients, A and B (Alice and Bob), and a cloud server S. Alice and Bob have datasets D_A and D_B, respectively. In the multi-party case, we define n clients $A_1, \ldots, A_{n-1}$, and B and a cloud server S. In both cases w.l.o.g. client B is often the party requesting the intersection computation. The computational security parameter λ determines the level of security against computationally bounded adversaries. A PPT (probabilistic polynomial time) algorithm is defined as having a running time polynomial in λ, and a function ϵ :

$\mathbb{N} \to \mathbb{R}$ is said to be negligible if for sufficiently large λ, $\epsilon(\lambda)$ approaches zero faster than the inverse of any polynomial in λ.

Security Model The security of outsourced PSI protocols typically relies on *simulation-based* security [24]. As discussed further in § 4, adversaries may be either semi-honest (follows the protocol, but tries to learn extra information), or malicious (can arbitrarily deviate from the protocol).

Whilst existing works on outsourced PSI are proven solely in the stand-alone simulation framework (e.g., [3, 18, 63]), this implies that they are secure under sequential composition [34]. Future work is needed to design outsourced PSI protocols with security under the universal composability (UC) framework [11].

3.2 Advanced Encryption and Sharing Primitives

Beyond standard symmetric and asymmetric encryption schemes, including digital signatures [14], there are several more advanced encryption primitives that are involved in the existing outsourced PSI literature.

Homomorphic Encryption. One frequently used building block for PSI is homomorphic encryption (HE). It enables the computation of certain operations, such as addition and/or multiplication, on data encrypted under a key K, producing an encrypted result where the computation carries through to the original data once decrypted. Here the symbol $\star$ represents the operation on the ciphertexts, and the symbol $*$ the operations with the plaintexts: $\mathrm{Dec}_K(\mathrm{Enc}_K(x) \star \mathrm{Enc}_K(y)) = x * y$.

Additive HE. An additive HE (AHE) scheme on ciphertexts $\mathrm{Enc}_K(x)$, $\mathrm{Enc}_K(y)$ must be homomorphic in addition and scalar multiplication, such that multiplication (or more generally some operation $\star$) performed on the ciphertexts corresponds to the sum of the underlying plaintexts: $\mathrm{Dec}_K(\mathrm{Enc}_K(x) \cdot \mathrm{Enc}_K(y)) = x + y$, $\mathrm{Dec}_K(\mathrm{Enc}_K(x)^a) = x \cdot a$. The Paillier public key cryptosystem is a popular AHE scheme used in many outsourced PSI protocols [3–5, 29]. Another example is exponential ElGamal (exElGamal), as used by Miyaji et al. [40].

Multiplicative HE. A multiplicative HE (MHE) scheme must satisfy: $\mathrm{Dec}_K(\mathrm{Enc}_K(x) \cdot \mathrm{Enc}_K(y)) = x \cdot y$, $\mathrm{Dec}_K(\mathrm{Enc}_K(x)^a) = x^a$. For outsourced PSI, both RSA [58] and ElGamal [14] schemes have been used in this way.

Fully HE. Fully HE (FHE) allows any number (subject to noise limitations) of both addition and multiparty operations. The noise limitations can be alleviated by a process called bootstrapping. Although bootstrapping is computationally intensive, fast bootstrapping is possible for FHE over the torus (TFHE), as used in CMPSI [18]. However, somewhat homomorphic encryption (SHE) permits both but only a finite number, e.g., the BGN scheme used by Kerschbaum [31] can take arbitrary additions but only one multiplication.

Secret Sharing. Secret sharing (SS) is a cryptographic method in which a party splits a secret into *shares* and distributes them among a group of participants such that no participant can reconstruct the secret on their own. This could be simple additive SS or Shamir's SS as in PRISM [33].

Threshold Cryptography. In threshold cryptography not all shares are necessarily required to reconstruct the secret, instead there is some minimum or

threshold number of secret shares required to reconstruct the key and decrypt. In particular, parties can agree on a shared key without a dealer [33, 40].

3.3 Efficient Data Structures and Representations

There are different strategies that outsourced PSI protocols use to represent their sets to improve their efficiency.

Polynomial Point-Value Set Representation. This technique was introduced as a novel representation for datasets in PSI by Abadi et al. in their initial outsourced PSI paper [3] and subsequently used in their later works [1, 4, 5, 29]. Suppose we define the universe of all possible elements of a set D_A as a field $R_{D_A} = \mathbb{F}_p$, then we can let all the elements a_i of D_A be all the roots of a polynomial $\rho_{D_A}(x)$ in the polynomial ring $R_{D_A}[x]$. Thus, we get the following representation of the set D_A as a polynomial: $\rho_{D_A}(x) = \prod_{i=1}^{d}(x - a_i) \in R_{D_A}[x],\ d = |D_A|$. Any polynomial $\rho(x)$ of degree strictly less than n can be defined by n point co-ordinate pairs (x_i, y_i) and recovered by polynomial interpolation. This representation reduces multiplication complexity from $O(d^2)$ to $O(d)$ and is achieved by publishing fixed x_i values, hence the point-value form consists of simply a vector of the y_i values.

Hash Tables. Computing over polynomials of high degree is costly. Hash tables in PSI provide a method for reducing computational complexity, by splitting large sets into smaller subsets called *hash bins*. A hash table tells you the *address* of an element, i.e., which bin it maps to [1, 5, 29]. If all clients use the same hash function, then elements in the intersection will be mapped to the same bin, allowing the PSI protocol to perform bin-wise operations, greatly reducing the number of comparisons required.

Bloom Filter. To efficiently check whether an element is a member of a specific set, many outsourced PSI schemes [1,14,31,40,63] utilize Bloom filters (BFs), a probabilistic data structure that allows false positives (at a controllable rate) but never false negatives. A BF is created with an empty m-bit string, set to all zeros, and k hash functions h_i. The bit positions corresponding to the hashes when applying the hash functions to every set element are set to 1. If all of the h_i value positions for a given element x are set to 1, then x is considered a member of the set.

Cuckoo Filter. More recently developed and with proven better performance [48], Cuckoo filters boast constant time lookups even in the worst case and, unlike Bloom filters, support deletion.

Elastic BF. An Elastic BF (EBF) [57] adds a cooperative array of m buckets at the index position of each standard m-bit Bloom filter. The output of the k independent hash functions is separated into two parts (the quotient and remainder of division by m). The remainder becomes the BF index and the quotient is called the *elastic fingerprint* and stored in the corresponding bucket. Besides deletion, EBF supports expansion when buckets become too full.

Oblivious Key Value Store. As introduced by Garimella et al. [20], OKVS is an abstraction of data structures that map key-value pairs (k_i, v_i) [42]. A *key-value store* (KVS) consists of two algorithms: Encode outputs an object S from n

key-value pairs and Decode outputs a value v from input object S and key k. We say a KVS is *oblivious* if the encodings of random values under two different key sets are computationally indistinguishable.

3.4 Permutations and Pseudorandomness

For many cryptographic primitives, permutations and pseudorandomness are key concepts that provide the basis for constructing secure and efficient protocols.

Permutation Function. A permutation function (PF) on a set $D_A = \{a_i\}$ is a bijective mapping from one permutation of D_A to another, rearranging the elements into a new order [33]: $\pi : D_A \mapsto D_A, \quad \pi(D_A) = \{a_{\pi(i)} : a_i \in D_A\}$.

Pseudorandom Number Generator. A PRG (or PRNG) takes an initial value (a *seed*) and generates a deterministic sequence of numbers that is indistinguishable in polynomial time from a uniform distribution [23, 33].

Pseudorandom Function. A pseudorandom function (PRF) takes a specific bit-string input and outputs a value in the output space $\mathcal{U}$ that is deterministic given a key K of length λ, yet indistinguishable from random to any party that does not know the secret key $K \in \{0,1\}^\lambda$ [1]: $F_K : K \times \{0,1\}^* \mapsto \mathcal{U}$.

Pseudorandom Permutation. A pseudorandom permutation (PRP) is a PRF where the function is a permutation, hence bijective. It takes a key K of length λ and permutes the elements of a set D_A in a deterministic and pseudorandom manner [1]: $\pi_K : K \times D_A \mapsto D_A, \;\; \pi_K(D_A) = \{a_{\pi_K(i)} : a_i \in D_A\}$.

3.5 Oblivious Protocols

Obliviousness in protocols prevents parties from revealing their inputs to each other, whilst helping to enable secure computation and data sharing.

Oblivious Transfer. Oblivious transfer (OT) refers to a protocol where the sender Alice wishes to send a message to the receiver Bob, without learning what message he receives. In Rabin's original OT protocol [45], Bob either receives the message m or no message $\perp$, and Alice does not know which. In 1-out-of-2 OT [17], Alice has two messages, m_0 and m_1, and Bob receives only one message m_b ($b \in \{0,1\}$), with Alice remaining oblivious to his choice. In practice, *OT extension* is required to reduce the computational overhead by leveraging a few expensive base OTs to generate a large number of cheaper ones [48].

Oblivious Polynomial Evaluation. Oblivious polynomial evaluation (OPE) is the evaluation of a polynomial ρ (known by the sender) on an input value x (known by the receiver). The sender learns nothing, and the receiver learns $\rho(x)$ [42].

Oblivious Pseudorandom Functions. An oblivious pseudorandom function (OPRF) is a protocol where Bob receives the result $\text{OPRF}_K(y_1), \ldots, \text{OPRF}_K(y_n)$ of his inputs under Alice's key K, yet Bob learns nothing about K and Alice learns nothing about Bob's inputs [48]: $F_{\text{OPRF}} : (K, y) \mapsto (\perp, F_K(y))$.

Proxy Re-Encryption. Proxy re-encryption (PRE) allows comparing the plaintexts of ciphertexts encrypted under different keys. The idea is that Alice and Bob generate a "re-key", which they send to a third-party or proxy.

This re-key enables the proxy to transform Alice's ciphertext into a ciphertext under Bob's key, allowing him to decrypt "on behalf" of her, with the proxy remaining oblivious. In the server setting, both parties may send intermediary rekeyed ciphertexts to the server to test whether their plaintexts match [64]. PRE can be based on asymmetric or symmetric encryption schemes. A symmetric-key PRE (SKPRE) can be constructed using a key homomorphic PRF [61].

Zero-Knowledge Proofs. A zero-knowledge proof (ZKP) is a cryptographic method by which one party (prover P) can prove to another (verifier V) that some statement is true, without revealing any further information [23]. A ZKP of knowledge (ZKPoK) can specifically prove knowledge of some value [14].

4 Properties

In this section, we define the key characteristics observed across existing outsourced PSI protocols. Particularly when selecting a PSI protocol for a specific real-world application, there are nuances to carefully consider which properties are most important. Through comprehensive analysis of a wide range of protocols, summarized in Tab. 1, we derived the following properties.

(Intersection) Cardinality Hiding. One of the main objectives of PSI is to ensure that the server remains unaware of any information regarding the clients' datasets or their intersection. The size or *cardinality* of the intersection may reveal sensitive information if learned by the server, for example, simply knowing the cardinality is greater than zero discloses the existence of an intersection between datasets. If a protocol does not reveal the size of the intersection to the server or any unauthorized third party, we say it is *cardinality hiding*.

Input Cardinality Leakage. Similarly, in some applications it may be unacceptable for the server to learn the exact size of parties' private datasets, known as *input cardinality leakage*. In most applications, it is unlikely that all of the clients' datasets will have exactly the same number of elements. Therefore, an important design feature can be that clients' datasets can have different cardinalities, independent or even unknown from each other [40]. A trivial technique would be padding all set lengths to equal some upper bound, simultaneously providing an effective means of hiding the true size of parties' datasets and avoiding input cardinality leakage. However, in the unbalanced PSI setting, where one dataset is much smaller than the other, it may also be important that the computation and communication complexities avoid the inefficiency of trivially padding the smaller set to the larger set size.

Repeatability. When clients outsource the computation of PSI, they need to transfer their encrypted dataset to the server. If the client cannot reuse the outsourced data with a different client (or group of clients) without compromising security, and has to re-encrypt the data locally for each intersection, it is referred to as *one-off delegation*. This approach requires the client to either constantly download or maintain a local copy of their data, limiting their ability to fully benefit from outsourcing to reduce storage and communication costs. As a result, achieving *repeated delegation* becomes desirable, as it allows clients

to outsource their dataset once and then perform multiple PSI computations, sometimes, but not always (e.g., Ruan and Zeng [48]), without a local copy.

Non-interactive Setup. In some settings, clients must communicate with each other during the setup phase, e.g., to establish a shared secret key. However, in the outsourced setting it is often the case that clients may only interact with each other via the server, perhaps asynchronously. A *non-interactive* setup phase is where clients are able to prepare and outsource their datasets to the server without any knowledge or interaction with the other clients.

Authorization. In a non-interactive protocol, clients may wish to have some control over who can perform intersections on their outsourced data, hence the desire for *authorization*. A protocol requires authorization if the client requesting the intersection must obtain permission from the other clients in order to receive the result. However, this stands in contrast to the ability to compute intersections with offline parties, which may be preferred in scenarios when a client perhaps has some trust in the other clients of the server and is happy to provide their outsourced data with no restrictions.

Multiple Clients. Another key property of PSI protocols is the number of clients whose intersection may be computed. In *two-party (2P-)PSI*, there are two clients, A and B. Client B initiates a request for the server to compute its intersection with A, i.e., $f(A, B) = A \cap B$. In *multi-party (MP-)PSI*, there are n clients, $A_1, ..., A_{n-1}$ and B. Client B requests its intersection with the sets of all the other clients $A_1, ..., A_{n-1}$, i.e., $f(A_1, ..., A_{n-1}, B) = \bigcap_1^{m-1} A_1, ..., A_{n-1} \cap B$.

Verifiability. The cloud server may not always be trusted not to tamper with the datasets or computation results, and therefore the clients want a method to *verify* the integrity of the data and the correctness of the result, with some proof that the server has behaved honestly and not deviated from the protocol.

Updatability. Many PSI protocols are designed for handling static sets, with a prohibitive cost of updating securely, despite many real-world environments that require frequent updates, such as stock market analysis and dynamic data systems. A protocol is considered *efficiently updatable* if it permits updates at sub-linear cost, i.e. eliminating the need to download the entire set.

Constant Client-side Storage. By *constant client-side storage*, we refer to the ability to fully outsource client datasets to storage in the cloud, to the point where no local copy of the dataset is required (for the computation and result), only some constant secret material.

Number of Servers. Although many of the existing outsourced PSI protocols consider the simple case of one cloud server, others are designed to share the delegated computation of the intersection between multiple cloud servers.

Adversary Model. The security of outsourced PSI protocols typically relies on *simulation-based* security [24]. We say a scheme has *semi-honest security* if it is secure against an adversary that follows the protocol exactly, but keeps a record of all messages they send and receive to try and learn extra information. In contrast, in the *malicious security* model, the adversary may deviate from the protocol, actively trying to compromise security.

Table 1. Synthesis of 20 papers across our framework of characteristics. **Properties:** CH = (intersection) cardinality hiding, IC = input cardinality leakage (d=upper bound (*implicit), D=size of attribute domain), R = repeatability, NI = non-interactive setup, A = authorized (—=interactive), 2P = two-party, MP = multi-party, V = verifiability, U = updatability, CS = constant client-side storage, #S = number of cloud servers, AM = adversary model (○=any party semi-honest, ◐=malicious client, semi-honest (^{R}=rational) server, ◑=semi-honest client, malicious server, ●=any party malicious), CR = collusion resistance (○=server and clients may not collude, ●=server may collude with some (but not all) client(s)).

Paper	Properties												
	CH	IC	R	NI	A	2P	MP	V	U	CS	#S	AM	CR
Kerschbaum (SAC'12) [30]	✓	$\|A\|$,$\|B\|$			—	✓	✓				1	○	●
Kerschbaum (ASIACCS'12) [31]	✓	$\|A\|$,d		✓		✓					1	○	○
Kamara et al. (FC'14) [28]	✓	$\|A_i\|$,$\|B\|$			—	✓	✓				1	●	○
Liu et al. (IC2E'14) *ESIP* [35]		$\|A\|$,$\|B\|$	✓	✓	✓	✓	✓			✓	1	○	○
Abadi et al. (SEC'15) *O-PSI* [3]	✓	$\|A_i\| = \|B\|$	✓	✓	✓	✓	✓			✓	1	○	○
Zheng and Xu (IC2E'15) *VDSI* [64]		$\|A\|$,$\|B\|$	✓	✓	✓	✓	✓	✓		✓	≥ 1	◐	○
Abadi et al. (FC'16) *VD-PSI* [4]	✓	d	✓	✓	✓	✓	✓	✓		✓	1	◑	○
Zhang et al. (Inf. Sci.'17) [61]		$\|A_i\|$,$\|B\|$		✓	✓	✓	✓			✓	2	$◐^R$	○
Miyaji et al. (J. Medical Syst.'17) [40]	✓	d*			—	✓	✓				1	○	○
Yang et al. (CCPE'18) [58]	✓	$\|A\| = \|B\|$	✓	✓	✓	✓	✓	✓		✓	1	○	○
Zhao and Chow (WPES'18) [63]	✓	$\|A\|$,$\|B\|$			—	✓	✓				1	○	○
Abadi et al. (TDSC'19) *EO-PSI* [5]	✓	d	✓	✓	✓	✓	✓	✓		✓	1	○	○
Kavousi et al. (ICEE'20) [29]	✓	d	✓	✓	✓	✓	✓			✓	1	○	○
Debnath et al. (DSC'21) *OPSI* [14]	✓	$\|A_i\|$,$\|B\|$	✓	✓	✓	✓		✓		✓	1	●	○
Li et al. (SIGMOD'21) *PRISM* [33]	✓	D	✓	✓			✓	✓		✓	≥ 2	◑	○
Abadi et al. (FC'22) *Feather* [1]	✓	d	✓	✓	✓	✓	✓		✓	✓	1	○	○
Ruan and Zeng (ICCIR'22) [48]	✓	$\|A\| = \|B\|$	✓	✓		✓	✓				1	○	○
Fan et al. (Mathematics'23) *CMPSI* [18]	✓	$\|A\|$,$\|B\|$	✓	✓		✓	✓				1	○	●
Zhang et al. (WAIM'24) [62]	✓	d	✓	✓	✓	✓	✓		✓	✓	1	○	○
Wei and Hu (ePrint'25) *D-PSI* [56]	✓	d*			—	✓	✓				1	○	○

Collusion-Resistance. It is very common in the existing outsourced PSI literature to include a non-collusion assumption, that is an assumption that the server does not collude with any of the clients. However, some protocols display so-called *collusion resistance* between clients, e.g., an honest majority of clients is required, or all but one client may collude without breaking security.

5 Protocols

Whilst all surveyed protocols have their differences in the way they execute outsourced PSI, in general they follow this structure:

1. **Outsourcing stage:** Security parameters and keys are generated, then clients Alice and Bob send an encrypted version of their sets to the server.
2. **Request:** Bob sends a request (either to the server or directly to Alice) to securely compute the intersection with Alice.

3. **PSI computation:** The server (obliviously) computes the intersection and sends the encoded result to Bob (and sometimes also Alice).

4. **Result retrieval:** Recipient(s) use their secret information to decode the result and reveal the intersection.

Much of this diversity between schemes stems from the choice of building blocks, hence we introduce the protocols below as categorized according to their underlying cryptographic primitives (e.g. homomorphic encryption, pseudorandom functions, secret sharing). We further evaluate the protocols against our framework of characteristics as defined above, and highlight key properties, including security and efficiency.

5.1 Public Key Encryption (PKE)

The first works on outsourcing intersection computations to untrusted cloud servers were two papers by Kerschbaum in 2012 [30, 31]. In the former, bitwise XOR operations are used to construct protocols under two scenarios: both a public output setting (where the server learns the intersection and sends this to the parties in plaintext) via keyed hash functions, and an oblivious service provider setting (where the server learns neither the intersection nor its size) via RSA encryption. Notably, this scheme offers some collusion resistance, in that the server and one client may collude without compromising security. However, the setup does not support more than two clients, and is not non-interactive, since Alice and Bob must jointly choose the RSA modulus.

In 2014, Liu et al. introduced the Encrypted Set Intersection Protocol (ESIP) for Outsourced Datasets [35], which has a non-interactive setup and authorization functionality, but does not hide the cardinality of the intersection from the server. Clients independently outsource their sets by hashing their elements, adding randomness and encrypting under symmetric keys. After authorization of a query, clients exchange re-randomization information encrypted under public keys via the server, enabling the server to privately identify the underlying matches, which it then sends to the requester for decryption. While the non-interactive setup means no prior knowledge of other clients is required to outsource data, clients must compute and send some new values to the server whenever an intersection is requested, increasing communication costs. Nevertheless, the authorization within the request ensures that clients have control over who can perform PSI with their outsourced data. However, the scheme is not fully private because it is not cardinality hiding, so the server learns the size of the intersection. Furthermore, there have been several issues discovered that undermine the security of this scheme, such as one-time pad re-use [5], and the possibility of revealing information about other intersections through repeated use of the same set, i.e., if $A \cap B$ and $A \cap C$ are computed with the same A, then the server can learn information about $B \cap C$ without their permission [3].

5.2 Homomorphic Encryption

Kerschbaum's latter 2012 work [31] introduced the first outsourced PSI based on HE, which has become the most popular building block from which to construct outsourced PSI schemes. There are schemes based on all three types of HE introduced in § 3.2, as discussed in the following.

AHE. Research led by Abadi et al. has explored multiple protocols for outsourced PSI using AHE, beginning in with *O-PSI* [3]. This protocol uses Paillier AHE and introduces a novel way of representing the clients' datasets (of maximum size d) as polynomials in point-value form. During the setup phase, the server publishes parameters including a PRF and a vector $\tilde{v}$ of $2d+1$ random points. Alice and Bob independently encode their datasets as polynomials and evaluate these at each point in $\tilde{v}$, resulting in vectors they blind under a random secret key for the PRF and send to the server. During the online phase, Bob requests to perform PSI with Alice by sending her his random blinding values encrypted with his public key. She authorizes the request by applying the inverses of her blinding values to Bob's encrypted values, and sending this with their IDs to the server, which uses HE to "switch" the blinding factors and compute the intersection. The result is sent to Bob, who decrypts with his private key and interpolates the resulting polynomial to learn the intersection values. Despite offering additional dataset integrity validation via homomorphic tags, the public key and HE operations of the overall approach have a "major impact on performance" [5]. Moreover, Oliaee et al. [43] show that O-PSI is vulnerable to man-in-the-middle (MITM) attacks.

The subsequent paper by Abadi et al. [4] follows on with Verifiable Delegated PSI on outsourced private datasets (VD-PSI), efficiently adding verifiability of correctness for the intersection result computation, with the extra cost only linear to the intersection size. Like O-PSI [3], clients send to the server vectors encoding their sets in point-value polynomial representation form, blinded with keyed pseudo-random values, then publish their Paillier public keys. Now, the requesting party Bob chooses a random value $\beta \in \mathbb{F}_p^*$, which he and Alice will both add to their sets in such a way that β will be in the intersection only if it has been computed honestly. This allows for verification with such a low overhead, as the only computation necessary is checking if β is included in the intersection. VD-PSI can be reused for unlimited computations, offering significant advantage over non-repeatable protocols that necessitate frequently re-encrypting and uploading the data.

Zhao and Chow [63] took a different approach, proposing privacy-preserving matchmaking using AHE-based OPE and BFs, in which most of the public-key operations are outsourced to a semi-honest server. Interestingly, they design a novel encrypted PSI cardinality (ePSI-CA) scheme, inspired by existential PSI (X-PSI), that underlies and unifies both their above-threshold and below-threshold PSI functionalities (called $t^{\geq}$-PSI and $t^{\leq}$-PSI). If we set $t = 0$ in $t^{\geq}$-PSI, then we get a general PSI scheme [63], however the communication costs are very large.

MHE. The first MHE-based scheme, and indeed the first HE scheme, was proposed by Kerschbaum [31]. It provides an initial semi-honest PSI protocol using Bloom filters, encrypted bit-wise using Goldwasser-Micali (GM) multiplicative homomorphic encryption [25] (based on quadratic residuosity), and the Sander-Young-Yung (SYY) technique [50] to achieve unbounded fan-in logical-AND on multiple encrypted bits in XOR-homomorphic schemes like GM. Thus, it is helpful for checking the presence of elements in a Bloom Filter. The PSI protocol is then extended to the malicious setting via client set certification. Finally there is a protocol for outsourced PSI in the semi-honest setting, incorporating a novel combination of the SYY technique and a simplification of the Boneh-Goh-Nissim (BGN) MHE scheme [10] modelled after the quadratic residues of GM. However, a local copy of the set must be kept in order to produce the result, hence this could be considered as not true outsourcing [3].

An improved variant of O-PSI [3] is given by Yang et al. [58] that contributes a large improvement in efficiency due to the use of RSA (multiplicatively homomorphic) instead of Paillier (additively homomorphic). Another advantage of their scheme is that it avoids the requirement that Bob must communicate with Alice directly to send his request, which is a barrier to true outsourcing in which all interactions between clients are delegated to the cloud.

Debnath et al. [14] devised an OPSI scheme using ElGamal MHE, ElGamal digital signatures, and Bloom filters. Their scheme achieves malicious security via ZKPs of discrete logarithms and they also extend their approach to a cardinality-only scheme OPSI-CA, through random permutation. They satisfy many attractive properties, like non-interactive setup, repeatability, and verifiable security against a malicious adversary. On the other hand, it should be noted that the design is only for two clients, a trusted third party is required to initialize identities and parameters, and particularly that the cardinalities of the private sets are made public, which might not be suitable for all use cases.

FHE. Fan et al. introduce CMPSI (Cloud-assisted Multi-party PSI) [18] using FHE over the torus (TFHE). They first give secure computation protocols for AND, OR and XNOR gates between multi-key LWE samples, and a secure comparison protocol (SCP) to test for equality between encrypted input data vectors. Similar to other schemes [14, 33, 64], public parameters are generated and shared by a Parameter Generation Centre (PGC), from which the clients generate a secret key and corresponding public key, bootstrapping key and key switching key. Both sending and receiving clients send their datasets (containing vectors of k Boolean values) encrypted under their secret key to the server, along with their public key set. The server performs SCP and secure OR on the ciphertexts, resulting in an encrypted Boolean vector of whether each element in the receiver's set was in the sender's, which is sent to the receiver who decrypts with the joint key. Unlike most schemes, CMPSI is secure even if the cloud server colludes with some of the clients since all keys are needed for decryption. Furthermore, it is secure against passive man-in-the-middle (MITM) attacks and has lower communication cost than Kamara et al. [28].

Similarly, Wei and Hu [56] use FHE, a privacy-preserving AND sub-protocol, and joint decryption. However, in this case each client set is represented by a Bloom filter where each encoded bit is encrypted under a jointly generated public key (a multi-key variant of BFV) and sent to the cloud server. The intersection of all Bloom filters is computed via a bitwise AND using only homomorphic additions. However, it requires a privacy-preserving zero-test to guarantee security against leakage of "differential" information about the intersection. The protocol does not meet its claim of repeatable and fully delegated outsourcing, since the jointly generated key material prevents repeated intersection computation with new clients, and at least the querier must maintain a local copy of their dataset.

5.3 Pseudorandom Protocols (PRP and/or PRF)

Kamara et al. [28] present four different server-aided PSI protocols based on PRPs to deliver varying security properties: semi-honest security, malicious (via dummy sets) and covert (via careful parameter setting) security, fairness, and cardinality hiding. These protocols all allow clients to jointly generate a key k under which they permute their sets, hence the setup is interactive. The fairness property is achieved by simply adding another layer of PRP, proving that the intersection computed and committed to by the server matches the intersection of clients' one-layered PRP values (revealed by applying the second PRP key). For the cardinality size-hiding variant, Alice and Bob share one keyed PRP, and Alice and the server share another; Bob obtains his set under both PRPs via the server and receives Alice's too and computes the intersection himself. Importantly, the computational costs scale linearly, enabling efficient PSI on sets with billions of entries. Consequently, this scheme has been used as a point of comparison for implementation performance in several later works [4, 18, 33, 44].

Abadi et al. improve upon their own O-PSI protocol [3] above by introducing a more efficient version called EO-PSI in [5]. This scheme is 10-100 times faster than O-PSI because it avoids any PKE or exponentiation by relying on PRFs, and it also utilizes hash tables, thus breaking down the polynomial into multiple polynomials of smaller degree. The protocol is extended to multiple clients, but does not scale well, with the polynomial evaluations becoming the limiting factor as the number of parties increases [1]. The need for secure channels between parties goes unmentioned in this work, hence Kavousi et al. [29] discuss passive attacks against EO-PSI and consequently amend the protocol in such a way as to remove the requirement that messages sent between parties are secret, furthermore reducing the computational complexity in the process.

Returning to another work by Abadi et al., they later proposed Feather [1], significantly improving the efficiency amongst multiple clients. Crucially, Feather was the first scheme allowing outsourced sets to be efficiently updatable, with 1000 times faster updates than EO-PSI [5] (for 2^{20} elements). Set elements are stored in bins within hash tables, hence clients need only to update, re-encode and send bins, instead of the entire set, thus the update complexity depends solely on the number of bins b, with $O(b)$ communication and $O(b^2)$ computation

overhead. However, there is only semi-honest security, secure communication channels are assumed, and some leakage to the cloud (e.g., about query and access patterns) is permitted for the sake of efficiency. Further efficiency gains come from clients evaluating random polynomials locally instead of in the cloud, Horner's method for polynomial evaluation, and replacing the padding in EO-PSI [5] with blinded Bloom filters. Feather is shown to support up to 100 clients with 2^{20} set elements, resulting in twice as fast delegated PSI and 26 times faster cloud runtime than EO-PSI [5].

5.4 Proxy Re-Encryption (PRE)

Verifiable Delegated Set Intersection (VDSI) by Zheng and Xu [64] was the first outsourced PSI protocol to be verifiable, in the sense that the server generates a proof that it has executed the PSI operation correctly, which can be verified by the receiving client(s) before decrypting the result. The scheme is based on bilinear maps, where the server is able to compare ciphertexts encrypted under different keys by using proxy re-encryption: clients Alice and Bob send re-keys to the server as part of their authentication request. A novel aspect is the creation of a cryptographic multi-accumulator, which allows to privately test the set membership of $D_A \cap D_B \subseteq D_A (\subseteq D_B)$ in one go. However, similarly to ESIP [35], VDSI is not cardinality-hiding, and is vulnerable to the same leakage in repeated use of outsourced data to compute intersection with different sets. In addition, VDSI is subject to plaintext guess attacks from the server, requires a trusted third party (TTP) to initialize public parameters, and requires an authenticated private communication channel between each client and the cloud.

Zhang et al. [61] introduced a reputation-based two-server aided protocol, which combines a symmetric key proxy re-encryption (SKPRE) scheme, based on the learning with errors (LWE) problem, with a social game that assigns clients a reputation score based on previous behavior and penalizes them for defecting or colluding. If we assume that all parties are rational and choose the strategy that optimizes their reputation score, then it is always better for them to choose non-collusion, even if they do not receive the intersection, which avoids the typical PSI fairness impossibility results, allowing for collusion-free computation with complete fairness. Along with non-interactive setup and support for multiple clients, all clients have different keys, so not all information is compromised if one party's key is. Furthermore, there is no user-to-user interaction or costly public key infrastructure (PKI) operations.

5.5 Secret Sharing

Miyaji et al. [40] design an outsourced multi-party PSI protocol for integrating medical data using secret sharing. Their two main criteria are that dataset cardinalities can be independent, and that the client-side computational complexity is independent of the number of clients. To start, the clients initialize exponential ElGamal pairs and publish their public keys, and compute the (n, n)-threshold public key, then send the Bloom filter of their set encrypted under

this key to the server. The server randomizes and broadcasts the result back to the clients, who jointly threshold decrypt the result and obtain the intersection by a Bloom filter check. Hence, the protocol is not repeatable for computing intersections with a new set of parties, and a local copy must be kept. They demonstrate practical performance of a maximum 450 seconds for up to 16,384 elements and 16 parties, at different bit security levels; whilst clearly infeasible for many real-world applications with millions or even billions of set elements [49], it may indeed be well-suited to its intended use on patient data between hospitals/clinics.

PRISM by Li et al. [33] extends secret sharing to not just PSI over multiple public outsourcing servers, but also PSU and aggregation functions, including PSI sum, average, maximum, median and cardinality, as well as PSI over multiple attributes. PRISM utilizes both additive and Shamir's SS, cyclic groups, permutation functions and a PRG. There is security for both semi-honest and malicious servers, including hiding access patterns from servers but they assume that no servers communicate with each other to collude, and a trusted initiator is required as in [14, 64]. PRISM scales even better than Kamara et al. [28] for two clients and a billion values, and also scales nicely for up to 50 clients and 20 million values.

5.6 Oblivious Primitives

Ruan and Zeng [48] introduced an offline outsourced PSI based on OT, OPRFs, and Cuckoo filters to achieve computation and communication complexity linear in the set size. However, the offline nature of the scheme means that there is no authentication stage, and there is no concrete performance evaluation to compare it to its contemporaries. Moreover, as with some of the earlier protocols, clients must keep a local copy of their dataset to match the result of their Cuckoo filter look-ups to intersection elements, hence the outsourcing is not complete.

More recently, Zhang et al. [62] proposed a new updatable outsourced multi-party PSI scheme, joining Feather [1] as the only other efficiently updatable protocol (similarly achieved via hash tables), but is has up to 30% faster runtimes in comparison (for multiple parties). They achieve this speed-up by avoiding the large number of polynomial computations that slow down Feather's intersection computation, instead using an EBF and OKVS structure. The protocol has semi-honest security and a non-collusion assumption, but one of the main limitations is the communication complexity since the cost goes from 0.2 MB for two clients with 2^{10} elements to over 280 MB for 2^{20} elements.

6 Performance

Performance of outsourced PSI protocols must be considered from two perspectives – asymptotic complexity and concrete benchmarks. Note that our analysis of practical runtimes relies on self-reported benchmark data, from non-uniform implementations with differing software, hardware and network environments.

Table 2. Taxonomy of protocols categorized by underlying cryptographic primitive, along with their asymptotic complexities; u=size of dataset A, v=size of dataset B, d=maximum size of datasets, λ=security parameter, m=size of intersection, n=number of clients, k=number of hashes, b=hash bin capacity, w=EBF bucket capacity.

Type	Paper	Computation Complexity	Communication Complexity
PKE	Kerschbaum (SAC'12) [30]	$O(uv) \approx O(d^2)$	$O(u+v)+O(uv) \approx O(d^2)$
	Liu et al. (IC2E'14) *ESIP* [35]	$O(d^2)$	$O(d)$
AHE	Abadi et al. (SEC'15) *O-PSI* [3]	$O(d^2)$	$O(d)$
	Abadi et al. (FC'16) *VD-PSI* [4]	$O(d)$	$O(d)$
	Zhao and Chow (WPES'18) [63]	$O(\log(\lambda)(u+v))$	$O(\log(\lambda)(u+v))$
MHE	Kerschbaum (ASIACCS'12) [31]	$O(v(k\lambda+k+2\lambda))$	$O(ku+\lambda v)$
	Yang et al. (CCPE'18) [58]	$O(d)$	$O(d)$
	Debnath et al. (DSC'21) *OPSI* [14]	$O(d)$	$O(d)$
FHE	Fan et al. (Mathematics'23) *CMPSI* [18]	$O(d)$	$O(d)$
	Wei and Hu (ePrint'25) *D-PSI* [56]	$O(nd\log(n))$	$O(nd)$
PRE	Zheng and Xu (IC2E'15) *VDSI* [64]	$O(d)$	$O(m)$
	Zhang et al. (Inf. Sci.'17) [61]	$O(2(u+v+m))$	$O(2(u+v)+3m)$
SS	Miyaji et al. (J. Medical Syst.'17) [40]	$O(u_i+nu)$	$O(u+n+nu)$
	Li et al. (SIGMOD'21) *PRISM* [33]	$O(nd)$	$O(nd)$
PRF	Abadi et al. (TDSC'19) *EO-PSI* [5]	$O(d)$	$O(d)$
	Kavousi et al. (ICEE'20) [29]	$O(d)$	$O(d)$
PRP	Kamara et al. (FC'14) [28]	$O(d)$	$O(d)$
	Abadi et al. (FC'22) *Feather* [1]	$O(nd)$	$O((n-1)d)$
	Zhang et al. (WAIM'24) [62]	$O(nd)$	$O(nd(bw))$
OPRF	Ruan and Zeng (ICCIR'22) [48]	$O(d)$	$O(d)$

Consequently, it is vital to bear in mind the limitations of this approach, as discussed further in sections 6.2 and 6.3.

6.1 Asymptotic Complexity

We summarize the asymptotic communication and computation complexity of all 20 outsourced PSI protocols in Tab. 2.

Communication Complexity. The communication complexity is the total amount of data sent between parties in a protocol. Several outsourced PSI protocols achieve linear communication complexity, proportional to the size of the clients' datasets. Where protocols experience quadratic communication complexity, e.g., in Kerschbaum [30], this poses challenges when dealing with large datasets or a vast amount of clients. On the other hand, VDSI [64] achieves even smaller communication than linear to dataset size, instead being proportional to the size of the intersection. For multi-party outsourced PSI, the number of clients can also influence the communication complexity, for example, Feather [1] is linear to both dataset cardinality and number of clients. However, other multi-party protocols may even be independent of the number of participants.

Computation Complexity. Computation complexity refers to the total computational resources required by the clients and server to execute a protocol. Similarly to communication, two of the main factors affecting the computa-

tion complexity are the size of the input datasets, and the number of clients in the multi-party case. This can be seen in Tab. 2, where the dependence on these variables is clear. Evidently, the type of primitive(s) used in an outsourced PSI protocol significantly influences the computation complexity. Earlier protocols depending on public-key operations suffered from quadratic computational complexity, whereas using only symmetric key operations helped protocols to achieve linear computation. For example, while O-PSI [3] introduced the point-value polynomial representation, crucial for reducing computational cost, its use of Paillier homomorphic encryption is expensive. This compared to the subsequent EO-PSI [5] that was designed to avoid public-key operations, hence requires only modular additions and multiplications, and gains further computational efficiency via utilizing hash tables. Beyond the adoption of hash tables, other improvements in set representation have reduced the computational overheads, from the initial use of Bloom filters to advancements like Cuckoo and EBFs. In general, homomorphic encryption (particularly FHE) brings significant computational costs, whereas OPRF-based PSI protocols are known for their efficiency.

Verification/Malicious Security Overhead. To gain additional properties such as verification or malicious security, it is usually necessary to introduce some overhead. The first verifiable outsourced protocol VDSI [64] uses a novel multi-accumulator scheme based on bilinear maps on cyclic group $|G|$ of order p, which contributes $5p$ output size and computational overhead linear to the size of both clients' dataset sizes and the size of the intersection. In VD-PSI [4], the computation complexity of verification is linear to only the intersection size $O(m)$, compared to the malicious security in Kamara et al. [28], which is linear to the total number of inputs, i.e., the sum of dataset sizes. Notably, even in the multi-party context, the verification overhead of VD-PSI does not depend on the number of clients. The security against collusion in Zhang et al. [61] comes from the reputation system, where parties rate each other then compute new trust values. PRISM [33] enables PSI result verification by each client secret sharing the complement of each element with a second server, hence twice the computation and communication is needed server-side, in addition to the client-side preparation of complement values and verification calculations, which are both linear to the number of elements. Both ElGamal signatures and ZKPs are used for verification and to obtain malicious security in Debnath et al. [14], however, they still achieve linear complexity.

Update Complexity. Efficient updatability is a major limitation for many existing protocols, as most require clients to download their entire dataset, perform local updates, and then re-upload the entire dataset. This incurs high communication and computation costs that scale linearly with the set size. Feather [1] proposed the first efficiently updatable protocol via its use of hash tables, since the client must only retrieve and reupload one bin. Thus, Feather has an update complexity of $O(b)$ for communication and $O(b^2)$ for computation, where b is the bin size. Another efficiently updatable protocol is Zhang et al. [62], where

like Feather its update complexity is independent of the set size, but rather depends on the selection of two parameters – bin size and EBF bucket capacity.

6.2 Practical Performance

Implementations. Whilst 14 out of 20 papers indicated to have implemented the proposed protocols, only 4 papers have open-source implementations available: Kamara et al. [28], O-PSI [3], EO-PSI [5], and Feather [1]; and 1 pending [56]. Despite reaching out to the remaining 9 authors, we were not able to gain access to additional implementations. Therefore, for these works, we have to rely on self-reported results. In terms of programming languages, C++ was by far the most frequently chosen, with 8 out of the 14 protocols [1,3,5,18,28,31,40,63], using libraries such as Crypto++, Paillier, PBC, NTL, GMP, and MKTFHE. There were also 2 implementations in Java [30, 64], 1 Rust [56], and 3 did not specify [33, 58, 62].

Memory Usage. As discussed in Morales et al. [42], memory consumption tends to be poorly addressed in the PSI literature. This is also the case with outsourced PSI, perhaps as one of its main goals is to shift the burden to the cloud, which is assumed to have highly capable resources. The RAM required on the client side depends largely on the choice of data structure used to encode the dataset, and for probabilistic options like Bloom filters, the chosen error rate is also a factor.

Runtime. The lack of open-source implementations unfortunately prevented us from conducting benchmarks in a unified environment for a fair comparison. Therefore, in Tab. 3, we synthesize the available self-reported runtimes across varying client numbers and dataset sizes, for which we refer to Tab. 13 of [33], Tab. 6 of [1], and Tabs. 2-4 of [62]. However, these tests were all run on different client hardware and cloud server environments – obviously, the technology available in 2014 will have been very different in capacity to that available 10 years later in 2024. The implementation of Feather for comparison in Zhang et al. [62] is already over 7 times faster, only two years later. Moreover, some papers only include graphical representations of their runtime data, making it impossible to retrieve accurate numbers for comparison. Nevertheless, from Tab. 3 we learn that for two clients [35] appears faster than any other scheme for smaller dataset sizes, as illustrated for 2^{10} elements in Fig. 1. Although quickest, ESIP [35] is not cardinality hiding or repeatable, highlighting that we must still consider performance in the context of the other properties defined in § 4. Their protocol is followed by Zhang et al. [62] and Feather, which satisfy several useful properties like efficient updates. Furthermore, by the time we get to $2^{20} \approx 1$ M elements, Fig. 1 shows that the fastest protocol is now Kamara et al. [28], almost three times faster than Zhang et al. [62].

For more than two clients, the winner for large sets is PRISM [33], which can handle 20M elements from 10 clients in 18 s. For smaller client numbers and set sizes, the slowest is consistently Miyaji et al. [40], taking 15 seconds for 2^{10} elements on 4 clients, compared to Zhang et al. [62] (0.22 s), which is always just slightly faster than Feather [1] (0.26 s).

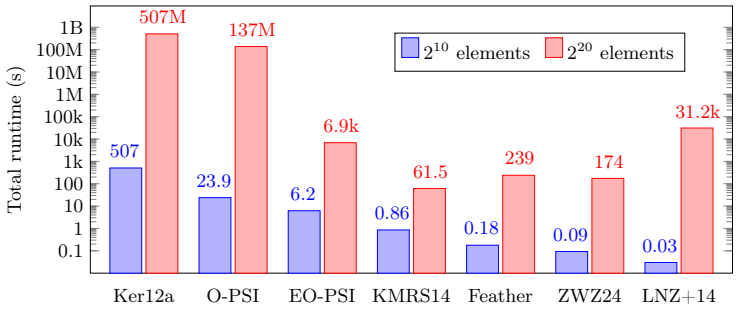

Fig. 1. Total (self-reported) runtime comparison for two clients and different set sizes. Results for Ker12a and O-PSI for 2^{20} elements extrapolated from original data.

6.3 Outsourced PSI Selection Tool

To provide a practical, data-driven resource to anyone choosing an outsourced PSI protocol, we develop an interactive selection tool. It enables easy filtering of protocols based on any combination of the 12 characteristics from our framework. Existing open-source implementations are linked. A key innovation is the transformation of a static collection of benchmark data dispersed across papers into one dynamic, predictive model for protocol analysis tailored to the target scenario. Our tool is available at https://github.com/hawkesse/SoK-Outsourced-PSI and a scaled screenshot is provided in Fig. 2.

Methodology. All available benchmark data was collated from the literature and evaluated, excluding any partial benchmarks, e.g., only encryption runtime. The result was 9 protocols benchmarked in the two-party setting, and 4 protocols benchmarked in the multi-party setting. The relationship between number of parties and runtime may be non-trivial, hence we only investigate multi-party runtimes for those protocols where multi-party benchmarks were provided.

The tool calculates an estimated runtime based on the dataset size input by the user and the extrapolation of collated benchmark data of the form (x =dataset

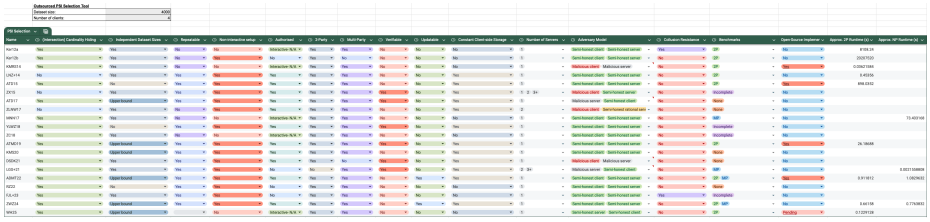

Fig. 2. Scaled screenshot of our interactive selection tool. Users can filter the individual properties based on their needs and obtain performance estimates based on the individually chosen input set sizes and number of parties.

Table 3. Collation of self-reported runtimes across varying numbers of clients (#C), for the minimum and maximum set size in each case. Runtimes given to two decimal places where possible; -g- = graphical representation only, s^+ = encryption time only.

Type	Protocol	#C	Minimum		Maximum	
			set size	runtime	set size	runtime
PKE	Kerschbaum (SAC'12) [30]	2	100	5.15 s	1000	507.29 s
	Liu et al. (IC2E'14) *ESIP* [35]		2^{10}	0.03 s	2^{20}	31,186.40 s
AHE	Abadi et al. (SEC'15) *O-PSI* [3]	2	2^{10}	23.9 s	2^{15}	57,357.5 s
	Zhao and Chow (WPES'18) [63]		100	979.28 s	—	—
MHE	Kerschbaum (ASIACCS'12) [31]		2^4	336.72 s	2^7	20,667.76 s
	Yang et al. (CCPE'18) [58]		100	-g-	5000	-g-
FHE	Fan et al. (Mathematics'23) *CMPSI* [18]		2^8	0.01 s^+	2^{20}	47.99 s^+
	Wei and Hu (ePrint'25) *D-PSI* [56]		2^8	0.01 s^+	2^{20}	32.53 s^+
PRE	Zheng and Xu (IC2E'15) *VDSI* [64]	2	2^{10}	-g-	2^{15}	-g-
SS	Miyaji et al. (J. Medical Syst.'17) [40]	4	2^6	1.02 s	2^{14}	297 s
		8	2^6	1.5 s	2^{14}	355 s
		16	2^6	1.98 s	2^{14}	450 s
	Li et al. (SIGMOD'21) *PRISM* [33]	10	5 M	4.2 s	20 M	18 s
		20	5 M	8.6 s	20 M	≈ 33 s (-g-)
		50	5 M	20 s	20 M	≈ 71 s (-g-)
PRF	Abadi et al. (TDSC'19) *EO-PSI* [5]	2	2^{10}	6.2 s	2^{20}	6,864.20 s
PRP	Kamara et al. (FC'14) [28]		1000	0.86 s	1 M	61.54 s
	Abadi et al. (FC'22) *Feather* [1]		2^{10}	1.33 s	2^{20}	1,801.6 s
		4	2^{12}	5.43 s	2^{20}	1,835.51 s
		15	2^{12}	6.3 s	2^{20}	2,443.61 s
		16 K	2^{10}	328.85 s	2^{11}	909.46 s
	↑(*Feather* results reported in [62])	2	2^{10}	0.18 s	2^{20}	239.12 s
		4	2^{10}	0.26 s	—	—
		16	2^{10}	0.79 s	—	—
		1000	2^{10}	41.72 s	—	—
	Zhang et al. (WAIM'24) [62]	2	2^{10}	0.09 s	2^{20}	173.55 s
		4	2^{10}	0.22 s	—	—
		16	2^{10}	0.60 s	—	—
		1000	2^{10}	28.16 s	—	—

size, y =runtime). All regression models are constrained to pass through the origin (since a dataset of zero items should result in zero runtime), and are determined by the scheme's known asymptotic computation cost. For quadratic time algorithms, we compute the quadratic regression equation through the origin, i.e., of the form $y = mx^2$. For linear time algorithms, we use $y = mx$. The slope m represents the per-item cost and is fixed for each protocol.

In the multi-party setting with more than two clients, the estimation is dependent on both a given choice of dataset size and given number of parties, i.e., (x =dataset size, y =number of parties, z =runtime). Two different methods were used, depending on the format of reported benchmark data. For protocols Miyaji et al. [40] and PRISM [33], where measurements for multiple dataset sizes were available for each number of parties, a two-stage 2D regression

model ($z = (gy + c)x$) was chosen. Separate initial linear regression equations were computed as in the 2P setting for each number of parties benchmarked, with slopes m_n ($n = 4, 8, 16$ and $n = 10, 20, 50$ resp.), then a second linear regression was computed between the number of parties n and their slopes m_n to create the gradient equation ($gy + c$). Here, c is the baseline per-item cost and g illustrates the rate at which, e.g., more parties make each item more expensive to process. For Feather [1] and Zhang et al. [62], where only one dataset size per party size was reported, a direct 3D regression through the origin ($z = ax + by$) was used instead, due to data sparsity. Thus, a and b express the per-item and per-party costs, respectively.

Limitations. The benchmark data was gathered from a range of academic papers published over more than a decade. While this approach ensures the most extensive list of protocols, the challenges to comparability due to both lack of uniform testing environments and increase in computing and networking speeds over time must be acknowledged. Therefore, our tool is not intended to predict exact runtimes but to capture the underlying trends, for example, the regression coefficients, as a meaningful complement to the paper's extensive qualitative analysis. We propose that the tool itself and the methodology behind it, could serve as a foundation for future, more standardized benchmarking efforts.

7 Discussion and Future Directions

One of the key properties that we identify is hiding the cardinality of the intersection from the server, but what about the cardinality of the input sets? In certain cases, the size of a dataset could reveal sensitive information, e.g., when computing the intersection between flight passengers and a government no-fly watch list [60]. It could be argued that a PSI protocol does not meet the goal of learning nothing about the other party's set if they can learn the set size.

In many scenarios, one party's set may be much larger than the other, known as *unbalanced* PSI. For example, in mobile contact discovery a client's small set of contacts is compared to a large database of registered users. Ideally, outsourced PSI protocols would allow for unbalanced set sizes with communication and computation complexities that scale linearly in the size of the smaller set. This has been achieved for the regular PSI (with pre-processing) [53], yet has not been explored in the outsourced setting – we leave this as future work.

Another area of PSI research currently receiving a lot of interest is *fuzzy PSI*. This is a specific case of *structurally-aware PSI* [21, 22], in which the receiver has N balls of radius δ and dimension d, and the sender learns which of their points lie within one of receiver's balls [9]. There are many compelling use cases for private matching between inexact values within a certain threshold such as comparing biometric data or GPS locations. Whilst the algorithm by van Baarsen and Pu [9] has comparably good efficiency, outsourcing the computations could still bring benefits, thus outsourced fuzzy PSI is another promising direction of research.

Fairness in PSI ensures that either all honest parties receive the intersection result, or no one does (cf. § 2.1). Only two of the outsourced PSI protocols consider this property – Kamara et al. [28] via applying a second PRP layer, and Zhang et al. [61] via a reputation-based game under the assumption of rational servers. Fairness will be essential to consider if considering future applications of outsourced PSI to auctions and financial analysis, to ensure that no party is able to act on information ahead of the others and gain an unfair advantage.

As recommendations for future work, we would urge researchers to contribute open-source implementations of their protocols where possible, to encourage future benchmarking and adoption for real-world use cases. In a similar manner, we encourage sharing open access data on benchmark experiments, and to report results on concrete communication costs, not just runtimes. Another beneficial action is to provide asymptotic and concrete performance results for each individual party (e.g. client A_i, client B, server) and/or phase (e.g. encryption, computation, decryption) to more clearly convey the resource requirements.

8 Conclusion

In this SoK, we have carefully reviewed 20 existing outsourced PSI protocols. Our work includes a comparison of primitives, performance, and security models, and additionally provides a framework of 12 properties that characterize crucial yet overlooked nuances between different protocols. Furthermore we provide an interactive selection tool. Finally, we discuss the research gap between regular PSI and outsourced PSI, and identify trends such as fuzzy PSI, unbalanced PSI, and fairness for further study in the outsourced setting.

Acknowledgements Sophie Hawkes was supported by the EPSRC and the UK government as part of the Centre for Doctoral Training in Cyber Security for the Everyday at Royal Holloway, University of London (EP/S021817/1).

References

1. Abadi, A., Dong, C., Murdoch, S.J., Terzis, S.: Multi-party Updatable Delegated Private Set Intersection. In: FC. LNCS, vol. 13411, pp. 100–119. Springer (2022)
2. Abadi, A., Doyle, B., Gini, F., Guinamard, K., Murakonda, S.K., Liddell, J., Mellor, P., Murdoch, S.J., Naseri, M., Page, H., Theodorakopoulos, G., Weller, S.: Starlit: Privacy-Preserving Federated Learning to Enhance Financial Fraud Detection. IACR ePrint https://ia.cr/2024/90
3. Abadi, A., Terzis, S., Dong, C.: O-PSI: Delegated Private Set Intersection on Outsourced Datasets. In: SEC. IFIP AICT, vol. 455, pp. 3–17. Springer (2015)
4. Abadi, A., Terzis, S., Dong, C.: VD-PSI: Verifiable Delegated Private Set Intersection on Outsourced Private Datasets. In: FC. LNCS, vol. 9603, pp. 149–168. Springer (2016)
5. Abadi, A., Terzis, S., Metere, R., Dong, C.: Efficient Delegated Private Set Intersection on Outsourced Private Datasets. IEEE TDSC **16**(4), 608–624 (2019)

6. Asharov, G., Hamada, K., Kikuchi, R., Nof, A., Pinkas, B., Tomida, J.: Secure Statistical Analysis on Multiple Datasets: Join and Group-By. In: CCS. pp. 3298–3312. ACM (2023)
7. Asharov, G., Lindell, Y., Schneider, T., Zohner, M.: More Efficient Oblivious Transfer and Extensions for Faster Secure Computation. In: CCS. pp. 535–548. ACM (2013)
8. Asokan, N., Schunter, M., Waidner, M.: Optimistic Protocols for Fair Exchange. In: CCS. pp. 7–17. ACM (1997)
9. van Baarsen, A., Pu, S.: Fuzzy Private Set Intersection with Large Hyperballs. In: EUROCRYPT. LNCS, vol. 14655, pp. 340–369. Springer (2024)
10. Boneh, D., Goh, E., Nissim, K.: Evaluating 2-DNF Formulas on Ciphertexts. In: TCC. LNCS, vol. 3378, pp. 325–341. Springer (2005)
11. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS. pp. 136–145. IEEE (2001)
12. Chen, Y., Huang, K.: JEDI: Joint and Effective Privacy Preserving Outsourced Set Intersection and Data Integration Protocols. IEEE TIFS **18**, 4504–4514 (2023)
13. Chung, H., Kim, M., Yang, P.C.: Multiparty Delegated Private Set Union With Efficient Updates on Outsourced Datasets. IEEE TDSC pp. 1–16 (2024)
14. Debnath, S.K., Sakurai, K., Dey, K., Kundu, N.: Secure Outsourced Private Set Intersection with Linear Complexity. In: DSC. pp. 1–8. IEEE (2021)
15. Dong, C., Chen, L., Camenisch, J., Russello, G.: Fair private set intersection with a semi-trusted arbiter. In: DBSec. LNCS, vol. 7964, pp. 128–144. Springer (2013)
16. Duong, T., Phan, D.H., Trieu, N.: Catalic: Delegated PSI Cardinality with Applications to Contact Tracing. In: ASIACRYPT. LNCS, vol. 12493, pp. 870–899. Springer (2020)
17. Even, S., Goldreich, O., Lempel, A.: A Randomized Protocol for Signing Contracts. CACM **28**(6), 637–647 (1985)
18. Fan, C., Jia, P., Lin, M., Wei, L., Guo, P., Zhao, X., Liu, X.: Cloud-Assisted Private Set Intersection via Multi-Key Fully Homomorphic Encryption. Mathematics **11**(8) (2023)
19. Freedman, M.J., Nissim, K., Pinkas, B.: Efficient Private Matching and Set Intersection. In: EUROCRYPT. LNCS, vol. 3027, pp. 1–19. Springer (2004)
20. Garimella, G., Pinkas, B., Rosulek, M., Trieu, N., Yanai, A.: Oblivious Key-Value Stores and Amplification for Private Set Intersection. In: CRYPTO. LNCS, vol. 12826, pp. 395–425. Springer (2021)
21. Garimella, G., Rosulek, M., Singh, J.: Structure-Aware Private Set Intersection, with Applications to Fuzzy Matching. In: CRYPTO. LNCS, vol. 13507, pp. 323–352. Springer (2022)
22. Garimella, G., Rosulek, M., Singh, J.: Malicious Secure, Structure-Aware Private Set Intersection. In: CRYPTO. LNCS, vol. 14081, pp. 577–610. Springer (2023)
23. Goldreich, O.: The Foundations of Cryptography – Volume 1: Basic Tools. Cambridge University Press (2001)
24. Goldreich, O.: The Foundations of Cryptography – Volume 2: Basic Applications. Cambridge University Press (2004)
25. Goldwasser, S., Micali, S.: Probabilistic Encryption. J. Comput. Syst. Sci. **28**(2), 270–299 (1984)
26. Jolfaei, A.A., Mala, H., Zarezadeh, M.: EO-PSI-CA: Efficient Outsourced Private Set Intersection Cardinality. J. Inf. Secur. Appl. **65**, 102996 (2022)
27. Kamara, S., Mohassel, P., Raykova, M.: Outsourcing multi-party computation. IACR ePrint https://ia.cr/2011/272

28. Kamara, S., Mohassel, P., Raykova, M., Sadeghian, S.S.: Scaling Private Set Intersection to Billion-Element Sets. In: FC. LNCS, vol. 8437, pp. 195–215. Springer (2014)
29. Kavousi, A., Mohajeri, J., Salmasizadeh, M.: Improved Secure Efficient Delegated Private Set Intersection. In: ICEE. pp. 1–6. IEEE (2020)
30. Kerschbaum, F.: Collusion-Resistant Outsourcing of Private Set Intersection. In: SAC. pp. 1451–1456. ACM (2012)
31. Kerschbaum, F.: Outsourced Private Set Intersection Using Homomorphic Encryption. In: ASIACCS. pp. 85–86. ACM (2012)
32. Kolesnikov, V., Kumaresan, R., Rosulek, M., Trieu, N.: Efficient Batched Oblivious PRF with Applications to Private Set Intersection. In: CCS. pp. 818–829. ACM (2016)
33. Li, Y., Ghosh, D., Gupta, P., Mehrotra, S., Panwar, N., Sharma, S.: PRISM: Private Verifiable Set Computation over Multi-Owner Outsourced Databases. In: SIGMOD. pp. 1116–1128. ACM (2021)
34. Lindell, Y.: How to simulate it - A tutorial on the simulation proof technique. In: Tutorials on the Foundations of Cryptography, pp. 277–346. Springer International Publishing (2017)
35. Liu, F., Ng, W.K., Zhang, W., Giang, D.H., Han, S.: Encrypted Set Intersection Protocol for Outsourced Datasets. In: IC2E. pp. 135–140. IEEE (2014)
36. Liu, Z., Wang, L., Bao, H., Cao, Z., Zhou, L., Liu, Z.: Efficient and Privacy-Preserving Cloud-Assisted Two-Party Computation Scheme in Heterogeneous Networks. IEEE Trans. Ind. Informatics **20**(5), 8007–8018 (2024)
37. López-Alt, A., Tromer, E., Vaikuntanathan, V.: Cloud-Assisted Multiparty Computation from Fully Homomorphic Encryption. IACR ePrint https://ia.cr/2011/663
38. López-Alt, A., Tromer, E., Vaikuntanathan, V.: On-The-Fly Multiparty Computation on the Cloud via Multikey Fully Homomorphic Encryption. In: STOC. pp. 1219–1234. ACM (2012)
39. Meadows, C.: A More Efficient Cryptographic Matchmaking Protocol for Use in the Absence of a Continuously Available Third Party. In: S&P. pp. 134–137. IEEE (1986)
40. Miyaji, A., Nakasho, K., Nishida, S.: Privacy-Preserving Integration of Medical Data - A Practical Multiparty Private Set Intersection. J. Medical Syst. **41**(3), 37:1–37:10 (2017)
41. Mohassel, P., Rindal, P., Rosulek, M.: Fast Database Joins and PSI for Secret Shared Data. In: CCS. pp. 1271–1287. ACM (2020)
42. Morales Escalera, D., Agudo, I., López, J.: Private Set Intersection: A Systematic Literature Review. Comput. Sci. Rev. **49**, 100567 (2023)
43. Oliaee, M.M., Delavar, M., Ameri, M.H., Mohajeri, J., Aref, M.R.: On the Security of O-PSI a Delegated Private Set Intersection on Outsourced Datasets. In: ISCISC. pp. 77–81. IEEE (2017)
44. Pinkas, B., Schneider, T., Zohner, M.: Scalable Private Set Intersection Based on OT Extension. ACM TOPS **21**(2), 7:1–7:35 (2018)
45. Rabin, M.O.: How To Exchange Secrets with Oblivious Transfer. Technical Report TR-81, Aiken Computation Lab, Harvard University (1981)
46. Raghuraman, S., Rindal, P.: Blazing Fast PSI from Improved OKVS and Subfield VOLE. In: CCS. pp. 2505–2517. ACM (2022)
47. Rindal, P., Schoppmann, P.: VOLE-PSI: Fast OPRF and Circuit-PSI from Vector-OLE. In: EUROCRYPT. LNCS, vol. 12697, pp. 901–930. Springer (2021)

48. Ruan, O., Zeng, J.: A Delegated Offline Private Set Intersection Protocol for Cloud Computing Environments. In: ICCIR. pp. 735–739. ACM (2022)
49. Sadat, M.N., Aziz, M.M.A., Mohammed, N., Pakhomov, S., Liu, H., Jiang, X.: A Privacy-Preserving Distributed Filtering Framework for NLP Artifacts. BMC Medical Informatics Decis. Mak. **19**(1), 183:1–183:10 (2019)
50. Sander, T., Young, A.L., Yung, M.: Non-Interactive CryptoComputing For NC^1. In: FOCS. pp. 554–567. IEEE (1999)
51. Shakya, R., Yasumura, Y., Suzuki, T., Ishimaki, Y., Yamana, H.: Outsourced Private Set Union on Multi-Attribute Datasets for Search Protocol using Fully Homomorphic Encryption. In: iiWAS. pp. 55–62. ACM (2019)
52. Shi, Y., Yang, W., Xu, W., Li, Q.: Research on Outsourced PSI Protocols for Privacy Preserving Data Sharing. In: ICBDS. CCIS, vol. 1563, pp. 125–136. Springer (2021)
53. Son, Y., Jeong, J.: PSI with computation or Circuit-PSI for Unbalanced Sets from Homomorphic Encryption. In: ASIACCS. pp. 342–356. ACM (2023)
54. Tajima, A., Sato, H., Yamana, H.: Outsourced Private Set Intersection Cardinality with Fully Homomorphic Encryption. In: ICMCS. pp. 1–8. IEEE (2018)
55. Vos, J., Conti, M., Erkin, Z.: Sok: Collusion-resistant multi-party private set intersections in the semi-honest model. In: SP. pp. 465–483. IEEE (2024)
56. Wei, S., Hu, J.: Delegated PSI from homomorphic encryptions. IACR ePrint https://ia.cr/2025/868
57. Wu, Y., He, J., Yan, S., Wu, J., Yang, T., Ruas, O., Zhang, G., Cui, B.: Elastic bloom filter: Deletable and expandable filter using elastic fingerprints. IEEE Trans. Computers **71**(4), 984–991 (2022)
58. Yang, X., Luo, X., Wang, X.A., Zhang, S.: Improved Outsourced Private Set Intersection Protocol Based on Polynomial Interpolation. CCPE **30**(1) (2018)
59. Yeo, F.Y., Ying, J.H.M.: Third-Party Private Set Intersection. In: ISIT. pp. 1633–1638. IEEE (2023)
60. Zhan, Y., Zhang, Z., Liu, Q., Wang, B.: Hiding the Input-Size in Multi-Party Private Set Intersection. DESI **91**(9), 2893–2915 (2023)
61. Zhang, E., Li, F., Niu, B., Wang, Y.: Server-aided private set intersection based on reputation. Inf. Sci. **387**, 180–194 (2017)
62. Zhang, Y., Wang, L., Zhou, L.: Efficient Updateable Private Set Intersection on Outsourced Datasets. In: APWeb/WAIM. LNCS, vol. 14964, pp. 84–99. Springer (2024)
63. Zhao, Y., Chow, S.S.M.: Can You Find The One for Me? In: WPES@CCS. pp. 54–65. ACM (2018)
64. Zheng, Q., Xu, S.: Verifiable Delegated Set Intersection Operations on Outsourced Encrypted Data. In: IC2E. pp. 175–184. IEEE (2015)