Summation-based Private Segmented Membership Test from Fully Homomorphic Encryption

***Abstract*—In many real-world scenarios, clients often need to verify if their data element is present in a set segmented across a large number of distributed data holders. Some of such scenarios include collaborative efforts among financial institutions for fraud detection, intelligence agencies sharing/querying watchlists across jurisdictions, and healthcare institutions collaborating on medical research while preserving patient privacy. To ensure user privacy, both the client's query and the sets of data holders should remain encrypted throughout the protocol. Prior work on Private Set Intersection (PSI), Multi-Party PSI (MPSI), and Private Membership Test (PMT) have limitations in this context. They require data holders to have plain text sets, incur excessively high latency for aggregating results from a large number of data holders, disclose information about the party with the intersection element, or induce a notable false positive rate.**

This paper introduces the Private Segmented Membership Test (PSMT) primitive, constructed using approximate-arithmetic fully homomorphic encryption. We prove its security in a semi-honest model with a high collusion level. Our approach addresses existing challenges to construct a PSMT protocol, avoiding information leakage about the intersection element's holder and minimizing false positives for a large number of data holders. Our novel idea is based on the summation-based homomorphic membership check that operates directly on encrypted data sets and enhances scalability, which is crucial for highly distributed datasets. Our PSMT protocol supports many more parties (up to 4096 in experimental comparisons) compared to prior related work that supports only around 100 efficiently. Experimental results demonstrate efficient aggregation with minimal overhead taking 92.5s for 1024 data holders and a 2^{25} set size, showcasing the efficiency of summation-based homomorphic aggregation. and privacy models compared to state-of-the-art PSI and MPSI protocols. We compare our PSMT protocol to state-of-the-art PSI and MPSI protocols, highlighting improvements in usability with a more robust privacy model and support for a larger number of parties.

***Index Terms*—Private Set Intersection, Private Information Retrieval, Multi-Party PSI**

I. Introduction

Privacy concerns often limit the collaboration of many entities in cases where each has private data that must be shared for joint usage. In many cases, the private data is generated and stored in a distributed manner, and methods of distributively and privately computing on data in such scenarios open avenues for new applications. There are many real-world scenarios where this problem needs to be solved.

Federal tax authorities seek information on potential tax evaders holding accounts in both domestic and foreign banks. However, due to jurisdictional constraints and privacy concerns, banks cannot openly disclose account holder details, and tax authorities clearly cannot reveal their suspect list. The authorities also face the challenge of scaling collaboration with the high/growing number of participants. Anonymous collaboration is desired by such institutions to avoid bad publicity. In many cases, these institutions are willing to collaborate with the tax authorities [24], and they themselves also wish to exercise rigorous scrutiny when extending loans to new and existing customers to mitigate potential risks. Unfortunately, many financial privacy laws [31] prohibit banks from revealing customer data to third parties without consent. There are currently over 4,700 FDIC-insured banks in the United States only. When a customer applies for any kind of financial assistance within one of them, efficient collaboration and sharing of fraud lists can make the decision process significantly more trustworthy. In such cases, banks do not wish to share their specific fraud lists, and they do not even wish to disclose whether the queried customer is on their own fraud list due to various privacy and legal concerns. Such a secure membership query scenario could extend to credit card companies, tax-collection agencies, and similar entities, necessitating the involvement of many parties in the decision-making process. Efficient collaboration is needed in these circumstances as the number of collaborating entities grows, emphasizing the challenges of distributed computing in such large-scale scenarios. A data-sharing protocol that allows queriers to efficiently learn only whether a queried entity exists in distributed fraud lists would help such institutions examine a person's credibility beyond the nation.

Many government agencies (e.g., FBI, CIA) maintain sensitive lists of secret agents or watchlists distributed across its multiple divisions and branches. Data sharing for identity verification, background checks, security clearance, and watchlist screening requires secure and efficient collaboration without revealing membership details. Doing so over distributed databases is nontrivial due to the growing scale of the number of databases involved and the challenges of sharing personally identifiable information (PII). Therefore, it is desirable and even imperative that the records be stored in encrypted forms across a large number of distributed servers such that each record is under strict security/privacy protection (e.g., DHS Use Cases [32]). Similarly, the auto insurance industry faces a challenge where obtaining a customer's complete driving violation history from a growing number of companies or the Department of Motor Vehicles can minimize risks. In the medical insurance industry, highly distributed storage of medical insurance records presents a similar challenge.

In the given examples, a privacy challenge arises due to the substantial scale of entities involved in distributed data-sharing applications (hundreds of thousands in the tax fraud [36]).

This underscores the increasing need for privacy-preserving set intersection protocols with a high number of distributed dataset holders. The objective is to perform queries without revealing the source set of the intersection, termed *provenance privacy*. These PII containing datasets frequently updated and held by distinct parties, require strict protection to ensure individual privacy. Storing and utilizing records in encrypted form is thus essential to safeguard against data breaches and insider threats. Differing from traditional PSI, MPSI, or PMT scenarios in multiple dimensions, we term this problem *Private Segmented Membership Test* (PSMT) and provide its formal definition.

Existing approaches, such as private set intersection (PSI), are inadequate for the mentioned problems. PSI protocols allow two parties (referred to as *receiver* and *sender* in accordance with [29]) to compute set intersections without revealing any additional information. While existing PSI protocols can address the PSMT, they necessitate dataset holders accessing the elements of their sets in plaintext format for optimizations and pose several scalability issues. Thus, for highly distributed encrypted databases with sensitive information such as medical, financial, or criminal records, PSI is unsuitable. Generic multiparty-PSI (MPSI) protocols like [8], [27], [30] are also less efficient for PSMT due to high interactions (in OT-based), privacy concerns against the senders, and communication runtime, bandwidth or storage needs arising from distributed and private computing needs. Additionally, Private Membership Test (PMT) protocols [37], optimized for single elements, require unencrypted databases, may produce high false positives, and are not designed for multiparty use without compromising the provenance privacy of the senders.

Fully Homomorphic Encryption (FHE) is a widely used building block for constructing PSI protocols due to its single-round communication requirement. However, PSI protocols based on FHE, [10], [11], [18], face several challenges in addressing the PSMT problem. Theoretically, given a set held by the sender X and receiver query y, existing PSI protocols homomorphically compute a *sender polynomial*, $f(y) = r \prod_{x \in X}(x - y)$, where r is a random mask. To apply this method for PSMT, each sender calculates their own sender polynomial, requiring FHE parameters to tolerate $O(log(l))$ *additional* multiplicative depth for even a moderately large number of senders l (e.g., $l = 64$). Sender partitioning, although reducing multiplicative depth during polynomial computation, increases depth during result aggregation, particularly with numerous sender polynomials or multiple senders.

Our PSMT protocol operates on encrypted input for both the receiver's element and the sets held by the sender, eliminating the need for handling sensitive data (besides encryption) in plaintext. This simplicity allows handling frequent set updates without repeating preprocessing. Our protocol is based on FHE, and it employs cheap homomorphic additions for ciphertext aggregation from multiple senders and does not suffer from scalability issues from a growing number of senders in a distributed environment. The security of our protocol relies on post-quantum FHE security, and it does not compromise the provenance privacy of the senders to the receiver. We assume a semi-honest model, provide a security proof, and utilize threshold-FHE that can tolerate up to l collusion among l senders. In summary, we construct a protocol that can tolerate a large number of senders and efficiently compute on encrypted sender sets that require frequent set updates.

The contributions of this work are summarized as follows:

- We define the Private Segmented Membership Test (PSMT) problem, which is widely relevant to real-world scenarios. Existing approaches result in various limitations, and we present a novel solution to address them.
- We address the shortcomings of existing approaches using finite-field FHE, which results from encrypted user data segmented across many senders and high aggregation latency. We provide a novel summation-based set intersection protocol with approximate arithmetic FHE that overcomes these limitations.
- For the technical challenges in solving PSMT with our novel solution, such as plaintext domain size and function approximation accuracy/latency, we provide novel strategies to deal with such issues and achieve good performance even for a large, growing number of senders.
- We implement our method and present an experimental evaluation of our solution to show its significant performance advantage in the case of a large number of senders. Our anonymized source code is available for reproducibility and future research at https://anonymous.4open.science/r/psmt-1323/. We show up to 4-11$\times$ performance improvement over previous works.

II. Related Work

A. Private Set Intersection (PSI)

The first PSI protocol was presented based on the Diffie-Hellmann (DH) key agreement scheme. This protocol leveraged the commutative properties of the DH function and offered security against the random oracle model. Its low communication cost continues to serve as a foundation for many modern PSIs. [23] introduced PSI protocols based on oblivious polynomial evaluation (OPE) where sets are represented as polynomials. Additionally, PSI protocols have been constructed using Oblivious Pseudo-Random Functions (OPRFs) [22], garbled circuits [40], and oblivious transfer (OT), and OT-extension [35]. Recently, FHE with post-quantum security has been employed for many PSI protocols for unbalanced set sizes using OPE, where the difference in sets held by the sender and receiver is very high.

The two-party Private Set Intersection (PSI) model is extensively studied due to its wide real-world applications. Several variants of this model exist, where either both parties learn the intersection (mutual PSI) [20] or only one of the parties learns the intersection (one-way PSI). [35] utilize (1-out-of-n) OT based on [26] for PSI. The limitation of their approach is that OT step requires the sender to access elements in the hash table's bins, and extending it to substantial parties requires multiple Oblivious Pseudo-Random Function (OPRF) evaluations via OT, greatly increasing communication overhead. CLR17 [11] protocol, and its improved variants [10], [18] are

TABLE I
COMPARISON OF EXISTING WORKS TO PSMT. S.A.S.: SECURITY AGAINST SENDERS; L.F.P.: LOW FALSE POSITIVE RATE (LESS THAN 10^{-3}). OPRF, OPPRF, OKVS AND HE DENOTE OBLIVIOUS PSEUDORANDOM FUNCTION, OBLIVIOUS PROGRAMMABLE PSEUDORANDOM FUNCTION, OBLIVIOUS KEY VALUE STORE AND HOMOMORPHIC ENCRYPTION RESPECTIVELY. POST-QUANTUM REFERS TO THE SECURITY AGAINST QUANTUM COMPUTER CAPABLE ADVERSARIES.

Protocol	Construction	Class	Post-Quantum	S.A.S.	L.F.P.	Adversary Model
Chen *et al.* [11]	FHE	PSI	✓	×	✓	Semi-honest
Chen *et al.* [10]	FHE, OPRF	PSI	✓	×	✓	Malicious
Cong *et al.* [18]	FHE, OPRF	PSI	✓	×	✓	Malicious
Kolesnikov *et al.* [27]	OPPRF	MPSI	×	×	✓	Semi-honest
Ramezanian *et al.* [37]	Bloom/Cuckoo Filter, HE	PMT	×	×	×	Semi-honest
Pinkas *et al.* [35]	Oblivious Transfer	PSI	×	×	✓	Semi-honest
Bay *et al.* [4]	Bloom Filter	MPSI	×	×	×	Semi-honest
Nevo *et al.* [30]	OPPRF, OKVS	MPSI	×	×	✓	Malicious
This work	FHE	PSMT	✓	✓	✓	Semi-honest

state-of-the-art FHE-based PSI protocols to the best of our knowledge. The basic protocol in [11] has the sender sample a uniformly random non-zero element r_i and homomorphically compute the intersection polynomial $z_i = r_i \prod_{x \in X}(c_i - x)$ using encrypted receiver's set $(c_1, c_2, \ldots, c_n)$ and the sender's unencrypted elements $x \in X$. z_i is returned to the receiver, who concludes that $y \in X$ iff $z_i = 0$. The receiver only learns the presence of an intersection. The CLR17 protocol applies many optimizations, including receiver and sender side batching using cuckoo hashing and binning, SIMD (Single Instruction Multiple Data) using FHE, and windowing. Later works [10], [18] used an OPRF preprocessing on the encoded sender set, achieved malicious security, applied the Paterson-Stockmeyer algorithm for evaluating the intersection circuit, and reduced the communication cost by using extremal postage-stamp bases. These protocols require the sender to access the set in plain text for the aforementioned optimizations and encodings for creating the polynomial for interpolation. Consequently, the privacy of data points in the sets held by the sender is only protected against the receiver and not against the sender. Furthermore, adapting these methods to the multi-party scenario drastically increases the multiplicative depth required to obtain the query result.

B. Multi-party PSI (MPSI)

Multiparty Private Set Intersection (MPSI) extends the two-party PSI problem to scenarios involving more than two parties. Two-party PSI protocols can be extended to multiple parties to handle the MPSI scenario; however, these solutions often lead to privacy and performance issues. Several techniques have been employed to design MPSI, such as circuit-based computations [34], bloom filters [28], OPE [23], and OT and permutation-based hashing [33]. [27] used a technique based on oblivious evaluation of a programmable pseudorandom function (OPPRF) to implement a time-efficient MPSI protocol for large amounts of items. However, their time complexity scales quadratically w.r.t the number of parties in the protocol. [8] improves upon [27] in terms of communication and extends it to circuit-based PSI and quorum-PSI. Notably, these protocols provide intersection results to all or some parties based on the intersection outcome and do not support a substantial number of parties, making them incomparable to our work.

Recent work by [2] uses threshold FHE to construct threshold MPSIs, which have sublinear communication complexities with a threshold number that is sublinear in the number of elements in datasets. They use a similar polynomial encoding of each set element as in [11]. [4] provide two MPSI protocols based on bloom filters and threshold homomorphic public-key techniques. Their protocol performs better than previous state-of-the-art in terms of run time, given that the sets are small and a large number of senders exist. [30] construct concretely efficient malicious MPSI protocols based on the recently introduced primitives such as OPPRF and oblivious key-value store (OKVS).

C. Private Membership Test (PMT)

Private Membership Test (PMT, or Private Set Inclusion) is a similar problem to PSI, in which a single receiver learns the intersection of their element among several senders. To solve PMT, many works apply Private Information Retrieval (PIR) based protocols that allow a user to retrieve an item from a database without the database owner learning anything about the item. PSMT closely matches the PIR; however, the sender's database is public in PIR. PMT has been extensively studied, particularly for two-party PSI in malware detection [37]. While hashing seems a naive solution for low-latency multi-party PMT, it becomes insecure with low-entropy input domains, and even high-entropy input domains, it may leak repeated elements upon consecutive executions. One can solve PSMT using individual PMT protocols with all senders via 1-out-of-n OT-based PMTs, followed by a secure XOR computation by the client. However, this approach has several drawbacks. Firstly, it necessitates sender access to plaintext sets for OT, losing privacy. Secondly, it requires the client to run n PMT protocols with n senders, adding extra communication and computation. While OT extension-based protocols can reduce communication complexity, they also demand access to plaintext sets, and any updates in databases result in significant performance and communication penalties. [39] propose a carousel method for PMT for solving malware detection based on Trusted Execution Environments (TEEs). However, TEEs suffer from side-channel attacks and other kinds of hardware-based attacks [9], which have decreased their confidence for use recently.

In summary, addressing PSMT through general PMT-solving methods based on PIR, OT, or TEEs provides only partial solutions. The considerable overhead for a large number of senders, a lack of privacy for datasets held by senders, either from the receiver or the sender(s), along with high latency and low throughput typically associated with these protocols render them impractical for efficient PSMT solutions. We compare representative works to PSMT in Table I.

III. Preliminaries

A. Fully Homomorphic Encryption

Fully Homomorphic Encryption (FHE) is a cryptographic primitive that allows computation on encrypted data with post-quantum security. Noise associated with an FHE ciphertext grows corresponding to each homomorphic operation, i.e., additively with additions and multiplicatively with multiplications. The most prominent FHE schemes are BGV [7], B/FV [21], CKKS [13], and Torus-FHE [17]. In practice, FHE schemes are often implemented as Somewhat Homomorphic Encryption (SHE) schemes with a limited multiplicative depth

In this work, we use the CKKS scheme, which uses a fixed-point complex number encoding to enable homomorphic computations on real numbers. CKKS has operands in $R = \mathbb{Z}[X]/\langle \Phi_M(X) \rangle$, where $\Phi_M(X)$ is the cyclotomic polynomial (x^N+1) of order $M = 2N$ (cyclotomic index) and degree $N \in \mathbb{Z}$ which is the ring dimension. CKKS parameters include the ring dimension, ciphertext modulus, and standard deviation of the error, represented as (N, q, δ). We employ the CKKS parameters to maintain 128-bit security in both classical and quantum contexts. A standard security notion for FHE is indistinguishability under the chosen plaintext (IND-CPA) attack. It ensures that an attacker cannot gain any useful information about the plaintext by observing the corresponding ciphertexts, even if the attacker has the ability to choose the plaintexts for encryption. For the threshold functionality in our protocol, we utilize TFHE, which allows for distributed decryption of ciphertexts [6]. Using TFHE, our protocol can handle up to $l-1$ collusion among the parties where the receiver possesses one of the secret key shares. As in standard homomorphic encryption (HE) schemes, we require that a TFHE scheme satisfies compactness, correctness, and security.

For the threshold functionality in our protocol, we utilize TFHE, which allows for distributed decryption of ciphertexts [6]. Using TFHE, our protocol can handle up to $l-1$ collusion among the parties where the receiver possesses one of the secret key shares. As in standard homomorphic encryption (HE) schemes, we require that a TFHE scheme satisfies compactness, correctness, and security.

SIMD: In FHE, we can consider the factorization of (x^N+1) modulo p such that p is a prime. We can then write the message space as a direct product of small fields, encrypt a vector of elements of these fields, and operate in parallel on the entries of these vectors, thus obtaining single instruction, multiple data (SIMD) capabilities [38].

Parameters: The protocol involves $l+1$ entities, namely $P_0, P_1, \dots, P_{l-1}$ and P_y where, P_y is the receiver, P_{l-1} is the senders' leader and the rest of the parties are the senders. All senders possess sets of items with a bit-length of σ. The receiver holds an element y with a bit-length of σ; senders $P_0, \dots, P_{l-1}$ own sets $X_0, \dots, X_{l-1}$, and we define $\mathcal{X} = \bigcup_{i=0}^{l-1} X_i$.
Functionality: Inputs are encryption of y and encryptions of the sets $X_0, \dots, X_{l-1}$. The receiver gets $\{y\} \cap \mathcal{X}$, and the senders, including the sender's leader, get $\perp$.

Fig. 1. Ideal functionality $\mathcal{F}_{PSMT}$ for PSMT for one-sided output.

B. Private Segmented Membership Test (PSMT)

For a party P_y, with a data element y, and l parties $P_0, P_1, \dots, P_{l-1}$, each with a set X_i such that $\mathcal{X} = \sum_{i=0}^{l-1} X_i$, a PSMT allows the party P_y to learn $\{y\} \cap \mathcal{X}$, without leaking any elements not in any X_i or which party holds an element in the intersection. The sets held by different senders in the PSMT protocol are disjoint. The parties involved in the protocol are referred to as the *sender* and the *receiver*. The ideal functionality $\mathcal{F}_{PSMT}$ associated with the PSMT protocol is presented in Figure 1. It is a one-way protocol that outputs $\{y\} \cap \bigcup_{i=0}^{l-1} X_i$ to the receiver and nothing to the senders. We note that in the event where $y \in X_i$, the set X_i is hidden from the receiver. This functionality ensures the provenance privacy of the senders. Additionally, each sender P_i only has access to encryptions of $x \in X_i$. We note that in the rest of this paper when we make reference to the senders' sets, we refer to the encrypted sets of users that are held by the senders.

IV. PSMT Protocol

A. Notation

We use λ to denote the security parameter for the FHE scheme. We use $[l]$ to denote the integers in $[0, l-1]$. Logarithms are base-two unless otherwise stated.

- $X_0, X_1, \dots, X_{l-1}$ are the sender's set for l senders. y is the receiver's query element. Without loss of generality (WLOG), we assume X_{l-1} is the leader sender.
- c_{x_i} denote the ciphertext of the i-th sender: $i \in [0, l-1]$. The senders' set elements are encrypted in a SIMD fashion. c_y is the ciphertext of the receiver's element.
- N is a power of 2 and is the ring dimension in our FHE scheme; q is the ciphertext modulus; d is the homomorphic multiplicative depth of the computation.
- L, R, and n are parameters used in the application of Domain Extension Polynomials. c is the degree of polynomial used for Chebyshev polynomial approximation.
- j, k and ρ are optimization parameters to reduce the false positive rates and τ is the threshold required to confirm an intersection.

B. Basic Protocol Without Approximation

Protocols using FHE to implement PSI [11] use zero as a multiplicative annihilator. Multiplications are not scalable, so we propose the novel idea of using additive aggregation.

We consider set elements members of $\mathbb{Z}$ (e.g., hashed elements). In Figure 3, we outline our basic protocol. The receiver has access to a query element y, and s/he creates

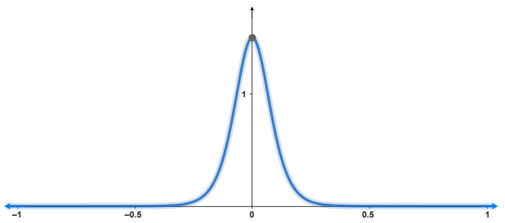

Fig. 2. Graph of the VAF $K \cdot (1 - \tanh^2(S \cdot x))$ where $K = 1.5$ and $S = 10$

replicas of y such that the number of replicas equals the batch size of the FHE scheme and encrypts them to obtain c_y such that all slots of c_y contain y. The receiver then sends c_y to l senders. Each sender $i \in [0, l-1]$ has access to a set X_i, and they encrypt X_i using FHE batching to obtain c_{x_i} such that each slot of c_{x_i} contains different elements of X_i. In case $|X_i| >$ batch size, the sender possesses multiple c_{x_i} otherwise, the sender possesses a single c_{x_i}. Each sender i then computes a homomorphic difference of c_y, and its ciphertext(s) c_{x_i}, diff_i in a SIMD fashion. Each sender i then computes a piecewise function $etan_i$: $\mathit{etan}_i(\mathit{diff}_i) = \begin{cases} K, & \text{if } \mathit{diff}_i = 0 \\ 0, & \text{if } \mathit{diff}_i \neq 0 \end{cases}$.

Namely, etan_i maps the input to K if the input is zero and zero otherwise. If equipped with multiple diff_i, the sender i homomorphically summates multiple etan_i into a single etan_i. Results obtained by the senders can be additively aggregated by computing $\sum_{i=0}^{l-1} \mathit{etan}_i$, whose value is non-zero for an intersection and zero for a non-intersection. If the senders' sets are disjoint, this sum will be K for an intersection; otherwise, it will be some multiple of K. We note that in practical deployments, it is most probable that some leader sender will take on the additional burden of aggregation or that different senders will take turns doing this. In our protocol, a leader sender aggregates the results.

This basic protocol can be implemented with any FHE scheme, and the aggregation would be composed of efficient homomorphic additions only. Unfortunately, this version would suffer from high multiplicative depths in computing "if $\mathit{diff}_i = 0$." This leads to the need to approximate etan_i.

C. Novel Value Annihilating Function (VAF)

One naïve thought is to approximate the function $etan_i$ with a polynomial, but doing so would require polynomials with very high degrees. We begin by considering the computation a single sender must perform. Suppose there exists a small negligible value ϵ and a sender has a set X_i with size n such that $x_j \in X_i$ for $j \in [n]$ and is given an FHE ciphertext c_y encrypting a receiver's input $y \in Y$. Consider the function $f(y, X_i) = \sum_{j=0}^{n-1} g(y, x_j)$, where $g(y, x_j) = \frac{1}{(x_j - y) + \epsilon}$. Then, $f(y, X_i)$ can be thought to exhibit the following behavior similar to that of $etan_i$:

$$f(y, X_i) = \begin{cases} 1/\epsilon + \sum_{j=1}^{n-1} g(y, x_j), & \text{if } y \in X_i \\ \sum_{j=0}^{n-1} g(y, x_j) \quad \ll K, & \text{if } y \notin X_i \end{cases}$$

Input : Receiver provides a query element y; senders provides sets $X_0, \ldots, X_{l-1}$ such that $\mathcal{X} = \bigcup_{i=0}^{l-1} X_i$. Each set contains bit strings of length σ. K is the output of the function *etan* when input is zero. σ, and K are public parameters.
Output: Receiver outputs $\{y\} \cap \mathcal{X}$; senders outputs $\perp$.

1. **Setup**: The senders and the receiver jointly agree on a FHE scheme with the same batch size . Senders are provided a public key pk, and the receiver is given a secret key sk.
2. **Encryption**: The receiver encrypts its element y such that all slots of the resulting ciphertext c_y contain replicas of y and sends c_y to each of the senders. Each sender $i \in [0, l-1]$ encrypts their sets $X_0, \ldots, X_{l-1}$ using the public key and obtain $c_{x_0}, \ldots, c_{x_{l-1}}$. If $|X_i| >$ batch size of the scheme, then the sender has multiple ciphertexts.
3. **Intersection**: Using c_y each sender i:
   a. Homomorphically computes $\mathit{diff}_i = c_y - c_{x_i}$ and $\mathit{etan}_i(\mathit{diff}_i)$. If possessing multiple ciphertexts, the sender computes multiple diff_i and $\mathit{etan}_i(\mathit{diff}_i)$ with the remaining ciphertexts and homomorphically summates them to a single etan_i.
   b. Sends etan_i to a leader sender, who then uses the FHE scheme to homomorphically compute $z = \sum_{i=0}^{l-1} \mathit{etan}_i$.
   The leader sender returns the ciphertext z to the receiver.
4. **Result interpretation**: The receiver decrypts z using its secret key sk and outputs, result which indicates an intersection if $z = K$ otherwise, not an intersection.

Fig. 3. Basic PSMT protocol for multiple parties

Here, when $y \in X_i$ and $\epsilon \to 0$, $f(y, X_i)$ increases without bound and its limit approaches infinity. This misuses the notion of infinity in terms of computation, as infinity is not a numerical quantity. Nevertheless, it leads us to some intuition on how to construct a protocol that is amenable to PSMT: we should have a function that each sender computes that outputs very large values (close to K) to annihilate the summation on an intersection but outputs a negligible summation value compared to K on a non-intersection. Then, additively summing these results from each sender would yield a result indicating if the receiver's element y is in $\bigcup_{i=0}^{l-1} X_i$, i.e., the large value indicates intersections and small negligible values indicate non-intersections.

To realize this idea, we consider a candidate function $D_K^{Exact}(x) : \mathbb{R} \to \mathbb{R}$ that exhibits the useful properties described above without the problematic behavior of a potential division by zero: $D_K^{Exact}(x) = \begin{cases} K, & \text{if } x = 0 \\ 0, & \text{if } x \neq 0 \end{cases}$. This function easily satisfies these constraints and has the exact same behavior as etan_i where $K \in \mathbb{R}$. We call such functions Value Annihilating Functions (VAFs). A VAF maps zero to K and all other values to zero. Computing VAFs homomorphically is challenging because *if* conditions cannot be evaluated efficiently in FHE. We thus consider another function approximating $D_K^{Exact}(x)$. For such a function, approximate-arithmetic FHE is preferable, as computing such approximate functions on finite fields is difficult. It is challenging to find a function that accurately approximates such kind of behavior because functions approximating such kind of behavior require computing the inverse or the division operation, which is fundamentally challenging in FHE. With all the considerations, we discovered the following function exhibits satisfactory behavior for our purpose: $D_{K,S}^{Approx}(x) = K \cdot (1 - \tanh^2(S \cdot (x)))$. This function is shown in Figure 2, approximating a VAF.

Parameters: X_i is the input set for l senders: $i \in [0, l-1]$. y, τ, and κ are the receiver's input, threshold value, and limit for the random mask, respectively. λ denotes the computational security parameter, and σ denotes set element's bit strings length. λ, τ, κ and σ are public parameters. K and S are public VAF parameters. L, R and n are public DEP parameters. c is the degree of the polynomial used for Chebyshev approximation.

1. [**Parameters**]
   a. **FHE**: Parties agree on parameters (N, q, δ, d) for the CKKS FHE scheme with λ computational security.
   b. **Key Distribution**: If using the TFHE functionality, a trusted setup provides each key shareholder $i \in [0, \varphi)$ a share of the secret key sk as one of $\{sk_0, sk_1, \ldots, sk_{\varphi-1}\}$, and broadcast the public key pk. If TFHE is not being used, the functionality can simply generate a public key pk for the senders and give a secret key sk to the receiver.
2. [**Encryption**]
   a. **Encrypt X & batch** : For all $x \in X_i$, such that $i \in [0, l-1]$, each sender i groups its values x into vectors in $\mathbb{R}^m$ of length m with elements in $\mathbb{R}$ and batches each vector into $2 \cdot m/N$ plaintexts and encrypts them using pk to obtain c_{x_i}. As a result, each slot of the c_{x_i} contains individual x values. In case $|X_i| >$ batch-size, the sender possesses multiple ciphertexts encrypting X_i. Note that FHE batching can only be applied to either the receiver or senders, but not both.
   b. **Encrypt replicas of y**: Receiver constructs a vector of length $N/2$, with each element being $y \in \mathbb{R}$. Then, the receiver encodes the vector into a CKKS plaintext and encrypts it to obtain c_y. As a result, each slot of c_y contains y.
4. [**Compute Intersection**]
   a. **Homomorphically compute subtractions**: Each sender i receives the ciphertext c_y and computes $diff_i = c_y - c_{x_i}$.
   b. **Apply DEP to reduce the domain size**: Each sender i iteratively applies the domain extension process n times using DEP to shrink the domain interval of values $diff_i$ from step $4a$ into the interval $[-R, R]$.
   c. **Homomorphically evaluate VAF**: Each sender i uses the Chebyshev approximation method to approximate $etan_i = K \cdot (1 - \tanh^2(S \cdot diff_i))$ in the interval $[-R, R]$ using a degree c polynomial.
   d. **First homomorphic squaring**: Each sender i homomorphically squares $etan_i$ and obtains $etan_i^{2^j}$ such that $j \in [1, 12]$.
   e. **Scaling and second homomorphic squaring**: Each sender i multiplies $etan_i^{2^j}$ using a scaling factor, ρ. Step $4d$ can be applied again to obtain $(\rho \cdot etan_i^{2^j})^{2^k}$ with $k \in [3, 4]$ to exponentially increase the number of parties the protocol can handle and reduce the false positive rates to negligible or zero.
   f. **Homomorphically evaluate summation**: If a sender possesses multiple ciphertexts, Step $4a$ to $4e$ is applied to all the remaining ciphertexts in parallel, and homomorphically summated to $(\rho \cdot etan_i^{2^j})^{2^k}$. An aggregator collects $(\rho \cdot etan_i^{2^j})^{2^k}$ from all other senders, samples a random non-zero plaintext element $r \in [-\kappa, \kappa]$; and homomorphically evaluates $z = r + \sum_{i=0}^{l-1} (\rho \cdot etan_i^{2^j})^{2^k}$. z is then broadcast to other each key holder sender $i \in [0, \varphi)$ if TFHE is used. Otherwise, z is transmitted to the receiver.
5. [**Decrypt and get result**]
   a. **Partial decryption**: If using the TFHE functionality, upon receiving z, each key share holder $i \in [0, \varphi)$ partially decrypts the message to obtain $part_i$ using its share of the secret key sk_i and sends $part_i$ to the receiver.
   b. **Final decryption**: The receiver decrypts the ciphertext z and obtains a resulting vector of length $N/2$. If using TFHE, the receiver combines the partial descriptions received from the φ senders, $\{part_i : sk_i\}_{i \in [0, \varphi)}$ using pk and its share of the secret key sk_i and obtains the resulting vector of length $N/2$.
   d. **Interpretation of result**: The receiver decrypts z (either with basic FHE decryption or by combing TFHE partial descriptions), and checks if the magnitude of any element of the resulting vector exceeds τ, $\{y\} \cap \bigcup_{i=0}^{l-1} X_i = \{y : \text{FHE.Decrypt(z)} \geq \tau\}$.
   If any value in a ciphertext slot is greater than the threshold τ, an intersection exists; otherwise, no intersection exists.

Fig. 4. Full PSMT protocol

Since piece-wise functions cannot be evaluated efficiently in FHE, we designed a function that gets larger quickly as the input gets closer to zero. In $D_{K,S}^{Approx}$, K acts as the maxima, and S controls the input range for zero (or width of the function's peak) such that the function outputs zero for inputs in the range $(-\infty, -1) \cup (1, \infty)$. Using this function for approximation, senders acquire a summation value that is equal to K when an intersection is detected; otherwise, they obtain a summation value that is close to zero.

When computing functions such as $D_{K,S}^{Approx}(x)$ in approximate HE, only additions and multiplication operations are available. However, such approximations are not zero-valued for all nonzero arguments or even all arguments outside some interval centered at zero. In the following sections, we described how we compute a polynomial approximation $D_{K,S}^{Approx}(x)$ on a large interval whose values on $\mathbb{R}$ except for a small interval near zero can be bounded by some small number κ. Then, we can set parameters K, S, κ, ν such that $(l+1) \cdot \kappa < \tau$, where $\tau = \nu \cdot K$. $\nu \in (0, 1]$ is a threshold proportion used to account for the case where error from approximate-arithmetic FHE may cause a sum of l outputs of $D_{K,S}^{Approx}(x)$ on nonzero arguments to be greater than $(l+1) \cdot \kappa$.

D. Polynomial Approximation of VAF

The approximation of complex non-polynomial functions is a well-studied topic. Existing works such as [14], [15] use polynomial composition with iterative algorithms to approximate functions like min/max and comparison using FHE. Other works have used techniques like Taylor series [25], minimax approximation [12], look-up tables [19] and conversion between FHE schemes [17]. A major challenge for approximation techniques is finding polynomial approximations that work on large domains. Existing approaches for approximation [13], [17], [25] struggle for large domain intervals, inducing very high approximation errors. Even domains of thousands can be challenging. This would require homomorphically evaluating polynomials of an extremely large degree. For instance, simply using Chebyshev-based approximation for approximating non-linear functions for domains such as $[-1000, 1000]$, requires almost 20.5 minutes of computation time and an additional 12 multiplicative depth for acceptable accuracy. Hence, the error induced by the approximation and the computational cost of approximation are the major factors [16] to be considered.

Domain Extension Polynomials (DEPs). DEPs enable the effective shrinking of a large domain interval $[-L^n R, L^n R]$

to a smaller subinterval $[-R, R]$ such that the property of the VAFs around zero in the smaller subinterval is preserved. To compress inputs from an interval $[-L^n R, L^n R]$ to an interval $[-R, R]$, we can iteratively apply a DEP with $O(n)$ operations and $2n$ additional depth. Using this method we can convert inputs $z \in [-L^n R, L^n R]$ into values $D(z)$ such that $z \in [-R, R] \implies D(z) \approx z$ and $z \notin [-R, R] \implies D(z) \approx sign(z)$. Specifically, we utilize Algorithm 1 from [16]. To handle inputs in $[-R, R]$ using the DEP $B(z) = z - \frac{4}{27}z^3$ at an iteration n of Algorithm 1, we divide inputs to $B(z)$ by $L^n R$, and scale its ouputs by $L^n R$. This converts $B(z)$ from a DEP on $[-1, 1]$ to a DEP on $[-R, R]$. If the accuracy of $B(z)$ is not good enough, then squaring its outputs can make $B(0)$ larger and $B(z)$ for $z \neq 0$ smaller, at the cost of additional depth and runtime. DEPs enable the approximation of values within a large domain interval, allowing the protocol to efficiently handle potentially millions of inputs.

Chebyshev Approximation. Chebyshev polynomial approximation is are minimax-based polynomial approximation method that achieves the smallest possible polynomial degree with minimal approximation errors. Chebyshev polynomials reduce the Runge phenomenon that causes an approximation to yield poor accuracy at the edges of the interval and provide an approximation that is close to the best polynomial approximation to a continuous function.

We applied DEPs and Chebyshev-based approximation to approximate a VAF in a much smaller interval $[-R, R]$ while preserving the original larger domain $[-L^n R, L^n R]$ with low FHE multiplicative depth. This leads our protocol to approximate $etan_i$ with low approximation errors and computational cost for a significantly large domain. Over the course of our experiments, we encountered several challenges in determining suitable parameters for the DEP and Chebyshev-based approximation while tailoring them to the specific sizes of sender sets. Using $B(x) = x - \frac{4}{27}x^3$, it is imperative to maintain L below $1.5 \times \sqrt{3}$ while using DEPs. Expanding the domain's range involves increasing R and n so that $[-L^n R, L^n R]$ adequately accommodates large input sets without amplifying errors and computation. Thus, finding the right balance for DEP parameters and trying to simultaneously make both the Chebyshev polynomial approximation degree and computational depth as small as possible was quite challenging. We list our parameter findings in Table II for different senders' set sizes denoted by $|\mathcal{X}|$. We note that the values described in Table II are useful for the case where $\bigcap_{i=1}^{l} X_i = \varnothing$, i.e., each sender's set has no element in common with other sender sets.

E. Set Updates

As we do not require any preprocessing of senders' sets, updates to the encrypted senders' sets are trivial in our protocol. Most computations in our protocol take place during the online phase of the protocol. This allows the senders to easily add or delete any element in their sets, as no preprocessing is necessary during the offline phase. Parties can compute the intersection of their private sets on a regular basis with sets that are often updated. In previous works, when parties update their sets, various preprocessing operations were essential [3], which is difficult on sets held in the encrypted form. Our protocol does not suffer from such issues.

F. Security and Correctness

Using our basic protocol in Figure 3, the receiver can learn the aggregation value $\sum_{i=0}^{l-1} etan_i$ to infer information about the difference between their query and the sender's value to get information about the non-intersection values. To fix this issue, we require the senders to obscure the aggregation value by adding a small random masking term $r \in [-\kappa, \kappa]$, where κ is public and the bound for the approximation value of VAF on a non-intersection.

We prove the security of our protocol in the *semi-honest model* with *static adversaries*, assuming that all protocol participants run Probabilistic Polynomial Time (PPT). Adversaries can be any subset of corrupted parties and static adversaries are a set of corrupted parties that are determined before the execution of the protocol. We do not consider membership inference attacks with repeated adaptive queries or integrity-based attacks where an adversary may homomorphically modify the sender's set. The semantic security of the PSMT protocol follows directly from the IND-CPA security of the underlying FHE (or TFHE) scheme, and we provide proof for such security in the following sections.

1) Protocol Correctness: It is easy to see that the protocol correctly computes the intersection conditioned on the VAF approximation succeeding in approximating the function $etan_i$. The approximation succeeds if proper DEP and Chebyshev parameters for a given senders' set size are used from Section IV-D, and homomorphic squaring and scaling techniques are applied to eliminate false positives (see Section V-C).

2) Security Against Semi-Honest Adversaries: We prove the security of the protocol under the assumption of the existence of an IND-CPA secure TFHE scheme Π with a threshold l for l parties. Here, we consider the protocol participants as corrupted by the *adversary* where they are honest but curious. Loosely put, we say that the protocol Π_{PSMT} of Figure 4 realizes the functionality of $\mathcal{F}_{PSMT}$, if it is correct.

Theorem 1. *The protocol in Figure 4 is a secure protocol for $\mathcal{F}_{PSMT}$ in the semi-honest setting.*

Proof. We consider two classes of adversaries. The first class involves adversaries that corrupt a subset of the sender parties, including the receiver P_y. The second class involves adversaries that corrupt a subset of the sender parties, not including the receiver P_y. The number of corrupted parties is denoted by η and by definition $\eta < l$. We provide a separate simulation for each class.

Consider an adversary $\mathcal{A}$ that corrupts a strict subset $\mathcal{I}$ of parties from the set $\{P_y, P_0, \ldots, P_{l-1}\}$, including P_y. We define a simulator $\mathcal{S}$ equipped with a functionality $\mathcal{S}_{\text{GEN}}$ to generate keys of the underlying encryption scheme as follows:

TABLE II
PSMT PROTOCOL PARAMETERS FOR DIFFERENT SENDERS' SET SIZES WHERE K=1 AND S=10

$\|\mathcal{X}\|$	DEP			Chebyshev Parameters		Optimization			Aggregation Result	
	L	R	n	Degree	κ	j	k	ρ	Non-intersection	Intersection
2^7	2.50	21.0	2	27	3.4×10^{-5}	3	3	2.5	3.5×10^{-5}	1528.18
2^8	2.56	16.0	3	27	3.2×10^{-1}	1	3	2.5	3.3×10^{-1}	1526.54
2^{10}	2.58	24.0	4	27	1.8×10^{-6}	4	3	2.5	1.5×10^{-5}	1525.88
2^{13}	2.58	27.5	6	27	5.6×10^{-3}	4	3	2.5	5.7×10^{-3}	1525.88
2^{15}	2.58	43.5	7	27	3.7×10^{-1}	4	3	2.5	3.8×10^{-1}	1526.63
2^{20}	2.59	200.0	9	247	1.2×10^{-1}	2	3	2.5	1.2×10^{-1}	1526.11
2^{21}	2.59	400.0	9	247	7.3×10^{-1}	4	3	2.7	7.4×10^{-1}	2824.66
2^{22}	2.59	800.0	9	247	7.6×10^{-1}	6	3	2.7	7.8×10^{-1}	2824.29
2^{23}	2.59	1600.0	9	247	6.3×10^{-1}	8	3	2.7	6.5×10^{-1}	2824.00
2^{24}	2.59	3200.0	9	247	7.9×10^{-1}	10	3	2.7	8.4×10^{-1}	2824.20
2^{25}	2.59	6400.0	9	247	2.7×10^{-1}	12	3	2.7	3.0×10^{-1}	2823.92

1. Given $\{X_i\}_{i\in\mathcal{I}}$, which also includes P_y's input, and $Z = \cap_{i=0}^{l-1} X_i$, the simulator $\mathcal{S}$ invokes the corrupted parties on their corresponding inputs.
2. $\mathcal{S}$ generates $(pk,\ sk_i)$ by invoking the simulator's functionality $\mathcal{S}_{\text{GEN}}$ during the key generation phase.
3. Next, $\mathcal{S}$ plays the role of honest parties against P_y on arbitrary sets of inputs. Namely, $\mathcal{S}$ computes the subtractions, applies the DEP procedure, evaluates the VAF, and performs squaring and scaling. $\mathcal{S}$ then sends ciphertexts encrypting these inputs to the leader, who summates them and transmits the summated values.
4. Upon receiving the result, when P_y decrypts and interprets it, the simulator completes the decryption procedure as follows. If $y \in Z$, $\mathcal{S}$ homomorphically forces the decryption outcome to exceed the threshold value in one of the plaintext slots by adding the result ciphertext to another ciphertext containing a value greater than τ. If $y \notin Z$, the simulator ensures the decryption outcome does not exceed the threshold value in any plaintext slot by homomorphically multiplying the resulting ciphertext with a ciphertext encrypted with zeroes and then adding a random element $r \in [-\kappa, \kappa]$. This guarantees that all values of the ciphertext remain below threshold τ.

The simulator $\mathcal{S}$ needs to produce simulated ciphertexts towards η corrupted parties that are homomorphically evaluated and indistinguishable from the real ciphertexts because this is the only message that appears in the view of the corrupted party. Note that η is smaller than the threshold l, and there is at least one honest party, in particular $l-\eta$ honest parties. This makes sure that the ciphertexts will be indistinguishable even though η adversaries collude with each other because there will be at least one honest party holding a share of the secret key. The indistinguishability follows from the IND-CPA security of the underlying TFHE scheme Π, since both the simulated ciphertext and real ciphertext look like fresh encryptions of the scheme Π. Hence, the view of the corrupted receiver to the ciphertext received from the simulator is indistinguishable from the receiver's view in the real execution of the protocol.

Next, we consider an adversary that does not corrupt P_y. In this case, the simulator $\mathcal{S}$ is defined as follows.

1. Given $\{X_i\}_{i\in\mathcal{I}}$, which does not include P_y's input, and $Z = \cap_{i=0}^{l-1} X_i$, the simulator invokes the corrupted parties on their corresponding inputs.
2. $\mathcal{S}$ generates $(pk,\ sk_i)$ by invoking the simulator's functionality $\mathcal{S}_{\text{GEN}}$ during the key generation phase.
3. Next, $\mathcal{S}$ plays the role of P_y against the corrupted senders on an arbitrary set of inputs and concludes the simulation by playing the role of P_y on these arbitrary inputs. This corruption case is simpler as only P_y learns the output.

In this case, where we have only η corrupt senders, the simulator $\mathcal{S}$ can simply generate new encryptions of zero in the place of real encryptions of the receiver's query element y. By the IND-CPA security of the TFHE scheme, this ciphertext is indistinguishable from the senders' view in the real protocol. The ciphertexts will be indistinguishable even though η adversaries collude with each other because there exists at least one honest party, including the party P_y who holds a share of the secret key. This concludes the proof for the second case.

□

G. Asymptotic Complexity Analysis

Each sender must compute the procedures described in Figure 4. This requires $2n + O(log(c)) + j + k$ homomorphic multiplications. These procedures require a multiplicative depth of $2n$ for the application of the DEP, approximately $log(c)+1$ for Chebyshev polynomial approximation, and $j+k$ for repeated squaring.

CKKS operations, besides multiplication, e.g., addition and scaling, contribute relatively small amounts of overhead and noise. The additive aggregation of each sender's result remains cost-effective, not increasing multiplicative depth even with a growing number of senders. In the case of TFHE with α parties holding keys, communication between the receiver and other parties is capped at $l+\alpha$ ciphertexts, while inter-sender communication is capped at $l+\alpha-1$ ciphertexts. Without TFHE, there are three communication rounds between the receiver and senders, increasing to four with TFHE due to an extra round for partial decryption by keyholding senders. The computation requirement for PSMT is determined by the total number of homomorphic multiplications needed by the senders for the intersection steps in Figure 4.

Tradeoffs: The process of applying a DEP followed by Chebyshev polynomial evaluation requires significant depth, but this depth does not depend on the number of senders involved. Each sender can perform this computation in parallel, but this step may be expensive. Thus, even though we require a higher computational complexity compared to other methods for a small number of senders, this complexity does not depend on the number of senders and stays low even for a significantly higher number of senders. PSMT's computation and FHE multiplicative depth only depend on the size of the sender's set and not the number of senders.

V. Challenges

A. Increasing Throughput

The throughput of our PSMT protocol can be improved using the SIMD feature available in FHE schemes. The CKKS scheme supports the packing of multiple message vectors into a single ciphertext where the slots of the ciphertexts hold different values. This allows slot-wise addition and multiplication [38]. Using this mapping, the CKKS scheme can encrypt $b = N/2$ elements in $\mathbb{R}$ into a single ciphertext. We note that FHE batching can only be applied to, at most, one of the receiver or the senders. To enable batching when not all slots are used, we fill the remaining slots with a dummy value, which is negligible and does not alter the intersection results.

B. Supporting Larger Sets

Our protocol can be extended to use very large sender sets. However, we note that it will require more depth and computing power due to the large FHE parameters required for the DEP procedure and Chebyshev approximation. Since our final result is the summation of individual results from the senders, increasing the senders' set sizes does not dramatically increase the communication size.

C. Reducing False Positives

In PSMT, false positives can occur due to bad approximation accuracy of the VAF. After applying the DEP procedure, we found that the homomorphic approximation for the VAF was not accurate enough within the range $[-3, 0] \cup [0, 3]$. While the approximation would correctly map zero values to K, the non-zero values in the range $[-3, 0] \cup [0, 3]$ would be mapped to values far too close to K. Hence, instead of producing a hump at 0, we observed that the approximation would produce a flatter curve. These values would later contaminate the summation outcome, leading to false positives. To solve this issue, we used a technique involving homomorphic squaring and scaling. Homomorphically squaring the values in the range $(-1, 0) \cup (0, 1)$ would map them to smaller and smaller values. Then, we would scale them by a scaling factor $\rho \in (2, 2.8)$ such that they remain within $(-1, 0) \cup (0, 1)$. Finally, we apply homomorphic squaring again, which would square ρ for intersection and augment the difference between values mapped to zero and non-zero. After using the aforementioned techniques, κ is the highest value approximated by the Chebyshev when diff_i (see Figure 4) is at its minimum value of one.

D. Decentralizing Data Security

To decentralize the data security in PSMT, we propose to use TFHE. During setup, instead of giving an FHE secret key to the receiver, a trusted setup procedure (which can be run via trusted hardware or secure multiparty computation) will distribute shares of the TFHE secret key to φ parties, which may be any of the senders, receiver, or other parties. After the senders' results have been homomorphically summed, the resulting ciphertext z is broadcast to the φ key shareholders. Each of these parties performs partial decryption and then sends their result to the receiver, which combines the partial decryptions to get the final result. In our protocol, the receiver is given one of the TFHE secret keys to handle up to $\varphi - 1$ collusion level among the parties.

Our complete protocol after incorporating the aforementioned solutions is detailed in Figure 4.

VI. Evaluation

We implemented our proposed protocol using C++17 and OpenFHE v1.0.3 [1]. The anonymized source code is available at https://anonymous.4open.science/r/psmt-1323/. We use the CKKS encryption scheme [13] and OpenFHE's default parameter settings to ensure 128-bit security ($\lambda = 128$). Our experiments were run on a server with an AMD EPYC 7313P processor and 128GB of memory, running Ubuntu 20.04.

We evaluated the latency of per-sender query runtime and the runtime for aggregating results from multiple senders by performing 5 PSMT runs for each combination of parameters and taking the average. The bit-length δ of the set elements is matched to the maximum plaintext space accommodated by the DEP and Chebyshev parameters. The senders' individual computations are independent in our protocol. We assume that after receiving the query ciphertext from the receiver, each sender operates on their own set in parallel.

Baselines. Many variants of PSI, MPSI, and PMT protocols are designed to handle specific or more general scenarios. Selecting a specific protocol for comparison with our work is a complex task, given our novel privacy model with server-side encryption and a high number of senders. Considering the existing works that we discussed in Section II, [27], [2], [37], [35] and [8] either lack scalability for a large number of senders, do not provide a public implementation or lack post-quantum security. [11] is an older work and serves as the foundation for [10]. Thus, we primarily compare our protocol to the following state-of-the-art baseline FHE-based and non-FHE-based PSI and MPSI protocols: Cong *et al.*, [18], Bay *et al.* [4] and Nevo *et al.* [30]. Cong *et al.*'s protocol is based on [10], which uses the BFV FHE scheme [21]. We implemented their protocol for a multiparty setting using OpenFHE (based on C++), which was originally implemented using the SEAL library. Similarly, we ran the public implementation of the MPSI protocols of Bay *et al.* and Nevo *et al.* (both in C++) on our server for performance comparison.

TABLE III

COMMUNICATION AND COMPUTATION OVERHEAD FOR PSMT. AGGREG. DENOTES THE TOTAL TIME REQUIRED TO AGGREGATE THE CIPHERTEXTS FOR THE CORRESPONDING NUMBER OF SENDERS. QUERY REFERS TO THE QUERY COMPUTATION TIME FOR SENDERS.

$\|\mathcal{X}\|$	**Msg. size (MB)**		**# of Senders**	**Aggreg.** (second)	**Query** (second)	**Depth**
	R-to-S	S-to-R				
2^7	43	4.1	128	6.53	13.20	21
2^8	49	4.1	256	14.12	15.24	23
2^{10}	63	4.1	1024	40.29	19.09	30
2^{13}	79	4.1	1024	67.51	26.00	38
2^{15}	87	4.1	1024	71.16	30.07	42
2^{20}	107	4.1	1024	87.78	59.67	52
2^{21}	111	4.1	1024	103.98	64.65	54
2^{22}	115	4.1	1024	84.63	68.90	56
2^{23}	119	4.1	1024	90.34	75.80	58
2^{24}	123	4.1	1024	84.09	81.08	60
2^{25}	127	4.1	1024	92.53	86.76	62

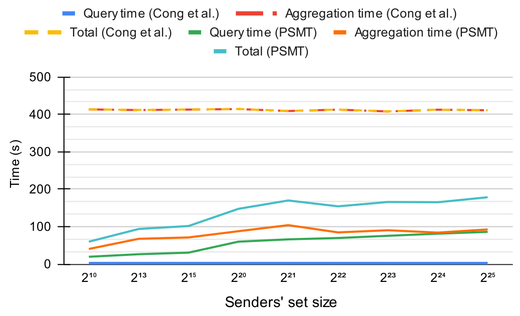

Fig. 5. Computation time comparison for Cong *et al.* [18] vs PSMT for different senders' set size. Number of senders is set to 1024.
**Bay *et al.* and Nevo *et al.* were omitted due to inefficiency for larger set sizes and a high number of senders, respectively.

Table III shows query computation time, which is the total time required to complete the DEP and Chebyshev approximation using the parameters in Section IV-D. In terms of computation, for 1024 senders, the aggregation time (not counting encryption/decryption) of PSMT stays within 100 seconds for senders' set sizes up to 2^{25} and our protocol can be easily extended for larger presets (4096 senders with a 2^{25} set size in Figure 6). We note that we only report computational latency for only up to 4096 parties, but this is only due to memory constraints when running the protocol with a huge number of senders in a single system.

Comparison with FHE-based protocol. We compare the computational latency of PSMT and Cong *et al.*'s protocol in Figures 5 to 8. For the number of senders less than 150, Cong *et al.* 's protocol is faster than PSMT, but for higher senders, PSMT outperforms Cong *et al.* As the number of senders increases, aggregation time for Cong *et al.* becomes the bottleneck due to *multiplicative* aggregation. The query computation time for PSMT is slower than Cong *et al.* 's in Figure 5, but PSMT is almost $11\times$ faster when the aggregation time is also taken into account for the overall computational latency. In terms of communication, Cong *et al.* 's protocol is better than PSMT due to the small FHE parameters required for query computation. PSMT, however, operates on encrypted senders' sets, unlike Cong *et al.* , where senders' sets are needed in plaintext for enabling various optimizations. This also makes

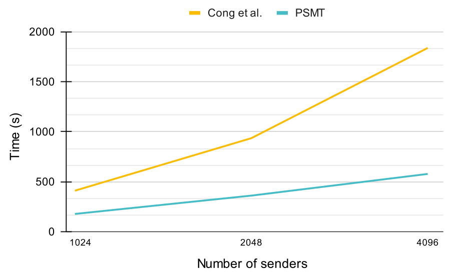

Fig. 6. Total computation time comparison for Cong *et al.* [18] vs. PSMT for different numbers of senders up to 4096. Senders' set size is set to 2^{25}.
** Bay *et al.* and Nevo *et al.* omitted due to inefficiency for set size of 2^{25}.

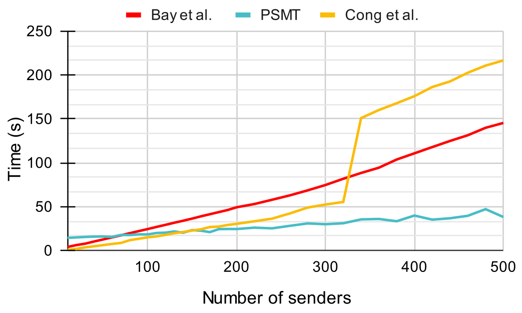

Fig. 7. Runtime comparison for Bay *et al.*'s MPSI [4] protocol with PSMT and Cong *et al.* [18]
** Nevo *et al.* omitted due to inefficiency for high number of senders

their scheme undesirable for applications that require frequent database updates. Both protocols have negligible false positive rates when proper scheme parameters are employed.

Comparison with non-FHE-based protocols. For the non-FHE-based protocols, multiplicative depth is not a significant issue, but most of these protocols often rely upon high communication rounds. These protocols either support large senders' set size or a large number of senders but not both due to huge communication overhead. We compared both our work and *Cong et al.* to Bay *et al.* [4] in the MPSI setting. It is important to note that their protocol induces a false positive probability of 1% due to bloom filters, while PSMT has negligible false positives. Similarly, they require five rounds of communication, and PSMT requires four rounds of communication when TFHE is employed, which can handle up to l collusion among the sender parties.

In terms of communication, PSMT is better than Bay *et al.*, and the overall communication in PSMT is bounded by $\mathcal{O}(l+\alpha)$ ciphertexts. In Bay *et al.*, senders can have duplicate elements among each other, but in PSMT, we assume that all senders have distinct elements. Hence, for the number of parties higher than 128 and 256 in PSMT and Cong *et al.* , we set the senders' set size of 2^8 and 2^9, respectively, even though this results in higher computational latencies for the two protocols. Finally, Bay *et al.* accesses the senders' set element in plain for encoding, while PSMT operates on encrypted sets. We report the runtime comparison in Figure 7. Cong *et al.* and Bay *et al.* are more efficient than PSMT for parties less than 50 and 150, respectively. PMST is up to $4\times$ and $5.5\times$ faster than Bay *et al.* and Cong *et al.* respectively for 500 senders. Nevo *et al.* [30] provide an MPSI protocol that

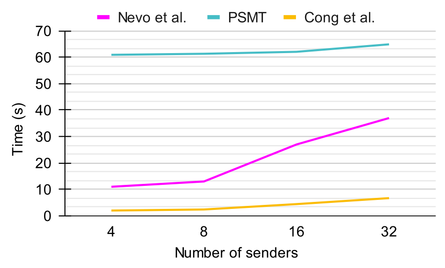

Fig. 8. Runtime comparison for Nevo *et al.*'s MPSI [30] with PSMT and Cong *et al.* [18]. Senders' set size is set to 2^{20}.
**Bay *et al.* 's omitted due to inefficiently for larger set sizes.

is secure in malicious setting. We used their highest available preset for the protocol by setting the number of senders to 32, senders' set size to 2^{20}, and collusion rate to 16. Figure 8 shows the runtime comparison for Nevo *et al.* with PSMT and Cong *et al.* where PSMT is slower than both protocols, but PSMT supports a very high number of parties, with a low overhead for an increasing number of parties. In terms of communication, *Nevo et al.* requires four communication rounds and nearly 8.4GB of data to be sent/received data across all parties for 15 parties (Table 4 in [30]) with a collusion level set to 14 and a sender set of 2^{20}. In comparison, PSMT achieves a significantly lower communication cost of a total of 4.6GB, showing an improvement of nearly $1.8\times$ for the same set of presets.

Comparison with other works. [27] is close to our work in the MPSI setting; however, it scales very poorly for a large number of parties, taking almost 300 seconds of computation for only 15 parties. [5] provide a maliciously secure MPSI based on garbled bloom filters and k-out-of-N OT. [5] and [27] has been compared against [30] hence we do not compare against [5]. [41] achieve maliciously secure multiparty PSI using bloom filters for large inputs; however, they do not provide a public implementation for comparison.

VII. Conclusion

In this work, we introduce the concept of a Private Segmented Membership Test (PSMT) for cases where users wish to query a set held distributed among many data holders. We show a basic PSMT protocol based on approximate-arithmetic FHE and provide details about overcoming various technical challenges to make our solution feasible. Our method protects the values of senders' elements from even the senders and allows for preprocessing-free updates. Our experiments demonstrate the scalability of our protocol that aggregates the penultimate results from multiple data holders. In future work, we aim to explore new avenues for further optimizations and better communication overhead in PSMT.

References

[1] Ahmad Al Badawi, Jack Bates, Flavio Bergamaschi, David Bruce Cousins, Saroja Erabelli, Nicholas Genise, Shai Halevi, Hamish Hunt, Andrey Kim, Yongwoo Lee, et al. Openfhe: Open-source fully homomorphic encryption library. In *WAHC'22*, pages 53–63, 2022.

[2] Saikrishna Badrinarayanan, Peihan Miao, Srinivasan Raghuraman, and Peter Rindal. Multi-party threshold private set intersection with sublinear communication. pages 349–379. Springer, 2021.

[3] Saikrishna Badrinarayanan, Peihan Miao, and Tiancheng Xie. Updatable private set intersection. *PoPETS*, 2022(2), 2022.

[4] Aslı Bay, Zekeriya Erkin, Jaap-Henk Hoepman, Simona Samardjiska, and Jelle Vos. Practical multi-party private set intersection protocols. *IEEE Trans. Inf. Forensics Secur.*, 17:1–15, 2021.

[5] Aner Ben-Efraim, Olga Nissenbaum, Eran Omri, and Anat Paskin-Cherniavsky. Psimple: Practical multiparty maliciously-secure private set intersection. In *ASIA CCS '22*, pages 1098–1112, 2022.

[6] Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter MR Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In *Advances in Cryptology – CRYPTO 2018 , Part I 38*, pages 565–596. Springer, 2018.

[7] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *TOCT*, 6(3):1–36, 2014.

[8] Nishanth Chandran, Nishka Dasgupta, Divya Gupta, Sai Lakshmi Bhavana Obbattu, Sruthi Sekar, and Akash Shah. Efficient linear multiparty psi and extensions to circuit/quorum psi. In *CCS '21*, pages 1182–1204, 2021.

[9] Guoxing Chen, Sanchuan Chen, Yuan Xiao, Yinqian Zhang, Zhiqiang Lin, and Ten H Lai. Sgxpectre: Stealing intel secrets from sgx enclaves via speculative execution. In *2019 EuroS&P*, pages 142–157. IEEE, 2019.

[10] Hao Chen, Zhicong Huang, Kim Laine, and Peter Rindal. Labeled psi from fully homomorphic encryption with malicious security. In *CCS '18*, pages 1223–1237, 2018.

[11] Hao Chen, Kim Laine, and Peter Rindal. Fast private set intersection from homomorphic encryption. In *CCS '17*, pages 1243–1255, 2017.

[12] Jung Hee Cheon, Jinhyuck Jeong, Joohee Lee, and Keewoo Lee. Privacy-preserving computations of predictive medical models with minimax approximation and non-adjacent form. In *International Conference on Financial Cryptography and Data Security*, pages 53–74. Springer, 2017.

[13] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology - ASIACRYPT 2017, Part I 23*, pages 409–437. Springer, 2017.

[14] Jung Hee Cheon, Dongwoo Kim, and Duhyeong Kim. Efficient homomorphic comparison methods with optimal complexity. In *Advances in Cryptology – ASIACRYPT 2020, Part II 26*, pages 221–256. Springer, 2020.

[15] Jung Hee Cheon, Dongwoo Kim, Duhyeong Kim, Hun Hee Lee, and Keewoo Lee. Numerical method for comparison on homomorphically encrypted numbers. In *Asiacrypt 2019*, pages 415–445. Springer, 2019.

[16] Jung Hee Cheon, Wootae Kim, and Jai Hyun Park. Efficient homomorphic evaluation on large intervals. *IEEE Trans. Inf. Forensics Secur.*, 17:2553–2568, 2022.

[17] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Tfhe: fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91, 2020.

[18] Kelong Cong, Radames Cruz Moreno, Mariana Botelho da Gama, Wei Dai, Ilia Iliashenko, Kim Laine, and Michael Rosenberg. Labeled psi from homomorphic encryption with reduced computation and communication. In *CCS '21*, pages 1135–1150, 2021.

[19] Jack LH Crawford, Craig Gentry, Shai Halevi, Daniel Platt, and Victor Shoup. Doing real work with fhe: the case of logistic regression. In *WAHC '18*, pages 1–12, 2018.

[20] Sumit Kumar Debnath and Ratna Dutta. Provably secure fair mutual private set intersection cardinality utilizing bloom filter. In *Inscrypt 2023*, pages 505–525. Springer, 2016.

[21] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *Cryptology ePrint Archive*, 2012.

[22] Michael J Freedman, Yuval Ishai, Benny Pinkas, and Omer Reingold. Keyword search and oblivious pseudorandom functions. In *TCC*, volume 3378, pages 303–324. Springer, 2005.

[23] Michael J Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *Eurocrypt*, pages 1–19. Springer, 2004.

[24] Irina Ivanova. Rich americans hide "billions" offshore thanks to tax loophole, senate panel finds, Aug 2022. CBS News.

[25] Miran Kim, Yongsoo Song, Shuang Wang, Yuhou Xia, Xiaoqian Jiang, et al. Secure logistic regression based on homomorphic encryption: Design and evaluation. *JMIR medical informatics*, 6(2):e8805, 2018.

[26] Vladimir Kolesnikov and Ranjit Kumaresan. Improved ot extension for transferring short secrets. In *Advances in Cryptology–CRYPTO 2013. Proceedings, Part II*, pages 54–70. Springer, 2013.

[27] Vladimir Kolesnikov, Naor Matania, Benny Pinkas, Mike Rosulek, and Ni Trieu. Practical multi-party private set intersection from symmetric-key techniques. In *CCS '17*, pages 1257–1272, 2017.

[28] Atsuko Miyaji, Kazuhisa Nakasho, and Shohei Nishida. Privacy-preserving integration of medical data: a practical multiparty private set intersection. *Journal of medical systems*, 41:1–10, 2017.

[29] Daniel Morales, Isaac Agudo, and Javier Lopez. Private set intersection: A systematic literature review. *Computer Science Review*, 49:100567, 2023.

[30] Ofri Nevo, Ni Trieu, and Avishay Yanai. Simple, fast malicious multiparty private set intersection. In *CCS '21*, pages 1151–1165, 2021.

[31] John Newman and Ritchie Amy. Financial privacy rule, Nov 2023.

[32] Department of Homeland Security. Dhs use cases of privacy enhancing technologies, June 2022.

[33] Benny Pinkas, Thomas Schneider, Gil Segev, and Michael Zohner. Phasing: Private set intersection using permutation-based hashing. In *USENIX Security '15*, pages 515–530, 2015.

[34] Benny Pinkas, Thomas Schneider, Oleksandr Tkachenko, and Avishay Yanai. Efficient circuit-based psi with linear communication. In *Advances in Cryptology – EUROCRYPT 2019*, pages 122–153. Springer, 2019.

[35] Benny Pinkas, Thomas Schneider, and Michael Zohner. Scalable private set intersection based on ot extension. *ACM TOPS*, 21(2):1–35, 2018.

[36] The Washington Post. Foreign banks to help U.S. fight tax evasion, Apr 2023.

[37] Sara Ramezanian, Tommi Meskanen, Masoud Naderpour, Ville Junnila, and Valtteri Niemi. Private membership test protocol with low communication complexity. *Digital Comm. and Net.*, 6(3):321–332, 2020.

[38] Nigel P Smart and Frederik Vercauteren. Fully homomorphic simd operations. *Designs, codes and cryptography*, 71:57–81, 2014.

[39] Sandeep Tamrakar, Jian Liu, Andrew Paverd, Jan-Erik Ekberg, Benny Pinkas, and N Asokan. The circle game: Scalable private membership test using trusted hardware. In *ASIA CCS '17*, pages 31–44, 2017.

[40] ACC Yao. How to generate and exchange secrets. in—27th annual symposium on foundations of computer science, 1986.

[41] En Zhang, Feng-Hao Liu, Qiqi Lai, Ganggang Jin, and Yu Li. Efficient multi-party private set intersection against malicious adversaries. In *CCSW'19*, pages 93–104, 2019.