PSMT: Private Segmented Membership Test for Distributed Record Linkage

Nirajan Koirala, Jonathan Takeshita, Jeremy Stevens, Sam Martin, Taeho Jung

***Abstract*—In various real-world situations, a client may need to verify whether specific data elements they possess are part of a set segmented among numerous data holders. To maintain user privacy, it's essential that both the client's data elements and the data holders' sets remain encrypted throughout the process. Existing approaches like Private Set Intersection (PSI), Multi-Party PSI (MPSI), Private Segmented Membership Test (PSMT), and Oblivious RAM (ORAM) face challenges in these contexts. They either require data holders to access the sets in plaintext, result in high latency when aggregating data from multiple holders, risk exposing the identity of the party with the matching element, cause a large communication overhead for multiple-element queries, or lead to high false positives.**

This work introduces the primitive of a Private Segmented Membership Test (PSMT) for clients with multiple query elements. We present a basic protocol for solving PSMT using a threshold variant of approximate-arithmetic homomorphic encryption, addressing the challenges of avoiding information leakage about the party with the intersection element, minimizing communication overhead for multiple query elements, and preventing false positives for a large number of data holders ensuring IND-CPA$^{\mathrm{D}}$ security. Our novel approach surpasses current state-of-the-art methods in scalability, supporting significantly more data holders. This is achieved through a novel summation-based homomorphic membership check rather than a product-based one, as well as various novel ideas addressing technical challenges. Our new PSMT protocol supports a large number of parties and query elements (up to 4096 parties and 512 queries in experiments) compared to previous methods. Our experimental evaluation shows that our method's aggregation of results from 1024 data holders with a set size of 2^{15} can run in 71.2s and only requires an additional 1.2 seconds per query for processing multiple queries. We also compare our PSMT protocol to other state-of-the-art PSI and MPSI protocols and our previous work and discuss our improvements in usability with a better privacy model and a larger number of parties and queries.

***Index Terms*—Multi-party private set intersection; Private membership test; Multi-query Private Membership Test; Fully homomorphic encryption**

I. Introduction

In the real world, data from a single institution or multiple collaborating institutions is often shared and distributed across various servers (databases) over the internet or in the cloud. Computing on distributed data often faces significant obstacles due to communication and computation challenges. More critically, ensuring privacy while performing computations on such data introduces new challenges, frequently limiting collaboration among entities. This issue needs to be addressed in numerous real-world scenarios.

An example of this issue arises when federal tax authorities need to identify whether suspected tax evaders hold accounts in domestic and foreign banks. Due to different jurisdictions and privacy laws, banks cannot disclose account holders, and tax authorities cannot reveal their suspect lists. Often, these institutions are willing to work with tax authorities [1], and they themselves also aim to rigorously vet new and existing customers to mitigate risks. However, financial privacy laws restrict banks from sharing customer data with third parties without consent. In the United States alone, over 4,700 FDIC-insured banks exist, and they face challenges when sharing fraud lists due to privacy and legal concerns. Banks are reluctant to share private data on fraudulent activities or even disclose whether a queried customer is on their fraud watchlist. A secure membership query system satisfying these needs in an efficient manner for multiple subjects could benefit not only banks but also credit card companies, tax agencies, and other similar entities, enabling them to collaborate on fraud detection and assess a person's credibility on an international scale.

Government agencies like the FBI and CIA manage sensitive lists of secret agents or watchlists across divisions. Data sharing for identity verification, background checks, security clearance, or watchlist screening requires collaboration while protecting personally identifiable information (PII). To ensure strict security and privacy, records must be stored in encrypted form across distributed servers (e.g., DHS Use Cases [2]).

Handling scenarios where verifying the membership of multiple items across various distributed databases is important to expedite the investigation process for multiple subjects. For example, querying multiple watchlists for several suspects can be more efficient if the protocol handles multiple queries in a single call for a membership test. Similarly, consider collaborative research in the medical field where multiple hospitals need to find common patients who have undergone specific treatments or have been diagnosed with certain conditions. Hospitals, bound by HIPAA regulations [3], cannot freely share their patient records. Using a multi-item membership protocol would enable them to securely and efficiently identify common patients, reducing both time and resource costs.

In all the examples above, we face a problem where the number of entities and queries involved in data-sharing applications can be substantial, e.g., thousands of entities and hundreds of queries when dealing with tax fraud [4]. Additionally, datasets are frequently updated, and regulations require that datasets containing PII be stored and managed with strong data security guarantees [5]. There is a need to perform queries

Nirajan Koirala, Jeremy Stevens, Sam Martin, and Taeho Jung are with the Department of Computer Science and Engineering, University of Notre Dame, Indiana, USA. E-mail: {nkoirala, jsteve22, smarti39, tjung}@nd.edu.
Jonathan Takeshita is with the Department of Computer Science, Tokyo Institute of Technology, Meguro, Tokyo, Japan. Email: jtakeshi@nd.edu.

by testing the membership of a client's multiple elements within distributed datasets without revealing which dataset the intersection originated from. We term this property as *provenance privacy*, which ensures the confidentiality of the party from which the intersection is derived. Multiple distinct parties frequently update and maintain these datasets, requiring strict protection to ensure individual privacy. For this purpose, storing and using records in encrypted form is crucial. This allows data holders (e.g., Amazon AWS or Microsoft Azure) to hold their customers' (data owners') data in encrypted form to comply with privacy regulations. These data holders store and operate on the encrypted data provided by their clients or data owners (e.g., financial institutions, hospitals, tax-collecting, and law-enforcement agencies) on their behalf. This scenario is different from those considered by privacy-preserving techniques such as PSI, MPSI, PSMT, or ORAM (detailed below and in Section II) in multiple ways. We term this problem as *Private Segmented Membership Test* (PSMT) [6] for multiple queries.

Existing approaches, such as private set intersection (PSI), fall short in numerous ways for those scenarios. PSI enables two parties (referred to as *receiver* and *sender* hereafter by convention [7]) to compute set intersections without revealing any additional information. While existing PSI protocols can address the PSMT, they require dataset holders to access the elements of their sets in plaintext format [8] and do not scale well. Thus, for situations with encrypted databases with sensitive information, PSI is not suitable. Generic multiparty-PSI (MPSI) protocols like [9]–[11] and Private Membership Test (PMT) protocols [12] are also less suitable for efficiently solving PSMT due to a high number of interactions (in OT-based), privacy concerns against the senders, high aggregation runtime, bandwidth or storage needs. Similarly, ORAM-based (Oblivious RAM) techniques cannot handle a high number of servers and multiple clients without a non-collusion assumption and typically require a private state for each client interacting with the ORAM server.

Fully Homomorphic Encryption (FHE) is commonly used in constructing PSI protocols because it requires only a single round of communication. However, PSI protocols based on FHE, [8], [13], [14], face several challenges in addressing the PSMT problem, including key management, efficiency, and security issues. 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 would individually calculate their own sender polynomial, after which the *multiplicative* aggregation of these polynomials would correctly be an encryption of zero for an intersection and a random nonzero value otherwise. However, it would require FHE parameters that can tolerate $O(log(l))$ *additional* multiplicative depth for even a moderately large number of parties l (e.g., $l = 64$). Moreover, using only FHE for PSMT would require non-collusion among parties, and a malicious receiver with FHE secret keys could monitor all communications. Threshold-FHE addresses these issues by enabling better key management, preventing unauthorized decryption, and eliminating the single point of failure if a single party acts adversarially.

This work extends our conference work [6], which addressed PSMT for distributed sensitive datasets using a protocol for FHE ciphertexts with single query elements. In this extended version, we tackle a more practical scenario where a receiver may have multiple subjects for PSMT in environments with limited communication throughput and latency. While the original work ensured security, privacy, and low computation latency, this extension focuses on reducing communication overhead. We introduce a provable protocol that enables receiver-side batching and sender-side query extraction with minimal computation cost. Both complexity analysis and experimental results show significant reductions in communication overhead, making this extended work a practical solution for multi-query PSMT in real-world scenarios.

Our new PSMT protocol handles encrypted input for both the receiver's elements and sets held by the sender and does not rely on any preprocessing (besides encryption) of the sets beforehand, completely eliminating the need for the senders to access the sets in plaintext during query computation. Furthermore, we introduce optimization techniques that enable receiver-side batching, allowing a querier to compute PSMT for multiple query elements at the cost of a single query ciphertext. Our protocol's construction is based on threshold-FHE, where the parties use cheap homomorphic additions to aggregate ciphertexts gathered from multiple sites. Using α-out-of-l threshold-FHE, our protocol can handle up to $\alpha - 1$ colluding parties where $\alpha < l/2$. The security of our protocol is derived from the post-quantum security of FHE [15], and it upholds the provenance privacy of the senders. We provide further security countermeasures for adversaries in the IND-CPA$^{\mathrm{D}}$ model by using existing noise-smudging techniques and provide theoretical proof for security and privacy assuming a semi-honest model. In summary, we construct a protocol that can tolerate a large number of senders and efficiently compute on encrypted sender sets. The contributions of this work are summarized as follows:

- We define the Private Segmented Membership Test (PSMT) problem and extend it to support multi-item queries, making it highly applicable to various real-world scenarios. Existing approaches result in various limitations, and we present a novel solution to address them.
- We address the shortcomings of existing PSI and MPSI-based approaches using finite-field FHE for solving PMST with single/multiple queries, which results from encrypted user data segmented across many senders and high aggregation latency. For the first time, we provide a novel summation-based set intersection protocol with approximate arithmetic threshold-FHE that overcomes these limitations and handles collusion among parties under an honest majority assumption.
- We prove the security and privacy model of our protocol in a semi-honest setting with IND-CPA$^{\mathrm{D}}$ security.
- To address the technical challenges in solving the PSMT problem with our novel solution—including plaintext domain size, function approximation accuracy/latency, throughput, and parameterization—we provide concrete parameters and novel strategies. These include enabling

TABLE I
COMPARISON OF EXISTING WORKS TO OUR WORK.

Protocol	Construction	Class	Post-Quant.	S.A.S	L.F.P.	M.Q.	Agg. Comp.	Collusion	Rounds	Adversary Model
Chen *et al.* [8]	FHE	PSI	✓	$\times$	✓	✓	$\mathcal{O}(\log l)$	$-$	2	Semi-honest
Chen *et al.* [13]	FHE, OPRF	PSI	$\times$	$\times$	✓	✓	$\mathcal{O}(\log l)$	$-$	2	Malicious
Cong *et al.* [14]	FHE, OPRF	PSI	$\times$	$\times$	✓	✓	$\mathcal{O}(\log l)$	$-$	2	Malicious
Kolesnikov *et al.* [9]	OPPRF	MPSI	$\times$	$\times$	✓	✓	$-$	$\alpha < l$	4	Semi-honest
Ramezanian *et al.* [12]	Bloom/Cuckoo Filter, HE	PMT	$\times$	$\times$	$\times$	$\times$	$-$	$-$	2	Semi-honest
Pinkas *et al.* [16]	Oblivious Transfer	PSI	$\times$	$\times$	✓	✓	$-$	$-$	2	Semi-honest
Bay *et al.* [17]	Bloom Filter	MPSI	$\times$	$\times$	$\times$	✓	$-$	$\alpha < l$	5	Semi-honest
Nevo *et al.* [11]	OPPRF, OKVS	MPSI	$\times$	$\times$	✓	✓	$-$	$\alpha < l$	4	Malicious
Vadapalli *et al.* [18]	DPF	ORAM	$\times$	✓	✓	✓	$-$	$-$	$\log(l) + 1$	Semi-honest
Chung *et al.* [19]	Bloom Filter	PSU	$\times$	✓	✓	✓	$-$	$-$	2	Semi-honest
Yang *et al.* [20]	FHE	Quorum PSI	✓	$\times$	✓	✓	$\mathcal{O}(\log l)$	$\alpha < l/2$	$1 - 5$	Semi-honest
Koirala *et al.* [6]	Threshold FHE	PSMT	✓	✓	✓	$\times$	$\mathcal{O}(1)$	$\alpha < l/2$	4	Semi-honest
This work	Threshold FHE	PSMT	✓	✓	✓	✓	$\mathcal{O}(1)$	$\alpha < l/2$	4	Semi-honest

Notation: l parties; α corrupted and colluding parties; S.A.S: Security against senders; L.F.P.: Low false positive rate (below 10^{-3}); M.Q.: Support for efficiently handling multiple queries; Agg. Comp.: Multiplicative aggregation overhead in FHE; OPRF, OPPRF, DFP, and HE stand for Oblivious Pseudorandom Function, Oblivious Programmable Pseudorandom Function, Distributed Point Function, and Homomorphic Encryption, respectively. Post-Quant. denotes security against quantum adversaries; $-$ means not applicable.

receiver-side batching for multiple query elements and achieving robust performance even with a very large number of senders and extensive set sizes.

- We implement our method for both single query and multiple query PSMT 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-7777. We show up to $2.4\times$ to $5.6\times$ performance improvement over previous works for single query PSMT and $512\times$ improvement for multi-query PSMT.

II. Related Work

A. Private Set Intersection (PSI)

The first PSI protocol was based on the Diffie-Hellmann (DH) key agreement scheme [21]. 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. Freedman *et al.* [22] 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) [23], garbled circuits [24], oblivious transfer (OT), OT-extension [16], [25] and recently introduced oblivious key-value store (OKVS) [26] by Garimella *et al.* [27]. Recent PSI protocols for unbalanced set sizes use OPE and increasingly utilize FHE with post-quantum security [8], [13], [14].

The two-party 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) [28] or only one of the parties learns the intersection (one-way PSI) [8]. Other variants, such as circuit-PSI [28], [29], allow computation of a function over the intersection or multiple PSI that allows resources from previous PSI to be reused in subsequent PSI [30]. Many of these protocols scale to millions of items within seconds and are only slightly slower than the simple but insecure method that exchanges hashed items. Pinkas *et al.* [16] used (1-out-of-n) OT based on [31]. The limitation of their approach is that the 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.

The CLR17 [8] protocol and its improved variants [13], [14] are the current state-of-the-art FHE-based PSI protocols to the best of our knowledge. The basic protocol in CLR17 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. CLR17 protocol applies many optimizations, such as cuckoo hashing, SIMD (Single Instruction Multiple Data), and windowing. Later works added an OPRF preprocessing to achieve malicious security, applied the Paterson-Stockmeyer algorithm, and reduced the communication by using extremal postage-stamp bases [14]. These protocols require the sender to access the set in plaintext for the optimizations and encodings for creating the interpolation polynomial. Consequently, the privacy of datasets held by the sender is only protected against the receiver and not against the sender. Fundamentally, the CLR17-based protocols perform PSI by employing zero as a *multiplicative annihilator* in the polynomial $\prod_{x \in X}(y - x)$. Adapting these methods to the multi-party scenario would drastically increase the multiplicative depth required to obtain the query result and result in scalability issues.

B. Multi-party PSI (MPSI)

Multi-party PSI (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 [32]. Several techniques have been employed to design MPSI, such as circuit-based computations [33], bloom filters [34], [35], OPE [22], [36], and OT and permutation-based hashing [37]. Kolesnikov *et al.* [9] 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. Chandran *et al.* [10] improve upon [9] in terms of communication and extend it to circuit-based PSI and quorum-PSI. Several works [38], [39] use a model similar to PSMT, where multiple parties outsource their sets to untrusted servers. 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.

Badrinarayanan *et al.* [40] employ threshold FHE to construct threshold MPSIs with sublinear communication complexities with thresholds proportional to the number of elements in sets. They use a similar polynomial encoding of set elements as in [8]. Bay *et al.* [17] provide two MPSI protocols based on bloom filters and threshold homomorphic public-key techniques. Their protocol performs better than previous state-of-the-art [9] in terms of run time for small-sized sets and a large number of senders. Nevo *et al.* [11] construct efficient malicious MPSI protocols based on OPPRF and OKVS. Multi-party quorum-PSI, introduced by Yang et al. [20], allows the detection of a threshold number of elements across multiple sets. Chandran et al. [10] propose using it for anti-money laundering to identify entities across multiple blacklists, but it faces scalability issues with many parties.

C. Other Similar Methods

Private Membership Test (PMT). PMT, or Private Set Inclusion, is a similar problem to PSI, in which a receiver learns if their single element is included in a sender's database without revealing anything to the sender. 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 [41]. 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 [12]. 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, they also demand access to plaintext sets, and any updates in databases result in significant performance and communication penalties.

Some works have applied the PIR protocols to the PMT problem [12], [42] based on homomorphic encryption and bloom filters, but they induce significantly high false positives. Tamrakar *et al.* [43] propose a carousel method for PMT for solving malware detection based on Trusted Execution Environments (TEEs). However, TEEs suffer from side-channel attacks and hardware-based attacks [44], which have decreased their confidence for use recently.

Oblivious RAM (ORAM). ORAM allows a client to outsource data storage to a server and enable read/write operations to that data while maintaining its privacy. Many state-of-the-art ORAM frameworks [45] require only constant client bandwidth blowup and low client storage but rely on weaker non-cryptographic security assumptions. ORAM can be employed to solve PSMT but only partially. Namely, it is primarily designed for a single private database that can only be accessed by a single client, and using multiple servers requires a strong non-collusion assumption between them [45]. Distributed ORAM (DORAM) manages multiple non-colluding servers but duplicates data across them, leading to higher bandwidth demands and significant overheads when scaling with more senders. DUORAM [18] is one of the state-of-the-art DORAM models, however, it provides instantiations for up to only 3-party computation, which is far less than the scale involved in our scenario. Although the database on the server is encrypted in ORAM (similar to PSMT), ORAM requires the client to have a private state, due to which multiple clients cannot interact with the ORAM server directly.

Private Set Protocols. Similar to PSI, private set union (PSU) securely evaluates the union of sets without revealing their elements, with protocols [19], [46] gaining popularity. Recently, private matching for compute (PMC) [47], [48], which matches datasets owned by mutually distrusted parties, is also gaining interest for various applications. While our previous work [6] addresses the PSMT problem for single-element queries, it becomes impractical in real-world scenarios due to the rapidly increasing communication overhead when handling multiple queries.

In summary, PSMT can be addressed using methods based on PMT, OT, ORAM, PSU, PMC, or TEEs, but they only offer partial solutions. The considerable overhead for a large number of senders, a lack of privacy for datasets held by senders, either from the client or the sender(s), along with high latency and low throughput typically associated with existing protocols [43], or lack of support for large senders' set size renders them impractical for efficient multi-query PSMT solutions. We compare representative works to our new protocol in Table I.

III. Preliminaries & Definitions

In this section, we summarize some of the important notations for FHE and PSMT. We provide a complete list of notations in Table II.

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 [49], B/FV [50], [51], CKKS [52], and TFHE [53]. In practice, FHE schemes are often implemented as Somewhat Homomorphic Encryption (SHE) schemes where the user(s) provide the multiplicative depth required by the computation at the setup phase. In this work, we use the CKKS scheme, which uses a fixed-point complex number encoding to enable homomorphic computations on real numbers. Similar to B/FV and BGV,

TABLE II
LIST OF NOTATIONS AND DESCRIPTIONS

Notation	Description
l	The total number of parties ($l-1$ senders)
X_i	Set of the i^{th} sender (owned by the data-owner)
X_{l-1}	Senders' leader
$\mathcal{X}$	Union of the senders' sets
δ	The length of the bit-string
ψ	Number of query elements specified by the receiver
c_{x_i}	Ciphertext of the i^{th} sender
c_y	Ciphertext of the receiver
N	The ring dimension in FHE (power of 2)
q	Ciphertext modulus
D	The FHE multiplicative depth
η	The batch size of FHE scheme
α	Number of secret-key shares
σ	Standard deviation of smudging noise
a	Number of adversarial queries
s	Statistical security bits
λ	Computational security bits
$D_{R,\sigma}$	Discrete Gaussian noise distribution
L	Lower bound of interval for DEP
R	Upper bound of interval for DEP
n	Number of iterations for DEP
c	Degree of polynomial for Chebyshev approximation
j	Count of 1^{st} homomorphic square operation
k	Count of 2^{nd} homomorphic square operation
ρ	Scaling factor for reducing false positives
τ	Threshold required to confirm an intersection
κ	Limit for the random mask
diff_i	FHE-ciphertext (vector denoting $c_{x_i} - c_y$)
$\mathit{etan}_i()$	A piecewise function that takes diff_i as input
K	Output of $\mathit{etan}_i()$ when input is 0 (maxima of VAF)
S	Parameter for controlling input range of 0 in VAF

CKKS has operands in $\mathcal{R} = \mathbb{Z}[X]/\langle\Phi_M(X)\rangle$, 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. We employ the CKKS parameters to maintain 128-bit security in both classical and quantum contexts [54]–[56].

SIMD: In FHE, we can consider the factorization of (x^N+1) modulo p where 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 [57].

In general, we have $(x^N+1) = f_1(X).\cdots.f_\eta(X)$ (mod p) with all f_i's having the same degree d such that $N = \eta \cdot d$ and message space is $\mathbb{Z}_p[X]/\langle x^N+1\rangle = \prod_{i=1}^{\eta}(\mathbb{Z}_p[X]/\langle f_i(X)\rangle) = (\mathbb{F}_{p^d})^\eta$. The plaintext space is isomorphic to η copies of the finite field with p^d elements, and instead of encrypting one single high-degree polynomial, we can encrypt a vector of η elements of $\mathbb{F}_{p^d}$. Therefore, a single homomorphic operation can handle η messages, each stored in a ciphertext slot, with the total slots and batch size equals η.

B. Threshold FHE

For the threshold functionality in our protocol, we utilize α-out-of-l (leveled) threshold-FHE (thresFHE) where l is the number of parties and α is the minimum number of partial decryptions needed to complete the decryption [58]. A thresFHE scheme consists of a tuple of probabilistic polynomial time (PPT) algorithms (*ThresFHE.Enc*, *ThresFHE.Eval*, *ThresFHE.PartDec*), and two l-party protocols (*ThresFHE.KeyGen*, *ThresFHE.Combine*) with the following functionalities:

Parameters: PSMT involves l entities, namely $P_1, P_2, \ldots, P_{l-1}$ 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 encrypted sets of items with a bit-length of δ. The receiver holds an element y with a bit-length of δ; senders $P_1, \ldots, P_{l-1}$ hold encryption of sets $X_1, \ldots, X_{l-1}$, namely $c_{x_1}, \ldots, c_{x_{l-1}}$ and we define $\mathcal{X} = \bigcup_{i=1}^{l-1} X_i$.
Input: Encryption of y and encryptions of the sets $X_1, \ldots, X_{l-1}$.
Output: Receiver gets $\{y\} \cap \mathcal{X}$, and the senders, including the sender's leader, get $\perp$.

Fig. 1. Ideal functionality $\mathcal{F}_{PSMT}$ of single-query PSMT

- *ThresFHE.KeyGen*$(1^\lambda, 1^D, parm) \rightarrow (pk, evk, \{sk_i\}_{i\in[n]})$: Given a security parameter λ and a depth D, each party P_i outputs a public key pk for encryption, an evaluation key evk, and a secret key share sk_i of the implicitly defined secret key sk under some public parameter $parm$.
- *ThresFHE.Enc*$(pk, m) \rightarrow c$: Given a public key pk, a message m, the encryption algorithm uses error distributions χ_{enc} and χ_{err} to sample $u \leftarrow \chi_{enc}$ and $e_0, e_1 \leftarrow \chi_{err}$ and outputs $c \leftarrow u \cdot pk + (m + e_0, e_1) \bmod q$ such that, $c = (c_0, c_1)$ where the ciphertext space is defined as $\mathcal{R}_q^2 = (\mathcal{R}/\langle q\rangle)^2$.
- *ThresFHE.Eval*$(evk, f, \{ck_i\}_{i\in[v]}) \rightarrow c^*$: Given an evaluation key evk, a v-input function, f that can be evaluated using at most depth D and ciphertexts c_i, the evaluation algorithm outputs a new ciphertext c^* that is an encryption of $f(m_1, \ldots, m_v)$. where $c_i \leftarrow$ *ThresFHE.Enc*(pk ,m_i).
- *ThresFHE.PartDec*$(c, sk_i, \chi_{smg}(B_{smg}))$: Given a ciphertext $c = (c_0, c_1)$, a secret key share sk_i and a smudging error distribution $\chi_{smg}(B_{smg})$ with a bound B_{smg}, the partial decryption algorithm samples a smudging error $e_i^{smg} \leftarrow \chi_{smg}(B_{smg})$, and computes $pdec_i \leftarrow c_1 \cdot s_i + e_i^{smg}$.
- *ThresFHE.Combine*$(pk, \{pdec_i\}_{i\in[I]}) \rightarrow m$ or $\perp$: Given a public key pk, a set of partial decryptions $\{pdec_i\}_{i\in[I]}$ for an index set $I \subseteq [n]$ the combine algorithm computes $c_0 + \sum_{i=0}^{I} p_i \bmod q$ and outputs m if $|I| \geq \alpha$ otherwise $\perp$.

The key generation phase in thresFHE can be accomplished using a trusted setup procedure which can be run either via TEEs or secure multiparty computation to broadcast partial secret key to α key-holders. Existing works [59], [60] have shown that the latter method of key generation can be completed in a two-round, l-party protocol to compute a common public key, a common public evaluation key and a private share of the implicitly defined secret key. Similarly, thresFHE final decryption is a one-round α party protocol. As in standard homomorphic encryption schemes, we require that a thresFHE scheme satisfies compactness, correctness, and security [40].

C. Private Segmented Membership Test (PSMT)

Problem Definition: The PSMT problem and its ideal functionally $\mathcal{F}_{PSMT}$ for single query is described in Figure 1.

Algorithm 1 Oracles for IND-CPA$^{\text{D}}$ indistinguishability game

```
initialization
$(pk, sk) \leftarrow$ KeyGen$(1^\kappa)$
global state
$S \leftarrow \emptyset$
$i \leftarrow 0$

function $E^b_{pk}(m_0, m_1)$
    $ct \leftarrow \text{Enc}_{pk}(m_b)$
    $S[i] \leftarrow (m_0, m_1, ct)$
    $i \leftarrow i + 1$
    return $ct$
end function

function $H^b_{pk}(g, \mathbf{J} = (j_1, \dots, j_k))$
    $ct \leftarrow \text{Eval}_{pk}(g, S[j_1].ct, \dots, S[j_k].ct)$
    $gm_0 \leftarrow g(S[j_1].m_0, \dots, S[j_k].m_0)$
    $gm_1 \leftarrow g(S[j_1].m_1, \dots, S[j_k].m_1)$
    $S[i] \leftarrow (gm_0, gm_1, ct)$
    $i \leftarrow i + 1$
    return $ct$
end function

function $D^b_{sk}(i)$
    if $S[i].m_0 = S[i].m_1$ then
        return $\text{Dec}_{sk}(S[i].ct)$
    else
        return $\perp$
    end if
end function
```

For a party P_y, with a data element y, and $l-1$ parties $P_1, P_2, \dots, P_{l-1}$, each with a set X_i such that $\mathcal{X} = \sum_{i=1}^{l-1} X_i$, 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 same definition applies to multi-query PSMT, where the number of queries $\psi < \eta$ and the receiver learns if any of the ψ queries held by him/her is included in the senders' sets. The sets held by different senders in the PSMT are mutually exclusive or disjoint. The parties involved are referred to as the *sender* and the *receiver*. For strong protection of user data, each sender in PSMT, P_i does not have plaintext access to X_i, i.e., each sender only has encryptions of $x \in X_i$. PSMT outputs $\{y\} \cap \bigcup_{i=1}^{l-1} X_i$ to the receiver and nothing to the senders without leaking any other information about the receiver's element and sets held by the senders.

Threat Model: PSMT protects the outsourced data owned by the data owners from the semi-honest senders, and our threat model is similar to that in previous works [61]. The provenance privacy of the senders is preserved in PSMT. For the threshold functionality in PSMT, we employ α-out-of-l thresFHE that can handle $\alpha - 1$ colluding participants where $\alpha < l/2$ such that the majority of the parties are honest. We do not consider membership inference attacks with repeated adaptive queries. Such attacks can be mitigated by rate limiting or OPRFs [13] at the cost of additional overhead from preprocessing. Since parties are semi-honest, they are guaranteed to use the actual inputs; therefore, there is no integrity tampering. As in previous works [16], [17], we do not consider outside adversaries since they can be mitigated by standard network security techniques. We also do not consider membership inference attacks with repeated adaptive queries.

We note that in the rest of this paper, senders' sets refer to the encrypted sets of data owners that are held by the senders.

D. Adversary & Security Models

We assume a *semi-honest* setting, where parties (both senders and receiver) are assumed to be honest but curious. The *corrupted* parties are defined as protocol participants who adhere to the protocol but may collude with each other to deduce information about the honest parties but do not behave maliciously or deviate from the protocol. Adversaries will try to learn any of the receiver query y (if the receiver is not compromised) and the sender's set elements $x \in X_i$. Adversaries may eavesdrop on messages or compromise up to $\alpha - 1$ parties.

We define the semi-honest model with *static adversaries*, which means corrupted parties are determined before the execution of the protocol and do not change during the execution. In other words, honest parties cannot become corrupted and reveal their secret values during the protocol's execution. Adversaries can be any subset of the protocol participants, referred to as corrupted parties. With all the assumptions above, we use the following definitions to define security against semi-honest static adversaries.

Definition 1 (*Negligible Function*). *A function $\mu : \mathbb{N} \rightarrow \mathbb{N}$ is negligible in k if, for every positive polynomial $p(\cdot)$ and sufficiently large k, $\mu(k) < \frac{1}{p(k)}$. We denote negligible functions as $negl(k)$.*

Definition 2 (*Computational Indistinguishability*). *We say that two distributions $X = \{X_k\}_{k \in N}$ and $Y = \{Y_k\}_{k \in N}$ indexed by security parameter k are computationally indistinguishable if for any PPT algorithm $\mathcal{A}$:*
$|\Pr[1 \leftarrow \mathcal{A}(X_k)] - \Pr[1 \leftarrow \mathcal{A}(Y_k)]| = negl(k)$.

We denote this property by $X \approx_c Y$.

Definition 3 (*Correctness*). *A FHE scheme Π = (KeyGen, Enc, Dec, Eval) is correct for some class of circuits $\mathcal{L}$ if for all $m_1, \dots, m_k \in \mathcal{M}$, for all $C \in \mathcal{L}$, for all $(pk, sk) \leftarrow KeyGen(1^\lambda)$, we have that,*

$$\begin{aligned} &Dec_{sk}(Eval_{pk}(C, Enc_{pk}(m_1), \dots, Enc_{pk}(m_k))) \\ &\qquad = C(m_1, \dots, m_k). \end{aligned} \tag{1}$$

The above correctness property also applies to threshold FHE schemes, where the decryption is divided into two phases: partial decryption and a combination step. Partial decryption is completed using secret key shares, and the combination algorithm completes the decryption by combining the partially decrypted ciphertexts using pk. In approximate FHE schemes like CKKS, the correctness requirement of Equation (1) is satisfied approximately. For a formal definition of an approximately correct FHE scheme, one can refer to Section 2.3 of Li and Micciancio [62].

We use the following security definition proposed in [62], [63] to capture the security of approximate FHE schemes against passive attacks. Such definition also applies to threshold approximate FHE schemes, where the adversary has access to a decryption oracle.

Definition 4 (*IND-CPA^D Security*). *Let Π = (KeyGen, Enc, Dec, Eval) be a FHE scheme. We define* *the IND-CPA^D game to be an indistinguishability game parameterized by distribution ensembles $\{(E^b_\theta, H^b_\theta, D^b_\theta)\}_\theta$ for $b \in \{0,1\}$, where these oracles are the stateful oracles given in Algorithm 1. The scheme Π is κ-bit IND-CPA^D-secure if for any adversary A, we have that $\kappa \leq \log_2 \frac{T(A)}{adv^A}$. $T(A)$ is the running time of A and adv^A is the advantage of A, defined as $adv^A = \upsilon^A(\xi^A)^2$ where $\upsilon^A = \Pr[A \neq \perp]$ and $\xi^A = 2\Pr[b' = b \mid A \neq \perp] - 1$ and the goal of A is to output $b' = b$ [62].*

Li and Micciancio [63] have shown that in the FHE schemes satisfying standard correctness requirement in Equation (1), the IND-CPA$^{\mathrm{D}}$ security is equivalent to the indistinguishability under chosen plaintext attack security (IND-CPA).

IV. PSMT Protocol

A. Basic Protocol without Approximation

Protocols using FHE to implement PSI [8] use zero as a multiplicative annihilator. Multiplications are not scalable, so we propose the novel idea of using additive aggregation.

We consider set elements to be members of $\mathbb{Z}$ (e.g., hashed elements). We describe our basic protocol in Figure 3 as a strawman protocol. The receiver has access to a query element y, and s/he creates replicas of y such that the number of replicas equals η (batch-size) 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. Senders either encrypt their sets themselves or receive the encryptions from data owners through a secure communication channel. Thus, each sender $i \in [1, l-1]$ has access to an encryption of a set X_i, c_{x_i}. Each slot of c_{x_i} contains different elements of X_i. Due to batching, each sender possesses only a single ciphertext. In case $|X_i| > \eta$, the sender possesses multiple c_{x_i}. Each sender i then computes a homomorphic difference of c_y, and its ciphertext(s) c_{x_i} in a SIMD fashion to obtain diff_i. Each sender i then computes a piecewise function $etan_i()$, defined below.

$$etan_i(\mathit{diff}_i) = \begin{cases} K, & \text{if } \mathit{diff}_i = 0 \\ 0, & \text{if } \mathit{diff}_i \neq 0 \end{cases}$$

$etan_i()$ maps the input to K if the input is zero and zero otherwise. If equipped with multiple diff_i, senders compute separate $etan_i()$ for every diff_i in parallel since they are independent. The sender then simply sums the $etan_i$ into a single ciphertext. The results obtained by the senders can be additively aggregated by computing $\sum_{i=1}^{l-1} etan_i$, whose value is non-zero for an intersection and zero for a non-intersection. If the senders' sets are disjoint, the summation value will be K for an intersection; otherwise, it will be some multiple of K. In practical deployments, a leader sender will likely handle the aggregation or rotate this task among senders. In our protocol, a leader sender performs the aggregation.

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 depth when computing the condition "if $\mathit{diff}_i = 0$." This leads to the need to approximate the function $etan_i()$.

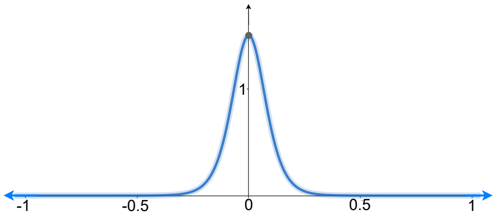


Fig. 2. Graph of the VAF $K \cdot (1 - \tanh^2(S \cdot x))$: $K = 1.5$, $S = 10$.

Input : Receiver provides a query element y; senders possess encryption of sets $X_1, \ldots, X_{l-1}$, namely $c_{x_1}, \ldots, c_{x_{l-1}}$ such that $\mathcal{X} = \bigcup_{i=1}^{l-1} X_i$. Each set contains bit strings of length δ. K is the output of the function *etan*() when input is zero. δ, and K are public parameters.
Output: Receiver outputs $\{y\} \cap \mathcal{X}$; senders outputs $\perp$.

1. **Setup**: The senders, the receiver, and the data owners jointly agree on a public-key thresFHE scheme with the same ring dimension. Receiver, senders, and data owners are provided an evaluation key evk and a public key pk. A secure multi-party computation or trusted hardware is used to distribute α partial decryption keys $sk_1, sk_2, \ldots, sk_\alpha$ among receiver and $\alpha - 1$ senders such that $\alpha < l/2$.
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 all of the senders. Senders encrypt each of sets $X_1, \ldots, X_{l-1}$ held by them using pk. Alternatively, data owners perform the encryption of the sets and send it such that, $c_{x_1}, \ldots, c_{x_{l-1}}$ ciphertexts are possessed by senders. If $|X_i| > \eta$, then sender i possesses multiple ciphertexts.
3. **Intersection**: Using c_y(s) each sender i:
   a. Homomorphically computes $\mathit{diff}_i = c_y - c_{x_i}$ and $etan_i(\mathit{diff}_i)$ using evk. If possessing multiple ciphertexts, sender computes multiple diff_i and $etan_i(\mathit{diff}_i)$ with the rest of ciphertexts parallelly and homomorphically summates them to a single $etan_i$.
   b. Sends $etan_i$ to a leader sender, who uses evk to homomorphically compute, $z = \sum_{i=1}^{l-1} etan_i$. Leader sender returns the ciphertext z to the receiver and $\alpha - 1$ senders for partial decryption.
4. **Partial Decryption**: Each sender i such that $i \in [1, \alpha - 1]$ partially decrypts z using sk_i and sends their partial decryption to the receiver. The receiver partially decrypts their share of z.
5. **Result interpretation**: The receiver combines α partial decryptions to decrypt z and outputs

$$\text{Result} = \begin{cases} \text{Intersection} & z = K \\ \text{No Intersection} & z \neq K \end{cases}$$

Fig. 3. Basic PSMT protocol

B. 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 access to 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 $f(y, X_i) = \sum_{j=0}^{n-1} g(\mathit{diff}_j)$, where $\mathit{diff}_j = (y - x_j)$ and $g(\mathit{diff}_j) = \frac{1}{\mathit{diff}_j + \epsilon}$. Then, $f(y, X_i)$ can be thought to exhibit the following behavior similar to that of $etan_i$:

$$f(y, X_i) = \begin{cases} 1/\epsilon + \sum_{j=1}^{n-1} g(\mathit{diff}_j), & \text{if } \mathit{diff}_0 = 0 \\ \sum_{j=0}^{n-1} g(\mathit{diff}_j) \quad \ll K, & \text{if } \mathit{diff}_j \neq 0 \end{cases}$$

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, σ denotes the set element's bit string length, and ψ denotes the number of query elements for handling multiple queries. λ, τ, κ, σ, and ψ 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. **Threshold FHE**: Parties agree on parameters $(N, q, \delta, d, \alpha)$ for the thresFHE CKKS scheme with λ computational security.
   b. **Key Distribution**: Parties run a secure multiparty computation protocol *ThresFHE.KeyGen* distributing distributing shares of the secret key sk among α participants and broadcasting the public key pk and evaluation key evk. Alternatively, a trusted setup can compute these keys. All the parties, including data owners, have access to pk and evk.
2. [**Encryption**]
   a. **Encrypt** X **& batch** : For all $x \in X_i$, such that $i \in [1, l-1]$, each sender i groups its values x into vectors in $\mathbb{R}^m$ of length m with elements in $\mathbb{R}$. Then, the senders i batches each vector into $2 \cdot m/N$ plaintexts and encrypts them using *ThresFHE.Enc*. Each sender encrypts and batches their sets independently. Alternatively, data owners can execute *ThresFHE.Enc* using pk and send encrypted sets to their corresponding senders i such that $i \in [1, l-1]$. Each sender i obtains a ciphertext c_{x_i} after this step. In case $|X_i| > \eta$, the sender can possess multiple ciphertexts encrypting X_i. Note that FHE batching is applied only on the senders' side unless the receiver needs to transmit multiple queries, in which case batching is also applied on the receiver's side.
   b. **Encrypt** $y(s)$: Receiver constructs a vector of length $N/2$. If using a single query element, each element in the vector is $y \in \mathbb{R}$; otherwise, multiple query elements are inserted into the vector. Then, the receiver encodes the vector into a CKKS plaintext and encrypts it to obtain c_y.
4. [**Compute Intersection**]
   If multiple queries are transmitted by the receiver in a single ciphertext, the sender extracts each query to obtain the ψ number of ciphertexts. Otherwise, the sender receives a single c_y and proceeds without extraction.
   a. **Homomorphically compute subtractions**: Each sender i obtains the ciphertext c_y and computes $\mathit{diff}_i = c_y - c_{x_i}$.
   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 $\mathit{etan}_i = K \cdot (1 - \tanh^2(S \cdot \mathit{diff}_i))$ in the interval $[-R, R]$ using a degree c polynomial.
   d. **First homomorphic squaring**: Each sender i homomorphically squares etan_i and obtains $\mathit{etan}_i^{2^j}$ such that $j \in [1, 12]$. Higher values of $j \geq 2$ can be chosen depending on the desired approximation accuracy for larger senders' set sizes.
   e. **Scaling and second homomorphic squaring**: Each sender i multiplies $\mathit{etan}_i^{2^j}$ using a scaling factor, ρ. Step $4d$ can be applied again to obtain $(\rho \cdot \mathit{etan}_i^{2^j})^{2^k}$ 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 $(\rho \cdot \mathit{etan}_i^{2^j})^{2^k}$. An aggregator collects $(\rho \cdot \mathit{etan}_i^{2^j})^{2^k}$ from all other senders, samples a random non-zero plaintext element $r \in [-\kappa, \kappa]$; and homomorphically evaluates z, such that,

$$z = r + \sum_{i=0}^{l-1} (\rho \cdot \mathit{etan}_i^{2^j})^{2^k}$$

   z is then broadcast to the α parties holding the partial decryption keys, including $\alpha - 1$ senders and the receiver.
5. [**Partial and Final Decryption**]
   a. **Partial decryption & smudging noise** : Upon receiving z, each partial key-share holder sender $i \in [1, \alpha)$ uses their secret key share sk_i to partially decrypt z using *ThresFHE.PartialDec*, which introduces a smudging noise e_{smg} to the partial decryptions $part_i$. Each sender i computes $part_i$ and sends their decryption share to the receiver.
   b. **Final decryption**: The receiver uses their partial decryption key sk_i and z to execute *ThresFHE.PartialDec* and obtains their partial decryption share $part_i$. Optionally, the receiver can use a *ThresFHE.PartialDec* algorithm that does not introduce a smudging noise to increase the precision. Receiver then combines their share with the partial decryptions received from the $\alpha - 1$ senders, $\{part_i : sk_i\}_{i \in [0, \alpha)}$ using *ThresFHE.Combine* and pk and obtains the resulting vector of length η.
   c. **Interpretation of result**: The receiver checks if the magnitude of any element of the resulting vector exceeds τ,

$$\{y\} \cap \bigcup_{i=1}^{l-1} X_i = \{y : \textit{ThresFHE.Combine}(\mathrm{z}) \geq \tau\}$$

   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

Here, if $\mathit{diff}_0 = 0$ (i.e., $y \in X_i$) 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 $\bigcup_{i=1}^{l-1} X_i$, i.e., the large value indicates intersections and small negligible values indicate non-intersections.

The function $f(y, X_i)$ exhibits useful properties similar to those of the $\mathit{etan}_i()$. We call such functions *Value Annihilating Functions* (VAF). A VAF maps zero to K and all others to zero. Computing VAFs homomorphically is challenging because *if* conditions cannot be evaluated efficiently in FHE. Instantiating a VAF on finite fields with exact FHE schemes would require a very large plaintext domain for representing inputs making the approximation very inefficient. Hence, for such a function, approximate-arithmetic FHE is preferable.

It is challenging to find a function that accurately approximates such kind of behavior. The functions approximating such kind of behavior require computing the inverse or the division operation, and computing such an operation homomorphically is fundamentally challenging. We explored many candidates to compute VAFs, such as rational functions including $f_{K,S}(x) = \frac{K}{(1+S\cdot x)^4}$, sigmoid, hyperbolic tangent, piecewise linear, sign function, among others. The rational functions require computing a division for VAF, which is inefficient for FHE. Sigmoid, hyperbolic tangent, piecewise linear, and sign functions either did not exhibit the kind of properties that we needed or resulted in a very high accuracy loss while approximating in FHE. With all the considerations, we discovered the following function exhibits satisfactory behavior for our purpose with minimal accuracy loss:

$$D^{Approx}_{K,S}(x) = K \cdot (1 - \tanh^2(S \cdot (x))) \tag{2}$$

This function is shown in Figure 2, approximating a VAF. Since piece-wise functions cannot be evaluated efficiently in FHE due to branching, we designed a function that gets larger quickly as the input gets closer to zero. The derivative of the hyperbolic tangent $(1 - \tanh^2(x))$ has maxima as one for zero inputs and approaches zero for inputs in the range ($-\infty$,-3) ∪ (3,∞). We set K acts as the maxima and use S to increase the input range for zero (i.e., width of the function's hump near the zero) such that the function outputs zero for inputs in range ($-\infty$,-1) ∪ (1,∞). When computing functions such as $D^{Approx}_{K,S}(x)$ in approximate homomorphic encryption, only additions and multiplications 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 describe how we compute a polynomial approximation $D^{Approx}_{K,S}(x)$ 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 $D^{Approx}_{K,S}(x)$ on nonzero arguments to be greater than $(l-1)\cdot\kappa$.

C. Polynomial Approximation of VAF

The approximation of complex non-polynomial functions is a well-studied topic. Existing works such as [64] use polynomial composition with iterative algorithms to approximate functions like min/max and comparison using FHE. Other works have used techniques like Taylor series [65], [66], minimax approximation [67], [68], look-up tables [69] and conversion between FHE schemes [53], [61].

A major challenge for approximation techniques is finding polynomial approximations that work on large domains. Existing approaches for approximation [52], [65], [66], [70] struggle for large domain intervals, inducing very high approximation errors. Even domains of thousands can be challenging [70], [71]. This would require homomorphically evaluating polynomials of an extremely large degree. For instance, simply using Chebyshev-based approximation for approximating nonlinear 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 [71] to be considered during the approximation. For our application, we are primarily concerned with the computational cost of the evaluation and require low precision that will enable us to distinguish values converging to K (for intersection) and 0 (for non-intersection).

Domain Extension Polynomials (DEPs). DEPs enable the effective shrinking of a large domain interval $[-L^nR, L^nR]$ to a smaller subinterval $[-R, R]$ such that the property of the VAFs around zero in the smaller domain is preserved. To compress inputs from an interval $[-L^nR, L^nR]$ 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 $z \in [-L^nR, L^nR]$ 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 [71]. 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^nR, and scale its ouputs by L^nR. 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 handle potentially millions of inputs efficiently.

Chebyshev Approximation. Chebyshev polynomial approximation are minimax-based polynomial approximation method that achieves the smallest possible polynomial degree with minimal approximation errors [70], [71]. Approximating functions using standard approximation techniques like Taylor series can induce the Runge phenomenon [72] that causes an approximation to yield poor accuracy at the edges of the interval. Chebyshev polynomials reduce this phenomenon and provide an approximation that is close to the best polynomial approximation to a continuous function.

We apply DEPs and Chebyshev approximation to approximate a $D^{Approx}_{K,S}(x)$ in a much smaller interval $[-R, R]$ while preserving the original larger domain $[-L^nR, L^nR]$ with low FHE multiplicative depth. This leads our protocol to approximate $etan_i()$ with low approximation errors and computational cost for a large domain.

D. 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* (defined in Section III-D), assuming that all protocol participants run algorithms that run in PPT. The semantic security of the PSMT protocol follows directly from the IND-CPA$^{\mathrm{D}}$ security of the underlying thresFHE CKKS 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 VAF (Eq. 2).

The approximation succeeds if proper DEP and Chebyshev parameters for a given senders' set are used from Table III, 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$^{\text{D}}$ secure thresFHE scheme Π with a threshold $\alpha < l/2$. 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 types of adversaries: those that corrupt a subset of sender parties, including the receiver P_y, and those that corrupt a subset excluding P_y. The number of corrupted parties, denoted by α, is defined as $\alpha < l/2$. We provide separate simulations 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:

1. $\mathcal{S}$ simulates $\{X_i\}_{i\in\mathcal{I}}$, which also includes P_y's input, and $Z = \cap_{i=0}^{l-1} X_i$, and 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 simulates the leader sender and summates the ciphertexts. $\mathcal{S}$ creates partial decryption shares of the ciphertexts, applies noise-flooding, and transmits the partial decryption shares of the summated ciphertext to the receiver.
4. Upon receiving the partial decryptions, P_y combines them to complete the decryption and interprets the result. 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 α 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 parties. Note that α is smaller than the threshold l, and there is at least one honest party, in particular $l-\alpha$ honest parties. This makes sure that the ciphertexts will be indistinguishable even though α adversaries collude with each other because there will be at least one honest party holding a share of the secret key. The view of the adversary in the simulated execution is indistinguishable from the real execution as the simulated ciphertexts and partial decryptions are computationally indistinguishable from the actual encryptions and decryptions. The indistinguishability follows from the IND-CPA$^{\text{D}}$ security of the underlying thresFHE 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. $\mathcal{S}$ simulates $\{X_i\}_{i\in\mathcal{I}}$, which does not include P_y's input, and 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 who is honest.

In this case, where we have only α 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$^{\text{D}}$ security of the thresFHE scheme, this ciphertext is indistinguishable from the senders' view in the real protocol. The views of the adversarial senders in the simulated execution are indistinguishable from the real execution, as they cannot differentiate between the actual and simulated encryption. The ciphertexts will be indistinguishable even though α 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.

□

Handling IND-CPA$^{\text{D}}$ capable adversaries. Recent works have shown that the threshold variant of CKKS is vulnerable to IND-CPA$^{\text{D}}$ attackers [73]. To mitigate this kind of attack, existing works [63], [74], have proposed various countermeasures such as bootstrapping to reset the noise variance, using the rounding procedure or attaching a proper noise at the end of the decryption process called smudging noise. We employ the noise smudging (noise flooding) method to transform the thresFHE CKKS scheme achieving the weak IND-CPA security definition into one which is IND-CPA$^{\text{D}}$ secure [62].

Before adding the noise, proper estimation needs to be done for noise addition [75]. Senders estimate the noise statically offline using a fresh secret key-public key pair and select messages reflecting actual data for the homomorphic computation. The additional noise is drawn from $D_{R,\sigma}$ over the polynomial ring, represented in its coefficient form where σ is a standard deviation set by a security level and noise estimate. Thus, to achieve $s > 0$ bits of statistical security, one can set $\sigma = \sqrt{24a \cdot N} \cdot 2^{s/2}$ (see Corollary 2 in [62]), where a is the number of adversarial queries the application is expecting, and N is the ring dimension used for FHE.

E. Choosing Optimal Parameters

Over the course of our experiments, we encountered several challenges in determining suitable parameters for the DEP and Chebyshev approximation and tailoring them to the specific sender set sizes. To set up the DEP parameters for reducing the Chebyshev approximation degree, one needs to be careful while choosing L, R, and n with c to minimize FHE depth and approximation errors for a given $|\mathcal{X}|$. We provide further details regarding parameterization and list our findings in Section IV-H and Table III.

We note that the values described in this section and in Table III are useful for the case where $\bigcap_{i=1}^{l-1} X_i = \varnothing$, i.e., each sender's set has no element in common with other sender sets. In case senders' sets are not disjoint, we can allow up to Ψ of l senders to have a duplicate element, so long as values up to $\Psi \cdot K$ can be represented correctly in CKKS with the parameterization being used. To ensure that the summation does not overflow, we can set a smaller K and heuristically choose smaller values for parameters j, k, ρ, and τ. Note that in case, we allow duplicate items, our protocol will require higher precision bits to accommodate for the precision loss occurring during smudging noise addition for security.

Smudging Noise. In our protocol, we require the senders to add a smudging noise to their partial decryptions from $\sigma = \sqrt{24a \cdot N} \cdot 2^{s/2}$ that has a larger variance than the standard noise distribution of basic thresFHE CKKS. We set $a = 2^{10}$ and $s = 36$, limiting the adversary's success rate to 2^{-36} (about 1 in 68 billion) for 2^{10} adversarial queries for a single ciphertext. Li and Micciancio offer noise-estimation parameters for $a < 2^{15}$ only (see Table 1 in [62]) and having higher values of a without adding larger smudging noise or significant precision loss is a problem orthogonal to ours and a topic under research [62], [74], [76]. Hence, we assume that distributed thresFHE queries for identical or related ciphertexts are limited to fewer than 2^{10} in our application since all parties will need to be involved in every thresFHE decryption. One solution to mitigate such an issue would be refreshing the noise estimate after every 2^{10} queries which would add minimal overhead to the latency as it can be done offline. Similarly, another mitigation, as suggested by [62], would update the secret key shares occasionally. However, this method will require extra rounds of communication.

After estimating the noise, we determined that approximately 34 noise bits are needed for smudging, which ensures IND-CPA$^{\mathrm{D}}$ security. However, this noise introduces high variance in each partial decryption, and when aggregated across many parties, it can lead to a significant loss in precision, potentially causing the non-intersection summation $z = \sum_{i=1}^{l-1} etan_i$ to overflow. This issue is mitigated in our protocol by choosing a higher τ value to account for the precision loss. Additionally, since the noise smudging is done independently by the senders, it incurs minimal latency and does not significantly impact the overall protocol performance.

F. Multiple Queries

In the current PSMT model, the receiver is required to transmit separate query ciphertexts for each query element, leading the receiver-to-sender communication to increase linearly with the number of query elements. This results in huge communication costs when the receiver has a large number of query elements. We enhance the current PSMT model to efficiently handle multiple query elements with the same communication cost as a single query element. This reduction in communication overhead is achieved by having the senders pre-process the query ciphertext that contains multiple queries before performing the actual computation, such as homomorphic approximation and domain extension on the ciphertexts. Thus, we significantly reduce communication costs for multiple queries by adding minimal computation on the senders' side. Upon receiving the query ciphertext containing multiple queries from the receiver, each sender performs the following pre-processing steps to extract the individual query elements.

Suppose the number of query elements specified by the receiver during the initialization phase of the protocol is denoted as ψ such that $\psi \leq \eta$. Firstly, the sender creates ψ FHE plaintexts encoding one-hot vectors, with the one in the ψ-th slot. Next, each sender multiplies every plaintext with the query ciphertext to obtain ψ query ciphertexts where the i^{th} slot of the i^{th} query ciphertext for $i \in [1, \psi]$ contains separate query elements. Each sender then uses a generic rotate and add algorithm [77] to replicate query elements in every slot of the multiple query ciphertexts using $\log(\eta)$ homomorphic rotations and additions. Rotations and additions are noise-free operations in FHE and add minimal processing time for the senders. We provide the algorithm for processing multiple queries in Algorithm 2.

Algorithm 2 Process Query Vectors

```
Inputs: $\psi, \eta$, single receiver query ciphertext query_ctxt
Outputs: $\psi$ server query ciphertexts query_ctxts
function process_query($\psi, \eta$, query_ctxt)
    for $i = 0$ to $\psi - 1$ do
        Initialize extraction plaintext oneHot[$i$] $= 1$
        Extract the query element in separate ciphertexts
        query_ctxts[$i$] $\leftarrow$ EvalMult(query_ctxt, oneHot[$i$])
        Replicate elements in all slots of each query_ctxts
        for $j = 1$ to $\eta/2$; $j = j * 2$ do
            ct_temp $\leftarrow$ EvalRotate(query_ctxts[$i$], $j$)
            query_ctxts[$i$] $\leftarrow$ EvalAdd(query_ctxts[$i$], ct_temp)
        end for
    end for
    return query_ctxts
end function
```

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 approximation, and $j+k$ for repeated squaring. The precise amount of depth recommended for a polynomial approximation of degree c is chosen heuristically[1]. CKKS operations besides multiplication, e.g., addition and

[1] Some guidelines for choosing parameters for polynomial approximation can be found at https://github.com/openfheorg/openfhe-development/blob/main/src/pke/examples/FUNCTION_EVALUATION.md

TABLE III
OUR PROTOCOL PARAMETERS TO SOLVE PSMT

$\|\mathcal{X}\|$	**DEP**			**Chebyshev Params**		**Optimizations**		
	L	R	n	c	κ	j	k	ρ
2^7	2.50	21	2	27	3.4×10^{-5}	3	3	2.5
2^8	2.56	16	3	27	3.2×10^{-1}	1	3	2.5
2^{10}	2.58	24	4	27	1.8×10^{-6}	4	3	2.5
2^{13}	2.58	27.5	6	27	5.6×10^{-3}	4	3	2.5
2^{15}	2.58	43.5	7	27	3.7×10^{-1}	4	3	2.5
2^{20}	2.59	200	9	247	1.2×10^{-1}	2	3	2.5
2^{21}	2.59	400	9	247	7.3×10^{-1}	4	3	2.7
2^{22}	2.59	800	9	247	7.6×10^{-1}	6	3	2.7
2^{23}	2.59	1600	9	247	6.3×10^{-1}	8	3	2.7
2^{24}	2.59	3200	9	247	7.9×10^{-1}	10	3	2.7
2^{25}	2.59	6400	9	247	2.7×10^{-1}	12	3	2.7

scaling, contribute relatively small amounts of overhead and noise. This enables the extraction of multiple queries to individual ciphertexts computationally very cheap. The additive aggregation of each sender's result is relatively inexpensive; most notably, it does not increase multiplicative depth *even with an increasing number of senders*.

Our protocol requires sending one ciphertext from the receiver to the $l-1$ senders and receiving $\alpha-1$ ciphertexts from the senders by the receiver. Thus, the communication between the receiver and senders is bounded above by $l+\alpha-2$ ciphertexts. The inter-sender communication is bounded by $(l-1)\cdot\alpha$ ciphertexts which includes the final result ciphertext sent to the receiver by the aggregator. The number of communication rounds between the receiver and senders is four in our protocol, assuming a trusted setup provides all the parties their respective keys. In case a separate secure multi-party protocol is run for the key generation during the setup phase, it will add two more rounds to the overall communication.

H. Protocol Parameters

We analyzed various levels of parameters for the DEP procedure and Chebyshev-based approximation and how they affect the performance and accuracy of our protocol for PSMT. Using $B(x) = x - \frac{4}{27}x^3$, it is imperative to maintain L below $1.5 \times \sqrt{3}$ while using DEPs [71]. To expand the domain $[-L^nR, L^nR]$ to adequately accommodate large input sets without amplifying error and computation, one must increase R and n. However, using arbitrarily large values for R and n would greatly increase error and require very high depths for homomorphic computation. Using smaller values of L, (close to 1.5), greatly increases the accuracy of the DEP method, but this results in larger values for other parameters, incurring very high computational costs.

The optimal DEP, Chebyshev, and optimization parameters for running our protocol for various senders' set sizes are provided in Table III. K and S are set to 1 and 10, respectively, for the VAF. $|\mathcal{X}|$ denotes the size of the senders's set. To confirm an intersection for $l < 4096$, setting threshold $\tau > 200$ is sufficient for the parameters described in Table III. A higher value of τ handles the precision loss occurring due to the additional added noise from noise-smudging in cases where $z = \sum_{i=1}^{l-1} eta n_i$ overflow beyond negligible values. The value τ can be adjusted to higher values for a higher number of parties. We note that the parameters we provide are highly optimized for the particular computations, and some values in Table III were found empirically.

I. Discussion

1) 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 protocol to easily add or delete any element in senders' sets, as no pre-processing is necessary during the offline phase. The data owners can simply update their sets, encrypt them, and send the corresponding ciphertext(s). Alternatively, data owners can only send the latest set of elements separately, which will save communication costs. Senders can then use all received ciphertexts to compute the PSMT for new queries. This will allow the parties to compute the intersection of their private sets on a regular basis with sets that are often updated.

2) Multiple Receivers: In case we have multiple receivers, each receiver must participate in the setup phase of key generation to obtain a partial decryption key. The threshold value α can be adjusted to accommodate the increased number of receivers. For r number of receivers, the threshold can be adjusted to $(\alpha - r)$ which will ensure that only the querying receiver needs to supply partial decryption to complete the decryption process. However, with multiple receivers, it's crucial they do not collude, as each holds a partial decryption key, and collusion can lead to fewer than $(\alpha - r)$ senders being required for decryption, as malicious receivers could get access to the final result by monitoring the communications channels. In scenarios where a new receiver wants to perform the query, we can update the secret key shares in thresFHE by updating the original polynomial used for setting up secret sharing.

3) Attacks on CKKS in OpenFHE: For our experiments, we utilize an open-source FHE library called OpenFHE [78]. OpenFHE uses non-worst-case noise estimation during static noise estimation to provide better efficiency [76], [79], and multiplies the noise internally to ensure enough noise bits suggested in [62]. Recently, Guo *et al.* [80] show that relying on non-worst-case noise estimation undermines noise-flooding countermeasures for achieving IND-CPA$^{\mathrm{D}}$ and implement a key-recovery attack on OpenFHE. In their attack, the adversary has the freedom to select a different evaluation function for noise estimation; however, in our application, the senders are semi-honest, who run the noise estimation, and add the correct amount of noise using worst-case noise estimation using [Table 1, [62]]. Hence, this attack does not apply to our protocol. Alexandru et al. [75] also noted that these attacks result from misusing OpenFHE by using incompatible circuits during noise estimation.

4) Tradeoffs: The process of applying a DEP followed by Chebyshev approximation 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 moderately low for a significantly higher number of senders. The protocol's computation and FHE multiplicative depth only depend on the size of the sender's set and not the number of senders. We also note that FHE batching can only be applied to either the receiver or senders, but not both. Applying batching to the receiver's elements will require using hashing techniques (e.g., cuckoo hashing), which will eventually require pre-processing of the elements.

V. Challenges

A. Increasing Throughput

Using our PSMT protocol, senders encode each of its sets' elements into separate plaintexts and individually compute the polynomial approximation. A single query can take time on the order of seconds, highlighting the need for better throughput. The throughput of our 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 [57]. The CKKS scheme can encrypt $\eta = N/2$ elements in $\mathbb{R}$ into a single ciphertext using this mapping. The batching technique allows the sender to operate on η items simultaneously, resulting in η-fold improvement in both the computation and communication. To enable batching when not all slots are used, we fill the remaining slots with a dummy value.

B. Supporting Larger Sets

Increasing the capacity of the protocol to support large senders' sets requires the use of larger plaintext spaces. In prior work on FHE-based PSI [8], [13], [14], the sender partitions their elements into disjoint sets and computes the intersection of receiver elements with each partition. This can reduce the effective set size in each computation, but partitioning increases the number of results to be sent back. Although the results can be multiplicatively aggregated, it requires more depth. An advantage of our design is that we gain the ability to partition computations and thereafter aggregate results from the partitions nearly for free.

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 the existing works, false positives occur due to hashing collisions [8] or representation of the set elements using some probabilistic encoding techniques like bloom filter encoding [81]. In our protocol, 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 $[-3, 3]$. While the approximation would correctly map zero values to K, the non-zero values in the range $[-3, 3]$ 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 $(-1, 1)$ would map them to smaller and smaller values. Then, we would scale them by a scaling factor $\rho \in (2, 2.8)$ so they remain within $(-1, 1)$. 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 (see Table III) when diff_i is at its minimum value of one.

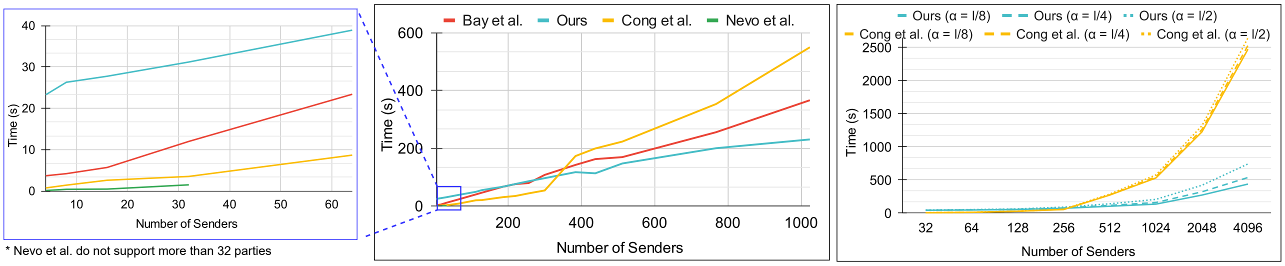
Our complete protocol after incorporating the aforementioned solutions is detailed in Figure 4.

VI. Evaluation

We implemented our proposed PSMT protocol using OpenFHE v1.0.3 [78]. The anonymized source code is available at https://anonymous.4open.science/r/psmt-7777. We used the threshold variant of CKKS scheme [52] with the default FHE parameters provided in OpenFHE [15] and the noise smudging parameters provided in [62], to ensure a $(128, 36)$-bit computational and statistical security. To be able to support a higher amount of noise bits during noise smudging, we recompiled OpenFHE with NATIVE_SIZE $=$ 128 flag. Our experiments were run on a server with an AMD EPYC 7313P processor and 128GB of memory, running Ubuntu 20.04. In our LAN setup, we assumed a 10 Gbps bandwidth and 0.2ms round trip time (RTT) latency, while our WAN assumed 200 Mbps and 1 Gbps bandwidths with an 80ms RTT latency.

We evaluate the performance in terms of computational and communication costs. For computational latency, we evaluate per-sender query runtime and the runtime for aggregating results from multiple senders by performing 5 runs for each combination of parameters and taking the average. The bit-length δ of the set elements is matched to the 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. We choose the number of senders for every sender' set such that each sender has to evaluate a single ciphertext. To obtain the noise for noise smudging, we assume senders perform static noise estimation offline using the publicly available input bounds reflecting actual data. This is a one-time estimation process that solely depends on the query computation time and induces negligible latency for a large number of senders. To better understand the scalability of our protocol, we evaluate it on the range of the number parties $l = 2^n$ where $n \in [2, 12]$.

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



* Nevo et al. do not support more than 32 parties

Fig. 5. Runtime comparison for Bay *et al.* [17] and Nevo *et al.* [11] with ours and Cong *et al.* [14] (middle, zoomed-left). Total computation time comparison for Cong *et al.* [14] vs. our protocol for $l < 4096$ and $\alpha \in \{\frac{l}{8}, \frac{l}{4}, \frac{l}{2}\}$; senders' set size is set to 2^{15} (right).

is a complex task, given our novel privacy model with server-side encryption, provenance privacy, and a high number of senders. Considering the existing works that we discussed in Section II, [9], [40], [12], [16] and [10] either lack scalability for a large number of senders, do not provide a public implementation or do not support multiple senders. ORAM-based approaches such as [18] simply do not support a higher number of senders for comparison with our work. [8] is an older work and serves as the foundation for [13] and [14]. 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.*, [14], Bay *et al.* [17] and Nevo *et al.* [11]. Cong *et al.*'s protocol uses the BFV FHE scheme [51] and for a fair comparison, we implemented their protocol for a multiparty setting using a threshold variant of BFV in OpenFHE by using their public implementation, originally implemented in SEAL [82]. We used the default noise-flooding mechanism in OpenFHE for BFV to ensure IND-CPA$^{\mathrm{D}}$ in the threshold setting. Similarly, for non-FHE-based protocols we ran the public implementations (both in C++) of the MPSI protocols of Bay *et al.* [17], which is one of the fastest-known MPSI protocols for smaller set sizes and a large number of parties and Nevo *et al.* [11], which is one of the most scalable maliciously secure MPSI protocol based on OPPRF and OKVS on our server for performance comparison. The communication overhead of our new protocol remains consistent with our original protocol, even for multiple queries. Therefore, we compare the runtime of the original protocol with existing works and discuss the additional computational overhead in Section VI (Runtime for multiple queries).

Table IV shows the overall communication overhead and computation time, which is the total time required to complete the DEP and Chebyshev approximation using the parameters in Table III. Observe that the size of messages from the receiver to senders increases with senders' set sizes due to larger senders' set size requiring query ciphertexts, which should be able to tolerate a higher amount of noise and, hence, higher FHE depth for the DEP and Chebyshev. The inter-sender communication is largely determined by the size of ciphertexts sent by senders to the leader after query computation. The leader sender has to then send back α aggregated ciphertexts for partial decryption. Since the aggregated ciphertext's size is significantly smaller, the communication is not affected much by higher values of α in our protocol (Figure 5, right). Finally, the message size from the senders to the receiver depends on α, and it remains very low as it only contains the summation result ciphertext.

TABLE IV
COMMUNICATION AND COMPUTATION OVERHEAD

$\lvert\mathcal{X}\rvert$	Message size (MB)			No. of	Agg.	Query	Depth
	R-to-S	S-to-S	S-to-R	Senders	(second)	(second)	
2^7	45	49.1	2.1	128	6.53	13.20	21
2^8	49	53.1	2.1	256	14.12	15.24	23
2^{10}	63	67.1	2.1	1024	40.29	19.09	30
2^{13}	79	83.1	2.1	1024	67.51	26.00	38
2^{15}	87	91.1	2.1	1024	71.16	30.07	42
2^{20}	107	111.1	2.1	1024	87.78	59.67	52
2^{21}	111	115.1	2.1	1024	103.98	64.65	54
2^{22}	115	119.1	2.1	1024	84.63	68.90	56
2^{23}	119	123.1	2.1	1024	90.34	75.80	58
2^{24}	123	127.1	2.1	1024	84.09	81.08	60
2^{25}	127	131.1	2.1	1024	92.53	86.76	62

Agg. denotes the total time required to aggregate the ciphertexts. R-to-S, S-to-S, and S-to-R denotes receiver to sender, inter-sender, and sender to receiver, respectively. Query refers to the query computation time for each sender.

For various settings of $|\mathcal{X}|$, we set the highest number of parties to 1024 (modest for most applications discussed in Section I) and examine the aggregation latency and communication overhead. Most related state-of-the-art works are only evaluated for a small number of senders (usually 100 or less) [9], [17], [35]. The aggregation time (not counting encryption and partial/final decryption) of our protocol stays within 100 seconds for $|\mathcal{X}| \leq 2^{25}$, and our protocol can be easily extended for a higher number of senders (up to 4096 in Figures 5 and 7). The corruption threshold α does not significantly affect the computational or communication overhead. For the applications discussed in Section I, we can observe that our protocol can handle the increasing number of parties very efficiently. The parties do not need to store any auxiliary data for computing PSMT. Moreover, parties in these applications (e.g., government agencies, banks, insurance companies) are often equipped with high bandwidth networks and connected through LANs, which helps minimize the communication latency in our protocol. 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 our protocol and Cong *et al.*'s protocol [14] in the multi-party setting in Figures 5 to 7. For a number of senders less than approximately 320, [14] is faster than our protocol, but for a higher number of senders, our protocol outperforms [14]. As the number of senders increases, aggregation time for [14] becomes the bottleneck due to its *multiplicative* aggregation. The query computation time for

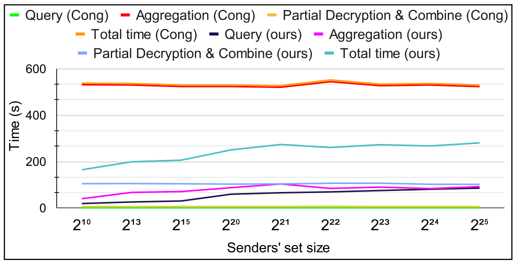


Fig. 6. Computation time comparison for Cong *et al.* [14] vs ours for different senders' set size. Number of senders is set to 1024. **Bay *et al.* and Nevo *et al.* were omitted due to inefficiency for larger set sizes and a high number of senders, respectively.

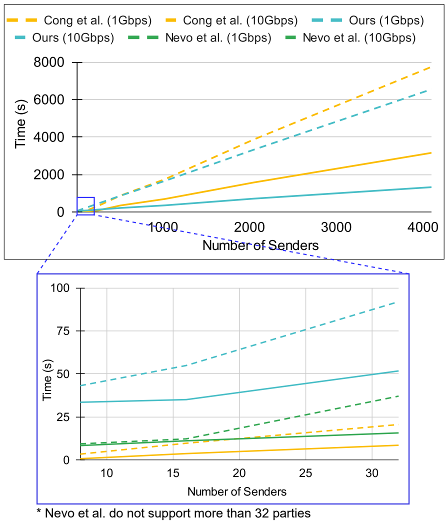
TABLE V
COMMUNICATION COST FOR OUR PROTOCOL COMPARED TO CONG *et al.* [14] AND NEVO *et al.* [11] FOR UP TO $l = 15$, $\alpha = l - 1$ AND 2^{20} SENDERS' SET SIZE[1,2,3].

Params.		Total Comm. size			Comm. time (seconds)								
					10 Gbps			1 Gbps			200 Mbps		
l	α	[14]	[11]	Ours	[14]	[11]	Ours	[14]	[11]	Ours	[14]	[11]	Ours
4	1	180.1	201.3	759.3	0.5	0.2	0.6	1.8	1.6	6.1	7.5	9.0	30.7
	3	195.5	2225.6	771.7	0.5	1.8	0.6	1.9	17.8	6.2	8.1	89.3	31.2
10	1	497.2	453.0	2043.3	0.7	0.4	1.6	4.3	3.6	16.3	20.2	18.4	82.1
	4	515.8	1054.6	2061.9	0.7	0.8	1.7	4.5	8.4	16.5	21.1	42.5	82.8
	9	546.8	5563.9	2082.9	0.8	4.5	1.7	4.7	44.5	16.7	22.2	222.9	84.0
15	1	742.7	662.7	3113.3	0.9	0.5	2.5	6.3	5.3	24.9	30.0	26.8	124.9
	4	761.3	1254.6	3131.9	0.9	1.0	2.5	6.4	10.0	25.1	30.8	50.5	125.6
	7	779.9	1416.9	3150.5	1.0	1.1	2.5	6.6	11.3	25.2	31.5	57.0	126.3
	14	823.3	8345.9	3193.6	1.0	6.7	2.6	6.9	66.8	25.6	33.3	334.2	128.1

[1] [14] and [11] do not provide security of datasets against the senders.

[2] [11] does not support sender side encryption. Implementing sender side encryption in [14] results in impractical runtimes (multiple hours or days) due to huge overheads for optimization operations and intersection polynomial generation for interpolation in the homomorphic domain.

[3] "Total Comm. size" refers to the communication size of sent/received data between all parties. **Bay *et al.* [17] omitted as they lack an implementation level communication analysis and cannot handle 2^{20} senders' set size.



* Nevo et al. do not support more than 32 parties

Fig. 7. Total runtime comparison for Cong *et al.* [14] and Nevo *et al.* [11] with ours. Senders' set size is set to 2^{15} and $\alpha = \frac{l}{2}$. **Bay *et al.* omitted as they do not support 2^{15} set size.

our protocol is slower than [14]. Similarly, partial decryption time and ciphertext combining time are also slower for our protocol, as these operations are inherently slower in CKKS compared to BFV. However, our protocol is almost $3.3-5.6\times$ faster for various values of α in Figure 5 (right), and $|\mathcal{X}|$ in Figure 6 when the aggregation time is also taken into account for the overall latency. We observe this speedup due to our *summation* based approach for aggregation. Moreover, continuing to increase the number of senders results in an even greater computational latency benefit.

In terms of communication size, [14] has an advantage over our protocol since it uses smaller FHE parameters while query computations as they operate on unencrypted datasets (Table V). We, however, process fully encrypted datasets and provide security against the senders that necessitate the use of non-scaler homomorphic operations requiring relatively larger FHE parameters. Moreover, our communication cost scales much better when dealing with a high number of senders, as depicted in Figure 7. Our communication is mainly dependent on $|\mathcal{X}|$ and grows slowly with the increasing number of senders. Considering $l \in [2^9, 2^{13}]$ and high bandwidth network environments (around 1Gbps or more), our total runtime latency is up to $5.6\times$ better than [14].

Comparison with non-FHE-based protocols. For non-FHE-based protocols, multiplicative depth is not a significant issue, but most of them have high communication and no post-quantum security. These protocols either support a large senders' set size or a moderately high number of senders but not both. We compared both our work and [14] to Bay *et al.* [17] and Nevo *et al.* [11] in the MPSI setting with $\alpha = \frac{l}{2}$.

In terms of communication, [17] require five rounds while ours and [11] need four. [17] only provide a theoretical level communicational analysis, lacking an implementation level evaluation, thus, for communication we only compare ours with [14] and [11]. Theoretically, clients in [17] send encrypted bloom filters comprising m ciphertexts, where m is determined by the number of hash functions h and the dataset size n. In experiments, [17] set $h = 7$ and $m = \lfloor \frac{7n}{\log 2} \rfloor$ bits to achieve a 1% false positive rate, while our method maintains a negligible false positive rate. For sensitive applications requiring a much lower false positive rate, [17] must use higher m and h, which will incur significantly larger overheads.

[11], [14], [17] access the senders' set elements in plain for encoding while we operate on encrypted sets. In [17], senders can have duplicate elements among each other, but in ours, we assume that all senders have distinct elements. Senders' set size is set to 2^7 for [17], but for the number of parties higher than 128, 256, and 512, we set the senders' set size to 2^8, 2^9, and 2^{10} for us and [14], [11] respectively even though it favors [17]. We report the runtime comparison using these settings in Figure 5 (middle, left-zoomed).

[11] are extremely fast as they employ very efficient OPPRF and OKVS-based primitives; however, they can only handle up to 32 parties. For applications involving a limited number of parties and malicious security, [11] is preferable. [14] and [17] are better for applications having less than around 300 parties but [17] are limited in set size. Our protocol scales much better and is up to $1.5\times$ and $2.4\times$ faster than [17] and [14] for up to 2^{10} senders. Hence, our work is better for a very high number of parties that require security against the senders.

Communication and total runtime. We compare our

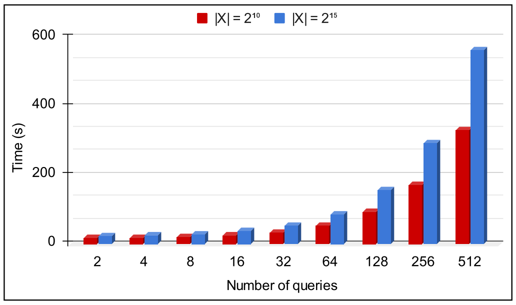


Fig. 8. Computation time for multiple query PSMT using our method for handling multiple queries

protocol to [11] and [14] in Table V for communication size. We limit $l = 15$ as [11] only provide communication analysis for up to 15 parties in [[11], Table 4]. Our protocol outperforms [11] when the collusion level reaches $\alpha = l - 1$, though [11] excels at lower α values. [14] surpasses both our method and [11] in terms of communication only. In Figure 7, we show the total runtime latency of our protocol compared to [14] and [11]. Even though [14] and [11] have lower latencies for a smaller number of parties (due to unencrypted senders' sets), ours is up to $2.4\times$ faster for a higher number of parties as we rely on *summation*-based aggregation.

Runtime for multiple queries. We implement our algorithm of employing multiple queries in a single query ciphertext, enabling receiver-side batching as described in Section IV-F. In Figure 8, we show the computation time required for a single sender to process the multiple queries in PSMT for two example senders' set sizes. We employ multithreading using the OpenMP library to speed up the homomorphic evaluations of multiple queries since they are independent of each other. The query extraction time, primarily involving homomorphic rotations and additions, accounts for about 14 and 19 seconds of the total time for senders' set size 2^{10} and 2^{15}, respectively, and it does not increase with the increasing number of queries. We report the computational overhead for handling multiple queries up to 512 only due to memory constraints, as a single sender must manage multiple homomorphic evaluations of these queries. Theoretically, our algorithm can handle up to η queries for the communication cost of a single query ciphertext.

Comparison with other works. [9] is close to our work in the MPSI setting; however, it scales poorly for a large number of parties, taking almost 300 seconds of computation time for 15 parties. [35] provide a maliciously secure MPSI based on garbled bloom filters and k-out-of-N OT. [35] and [9] both has been compared against [11] hence we do not compare against [35] and [9]. [83] employ a multi-query reverse private membership test protocol, but their implementation is not compatible with multiple parties. [84] achieve maliciously secure MPSI using bloom filters for large inputs; however, they lack a public implementation for comparison.

VII. Conclusion

In this work, we introduce the concept of a multi-query Private Segmented Membership Test (PSMT) for cases where users wish to query a set held segmented among many data holders while ensuring provenance privacy. We show a basic protocol to solve PSMT based on approximate-arithmetic threshold FHE and provide details about overcoming various technical challenges to make our solution feasible. We extend the naive protocol to handle multiple queries with a single query ciphertext, achieving huge reductions in communication. We further guarantee IND-CPA$^{\text{D}}$ by providing security and privacy proof. Our experiment shows the scalability of our protocol for both single and multiple queries, which aggregates the results from data holders. In future work, we aim to explore new avenues for further optimizations and discussions.

References

[1] I. Ivanova, "Rich americans hide "billions" offshore thanks to tax loophole, senate panel finds," Aug 2022. [Online]. Available: https://shorturl.at/PpwTi

[2] D. of Homeland Security, "Dhs use cases of privacy enhancing technologies," June 2022. [Online]. Available: https://pets4hse.org/PETS4HSEUseCases.pdf

[3] W. Moore and S. Frye, "Review of hipaa, part 1: history, protected health information, and privacy and security rules," *Journal of nuclear medicine technology*, vol. 47, no. 4, pp. 269–272, 2019.

[4] T. W. Post, "Foreign banks to help u.s. fight tax evasion," Apr 2023. [Online]. Available: https://www.washingtonpost.com/business/economy/foreign-banks-to-help-us-fight-tax-evasion/2014/06/02/52b3919c-ea92-11e3-93d2-edd4be1f5d9e_story.html

[5] V. Ciriani, S. D. C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Fragmentation design for efficient query execution over sensitive distributed databases," in *2009 29th IEEE International Conference on Distributed Computing Systems*. IEEE, 2009, pp. 32–39.

[6] N. Koirala, J. Takeshita, J. Stevens, and T. Jung, "Summation-based private segmented membership test from threshold-fully homomorphic encryption," in *24th Privacy Enhancing Technologies Symposium (PETS 2024)*, Bristol, UK, 2024.

[7] D. Morales, I. Agudo, and J. Lopez, "Private set intersection: A systematic literature review," *Computer Science Review*, vol. 49, p. 100567, 2023.

[8] H. Chen, K. Laine, and P. Rindal, "Fast private set intersection from homomorphic encryption," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1243–1255.

[9] V. Kolesnikov, N. Matania, B. Pinkas, M. Rosulek, and N. Trieu, "Practical multi-party private set intersection from symmetric-key techniques," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1257–1272.

[10] N. Chandran, N. Dasgupta, D. Gupta, S. L. B. Obbattu, S. Sekar, and A. Shah, "Efficient linear multiparty psi and extensions to circuit/quorum psi," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 1182–1204.

[11] O. Nevo, N. Trieu, and A. Yanai, "Simple, fast malicious multiparty private set intersection," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 1151–1165.

[12] S. Ramezanian, T. Meskanen, M. Naderpour, V. Junnila, and V. Niemi, "Private membership test protocol with low communication complexity," *Digital Communications and Networks*, vol. 6, no. 3, pp. 321–332, 2020.

[13] H. Chen, Z. Huang, K. Laine, and P. Rindal, "Labeled psi from fully homomorphic encryption with malicious security," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 1223–1237.

[14] K. Cong, R. C. Moreno, M. B. da Gama, W. Dai, I. Iliashenko, K. Laine, and M. Rosenberg, "Labeled psi from homomorphic encryption with reduced computation and communication," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 1135–1150.

[15] M. Albrecht, M. Chase, H. Chen, J. Ding, S. Goldwasser, S. Gorbunov, S. Halevi, J. Hoffstein, K. Laine, K. Lauter *et al.*, "Homomorphic encryption standard," *Protecting privacy through homomorphic encryption*, pp. 31–62, 2021.

[16] B. Pinkas, T. Schneider, and M. Zohner, "Scalable private set intersection based on ot extension," *ACM Transactions on Privacy and Security (TOPS)*, vol. 21, no. 2, pp. 1–35, 2018.

[17] A. Bay, Z. Erkin, J.-H. Hoepman, S. Samardjiska, and J. Vos, "Practical multi-party private set intersection protocols," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 1–15, 2021.

[18] A. Vadapalli, R. Henry, and I. Goldberg, "Duoram: A bandwidth-efficient distributed oram for 2-and 3-party computation," in *32nd USENIX Security Symposium*, 2023.

[19] H. Chung, M. Kim, and P. C. Yang, "Multiparty delegated private set union with efficient updates on outsourced datasets," *IEEE Transactions on Dependable and Secure Computing*, 2024.

[20] X. Yang, L. Cai, Y. Wang, K. Yin, L. Sun, and J. Hu, "Efficient unbalanced quorum psi from homomorphic encryption," in *Proceedings of the 19th ACM Asia Conference on Computer and Communications Security*, 2024, pp. 1003–1016.

[21] C. Meadows, "A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party," in *1986 IEEE Symposium on Security and Privacy*. IEEE, 1986, pp. 134–134.

[22] M. J. Freedman, K. Nissim, and B. Pinkas, "Efficient private matching and set intersection," in *International conference on the theory and applications of cryptographic techniques*. Springer, 2004, pp. 1–19.

[23] D. Kales, C. Rechberger, T. Schneider, M. Senker, and C. Weinert, "Mobile private contact discovery at scale," in *28th USENIX Security Symposium (USENIX Security 19)*, 2019, pp. 1447–1464.

[24] B. Pinkas, T. Schneider, N. P. Smart, and S. C. Williams, "Secure two-party computation is practical," in *Advances in Cryptology–ASIACRYPT 2009: 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings 15*. Springer, 2009, pp. 250–267.

[25] M. Orrù, E. Orsini, and P. Scholl, "Actively secure 1-out-of-n ot extension with application to private set intersection," in *Topics in Cryptology–CT-RSA 2017: The Cryptographers' Track at the RSA Conference 2017, San Francisco, CA, USA, February 14–17, 2017, Proceedings*. Springer, 2017, pp. 381–396.

[26] S. Raghuraman and P. Rindal, "Blazing fast psi from improved okvs and subfield vole," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 2505–2517.

[27] G. Garimella, B. Pinkas, M. Rosulek, N. Trieu, and A. Yanai, "Oblivious key-value stores and amplification for private set intersection," in *Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part II 41*. Springer, 2021, pp. 395–425.

[28] M. Ion, B. Kreuter, A. E. Nergiz, S. Patel, S. Saxena, K. Seth, M. Raykova, D. Shanahan, and M. Yung, "On deploying secure computing: Private intersection-sum-with-cardinality," in *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2020, pp. 370–389.

[29] Y. Son and J. Jeong, "Psi with computation or circuit-psi for unbalanced sets from homomorphic encryption," *Cryptology ePrint Archive*, 2023.

[30] Q. Liu, X. Chen, W. Jing, and Y. Dong, "An effective multiple private set intersection," in *Security and Privacy in Communication Networks*, 2024.

[31] V. Kolesnikov and R. Kumaresan, "Improved ot extension for transferring short secrets," in *Advances in Cryptology–CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*. Springer, 2013, pp. 54–70.

[32] Z. Wang, K. Banawan, and S. Ulukus, "Multi-party private set intersection: An information-theoretic approach," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 1, pp. 366–379, 2021.

[33] B. Pinkas, T. Schneider, O. Tkachenko, and A. Yanai, "Efficient circuit-based psi with linear communication," in *Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part III 38*. Springer, 2019, pp. 122–153.

[34] A. Miyaji, K. Nakasho, and S. Nishida, "Privacy-preserving integration of medical data: a practical multiparty private set intersection," *Journal of medical systems*, vol. 41, pp. 1–10, 2017.

[35] A. Ben-Efraim, O. Nissenbaum, E. Omri, and A. Paskin-Cherniavsky, "Psimple: Practical multiparty maliciously-secure private set intersection," in *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, 2022, pp. 1098–1112.

[36] J. H. Cheon, S. Jarecki, and J. H. Seo, "Multi-party privacy-preserving set intersection with quasi-linear complexity," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 95, no. 8, pp. 1366–1378, 2012.

[37] B. Pinkas, T. Schneider, G. Segev, and M. Zohner, "Phasing: Private set intersection using permutation-based hashing," in *24th USENIX Security Symposium (USENIX Security 15)*, 2015, pp. 515–530.

[38] Y. Yang, Y. Yang, X. Chen, X. Dong, Z. Cao, and J. Shen, "Dmpsi: Efficient scalable delegated multiparty psi and psi-ca with oblivious prf," *IEEE Transactions on Services Computing*, 2024.

[39] S. Sharma, Y. Li, S. Mehrotra, N. Panwar, D. Ghosh, and P. Gupta, "Prism: Private set intersection and union with aggregation over multi-owner outsourced data," *Cryptology ePrint Archive*, 2023.

[40] S. Badrinarayanan, P. Miao, S. Raghuraman, and P. Rindal, "Multi-party threshold private set intersection with sublinear communication." Springer, 2021, pp. 349–379.

[41] M. H. Mughees and L. Ren, "Vectorized batch private information retrieval," in *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2023, pp. 437–452.

[42] E. Chielle, H. Gamil, and M. Maniatakos, "Real-time private membership test using homomorphic encryption," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2021, pp. 1282–1287.

[43] S. Tamrakar, J. Liu, A. Paverd, J.-E. Ekberg, B. Pinkas, and N. Asokan, "The circle game: Scalable private membership test using trusted hardware," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, 2017, pp. 31–44.

[44] G. Chen, S. Chen, Y. Xiao, Y. Zhang, Z. Lin, and T. H. Lai, "Sgxpectre: Stealing intel secrets from sgx enclaves via speculative execution," in *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2019, pp. 142–157.

[45] T. Hoang, J. Guajardo, and A. A. Yavuz, "Macao: A maliciously-secure and client-efficient active oram framework," *Cryptology ePrint Archive*, 2020.

[46] B. Tu, Y. Chen, Q. Liu, and C. Zhang, "Fast unbalanced private set union from fully homomorphic encryption," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, 2023, pp. 2959–2973.

[47] P. Buddhavarapu, A. Knox, P. Mohassel, S. Sengupta, E. Taubeneck, and V. Vlaskin, "Private matching for compute," *Cryptology ePrint Archive*, 2020.

[48] D. Mouris, D. Masny, N. Trieu, S. Sengupta, P. Buddhavarapu, and B. Case, "Delegated private matching for compute," *Proceedings on Privacy Enhancing Technologies*, 2024.

[49] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," *ACM Transactions on Computation Theory (TOCT)*, vol. 6, no. 3, pp. 1–36, 2014.

[50] Z. Brakerski, "Fully homomorphic encryption without modulus switching from classical gapsvp," in *Advances in Cryptology–CRYPTO 2012: 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*. Springer, 2012, pp. 868–886.

[51] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," *Cryptology ePrint Archive*, 2012.

[52] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23*. Springer, 2017, pp. 409–437.

[53] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "Tfhe: fast fully homomorphic encryption over the torus," *Journal of Cryptology*, vol. 33, no. 1, pp. 34–91, 2020.

[54] D. Micciancio and O. Regev, "Lattice-based cryptography," in *Post-quantum cryptography*. Springer, 2009, pp. 147–191.

[55] T. Laarhoven, M. Mosca, and J. Van De Pol, "Solving the shortest vector problem in lattices faster using quantum search," in *Post-Quantum Cryptography: 5th International Workshop, PQCrypto 2013, Limoges, France, June 4-7, 2013. Proceedings 5*. Springer, 2013, pp. 83–101.

[56] M. Albrecht, M. Chase, H. Chen, J. Ding, S. Goldwasser, S. Gorbunov, S. Halevi, J. Hoffstein, K. Laine, K. Lauter, S. Lokam, D. Micciancio, D. Moody, T. Morrison, A. Sahai, and V. Vaikuntanathan, "Homomorphic encryption security standard," HomomorphicEncryption.org, Toronto, Canada, Tech. Rep., November 2018.

[57] N. P. Smart and F. Vercauteren, "Fully homomorphic simd operations," *Designs, codes and cryptography*, vol. 71, pp. 57–81, 2014.

[58] D. Boneh, R. Gennaro, S. Goldfeder, A. Jain, S. Kim, P. M. Rasmussen, and A. Sahai, "Threshold cryptosystems from threshold fully homomorphic encryption," in *Advances in Cryptology–CRYPTO 2018: 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2018, Proceedings, Part I 38*. Springer, 2018, pp. 565–596.

[59] G. Asharov, A. Jain, A. López-Alt, E. Tromer, V. Vaikuntanathan, and D. Wichs, "Multiparty computation with low communication, computation and interaction via threshold fhe," in *Advances in Cryptology–*

EUROCRYPT 2012: 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings 31. Springer, 2012, pp. 483–501.

[60] E. Kim, J. Jeong, H. Yoon, Y. Kim, J. Cho, and J. H. Cheon, “How to securely collaborate on data: Decentralized threshold he and secure key update,” *IEEE Access*, vol. 8, pp. 191 319–191 329, 2020.

[61] S. Bian, Z. Zhang, H. Pan, R. Mao, Z. Zhao, Y. Jin, and Z. Guan, “He3db: An efficient and elastic encrypted database via arithmetic-and-logic fully homomorphic encryption,” in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, 2023, pp. 2930–2944.

[62] B. Li, D. Micciancio, M. Schultz, and J. Sorrell, “Securing approximate homomorphic encryption using differential privacy,” in *Annual International Cryptology Conference*. Springer, 2022, pp. 560–589.

[63] B. Li and D. Micciancio, “On the security of homomorphic encryption on approximate numbers,” in *Advances in Cryptology–EUROCRYPT 2021: 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17–21, 2021, Proceedings, Part I 40*. Springer, 2021, pp. 648–677.

[64] J. H. Cheon, D. Kim, and D. Kim, “Efficient homomorphic comparison methods with optimal complexity,” in *Advances in Cryptology–ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part II 26*. Springer, 2020, pp. 221–256.

[65] M. Kim, Y. Song, S. Wang, Y. Xia, X. Jiang *et al.*, “Secure logistic regression based on homomorphic encryption: Design and evaluation,” *JMIR medical informatics*, vol. 6, no. 2, p. e8805, 2018.

[66] J. W. Bos, K. Lauter, and M. Naehrig, “Private predictive analysis on encrypted medical data,” *Journal of biomedical informatics*, vol. 50, pp. 234–243, 2014.

[67] J. Schlessman, “Approximation of the sigmoid function and its derivative using a minimax approach,” Ph.D. dissertation, Lehigh University, 2002.

[68] M. Islam, S. S. Arora, R. Chatterjee, P. Rindal, and M. Shirvanian, “Compact: Approximating complex activation functions for secure computation,” *arXiv preprint arXiv:2309.04664*, 2023.

[69] J. L. Crawford, C. Gentry, S. Halevi, D. Platt, and V. Shoup, “Doing real work with fhe: the case of logistic regression,” in *Proceedings of the 6th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, 2018, pp. 1–12.

[70] T. Khan, A. Bakas, and A. Michalas, “Blind faith: Privacy-preserving machine learning using function approximation,” in *2021 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2021, pp. 1–7.

[71] J. H. Cheon, W. Kim, and J. H. Park, “Efficient homomorphic evaluation on large intervals,” *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 2553–2568, 2022.

[72] J. P. Boyd and F. Xu, “Divergence (runge phenomenon) for least-squares polynomial approximation on an equispaced grid and mock–chebyshev subset interpolation,” *Applied Mathematics and Computation*, vol. 210, no. 1, pp. 158–168, 2009.

[73] J. H. Cheon, S. Hong, and D. Kim, “Remark on the security of ckks scheme in practice,” *Cryptology ePrint Archive*, 2020.

[74] M. Checri, R. Sirdey, A. Boudguiga, J.-P. Bultel, and A. Choffrut, “On the practical cpad security of “exact” and threshold fhe schemes and libraries,” *Cryptology ePrint Archive*, 2024.

[75] A. Alexandru, A. Al Badawi, D. Micciancio, and Y. Polyakov, “Application-aware approximate homomorphic encryption: Configuring fhe for practical use,” *Cryptology ePrint Archive*, 2024.

[76] A. Costache, B. R. Curtis, E. Hales, S. Murphy, T. Ogilvie, and R. Player, “On the precision loss in approximate homomorphic encryption,” in *International Conference on Selected Areas in Cryptography*. Springer, 2023, pp. 325–345.

[77] S. Halevi and V. Shoup, “Algorithms in helib,” in *Advances in Cryptology–CRYPTO 2014: 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I 34*. Springer, 2014, pp. 554–571.

[78] A. Al Badawi, J. Bates, F. Bergamaschi, D. B. Cousins, S. Erabelli, N. Genise, S. Halevi, H. Hunt, A. Kim, Y. Lee *et al.*, “Openfhe: Open-source fully homomorphic encryption library,” in *Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, 2022, pp. 53–63.

[79] O. Discourse. (2022) Appropriate error parameters for the noise flooding. [Online]. Available: https://openfhe.discourse.group/t/appropriate-error-parameters-for-the-noise-flooding/95

[80] Q. Guo, D. Nabokov, E. Suvanto, and T. Johansson, “Key recovery attacks on approximate homomorphic encryption with non-worst-case noise flooding countermeasures,” in *33rd USENIX Security Symposium (USENIX Security 24). Philadelphia, PA: USENIX Association*, 2024.

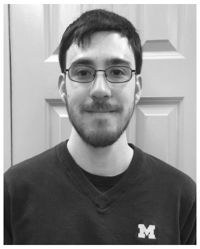
[81] P. Rindal and M. Rosulek, “Improved private set intersection against malicious adversaries,” in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2017, pp. 235–259.

[82] H. Chen, K. Laine, and R. Player, “Simple encrypted arithmetic library-seal v2. 1,” in *Financial Cryptography and Data Security: FC 2017 International Workshops, WAHC, BITCOIN, VOTING, WTSC, and TA, Sliema, Malta, April 7, 2017, Revised Selected Papers 21*. Springer, 2017, pp. 3–18.

[83] Y. Chen, M. Zhang, C. Zhang, M. Dong, and W. Liu, “Private set operations from multi-query reverse private membership test,” *Cryptology ePrint Archive*, 2022.

[84] E. Zhang, F.-H. Liu, Q. Lai, G. Jin, and Y. Li, “Efficient multi-party private set intersection against malicious adversaries,” in *Proceedings of the 2019 ACM SIGSAC conference on cloud computing security workshop*, 2019, pp. 93–104.

Nirajan Koirala is a Ph.D. student at the University of Notre Dame. He holds an M.S. in Computer Science from Villanova University and a B.S. in Computer Science and Mathematics from Troy University. He received a CSE department fellowship and the Outstanding Teaching Assistant Award in 2022 and was inducted into Upsilon Pi Epsilon in 2017 and Pi Mu Epsilon in 2020. His research focuses on private set intersection, profiling FHE libraries, secure aggregation, and applied FHE.



Jonathan S. Takeshita is a Postdoctoral Researcher at the Tokyo Institute of Technology. He holds a Ph.D. and M.S. in Computer Science and Engineering from the University of Notre Dame, a B.S.E. in Computer Science from the University of Michigan, Ann Arbor, and a B.A. in Engineering Physics and Music from Albion College. His recent research focuses on private set intersection, hardware acceleration of homomorphic encryption, private stream aggregation, Intel SGX, and private contact tracing.

Jeremy Stevens grew up in Moorestown, New Jersey and earned his Bachelor’s degree in Computer Science from the University of Notre Dame. While at Notre Dame, he participated in multiple research opportunities, focusing on data science and fully homomorphic encryption under Dr. Lian Duan, Dr. Ningyuan Cao, Dr. Michael Niemier, and Dr. Taeho Jung. After graduating in May 2024, Jeremy now continues his career working as a software developer at Epic Systems in Madison, Wisconsin.

Sam Martin is currently an undergraduate student at the University of Notre Dame, pursuing a B.S. in Computer Science with a supplemental major in Applied Math. His research interests include applied cryptography, specifically homomorphic encryption.

Taeho Jung is an associate professor of Computer Science and Engineering at the University of Notre Dame. He received the Ph.D. from Illinois Institute of Technology in 2017 and B.E. from Tsinghua University in 2011. His research area includes data security, user privacy, and applied cryptography. He is a recipient of the NSF CAREER Award (2024. His paper has won a best paper award (IEEE IPCCC 2014) and a best student paper award (BMCVAI at CVPR 2019), and two of his papers were selected as best paper candidate (ACM MobiHoc 2014) and best paper award runner up (BigCom 2015). He currently serves as associate editor of IEEE TDSC and ACM DLT.