Delegated PSI from Homomorphic Encryptions

Sicheng Wei[1] and Jingwei Hu[1]

Nanyang Technological University, Singapore, {davidhu}@ntu.edu.sg

Abstract. This paper presents an efficient protocol for private set intersection in a setting with multiple set owners and a semi-honest cloud server. The core idea is to reduce the intersection computation to secure operations over Bloom filters, enabling both scalability and efficiency. By leveraging this transformation, our protocols achieve strong privacy guarantees while minimizing computation and communication overhead.

Keywords: Multi-party Computation · Private Set Intersection · Cloud-assisted Computing

1 Introduction

This paper presents a novel solution to the Delegated Private Set Intersection (D-PSI) problem using Threshold Additive/Fully Homomorphic Encryption. Unlike the standard multi-party private set intersection (PSI) setting, an additional computing entitya cloud serveris introduced. The framework consists of three types of computing parties: a cloud server C, delegated users $A_1, A_2, \dots, A_N$, and a querying user $B \in \{A_i\}_i$. Each delegated user A_i delegates their private set to the cloud server C, while the querying user B not only delegates their set to C but also eventually learns the intersection result. The goal of D-PSI is to collaboratively compute the intersection while ensuring that individual sets remain private.

In the D-PSI model, the cloud server plays a crucial role by handling the majority of the computation and communication workload while remaining untrusted by the other computing parties, including $\{A_i\}$. The intersection computation is represented by a function f, such that:

$$\bigcap_i A_i \leftarrow f(A_1, \dots, A_N).$$

A natural approach is for all users $\{A_i\}$ to encrypt their sets and send them to the cloud server C, which then performs homomorphic computation of the intersection function:

$$\mathrm{Enc}(\bigcap_i A_i) \leftarrow f(\mathrm{Enc}(A_1), \dots, \mathrm{Enc}(A_N)).$$

Finally, the cloud server C returns the encrypted intersection result $\mathrm{Enc}(\bigcap_i A_i)$ to the querying user B, who decrypts it locally to obtain the intersection.

However, this framework presents several major challenges:

1. **Set Size Scalability.** Current practical Fully Homomorphic Encryption (FHE) schemes typically support nearly unlimited-depth homomorphic addition, but still suffer from significant limitations on multiplication depth. This constraint directly limits the size of polynomials that can be evaluated, and indirectly restricts the maximum size of the input sets that the PSI protocol can support. A core question arises: *How can we design an intersection function f that enables large-scale set intersection under the current multiplicative depth limitations of FHE?*

2. **Computing Party Scalability.** Most existing FHE-based PSI constructions are designed for two-party settings and cannot be naturally extended to multi-party computation. Thus, another critical question is: *How can we construct an intersection function f that preserves privacy while naturally scaling to multiple computing parties?*

3. **Efficiency of Set Intersection.** Even if an intersection function f satisfying the above two conditions can be constructed, its homomorphic evaluation efficiency remains a major concern. Given that addition is significantly cheaper than multiplication in FHE systems, an important design goal is: *How can we maximize the use of addition operations while minimizing the number of multiplications in order to achieve a more efficient intersection computation?*

After extensive analysis, we propose a concrete construction of the intersection function f and prove its existence. Our approach demonstrates that f can be constructed using only addition operations.

Our Contributions We summarize the contributions of this paper as follows:

- We propose a completely new construction for delegated PSI. Unlike previous works that rely on a primitive called *randomized intersection encoding*, our approach avoids this dependency and instead bases delegated PSI on Bloom filters, which are eventually transformed into simple addition operations.
- Our construction achieves sub-optimal $\Omega(Nk \log k)$ computational complexity and optimal $\Omega(Nk)$ communication complexity, where N denotes each party's set size and k is the number of parties.
- We introduce several optimization techniques to enhance concrete efficiency, such as lightweight FHE parameter instantiation, compact secure-AND protocol parameter design, and SIMD-style FHE batching. These optimizations may be of independent interest to the community.

2 Related works

The standard Private Set Intersection (PSI) model assumes that all computing parties hold their sets locally and perform computations independently, without relying on a cloud server. However, in practice, cloud computing is widely adopted and offers significantly greater computational and communication capabilities than individual users. An ideal solution should allow users to securely delegate their sets to a cloud server, which performs most of the computation. Since the cloud may be untrusted, user data must remain private to the cloud, and the computation process must not leak any information about the sets. This model is referred to as Delegated Private Set Intersection (Delegated-PSI).

We now review work related to delegated multi-party private set intersection. Delegated-PSI protocols leverage cloud computing for both computation and storage, and can be categorized into two types: protocols supporting one-off delegation and those enabling repeated delegation.

One-off delegation protocols (e.g., [Ker12, KMRS14]) require clients to locally re-encode their data for each intersection computation and do not allow reuse of previously outsourced encrypted data. However, these protocols are designed exclusively for two-party scenarios.

In contrast, repeated delegation PSI protocols allow clients to outsource their encrypted data to the cloud once, and subsequently perform multiple PSI computations under the data owners authorization, without needing to keep a local copy or re-encode data for each execution. Representative works in this category include [ATD15, ATD17, ATMD17]. Among them, [ATD15, ATD17] use public key encryption to represent the entire dataset as a "blinded" polynomial and outsource it to the cloud. In particular, [ATMD17] is the first to propose a more efficient Delegated-PSI protocol based on secret sharing, and it further improves PSI computation performance through a hash table structure.

It is worth noting that all these secure protocols mentioned above are designed for two-party settings with static datasets. In contrast, [ATD20] introduces a novel technique called Randomized Intersection Encoding (RIE) to enable intersection computation under secret sharing. This approach overcomes the limitations of two-party computation and static data, enabling support for multi-party scenarios and dynamic dataset updates. However, [ATD20] faces a scalability issue in practical deployment: user B must broadcast their shares to all delegated users A, effectively delegating to multiple cloud servers. This is not scalable when the number of delegated users grows, as user-to-user communication is typically less stable than connections to a centralized cloud server. To address this, [HLZ25] proposes an improved solution based on Beaver multiplication triples, mitigating the issue without increasing computational or communication complexity.

It is clearly seen that the only existing delegated PSI protocols that support the multi-party setting are based on secret sharing. This raises a natural question: Is it possible to design a secure and efficient multi-party delegated PSI protocol using other cryptographic primitives? In this paper, we address this question by leveraging threshold fully homomorphic encryption (FHE), secure-AND subprotocols, and other supporting techniques.

3 Preliminaries

Throughout this paper, we use the following notations:

- (N, k) The symbols N and k parameterize the main protocol, *i.e.*, private set intersection (PSI). Here, N denotes the number of participating parties computing the intersection, and each party holds a set containing k elements.

- (t, ℓ) The parameters t and ℓ are used in the privacy-preserving AND subprotocol $\mathbf{\Pi}_{AND}$. In this subprotocol, each bit is encoded as a vector over $\mathbb{F}_t^\ell$. Initially, we use binary vectors, and thus set $t = 2$.

- (h, ϵ) The parameters h and ϵ are used in the Bloom filter construction. Here, h denotes the number of independent hash functions used, and ϵ denotes the false positive rate of the Bloom filter.

- (n, q, t) We use the symbols n, q, and t to parameterize the fully homomorphic encryption (FHE) scheme. Specifically, n denotes the lattice dimension (which also corresponds to the degree of the polynomial in the polynomial ring $\mathcal{R}$), q denotes the ciphertext modulus, and t denotes the plaintext modulus.

3.1 Privacy-preserving AND operation

Given N bits $\{x_i\}_{i=1,\cdots,N}$,each of which is held by a party privately, we consider how N parties perform a logical AND operation on their bits without compromising privacy, denoted as protocol $\mathbf{\Pi}_{AND}$:

1. Each party A_i encrypts their bit $x_i \in \{0, 1\}$ using a public-key encryption scheme, denoted as $\mathrm{Enc}_{pk}(\mathbf{x}_i)$:

(a) If $x_i = 1$, then $\mathbf{x}_i$ is set as a zero vector of ℓ bits:

$$\mathbf{x}_i \leftarrow \mathbf{0} \text{ w.r.t. } \mathbf{0} = (0, \dots, 0).$$

(b) Else $\mathbf{x}_i$ is set as a random ℓ-bit vector:

$$\mathbf{x}_i \overset{\$}{\leftarrow} \{0,1\}^{\ell}.$$

2. Each A_i sends their ciphertext $\text{Enc}_{pk}(\mathbf{x}_i)$ to the cloud server C.

3. The cloud server C computes:

$$\sum_i \text{Enc}_{pk}(\mathbf{x}_i) = \text{Enc}_{pk}\left(\sum_i \mathbf{x}_i \mod 2\right).$$

4. The cloud server C sends $\text{Enc}_{pk}\left(\sum_i \mathbf{x}_i\right)$ to any arbitrary party A_i.

5. A_i decrypts locally to obtain:

$$\mathbf{r} = \sum_i \mathbf{x}_i.$$

6. If the vector $\mathbf{r}$ is a zero vector, then $\bigwedge_{i=1}^{N} x_i = 1$, otherwise $\bigwedge_i x_i = 0$.

If the encryption scheme $Enc_{pk}(\cdot)$ is semantically secure, then it is straightforward to prove the protocol is semi-honest secure. Thereby we focus on proving the correctness of the protocol.

Proof. If $\bigwedge x_i = 1$, then all $\mathbf{x}_i$ are zero vectors, implying:

$$\mathbf{r} = \sum_i \mathbf{x}_i = \sum_i \mathbf{0} = \mathbf{0}.$$

If $\bigwedge x_i = 0$, then at least one $\mathbf{x}_i$ is a random vector. Consequently, $\sum_i \mathbf{x}_i$ is also a random vector, and the probability that the vector $\mathbf{r}$ results in an all-zero vector is only $\frac{1}{2^\ell}$, which is negligible. Thus, the proof is complete. □

Remark Protocol $\mathbf{\Pi}_{AND}$ demonstrates that a homomorphic AND operation can be performed as long as the encryption scheme $\text{Enc}(\cdot)$ supports homomorphic addition. This technique originates from the well-known SYY approach [SYY99].

3.2 Set Intersection Operation

A given set A can be represented using a Bloom Filter (BF). The algorithm constructing the bloom filter HT_A from a set A, denoted as $\mathcal{A}_{BF}$, is as follows:

1. Initialize a hash table (one-dimensional array) HT_A, setting all storage positions to zero.

2. Prepare h independent hash functions, denoted as H_j for $j = 1, \dots, h$.

3. For each $a \in A$:

(a) Compute h hash values:

$$i_a^{(j)} \leftarrow H_j(a), \quad \forall j.$$

(b) Set the corresponding bits in the Bloom Filter:

$$HT_A[i_a^{(j)}] \leftarrow 1, \quad \forall j.$$

4. Return HT_A.

To determine whether an element x belongs to A, we use the Bloom Filter membership test:

1. Compute h hash values for x:

$$i_x^{(j)} \leftarrow H_j(x), \quad \forall j.$$

2. If $HT_A[i_x^{(j)}] = 1$ for all j, return $x \in A$; Otherwise, return $x \notin A$.

Remark Since Bloom Filters allow false positives (i.e., mistakenly identifying a non-member as a member), we define the false positive rate ϵ as:

$$\epsilon = \Pr(a \notin A \text{ and } HT_A[H_j(a)] = 1 \text{ for all } j).$$

In practice, system parameters are often chosen as $h = 40$, $|HT_A| = 57.67 \cdot |A|$, and $\epsilon = 2^{-40}$.

3.2.1 Bloom Filter-Based Set Intersection

Given N sets $\{A_i\}_i$ and their corresponding Bloom Filters HT_{A_i}, the intersection Bloom Filter $HT_{\cap A_i}$ is computed as:

$$HT_{\cap A_i} \stackrel{def}{=} \bigwedge_i HT_{A_i},$$

where the bitwise AND operation is applied across all Bloom Filters. We argue the correctness of the defined set intersection operation with the following proof.

Proof. If $x \in \cap A_i$, then HT_{A_i} has '1' at $H_j(x)$ for all j. Since $HT_{\cap A_i}$ is constructed using a bitwise AND operation across all HT_{A_i}, it also has '1' at these positions. Thus, x will always pass the membership test of $HT_{\cap A_i}$, proving the correctness of the operation when x is indeed in the intersection. Therefore, the remaining task is to argue that when $x \notin \cap A_i$, it cannot pass the membership test of $HT_{\cap A_i}$ except with negligible probability.

If $x \notin \cap A_i$ but still passes the membership test of $HT_{\cap A_i}$, it means that for a randomly chosen x, all h positions in $HT_X \cap HT_Y$ must be set to '1'.

The probability that a specific position in HT_{A_i} is set to 1 is:

$$p_{A_i} = 1 - \left(1 - \frac{1}{|HT|}\right)^{h|A_i|} \approx 1 - e^{-\frac{h|A_i|}{|HT|}}.$$

where it requires that $|HT_{A_i}| = |HT_A|, \forall i$. Thus, the probability that a specific position in $HT_{\cap A_i}$ is set to 1 is:

$$p_{\cap A_i} = \prod_i p_{A_i} \approx \prod_i \left(1 - e^{-\frac{h|A_i|}{|HT|}}\right).$$

The false positive rate for $HT_{\cap A_i}$ is then:

$$\epsilon_{\cap A_i} = (p_{\cap A_i})^h = p_{A_1}^h \cdots p_{A_N}^h = \epsilon_{A_1} \cdots \epsilon_{A_N},$$

where $\epsilon_{A_i} = p_{A_i}^h$ represents the false positive rate of HT_{A_i}. Clearly, $\epsilon_{\cap A_i} < \epsilon_{A_i}$, confirming that using $HT_{\cap A_i}$ for set intersection is feasible.

□

3.3 Two-Party Fully Homomorphic Encryption

We use one of the most popular Fully Homomorphic Encryption (FHE) schemes called BFV, suitable for two-party computation. In this setting, there exist two computing parties: P_1, who exclusively holds the secret key sk, and both parties share knowledge of the public key pk.

- P_1 uses the common public key to encrypt its data x, producing $Enc_{pk}(x)$.
- P_1 sends the encrypted data $Enc_{pk}(x)$ to P_2 for homomorphic evaluation of a function f.
- P_2 computes the homomorphic evaluation $f(x)$ and returns $Enc_{pk}(f(x))$ to P_1.
- Finally, P_1 decrypts $Enc_{pk}(f(x))$ using the secret key sk to obtain the result $f(x)$.

Key Generation Algorithm $\mathcal{A}_{\text{FHE.KeyGen}}(1^\lambda)$:

1. Sample $s \leftarrow \mathcal{DG}(\sigma^2)$, $a \leftarrow U(R_q)$, and $e \leftarrow \mathcal{DG}(\sigma^2)$.
2. Output public key $pk = (a, -(as+e))$ and secret key $sk = s$.

Here, pk corresponds to an RLWE ciphertext encrypting zero.

Encryption Algorithm $\mathcal{A}_{\text{Encrypt}}(pk, m \in R_t)$:

1. Sample $u, e_0, e_1 \leftarrow \mathcal{DG}(\sigma^2)$.
2. Return ciphertext:

$$ct = (pk[0] \cdot u + e_0, \quad pk[1] \cdot u + e_1 + \Delta \cdot m).$$

Let $\Delta = \lfloor q/t \rfloor$. Here, $(pk[0], ct[0])$ is one RLWE sample, and $(pk[1], ct[1])$ is another RLWE sample.

Decryption Algorithm $\texttt{Decrypt}(sk = s, ct)$:

1. Compute $m' \leftarrow ct[0] \cdot s + ct[1]$.
2. Return $m \leftarrow \left\lfloor \frac{t \cdot m'}{q} \right\rceil$.

Correctness We have:

$$s \cdot ct[0] + ct[1] = -eu + e_0 s + e_1 + \Delta \cdot m.$$

Define:

$$e' = -eu + e_0 s + e_1, \quad \varepsilon = \frac{q}{t} - \Delta < 1.$$

It is easy to see that when:

$$||e'||_\infty < \frac{q}{2t} - t,$$

then:

$$\left\lfloor \frac{t \cdot m'}{q} \right\rceil = \left\lfloor m + \frac{t}{q}(e' - m) \right\rceil = m.$$

Homomorphic Addition Algorithm BFV FHE can support both homomorphic additions and multiplications. We stay focused on homomorhpic addition for our purpose.

$\mathcal{A}_{\text{BFV.Add}}(ct_0, ct_1)$:

1. Perform vector addition:
$$ct_{\text{add}} \leftarrow (ct_0[0] + ct_1[0], \quad ct_0[1] + ct_1[1]).$$
2. Return ct_{add}.

Correctness We have:

$$s \cdot ct_{\text{add}}[0] + ct_{\text{add}}[1] = -eu' + e_0's + e_1' + \Delta \cdot (m_0 + m_1).$$

Here, the noise terms:

$$\mu', e_0', e_1' \sim DG(2\sigma^2),$$

thus the upper bound of the total noise is:

$$|| - eu' + e_0's + e_1'||_\infty \leq 16\sigma\rho N + 16\sigma h + 6\sigma.$$

Since this bound is much smaller than $\Delta/2$, decryption remains correct.

3.4 Multi-party Fully Homomorphic Encryption

(Joint) Public Key Generation Protocol The key generation protocol $\Pi_{\text{BFV.PKGen}}$ is as follows:

1. P_1 and P_2 agree on a common random polynomial $a \in R_q$ (common random string, CRS).
2. P_1 computes $b_1 = -as_1 + x$.
3. P_2 computes $b_2 = -as_2 + y$.
4. P_1 and P_2 exchange b_1 and b_2 and compute $b = b_1 + b_2$.
5. The final public key is returned as $pk = (a, b)$.

Based on the RLWE assumption, we have:

$$b_1 \approx_c U, \quad b_2 \approx_c U,$$

which ensures that the above algorithm securely computes pk without leaking the private keys s_1 and s_2.

Public Key Encryption Algorithm The secure public key encryption algorithm $\mathcal{A}_{\text{BFV.PKE}}$ is formally defined as follows:

1. Run the public key generation algorithm $PKGen(s_1, s_2)$ to obtain the (joint) public key:
$$pk = (a, b = -a(s_1 + s_2) + e).$$
2. Generate "small" noise values $u, e_0, e_1 \sim \chi$ satisfying $\|e_1 + e_u + e_2\|_1 < B_1$.
3. Generate "large" noise $e^* \overset{\$}{\leftarrow} [-B_2, B_2]$ such that $\frac{B_1}{B_2} = \epsilon(\lambda)$.
4. Compute $ct_0 \leftarrow au + e_0$.
5. Compute $ct_1 \leftarrow bu + e_1 + e^*$.
6. Return the ciphertext $ct = (ct_0, ct_1)$.

Security analysis First, we show that the encryption algorithm eliminates the information leakage issue in decryption noise. Observe that:

$$\mathrm{Enc}_{pk}(m) = (au + e_0, bu + e_1 + e^* + \Delta m).$$

Substituting b:

$$= (au + e_0, -as_1u - as_2u + (x + y)u + e_1 + e^* + \Delta m).$$

Since:

$$b = -as^* + e,$$

decryption using the global secret key s^* yields:

$$s^*(au + e_0) - aus^* + e^* + \Delta m = e_0s^* + e^* + \Delta m.$$

Since e^* follows a uniform distribution and is independent of s^*, information leakage is entirely eliminated.

To prove the encryption algorithm is semantically secure, we must show that:

$$\mathrm{Enc}_{pk}(m) \approx_c (U, U).$$

Proof. Using contradiction, assume an adversary A can distinguish between $\mathrm{Enc}_{pk}(m)$ and uniform randomness (U, U). We construct an adversary A' that can break the RLWE hardness assumption.

Consider an adversary A' in a Threshold FHE setting, where party P_2 receives $pk_1 = (a, b_1 = -as_1 + e)$ from P_1 where $e = x + y$. The adversary A' proceeds as follows:

1. P_2 generates its private key s_2.
2. P_2 encrypts m using pk_1:
$$\mathrm{Enc}_{pk_1}(0) = (ct_0 = au + e_0, ct_1 = b_1u + e_1).$$
3. P_2 sets $ct_0^* = ct_0$.
4. P_2 generates a "large" noise $e^* \xleftarrow{\$} [-B_2, B_2]$.
5. P_2 computes $ct_1^* = ct_1 - s_2 \cdot ct_0 + e^*$.
6. P_2 feeds (ct_0^*, ct_1^*) and random samples $(u_1, u_2) \sim U$ into adversary A.

Observe that $\mathrm{Enc}_{pk}(0) = (au + e_0, bu + e_1 + e^*)$. Substituting b:

$$\mathrm{Enc}_{pk}(0) = (au + e_0, -as_1u - as_2u + (x + y)u + e_1 + e^*).$$

Since:

$$ct_1^* = ct_1 - s_2ct_0 + e^* = (-as_1 + e_1)u - s_2(au + e_0) + e^*,$$

we obtain:

$$ct_1^* = -as_1u - as_2u + e_1u - e_0s_2 + e^* \approx_c -as_1u - as_2u + e^*.$$

Thus:

$$(ct_0^*, ct_1^*) \approx_c \mathrm{Enc}_{pk}(0).$$

Since (ct_0^*, ct_1^*) is constructed from an RLWE ciphertext (ct_0, ct_1), if A can distinguish (ct_0^*, ct_1^*), then A' can distinguish (ct_0, ct_1), contradicting the RLWE assumption. This completes the proof. □

(Joint) Decryption Protocol The formally defined secure public key decryption algorithm is as follows:

1. Each party P_i receives the ciphertext:

$$ct = (au + e_0, bu + e_1 + \Delta m),$$

and computes:

$$w_i \leftarrow s_i(au + e_0).$$

2. Each P_i generates a large noise $e^* \in [-B, B]$ satisfying $\frac{|e_0 s_i|_1}{B} = \epsilon(\lambda)$. and computes:

$$w_i \leftarrow w_i + e^*.$$

3. Assuming P_1 is the final plaintext recipient, it computes:

$$w_1 + w_2 + (bu + e_1 + \Delta m) \approx \Delta m.$$

Proof. We prove that w_i is pseudo-random (computationally indistinguishable from a uniform distribution). Suppose, for contradiction, that there exists an efficient algorithm A that can distinguish w_i from a truly random value. Then, we can construct an RLWE distinguisher to break the RLWE assumption:

1. P_i uses its public key: $pk_i = (a, b_i) = (a, -as_i + e)$, to encrypt zero, obtaining an RLWE sample:

$$\mathrm{Enc}_{pk_i}(0) = (au + e_0, (-as_i + e)u + e_1).$$

2. P_i applies masking with a large noise e^* to the ciphertext

$$c \leftarrow -((-as_i + e)u + e_1 + e^*).$$

3. P_i randomly generates $c' \xleftarrow{\$} R_q$, and applies another large noise $e^{**} \in [-B, B]$ as masking:

$$c' \leftarrow c' + e^{**}.$$

4. P_i provides both c and c' to A to distinguish.

Since $c \approx_s w_i$ and c' follows a uniform distribution, if A can distinguish between c and c', it implies that RLWE samples can be distinguished from random distributions, contradicting the RLWE assumption. □

Remark The above proof of w_i being pseudo-random also implies that joint decryption protocol is semi-honest secure. Detailed proof of this claim is omitted for brevity.

Homomorphic Addition Algorithm The homomorphic addition in MP-FHE is identical to its counterpart in TP-FHE, and we skip the descriptions here.

4 Privacy-Preserving Set Intersection, A Framework

At this point, we present the overview framework for the privacy-preserving set intersection (PSI) protocol $\mathbf{\Pi}_{PSI}$:

1. Each A_i (for all i) computes its private Bloom filter where each bit is encoded to either a zero vector or a random vector as mentioned in Π_{AND}, denoted as BF_{A_i}.

2. Each A_i encrypts its private Bloom filter, denoted as $\mathrm{Enc}_{pk}(BF_{A_i})$, and sends it to the cloud server C.

3. The cloud C applies the privacy-preserving AND operation ($\mathbf{\Pi}_{AND}$) to compute the encrypted Bloom filter of the intersection:
$$\mathrm{Enc}_{pk}(BF_{\cap A_i}) = \mathrm{Enc}_{pk}(\sum_i BF_{A_i}).$$

4. The party B performs the followings to finally recover $\cap A_i$

   (a) For each $b \in B$:

      i. $Enc(\mathbf{x}_b) \leftarrow \sum_{j=1}^{h} Enc(BF_{\cap A_i}[H_j[b]])$.
      ii. The parties $\{A_i\}$ and B perform decryption collaboratively, allowing B to learn $\mathbf{x}_b = \sum_j BF_{\cap A_i}[H_j[b]]$
      iii. If $\mathbf{x}_b = \mathbf{0}$, then b is in the intersection: $\cap A_i \leftarrow b$.

5. The final intersection $\cap A_i$ is returned.

Remarks on security: Unfortunately, the proposed framework protocol Π_{PSI} is insecure. The vulnerability arises in Step 5, where the exposure of $\mathbf{x}_b$ related to the Bloom filter $BF_{\cap A_i}$ corresponding to the intersection set $\cap A_i$ may leak certain differential information. For instance, suppose the first item $b_1 \in B$ maps to the Bloom filter entries $BF_{\cap A_i}[1], BF_{\cap A_i}[2], BF_{\cap A_i}[3]$, while the second item $b_2 \in B$ maps to $BF_{\cap A_i}[1], BF_{\cap A_i}[2], BF_{\cap A_i}[4]$. Then, the difference $\mathbf{x}_{b_1} - \mathbf{x}_{b_2}$ could reveal information about $BF_{\cap A_i}[3] - BF_{\cap A_i}[4]$. To address this potential information leakage, we propose a privacy-preserving zero test mechanism, which eliminates the leakage and ensures the security of our protocol.

Remarks on performance: Assuming each computing party's set has size k, Step 1 requires each bit of BF_{A_i} to be encoded as a vector according to the $\mathbf{\Pi}_{AND}$ protocol, incurring linear encoding and computational overhead of $\Omega(k)$. Step 2 requires each BF_{A_i} to be encrypted as $\mathrm{Enc}_{pk}(BF_{A_i})$, which costs each A_i an optimal computational overhead of $\Omega(k)$ and an optimal communication overhead of $\Omega(k)$. Step 3 requires the cloud to perform homomorphic additions to realize the $\mathbf{\Pi}_{AND}$ subprotocol, with a total computational complexity of $\Omega(Nk)$. Step 4 involves threshold decryption of FHE ciphertexts and thus incurs a computational cost of $\Omega(Nk)$.

To conclude, the performance bottlenecks of the protocol are as follows:

- **Computational bottleneck:** Arises in Step 3 during the privacy-preserving AND operation, with computational complexity $\Omega(Nk)$, matching the optimal theoretical lower bound.

- **Communication bottleneck:** Occurs in Step 2, with communication complexity $\Omega(Nk)$, matching the theoretical lower bound.

Although both computation and communication complexities have reached their theoretical lower bounds, this does not necessarily imply optimal practical performance, as asymptotic analysis does not consider privacy-preserving AND operation parameters ℓ or specific homomorphic encryption system parameters (n, q). Below are some techniques to enhance practical performance:

- **Utilizing SIMD-like encoding in fully homomorphic encryption:** This increases the parallelism of the AND operation and reduces communication overhead, optimizing performance.
- **Simplifying homomorphic encryption system parameters:** The new privacy-preserving set intersection protocol only requires additive homomorphism, allowing for smaller parameter settings in fully homomorphic encryption, thereby improving efficiency.

4.1 Privacy-preserving Zero Test

We now detail how to resolve the partial information leakage issue in the framework protocol. Starting from Step 5, where an FHE encryption of a membership test indicator $\mathbf{x}_b$ with respect to $b \in \cap_i A_i$ has already been computed on the cloud side (such that $\mathbf{x}_b = \mathbf{0}$ if $b \in \cap_i A_i$, and otherwise $\mathbf{x}_b \neq \mathbf{0}$), we proceed with the following steps:

1. The cloud generates a random vector of the same size as $\mathbf{x}_b$, denoted by
$$\mathbf{x}_b^{(C)} \overset{\$}{\leftarrow} \mathbb{Z}_t^\ell.$$
2. The cloud homomorphically computes
$$Enc_{pk}(\mathbf{x}_b - \mathbf{x}_b^{(C)}) \leftarrow Enc_{pk}(\mathbf{x}_b) - Enc_{pk}(\mathbf{x}_b^{(C)}).$$
3. All computing parties jointly decrypt $Enc_{pk}(\mathbf{x}_b - \mathbf{x}_b^{(C)})$ and provide the enquiring user B with its secret share
$$\mathbf{x}_b^{(B)} \overset{\text{def}}{=} \mathbf{x}_b - \mathbf{x}_b^{(C)}.$$
4. The cloud (holding $\mathbf{x}_b^{(C)}$) and the user B (holding $\mathbf{x}_b^{(B)}$) engage in a zero test protocol to compute shares $x_b^{(C)}$ and $x_b^{(B)}$ such that
$$x_b = x_b^{(C)} + x_b^{(B)},$$
where $x_b = 1$ if $\mathbf{x}_b = \mathbf{0}$ and $x_b = 0$ otherwise.
5. The cloud securely sends its share to the enquiring user B, who reconstructs x_b and thus learns whether $b \in \cap_i A_i$.

Remarks on Correctness: Steps 1–3 generate additive secret shares of $\mathbf{x}_b$. Step 4 computes the zero-test function

$$f(\mathbf{x}_b) = \begin{cases} 1, & \text{if } \mathbf{x}_b = \mathbf{0}, \\ 0, & \text{otherwise.} \end{cases}$$

The final value $x_b \overset{\text{def}}{=} f(\mathbf{x}_b)$ is binary: $x_b = 1$ indicates that b belongs to the intersection $\cap_i A_i$, and $x_b = 0$ indicates otherwise.

Remarks on Security: The proposed protocol purely relies on additive secret sharing and homomorphic encryption. Its security naturally follows from the underlying security guarantees of secret sharing and homomorphic encryption schemes.

4.2 Constructing Zero Test from Additive Homomorphic Encryption

The performance bottleneck lies step 4 of the zero test, i.e. how to efficiently test whether the secret-shared vector $\mathbf{x}_b = (\mathbf{x}_b[1], \ldots, \mathbf{x}_b[\ell]) \in \mathbb{Z}_t^\ell$ is zero. In this section, we propose a new construction only requiring the underlying homomorphic encryption scheme supports homomorphic additions. The construction proceeds as follows:

1. The cloud and the user B agree on a public prime p such that $p > 2^{40}$ and $p > 2(t^\ell - 1)$, and set the plaintext space of the homomorphic encryption scheme to $\mathbb{Z}_p$.
2. The cloud holds its share of the membership indicator vector $\mathbf{x}_b \in \mathbb{Z}_t^\ell$, denoted by $\mathbf{x}_b^{(C)} = (\mathbf{x}_b^{(C)}[1], \ldots, \mathbf{x}_b^{(C)}[\ell])$, and computes:
$$u^{(C)} \leftarrow \sum_{i=1}^{\ell} \mathbf{x}_b^{(C)}[i] \cdot t^{i-1} \in \mathbb{Z}_{t^\ell}.$$
3. Similarly, user B holds his share $\mathbf{x}_b^{(B)} = (\mathbf{x}_b^{(B)}[1], \ldots, \mathbf{x}_b^{(B)}[\ell])$ and computes:
$$u^{(B)} \leftarrow \sum_{i=1}^{\ell} \mathbf{x}_b^{(B)}[i] \cdot t^{i-1} \in \mathbb{Z}_{t^\ell}.$$
4. User B encrypts $u^{(B)}$ using his secret key s_B to get $Enc(u^{(B)})$, and sends the ciphertext to the cloud.
5. The cloud computes:
$$Enc(u') \leftarrow Enc(u^{(B)}) + (u^{(C)} - t^\ell).$$
6. The cloud samples a random $r \xleftarrow{\$} \mathbb{Z}_p^*$ and performs a homomorphic scalar multiplication:
$$Enc(r \cdot u') \leftarrow r \cdot Enc(u'),$$
and sends $Enc(r \cdot u')$ back to user B.
7. User B decrypts $Enc(r \cdot u')$ to recover $r \cdot u'$. If $r \cdot u' = 0$, set $x_b = 1$; otherwise, set $x_b = 0$.

Correctness of the Zero Test. Steps 2–3 convert the vector $\mathbf{x}_b$ into an integer $u = g(\mathbf{x}_b)$ using a one-to-one mapping $g : \mathbb{Z}_t^\ell \to \mathbb{Z}_{t^\ell}$ defined by
$$g(\mathbf{x}_b) = \sum_{i=1}^{\ell} \mathbf{x}_b[i] \cdot t^{i-1}.$$

This process yields two additive shares $u^{(B)}$ and $u^{(C)}$ such that $u = u^{(B)} + u^{(C)} \mod t^\ell$, and each share appears as a uniformly random element in $\mathbb{Z}_{t^\ell}$

After Step 5, we have:
$$u' \overset{def}{=} u^{(B)} + u^{(C)} - t^\ell = u - t^\ell \mod p,$$

where $u = u^{(B)} + u^{(C)}$. If $\mathbf{x}_b = \mathbf{0}$, then $u \in \{0, t^\ell\}$, and the probability that $u = 0$ is only $\frac{1}{t^\ell}$. By choosing a sufficiently large t^ℓ, e.g., $t^\ell \geq 2^{40}$, we can ensure that $u = t^\ell$ with overwhelming probability, implying $u' = 0$.

Conversely, if $\mathbf{x}_b \neq \mathbf{0}$, then $u \in \{1, 2, \ldots, t^\ell - 1\} \cup \{t^\ell + 1, \ldots, 2t^\ell - 2\}$, and hence $u' \neq 0$. Thus, in Step 7, the decrypted value $r \cdot u'$ is zero if and only if $\mathbf{x}_b = \mathbf{0}$. Otherwise, it is uniformly random in $\mathbb{Z}_p^*$, and hence non-zero with probability 1. This allows user B to determine x_b.

Security of the Zero Test. All communication is encrypted using homomorphic encryption, ensuring that privacy is preserved. Moreover, the final value $r \cdot u'$ reveals no information about $\mathbf{x}_b$ beyond whether it is the all-zero vector.

Performance. Although the homomorphic scalar multiplication in Step 6 can be interpreted as performing r homomorphic additions, the value of r is typically large, rendering this approach impractical. Instead, we assume that the underlying homomorphic encryption scheme supports scalar multiplication and that such operations are efficient—an assumption that holds for lattice-based homomorphic encryption schemes. Since the proposed zero-test requires only one homomorphic scalar multiplication over $\mathbb{Z}_p$, the overall protocol remains highly efficient in practice.

Furthermore, the computation cost in Step 6 can be amortized by leveraging the standard SIMD encoding technique. Specifically, we can batch the scalar multiplications by computing

$$\mathsf{Enc}(\mathbf{r} \odot \mathbf{u}') \leftarrow \mathbf{r} \cdot \mathsf{Enc}(\mathbf{u}'),$$

where $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_p^N$, and each element $u' \in \mathbf{u}'$ is computed as

$$u' = u - t^\ell = \sum_{i=1}^{\ell} \mathbf{x}_b[i] t^{i-1} - t^\ell \bmod p,$$

corresponding to a specific input bit vector $\mathbf{x}_b \in B$.

4.3 Applicability to Standard Private Set Intersection

The delegated private set intersection (Delegated PSI) scheme framework discussed above assumes the presence of an untrusted third party, namely the cloud server. In fact, the construction proposed in this paper can also be applied to the standard PSI setting: by simply executing the cloud-side computation locally by any one of the computing parties, the cloud server assumption can be removed, resulting in a third-party-free PSI protocol. Moreover, the scheme achieves the theoretical lower bounds for both computational and communication overhead in the asymptotic sense.

Previous approaches [CLR17, CMdG$^+$21] to constructing standard PSI protocols using homomorphic encryption typically rely on Oblivious Polynomial Evaluation (OPE). When additive homomorphic encryption (e.g., Paillier) is used, OPE cannot be performed efficiently, resulting in extremely poor performance for large sets (e.g., on the order of millions). While fully homomorphic encryption (FHE), such as BFV, does support efficient OPE, current FHE schemes only support limited multiplicative depth (typically 6—9), which still limits the supported set size to well below the million scale. Furthermore, such approaches only support the asymmetric set case (i.e., in Delegated PSI, the query user's set must be much smaller than the delegated users'), and cannot be extended effectively to symmetric set scenarios.

In contrast, our construction based on Bloom filters completely avoids the dependence on OPE and homomorphic multiplication, thereby significantly enhancing scalability to support set sizes of tens or even hundreds of millions. In addition, since FHE schemes perform addition much more efficiently than multiplication, our construction also demonstrates strong practical performance advantages.

The current most efficient standard PSI protocol [RS21] in practice is based on VOLE (Vector Oblivious Linear Evaluation), which also achieves the theoretical lower bounds in both computation and communication. However, its advantage is primarily observed in the two-party model and does not extend efficiently to settings with three or more parties. In comparison, our proposed construction naturally supports arbitrary numbers of computing parties, offering clear scalability advantages and making it applicable to a broader range of real-world scenarios.

5 Smaller FHE System Parameters

When implementing the proposed PSI protocol using Fully Homomorphic Encryption (FHE), one of the critical challenges is setting the FHE system parameters to align with the protocol's security and performance requirements. Specifically, it is necessary to ensure that the FHE scheme supports the required multiplication (or addition) depth while maintaining the desired level of security.

In practice, a security strength of 128 bits is widely considered sufficient for most applications. Therefore, this paper fixes the target security strength at 128 bits and further explores the relationship between multiplication/addition depth and system parameter size to optimize the proposed PSI protocol.

FHE System Parameters FHE ciphertexts are defined over the polynomial ring:

$$R_q \times R_q = \mathbb{Z}_q[x]/(x^n + 1) \times \mathbb{Z}_q[x]/(x^n + 1),$$

where plaintexts are defined in the polynomial ring:

$$R_t = \mathbb{Z}_t[x]/(x^n + 1).$$

The standard deviation of system noise is denoted as σ. Thus, the FHE system parameters can be defined as:

$$\text{params} \overset{def}{=} (n, q, t, \sigma \approx 3.2).$$

Setting the system parameters requires balancing security strength and computational depth, which are inherently conflicting goals:

- **Enhancing FHE Security:** This requires choosing a sufficiently large n (i.e., higher-dimensional lattices) and minimizing the signal-to-noise ratio q/σ (where q indirectly determines the plaintext space size, and σ represents ciphertext noise magnitude).

- **Increasing Addition/Multiplication Depth:** This necessitates keeping n as small as possible (since a large n leads to increased decryption and multiplicative noise) while maximizing the signal-to-noise ratio q/σ to provide adequate noise tolerance for deeper computations.

Thus, parameter selection is fundamentally an optimization trade-off: ensuring sufficient security while maximizing computational depth to meet the targeted application demands.

5.1 Two-party FHE

We begin by examining the simplest case, where only two parties exist alongside an untrusted cloud. Specifically, when ciphertexts are decrypted, added, or multiplied, the noise within the ciphertext increases. Therefore, it is crucial to analyze the noise growth bounds to estimate the number of additions and multiplications that can be performed while maintaining correct decryption. To achieve this, we study the noise behavior in three key operations: decryption noise, addition noise, and multiplication noise.

Decryption Noise Typically, Fully Homomorphic Encryption (FHE) operates in an asymmetric encryption setting, where encryption is performed using a public key, and decryption requires a private key. However, the process of decrypting a ciphertext using the private key inherently introduces additional noise. In a sense, encrypting a plaintext m with the public key can be viewed as encrypting the same plaintext with the private key but with increased noise, expressed as:

$$\mathrm{Enc}_{pk}(\Delta \cdot m) = \mathrm{Enc}_{sk}(\Delta m + e_{\mathrm{init}})$$

where $e_{\mathrm{init}} \overset{def}{=} -eu + e_0 s + e_1$
and the noise components follow a discrete Gaussian distribution:

$$e, u, e_0, e_1 \sim \mathcal{DG}_((\sigma^2).$$

To quantify noise levels, we use the canonical norm. We introduce several key results:

1. If $e \sim U(R_t)$, then $||e||_{\mathrm{can}} \leq 6\sqrt{\frac{nt^2}{12}}$.
2. If $e \sim \mathcal{DG}(\sigma^2)$, then $||e||_{\mathrm{can}} \leq 6\sqrt{n\sigma^2}$.
3. If $e_0, e_1 \sim \mathcal{DG}(\sigma^2)$, then $||e_0 e_1||_{\mathrm{can}} \leq 16 \cdot n\sigma^2$.
4. If $e \sim \mathcal{DG}(\sigma^2)$ and $m \sim U(R_t)$, then $||em||_{\mathrm{can}} \leq 16 \cdot n\sigma t/\sqrt{12}$.

Using these bounds, we can derive:

$$||e_{\mathrm{init}}||_{\mathrm{can}} \leq ||eu||_{\mathrm{can}} + ||e_0 s||_{\mathrm{can}} + ||e_1||_{\mathrm{can}} \leq 16 \cdot \sigma^2 \cdot n + 16 \cdot \sigma^2 \cdot n + 6 \cdot \sigma \cdot \sqrt{n}.$$

Addition Noise We analyze the relationship between system parameters and addition depth using the mainstream FHE scheme, BFV. Assume we have two BFV ciphertexts ct_0 and ct_1, which are encryptions of plaintexts m_0 and m_1, respectively:

$$ct_i = \mathrm{Enc}(m_i).$$

The discrete Gaussian noise within the ciphertexts ct_0 and ct_1 is denoted as:

$$e_0 \sim \mathcal{DG}_((\sigma_0^2), \quad e_1 \sim \mathcal{DG}_((\sigma_1^2).$$

Denote the additive depth as k. Since homomorphic addition results in linear noise growth, after a single homomorphic addition, the noise follows:

$$e_{\mathrm{add}} \sim \mathcal{DG}_((\sigma_0^2 + \sigma_1^2).$$

To ensure correct decryption, the noise in the ciphertext must be bounded by an upper limit:

$$6 \cdot ||e_{\mathrm{add}}|| < \frac{q}{2t}. \tag{1}$$

The above equation establishes a quantitative relationship between the noise of a single homomorphic addition and the FHE system parameters. Based on this, we outline the following algorithm to estimate the additive depth:

1. **Initialize** noise levels $\sigma_0, \sigma_1, \sigma_{\text{add}}$, ensuring:

$$\sigma_0 = \sigma_1 = \sigma_{\text{add}} = 16 \cdot \sigma^2 \cdot n + 16 \cdot \sigma^2 \cdot n + 6 \cdot \sigma \cdot \sqrt{n}.$$

2. **Set** $k = 0$.
3. **While** $6 \cdot \sigma_{\text{add}} < \frac{q}{2t}$, repeat the following steps:
   (a) Compute $\sigma_{\text{add}} \leftarrow \sqrt{\sigma_0^2 + \sigma_1^2}$.
   (b) Update $\sigma_0 \leftarrow \sigma_{\text{add}}, \quad \sigma_1 \leftarrow \sigma_{\text{add}}$.
   (c) Increment $k \leftarrow k + 1$.
4. **Return** k.

Proposed Concrete Parameters Currently, the most effective attacks against Fully Homomorphic Encryption (FHE) include primal attacks, dual attacks, and BKW attacks. We propose three sets of concrete FHE parameters, evaluating their respective security levels using the `LWE estimator` and estimating their additive depths using the approach proposed above. The results are presented in the following table.

Table 1: Concrete FHE Parameters with Security Levels and Estimated Depths

$(n, \log q, \log t, \sigma)$	**Security Level (bits)**	**Additive Depth**
$(1024, 28, 1, 3.19)$	≥ 128	10
$(2048, 54, 8, 3.19)$	≥ 128	46
$(4096, 110, 8, 3.19)$	≥ 128	≥ 99

5.2 Multi-Party FHE

For the Multi-Party FHE case, due to the distributed key generation, the internal noise in the public key depends on the number of parties and is therefore greater than its counterpart in the two-party setting. Moreover, to prevent secret key leakage, a relatively large noise is introduced to obscure the original internal decryption noise, ensuring compliance with security requirements. We give detailed analysis on these considerations as follows:

Decryption Noise After public key encryption, the ciphertext $ct \leftarrow Enc_{pk}(m)$ corresponds to the original decryption noise:

$$e_{\text{dec}} = eu + e_0 s^* + e_1$$

Since the noise contains a multiplicative component, a more precise estimation using the canonical norm is required. Given that:

$$u, e_0, e_1 \sim \mathcal{DG}(\sigma^2), \quad e, s^* \sim \mathcal{DG}(N\sigma^2)$$

we derive the upper bound:

$$B_1 = ||e_{\text{dec}}||_{\text{can}} \leq ||e_1||_{\text{can}} + ||eu||_{\text{can}} + ||e_0 s^*||_{\text{can}} \leq 6n\sigma + 16n\sqrt{N}\sigma^2 + 16n\sqrt{N}\sigma^2$$

where the parameter N represents the number of computing parties in the threshold FHE scheme, and the parameter n comes from the polynomial ring $R_q \overset{\text{def}}{=} \mathbb{Z}_q[x]/(x^n+1)$.

Finally, we determine the upper bound for the large noise e^*:

$$B_2 = 2^\lambda B_1 = 2^\lambda(6n\sigma + 16n\sqrt{N}\sigma^2 + 16n\sqrt{N}\sigma^2) \approx 2^\lambda \cdot 32n\sqrt{N}\sigma^2$$

Addition Noise Note that in the above analysis of decryption noise, we have assumed that no homomorphic operations are performed. However, if homomorphic operations are introducedas in our case, where homomorphic addition is appliedthe decryption noise will further increase. Consequently, an even larger noise must be used to obscure the decryption noise and prevent potential secret key leakage.

In more detail, the deeper the addition depth, the larger the smudging noise required to effectively obscure the decryption noise and maintain security. Based on this, we outline the following algorithm to estimate the additive depth:

1. **Initialize** noise levels $\sigma_0, \sigma_1, \sigma_{\text{add}}$, ensuring
$$\sigma_0 = \sigma_1 = \sigma_{\text{add}} = 6n\sigma + 16n\sqrt{N}\sigma^2 + 16n\sqrt{N}\sigma^2.$$
2. **Set** $k = 0$.
3. **While** $6\sigma_{\text{add}} + 2^\lambda \cdot 6\sigma_{\text{add}} < \frac{q}{2t}$, repeat the following steps:
   (a) Compute $\sigma_{\text{add}} \leftarrow \sqrt{\sigma_0^2 + \sigma_1^2}$.
   (b) Update $\sigma_0 \leftarrow \sigma_{\text{add}}, \quad \sigma_1 \leftarrow \sigma_{\text{add}}$.
   (c) Increment $k \leftarrow k + 1$.
4. **Return** k.

Remark In the algorithm above, $6\sigma_{\text{add}}$ serves as a high-probability bound for the original ciphertext noise after performing homomorphic addition. To ensure security, the bound for the superpolynomially scaled smudging noise is conservatively set to $2^\lambda \cdot 6\sigma_{\text{add}}$.

Proposed Concrete Parameters We propose two sets of concrete FHE parameters, evaluating their respective security levels using the `LWE estimator` and estimating their additive depths using the approach proposed above. The results are presented in the following table.

Table 2: Concrete FHE parameters with security levels and estimated depths. The statistical security parameter is set to $\lambda = 40$, and the number of parties is $N = 1,000,000$.

FHE Parameters $(n, \log q, \log t, \sigma)$	**Security Level (bits)**	**Additive Depth**
$(4096, 110, 8, 3.19)$	≥ 128	56
$(4096, 110, 16, 3.19)$	≥ 128	40

6 SIMD-like Working Mode

Modern Fully Homomorphic Encryption (FHE) operates over an algebraic structure known as a polynomial ring. In other words, instead of encrypting individual integers, FHE encrypts polynomials over the ring:

$$R_t = \mathbb{Z}_t[x]/(x^n+1).$$

As discussed in the previous subsection, we encode a single bit of information into a vector of ℓ bits, enabling homomorphic operations through the addition of two ℓ-bit vectors.

The key challenge now is how to efficiently perform homomorphic addition on two ℓ-bit vectors. We propose the following approach:

1. Each party A_i encodes its bit x into an ℓ-bit vector:
$$\mathbf{x} = (x_1, x_2, \ldots, x_\ell).$$
2. Each party A_i embeds the vector $\mathbf{x}$ into a plaintext polynomial:
$$\sum_j x_j X^j \in R_t.$$
3. All parties $\{A_i\}_i$ jointly invoke a privacy-preserving protocol to compute the logical AND operation.

Remark By encoding the entire vector $\mathbf{x}$ into a single ciphertext instead of encrypting each of the ℓ bits separately, both the computational complexity and communication overhead are reduced by a factor of ℓ.

Further note that ℓ is significantly smaller than the FHE parameter n. This characteristic allows us to encode at least $\lfloor n/\ell \rfloor$ bits into a single ciphertext, thereby reducing both computational and communication complexity by nearly a factor of n.

To summarize, the final SIMD working mode of the privacy-preserving AND protocol proceeds as follows:

1. Each party A_i encodes its $\lfloor n/\ell \rfloor$ bits, denoted as $\{x_{i,j}\}_{j=1,\ldots,\lfloor n/\ell \rfloor}$, into $\lfloor n/\ell \rfloor$ vectors, each of length ℓ:
$$\mathbf{x}_j = (x_{j,1}, x_{j,2}, \ldots, x_{j,\ell}), \quad \text{for } j = 1, \ldots, \lfloor n/\ell \rfloor.$$
2. Each party A_i embeds all $\lfloor n/\ell \rfloor$ vectors $\{\mathbf{x}_j\}_j$ into a single plaintext polynomial:
$$\sum_{j=1}^{\lfloor n/\ell \rfloor} \sum_{k=1}^{\ell} x_{j,k} X^{(j-1)\cdot\ell+k-1} \in R_t.$$
3. All parties $\{A_i\}_i$ collaboratively execute a privacy-preserving protocol to compute the logical AND operation.

7 More Efficient Privacy-preserving AND Protocol

As discussed in Subsection 3.1, a single bit x can be encoded into an ℓ-bit vector $\mathbf{x}$, enabling a privacy-preserving AND protocol via homomorphic addition. A closer examination reveals that, to achieve a negligible failure probability of the protocol (i.e., returning an incorrect result) is at most $2^{-\lambda}$, the vector length ℓ must be set equal to λ.

However, this encoding expansion rate λ can be relatively large in practice. For instance, a typical choice is $\lambda = 40$, and our experiments confirm that this setting significantly burdens the preprocessing phase of the entire PSI protocol. Specifically, each party must encode every bit of their Bloom filter into a long binary vector and then perform FHE encryption in a SIMD-like manner to produce the encrypted representation. Reducing the length of the encoded vector would thus substantially improve the efficiency of the preprocessing phase in our PSI protocol.

To achieve the goal of reducing vector length, we propose a modified privacy-preserving AND protocol that aligns seamlessly with the SIMD-like representation required by FHE encryption. The key intuition is to exploit the size of each SIMD slot: each slot resides in the integer ring $\mathbb{Z}_t$, where t can be greater than 2. This flexibility allows us to encode a single bit x into a much shorter vector $\mathbf{x}$ over $\mathbb{Z}_t$, significantly reducing the encoding overhead.

To summarize, the modified, more efficient privacy-preserving AND protocol $\mathbf{\Pi}_{\mathrm{AND}}$ proceeds as follows:

1. Each party A_i encrypts their bit $x_i \in \{0,1\}$ using an FHE scheme, denoted as $\mathrm{Enc}_{pk}(\mathbf{x}_i)$:

   (a) If $x_i = 1$, then $\mathbf{x}_i$ is set as a zero vector over $\mathbb{Z}_t$:

   $$\mathbf{x}_i \leftarrow \mathbf{0} \text{ w.r.t. } \mathbf{0} = (0, \ldots, 0) \in \mathbb{Z}_t^\ell \text{ where } \ell \overset{def}{=} \frac{\lambda}{\log t}.$$

   (b) Else $\mathbf{x}_i$ is set as a random $\mathbb{Z}_t^\ell$ vector :

   $$\mathbf{x}_i \overset{\$}{\leftarrow} \mathbb{Z}_t^\ell.$$

2. Each A_i sends their ciphertext $\mathrm{Enc}_{pk}(\mathbf{x}_i)$ to the cloud server C.

3. The cloud server C computes:

   $$\sum_i \mathrm{Enc}_{pk}(\mathbf{x}_i) = \mathrm{Enc}_{pk}\left(\sum_i \mathbf{x}_i \mod t\right).$$

4. The cloud server C sends $\mathrm{Enc}_{pk}\left(\sum_i \mathbf{x}_i\right)$ to any arbitrary party A_i.

5. A_i decrypts locally to obtain:

   $$\mathbf{r} = \sum_i \mathbf{x}_i \bmod t.$$

6. If the vector $\mathbf{r}$ is a zero vector, then $\bigwedge_{i=1}^N x_i = 1$, otherwise $\bigwedge_i x_i = 0$.

Remark The correctness and security of the proposed protocol follow from the analysis presented in Subsection 3.1, and we omit the details here. Instead, we focus on the performance improvements achieved. Our PSI protocol is computationally dominated by the encryption/decryption phase, where each party encrypts or decrypts their Bloom filter. With the proposed privacy-preserving AND protocol, the number of ciphertexts required to represent the Bloom filter is reduced by a factor of $\log t$, thereby significantly decreasing the encryption time. Moreover, since the main communication bottleneck lies in transmitting the encrypted Bloom filter to the cloud server, this ciphertext reduction also leads to a substantial decrease in communication overhead. As discussed in Section 5, we may set $t \approx 2^8$, resulting in $\log t = 8$. Consequently, both the computational and communication overheads are improved by nearly an order of magnitude, leading to a significant overall performance gain for the PSI protocol.

8 Conclusion

In this paper, we presented a novel and efficient framework for Delegated Private Set Intersection (D-PSI), leveraging threshold homomorphic encryption and Bloom filter-based encodings. Unlike prior work that relies on complex cryptographic primitives such as randomized intersection encoding, our approach reduces the intersection computation to a series of homomorphic additions, significantly simplifying the protocol design and improving its scalability.

Our construction achieves computational complexity of $\Omega(Nk)$ and communication complexity of $\Omega(Nk)$, matching known asymptotic lower bounds, while also supporting large-scale datasets and multiple parties in a semi-honest cloud setting. We further introduced a suite of concrete optimization techniquesincluding lightweight FHE parameter selection, faster secure-AND protocol design, and SIMD-style batching that greatly improve practical performance.

Beyond the D-PSI setting, we also demonstrated that our construction can be adapted to standard PSI scenarios without reliance on any trusted third party, thus broadening its applicability.

We are currently implementing the proposed protocol and will report our experimental results in a future version of this paper. We believe our work opens a promising direction toward scalable, efficient, and privacy-preserving set intersection protocols.

References

[ATD15] Aydin Abadi, Sotirios Terzis, and Changyu Dong. O-psi: delegated private set intersection on outsourced datasets. In *ICT Systems Security and Privacy Protection: 30th IFIP TC 11 International Conference, SEC 2015, Hamburg, Germany, May 26-28, 2015, Proceedings 30*, pages 3–17. Springer, 2015.

[ATD17] Aydin Abadi, Sotirios Terzis, and Changyu Dong. Vd-psi: verifiable delegated private set intersection on outsourced private datasets. In *Financial Cryptography and Data Security: 20th International Conference, FC 2016, Christ Church, Barbados, February 22–26, 2016, Revised Selected Papers 20*, pages 149–168. Springer, 2017.

[ATD20] Aydin Abadi, Sotirios Terzis, and Changyu Dong. Feather: Lightweight multi-party updatable delegated private set intersection. *Cryptology ePrint Archive*, 2020.

[ATMD17] Aydin Abadi, Sotirios Terzis, Roberto Metere, and Changyu Dong. Efficient delegated private set intersection on outsourced private datasets. *IEEE Transactions on Dependable and Secure Computing*, 16(4):608–624, 2017.

[CLR17] Hao Chen, Kim Laine, and Peter Rindal. Fast private set intersection from homomorphic encryption. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1243–1255, 2017.

[CMdG+21] Kelong Cong, Radames Cruz Moreno, Mariana Botelho da Gama, Wei Dai, Ilia Iliashenko, Kim Laine, and Michael Rosenberg. Labeled psi from homomorphic encryption with reduced computation and communication. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 1135–1150, 2021.

[HLZ25] Jingwei Hu, Zhiqi Liu, and Cong Zuo. Delegated multi-party private set intersection from secret sharing. *Cryptology ePrint Archive*, 2025.

[Ker12] Florian Kerschbaum. Outsourced private set intersection using homomorphic encryption. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, pages 85–86, 2012.

[KMRS14] Seny Kamara, Payman Mohassel, Mariana Raykova, and Saeed Sadeghian. Scaling private set intersection to billion-element sets. In *Financial Cryptography and Data Security: 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers 18*, pages 195–215. Springer, 2014.

[RS21] Peter Rindal and Phillipp Schoppmann. Vole-psi: fast oprf and circuit-psi from vector-ole. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 901–930. Springer, 2021.

[SYY99] Tomas Sander, Adam Young, and Moti Yung. Non-interactive cryptocomputing for nc/sup 1. In *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*, pages 554–566. IEEE, 1999.