Fully Homomorphic Encryption-Based Protocols for Enhanced Private Set Intersection Functionalities

Jingwei Hu[1], Junyan Chen[2], Wangchen Dai[2] and Huaxiong Wang[1]

[1] Nanyang Technological University, Singapore, {davidhu,hxwang}@ntu.edu.sg
[2] Zhejiang Lab, China, {chjuny,w.dai}@zhejianglab.com

Abstract. This study delves into secure computations for set intersections using fully homomorphic encryption (FHE) within the semi-honest setting. Our protocols facilitate joint computations between two parties, each holding a set of inputs denoted as N_s and N_r in size, respectively. The primary objective is to determine various functionalities, such as intersection size and sum, while maintaining data confidentiality. These functionalities extend the classic private set intersection (PSI) and have practical applications in contact tracing, ad conversion analysis, and online dating, each abstracted into specialized PSI protocols.

Our work demonstrates that these extended PSI functionalities are interconnected, with the PSI-cardinality protocol serving as the foundation. By adapting this protocol, we naturally arrive at PSI-sum-cardinality. Additionally, PSI-token-threshold is achieved by augmenting PSI-cardinality with FHE-based oblivious polynomial evaluation (OPE). The tPSI protocol combines PSI-token-threshold and standard PSI, allowing information sharing when the intersection size exceeds a threshold.

Our protocols excel in simplicity, enhancing ease of understanding, implementation, optimization, and long-term maintenance. They also exhibit sublinear communication complexity concerning the larger sender's set, rendering them well-suited for scenarios involving substantial data. Various optimization techniques further bolster their practical efficiency.

Keywords: Private Set Intersection, Threshold Functionality, Fully Homomorphic Encryption

Contents

1 Introduction

This study explores the construction of secure computations for set intersections using fully homomorphic encryptions within the semi-honest setting. Our protocol facilitates joint computations between two parties, often referred to as the sender and receiver, each holding a set of inputs denoted as N_s and N_r in size, respectively. This joint computation aims to determine various functionalities, such as the intersection size, without compromising the confidentiality of each party's set. This extended functionality, beyond the scope of the classic private set intersection (PSI), holds significant practical applications, including:

Contact Tracing: Contact tracing plays a pivotal role in controlling infectious diseases, such as COVID-19. However, conventional contact tracing typically entails the large-scale collection of personal contact information, raising substantial privacy concerns. A foundational element of privacy-preserving contact tracing applications is to enable two parties, each possessing a set of items, to determine the intersection size of their private sets without disclosing the specific items in the intersection. This information aids disease diagnosis, enabling the isolation and testing of infected individuals, thereby curbing further disease transmission. Similar applications can be abstracted under the "PSI-token-threshold" protocol.

Ad Conversion: Ad conversion occurs when a user views an online advertisement for a specific company on a website and subsequently makes a purchase from that company's online store. The company running the ads seeks to quantify the proportion of its revenue attributable to its online advertising campaign. However, the essential data for computing these attribution statistics is distributed across two entities: the ad provider, possessing information about users exposed to a particular ad, and the company itself, holding data on purchase transactions and expenditures. Both parties may face reservations or technical constraints preventing them from sharing the underlying data. Nonetheless, they share a common objective: deriving an aggregate, population-level measurement—identifying the number of users who both viewed an ad and completed a corresponding purchase, along with the total amount spent by these users. Their aim is to achieve this without disclosing any individual user's information beyond these aggregated values within the input datasets. Applications akin to ad conversion can be abstracted in this paper as the "PSI-sum-cardinality" protocol.

Online Dating: Online dating platforms aim to determine whether two individuals, a man and a woman, share common interests and hobbies. If there is a significant overlap in their interests and hobbies, it suggests the potential for further communication, and thus, they should reveal their common interests/hobbies to each other. Conversely, if their interests and hobbies do not align well, it is advisable not to disclose these details. Applications similar to online dating can be abstracted under the "tPSI" protocol. In the "tPSI" protocol, if the intersection size between two parties exceeds a specific threshold (e.g., a relatively large intersection), both parties can access information about the intersection. Otherwise, they receive no information about the other party's set.

In this paper, we demonstrate that by utilizing fully homomorphic encryptions, these extended functionalities for private set intersections are closely interconnected. The foundation for all threshold-related functionalities lies in the "PSI-cardinality" protocol. By adjusting the "PSI-cardinality" protocol, we naturally arrive at the "PSI-sum-cardinality" protocol through a double execution of the "PSI-cardinality." Additionally, it is possible to achieve the "PSI-token-threshold" by augmenting the "PSI-cardinality" protocol with an FHE-based oblivious polynomial evaluation (OPE). Finally, we show that the "tPSI" protocol can be realized by combining the "PSI-token-threshold" and the standard PSI protocol. Our designs for secure computation over PSI outperform in terms of simplicity. A simple protocol is easier to explain to the multiple stakeholders involved, simplifying the decision to adopt new technology. It is also easier to implement without errors, test, audit for correctness, and modify. Furthermore, it is often easier to optimize through parallelization or distributed processing. Simplicity also aids long-term maintenance since, over time, an expanding group of individuals needs to understand the details of how a solution operates.

1.1 Contributions

We outline the contributions of our work, which explores the potential of utilizing state-of-the-art fully homomorphic encryptions to address various computations related to Private Set Intersection (PSI) problems:

1. **Private Set Intersection Cardinality Protocol:** We introduce a novel protocol for calculating the cardinality of the intersection between two sets held by different parties when dealing with semi-honest adversaries. This protocol achieves a communication complexity of $\Omega(N_r \log N_s)$, which is sublinear concerning the larger sender's set.

2. **Private Set Intersection Sum with Cardinality Protocol:** Building upon our proposed Private Set Intersection Cardinality Protocol, we develop a new protocol for computing the sum of elements within the intersection while preserving cardinality information.

3. **Private Set Intersection Token with a Threshold on Cardinality Protocol:** We present a robustPrivate Set Intersection Token with a Threshold on Cardinality Protocol, constructed based on our previously established Private Set Intersection Cardinality Protocol.

4. **Threshold PSI Protocol:** We devise a Threshold PSI Protocol for two parties within the semi-honest secure model. This protocol combines a standard PSI protocol with our new Private Set Intersection with Threshold Protocol, offering sublinear communication complexity.

5. **Efficient Performance** The communication complexity of the proposed protocols is low, exhibiting a logarithmic scaling with respect to the size of the sender's set, which is particularly advantageous when dealing with relatively large sender sets. Furthermore, we have introduced various optimization techniques to ensure the efficiency of our protocols in practice.

6. **Simplicity and Modularity:** Throughout our work, we maintain a focus on conceptual simplicity and modularity. This approach streamlines both security analysis and implementation processes, enhancing the practicality of our protocols.

These contributions collectively advance the understanding and application of fully homomorphic encryptions in addressing complex computations within the context of PSI problems.

1.2 Technical Overview

The basis of the secure computations over set intersection is the private set intersection cardinality `PSI-cardinality` protocol. However, in contrast to previous related works [DCW13, DD15, DD15, ZC18], our new solution is to utilize the FHE-based oblivious polynomial evaluation technique for directly executing membership tests and later accumulating these test results to obtain intersection cardinality. Based on `PSI-cardinality`, three variant protocols, *i.e.*, prviate set intersection sum with cardinality (`PSI-sum-cardinality`), *i.e.*, private set intersection cardinality with threshold (`PSI-token-threshold`), and threshold private set intersection (`tPSI`) are proposed.

PSI-cardinality Protocol We assume that one party, referred to as the sender, holds a set $S = \{s_i\}_i$, and the other party, referred to as the receiver, holds another set $R = \{r_i\}_i$. The PSI-cardinality protocol securely computes the size of the intersection between these two sets without revealing any other information about the sets themselves. Initially, we construct a homomorphic membership test, which returns '1' if an element belongs to the intersection and '0' otherwise. Summing the results of this membership test in a fast homomorphic manner yields an encryption of the intersection's cardinality.

PSI-sum-cardinality Protocol In the PSI-sum-cardinality protocol, we consider a scenario where the sender holds a set $S' = \{s_i : v_i\}_i$, where s_i represents the i-th element ID, and v_i represents the associated payload. The receiver holds another set $R = \{r_i\}_i$. This protocol securely computes both the intersection size $|S \cap R|$ and the sum of payloads v_i for all $s_i \in S \cap R$ between the two parties without disclosing additional information about the sets. The intersection size can be directly computed using the PSI-cardinality protocol mentioned above. By encoding the values v_i into the interpolation polynomial used in the membership test, it is also possible to obtain $\sum_i v_i$ using the PSI-cardinality protocol.

PSI-token-threshold Protocol The goal of the PSI-token-threshold protocol is to enable the sender to share a common token with the receiver if and only if the intersection set size satisfies the threshold criterion. In more formal terms, two parties jointly and securely compute a token K_0 if the intersection size satisfies the condition and compute another token K_1 if it does not. Initially, we employ the PSI-cardinality protocol, allowing the sender to hold an encryption of the intersection size, i.e., $Enc(|S \cap R|)$. Then, the sender invokes an oblivious polynomial evaluation on the input $Enc(|S \cap R|)$ to obtain $Enc(K_i)$.

tPSI Protocol Our threshold private set intersection (tPSI) protocol consists of two subprotocols. First, the PSI-token-threshold protocol securely shares a secret token K' between the two parties if the size of the set intersection satisfies the specified requirement. Otherwise, K' remains completely random. Next, we employ a standard private set intersection protocol. In our modified PSI-token-threshold protocol, the interpolation polynomial used in homomorphic token generation is simpler, allowing for faster evaluation. At the end of the PSI-token-threshold protocol, the sender, who has his set S and a secret token K, modifies his set as $S^K = \{s_i||K\}_i$, where each element in the new set S^K consists of the original set element s_i concatenated with the token K. Similarly, the receiver uses his token K' to modify his set as $R^{K'} = \{r_i||K'\}_i$. Finally, the sender and receiver engage in a normal PSI protocol (note that there is no restriction on the PSI protocol used here) with their updated sets, resulting in the intersection $S \cap R$ if and only if the intersection size satisfies the threshold condition.

2 Related work

The standard PSI, *i.e.*, two parties securely computing the set intersection (does not require computing some function f on the intersection) while the privacy of each party is preserved, is extensively studied in the literature and is still an active research topic. We categorize these research work into 5 classes as follows.

Diffie-Hellman key exchange based The earliest PSI protocols are build upon the Diffie-Hellman key exchange [Mea86, HFH99] with semi-honest security, and later extended with malicious security [CKT10, JL10]. RSA accumulator is exploited in to implement PSI protocol [ACT11]. The merit of the DH-based PSI protocols is the low communication complexity. However, these protocols reply on the complicated computation of expoentiation, which makes the computational overhead rather large. In particular, when the set held by each party is large, the computation efficiency for the DH-based PSI is low.

Oblivious Transfer based OT-extension based constructions are the mainstream of PSI protocols. These constructions typically include the hash-to-bins technique by using various hashing algorithms, *i.e.*, simple hashing, permutation-based hashing, or Cuckoo hashing, which helps reduce the computational overhead. Since the OT extension technique which relies on fast symmetric encryptions and a few asymmetric encryptions is highly efficient, the OT-extension based constructions [PSZ14, KKRT16, PSSZ15, HV17, RR17, CLR17, OOS17, PSWW18, PSZ18] are also fast. [KKRT16] also points out that the OT extensions used in this class of PSIs can be viewed as oblivious pseudorandom function (OPRF).

Oblivious Key-value Store based OKVS-based construction emerges as a new branch of PSI protocols. It uses a special encoding method to encode the set into an oblivious data structure called oblivious key-value store(OKVS). The computing efficiency of OKVS-based constructions [PRTY19, CM20] is initially low since they use complicated polynomial interpolation technique. Later proposed the so called PaXoS data structure which significantly improves the computing efficiency. Improving on this result, the up-to-date fastest OKVS construction appears [RS21]. The concept and the method for OKVS is further refined in [GPR$^+$21].

Polynomial Manipulation The polynomial manipulation based constructions [FNP04, KS05, DSMRY09, HN10, Haz18, FHNP16, GS19, GN19] present the set as a polynomial, and use polynomial related operations, *e.g.*, oblivious polynomial evaluation (OPE), to equivalently perform set operations. These constructions are generally slower but also show advantages on other aspects. For example, they can argue the security in the standard model without replying on non-standard assumptions like random oracle model. They can provide more functionalities rather than simply computing the set intersection (refer to the threshold PSI [GS19] as one such instance). [CLR17] exploits fully homomorphic encryption to perform polynomial evaluation obliviously. Their method shows better communication complexity which only relies on the size of the smaller set if the size of set held by each party is different. [CMdG$^+$21] further optimizes [CLR17] by exploring Paterson-Stockmeyer algorithm, postage-stamp bases, and Frobenius mapping to reduce the computation and communication overhead.

Branching Program This is a newly emerging and intriguing approach [JTKA22], with an asymptotic communication complexity sublinear to the smaller set. It exhibits an advantage when computing between non-equal sets. The core concept involves transforming set membership tests into Branching Programs (BPs), and homomorphically executing the BP. The significance of this approach lies not only in its potential for calculating PSI but also in its convenience for various operations based on PSI results, such as private computation on set intersections (PCI). Currently, this method is limited to theoretical constructs and theoretical performance estimations.

Besides the study of standard PSI, the cryptographic research community also investigate variants of standard PSI. Circuit Private set Intersection, Private intersection cardinality and threshold private set intersection, among others, are probably most important variants.

Circuit Private Set Intersection: Differing from the standard Private Set Intersection (PSI), Circuit Private Set Intersection (Circuit-PSI) provides each party with a vector form of shares (secret shares). These shares can collectively reconstruct a complete 0/1 vector, where '1' signifies an element in the intersection, and '0' indicates an element outside the intersection. In the initial work on Circuit-PSI [HEK12], a generic garbled circuit was considered for constructing PSI, with computational and communication complexity of $\Omega(N \log N)$. [PSSZ15] employed Cuckoo-Hashing and Simple-Hashing for bin-wise comparison, slightly reducing the computational and communication complexity to $\Omega(N \log N / \log \log N)$. [PSTY19] introduced the concept of Oblivious Batched Programmable PRF (Batched-OPPRF) and, based on this, first constructed Circuit-PSI with linear communication complexity of $\Omega(N)$. [CGS21] further improves on [PSTY19] to construct a circuit-PSI with both linear communication and computation complexity. [RS21], besides introducing VOLE for constructing OPRF and ultimately standard PSI, explored the transformation of VOLE-based OPRF into Batched-OPPRF to construct Circuit-PSI. [LPR$^+$21] discussed the Circuit-PSI problem for asymmetric set sizes for the first time, introducing PIR techniques for resolution. [SJ23] similarly explored Circuit-PSI with asymmetric set sizes, leveraging Fully Homomorphic Encryption (FHE) and MPC techniques to construct a more efficient Circuit-PSI.

Private Intersection Cardinality Private intersection cardinality problem [KS05, HW06, DD15, EFG$^+$15, PSWW18], considers how two parties collaboratively compute the cardinality of the intersected set, rather than the intersection itself. [FNP04] points out the lower bound of communication complexity of private set intersection is $\Omega(n)$, where n is the size of the set held by each party. This lower bound naturally applies to the private intersection cardinality problem. Google's research team explored a variant of Private Intersection Cardinality problem called the two-party intersection-sum with cardinality problem for the first time in [IKN$^+$20]. This problem involves simultaneously calculating the size of the intersection and the sum of set payloads. According to Google's paper, the most practical methods currently still rely on classical Diffie-Hellman and additive homomorphic encryption schemes. In [MPR$^+$20], research was conducted on the intersection-sum with cardinality problem under a malicious security model. Additionally, [TSS$^+$20] proposed a novel method for estimating the size of privacy-preserving intersections in the context of designing a COVID-19 contact tracing system (utilizing a combination of Diffie-Hellman and Keyword-PIR).

Threshold Private Set Intersection In many applications, we do not target computing the intersection or the intersection cardinality. For example, the two parties in the fingerprint matching, *i.e.*, a receiver/client who has a small set of all his features in his fingerprint, and a sender/server who has a large set which contains fingerprint features for multiple users. The protocol returns yes if the number of matched features excess some prescribed threshold parameter t (in order to tolerate false negative errors), otherwise returns no. Another good example is the lover pairing in dating website. the two parties are the man who has a set of his preferences, and the women who also has a set of her preferences. The protocol should return the intersected preferences if the number of shared preferences are large such that this pair of man and woman can learn more about each other. Otherwise, it should return an empty set (or nothing) to avoid personal privacy leakage. These applications are abstractly referred to as threshold PSI where each of the two parties hold a set of size n, and engates to compute the set intersection if the number of overlapping is above a threshold value t. Otherwise, each of the two parties should learn nothing. The threshold PSI problem is investigated in [FNP04, HOS17, GN19, PSWW18, ZC18]. [HOS17] combines the phasing technique and the threshold key encapsulation mechanism to construct threshold PSI. [PSWW18] optimizes the generic MPC for comparsion circuit to construct threshold PSI. [GN19] constructs a malicious secure threshold PSI based on the oblivious linear function evaluation and robust secret sharing. [ZC18] presents Paillier partial homomorphic encryption based oblivious polynomial evaluation to construct semi-honest secure PSI protocol. [GS19] reveals that when the intersection is large (say the size of intersection is at least $n - 2t \approx n$), threshold PSI protocol has to have a communication complexity of $\Omega(t)$. Based on the results from [GS19], threshold Private Set Intersection for multiple parties is further studied in [BMRR21] which suggests the sublinear communication complexity still holds in the multi-party case.

3 BFV Fully Homomorphic Encryption

BFV, also known as one representative scheme in the second generation of fully homomorpic encryption (FHE), is advantageous for leveled homomorphic encryptions, *i.e.*, homomorphic evaluation of a circuit with shallow multiplication depth. Theoretically speaking, by using Gentry's bootstrapping technique, BFV can support evaluating a circuit with unlimited multiplication depth. Nevertheless, the state-of-arts of bootstrapping is still far from being efficient. In this paper, since we focus on constructing efficient for the threshold PSI related computations with optimal communication complexity, we restrict the usage of BFV to leveled HE mode.

3.1 RLWE

BFV ciphertext is essentially a RLWE ciphertext defined over polynomial ring $R_q = \mathbb{Z}_q[X]/(X^N + 1)$:

$$(a(X), b(X)) = (a(X), a(X) \cdot s(X) + \lceil \frac{q}{t} \rfloor m(X) + e(X)) \in R_q \times R_q$$

where s is a secret key, m is the message defined over $\mathcal{R}_t = \mathbb{Z}_t[X]/(X^N + 1)$ (*a.k.a* BFV plaintext), $e = \sum_i e_i X^i$ is an error polynomial where each coefficient is extracted from a discrete Gaussian distribution with small variance $e_i \sim \mathcal{N}(0, \sigma^2)$. Likewise, we use $Enc_s^{N,q}(m)$ to denote a RLWE encryption of message $m(X)$.

3.2 Addition and Multiplication

Homomorphic addition and multiplication are two basic homomorphic operations supported by BFV. We skip the algorithmic descriptions for these two primitives but outline the functionality abstractly. `HomoAdd` takes two inputs as a RLWE encryption of message m_0 and a RLWE encryption of message m_1 defined over $\mathcal{R}_q \times \mathcal{R}_q$ and outputs another RLWE encryption of $m_0 + m_1$ defined over $\mathcal{R}_q \times \mathcal{R}_q$:

$$Enc_s^{N,q}(m_0 + m_1) \leftarrow Enc_s^{N,q}(m_0) + Enc_s^{N,q}(m_1)$$

`HomoAdd` consumes trivial amount of noise budget and thus BFV can perform almost unlimited number of homomorphic additions. `HomoAdd` can also support addition of one RLWE plaintext and one RLWE ciphertext, *i.e.*, $Enc_s(m_0 + m_1) \leftarrow m_0 + Enc_s(m_1)$.

Likewise, `HomoMul` takes two inputs as a RLWE encryption of m_0 and a RLWE encryption of m_1 and outputs another RLWE encryption of $m_0 \cdot m_1$:

$$Enc_s^{N,q}(m_0 \cdot m_1) \leftarrow Enc_s^{N,q}(m_0) \times Enc_s^{N,q}(m_1)$$

It is worth noting that `HomoMul` consumes significantly more noise budget than `HomoAdd` does. A typical value for the depth of multiplcation that BFV supports is 8. `HomoMul` also supports multiplication of one RLWE plaintext and one RLWE ciphertext, *i.e.*, $Enc_s(m_0 \cdot m_1) \leftarrow m_0 \times Enc_s(m_1)$.

3.3 SIMD Encoding

`SIMD` (single instruction multiple data) encoding is a unique feature for the second generation of FHEs which encodes a vector $\mathbf{m} = (m_0, \cdots, m_{N-1})$ over $\mathbb{F}_q$ to a polynomial over $\mathcal{R}_q$:

$$\mathbb{F}_q^N \xrightarrow{\texttt{SIMD}} p(X) \in \mathcal{R}_t$$

The key idea in `SIMD` is that $x^N + 1$ can be factorized into $\prod_{i=0}^{N-1}(X - \zeta^{2i+1})$. By Chinese remainder theorem (CRT), we have:

$$R_t = \frac{\mathbb{Z}_t[x]}{x^N+1} = \frac{\mathbb{Z}_t[x]}{\prod_{i=0}^{N-1}(x-\zeta^{2i+1})} \simeq^{CRT} \prod_{i=0}^{N-1} \frac{\mathbb{Z}_t[x]}{x-\zeta^{2i+1}} = \prod_{i=0}^{i=N-1} \mathbb{Z}_t[\zeta^{2i+1}]$$

In other terms, every polynomial $p(X) = \sum_i p_i X^i$ over $\mathcal{R}_t$ can be equivalently represented by its evaluations at $X = \zeta^{2i+1}$ for $i \in [N]$. The evaluations can be written in matrix form as follows:

$$\begin{bmatrix} m_0 \\ \vdots \\ m_{N-1} \end{bmatrix} = \begin{bmatrix} \zeta^{1\cdot 0} & \cdots & \zeta^{1\cdot(N-1)} \\ \vdots & \vdots & \vdots \\ \zeta^{(2N-1)\cdot 0} & \cdots & \zeta^{(2N-1)\cdot(N-1)} \end{bmatrix} \cdot \begin{bmatrix} p_0 \\ \vdots \\ p_{N-1} \end{bmatrix}$$

The vector $[m_0, \cdots, m_{N-1}]^T$ is essentially the N evaluation points associated with the polynomial $p(X)$. For SIMD encoding purpose, this vector of evaluation points are interpreted as the BFV plaintext vector, and by using SIMD encoding technique, the BFV plaintext vector is transformed to BFV plaintex polynomial $p(X)$. It is easily seen that BFV SIMD encoding is the inverse process of evaluating $p(X)$ at points $X = \zeta^{2i+1}$, written in matrix form as follows:

$$\begin{bmatrix} p_0 \\ \vdots \\ p_{N-1} \end{bmatrix} = \begin{bmatrix} \zeta^{1\cdot 0} & \cdots & \zeta^{1\cdot(N-1)} \\ \vdots & \vdots & \vdots \\ \zeta^{(2N-1)\cdot 0} & \cdots & \zeta^{(2N-1)\cdot(N-1)} \end{bmatrix}^{-1} \cdot \begin{bmatrix} m_0 \\ \vdots \\ m_{N-1} \end{bmatrix}$$

Based on the isomorphism between the plaintext vector and the plaintext polynomial constructed using the CRT, performing addition or multiplication of two plaintext polynomials is equivalent to element-wise addition or multiplication of two plaintext vectors. In other words,

$$\text{If } p_0(X) \leftarrow \texttt{SIMD}(\mathbf{m_0}), p_1(X) \leftarrow \texttt{SIMD}(\mathbf{m_1}),\ \text{then } p_0(X) + p_1(X) \leftrightarrow \mathbf{m_0} + \mathbf{m_1},\ \text{and}$$
$$p_0(X) \cdot p_1(X) \leftrightarrow \mathbf{m_0} \otimes \mathbf{m_1}$$

3.4 Left Rotation

Regarding the rotation, we view the vector after SIMD encoding as a $2 \times \frac{N}{2}$ matrix. In particular, we set $\zeta_j = \zeta^{5^j (\bmod 2N)}$ where ζ is a 2N-th primitive root of unity. $\zeta_j = \zeta^{5^j (\bmod 2N)}$ for all j forms a multiplicative group with order $\frac{N}{2}$ and thus $p(\zeta_j)$ for $j \in [\frac{N}{2}]$returns the first row of the matrix while $p(\zeta_j^{-1} (\mathrm{mod} t))$ for $j \in [\frac{N}{2}]$ returns the second row of the matrix.

The left cyclic rotation on the ciphertext $\mathbf{ct}$ by r slots $LRot(\mathbf{ct}, r)$ is described as follows:

1. Pre-compute the rotation key $\mathbf{rk} \leftarrow KSGen(s, s(X^{5^r}))$
2. Let $\mathbf{ct} = (a(X), b(X))$, apply Frobenius mapping to obtain $\mathbf{ct'} = (a(X^{5^r}), b(X^{5^r}))$
3. Return $KS(\mathbf{rk}, \mathbf{ct'})$

where $KSGen(s, s(X^{5^r}))$ denotes the key-switching key generation algorithm which essentially encrypts $s(X^{5^r})$ under the secret key s and $KS(\mathbf{rk}, \mathbf{ct'})$ denotes the key switching algorithm which essentially converts a message encrypted under $s(X^{5^r})$ to the same message under a different secret key s.

It is worth mentioning that the left rotation applies to each row of the matrix separately. It is also possible to rotate the columns,*i.e.*, swap the rows by using the following algorithm $SwapRow(\mathbf{ct})$:

1. Pre-compute the rotation key $\mathbf{rk} \leftarrow KSGen(s, s(X^{-1}))$
2. Let $\mathbf{ct} = (a(X), b(X))$, apply Frobenius mapping to obtain $\mathbf{ct}' = (a(X^{-1}), b(X^{-1}))$
3. Return $KS(\mathbf{rk}, \mathbf{ct}')$

3.5 Oblivious Polynomial Evaluation (OPE)

Oblivious Polynomial Evaluation (OPE) is a simple two-party secure computation protocol where the sender possesses a secret polynomial $p(x) = \sum_{i=0}^{N_s} p_i x^i$, and the receiver has a secret input y. Together, they jointly evaluate the polynomial at a point $x = y$ without disclosing the sender's polynomial $p(x)$ and the receiver's value of y.

OPE can be implemented using the BFV fully homomorphic encryption scheme as follows: The receiver encrypts their secret input y as $Enc(y)$ and sends this encrypted value, denoted as $Enc(y)$, to the sender. The sender can then compute $Enc(p(y))$ by applying the BFV homomorphic addition (`HomoAdd`) and multiplication (`HomoMul`) operations iteratively as follows:

$$\sum_i p_i \cdot (Enc(y))^i = Enc(\sum_i p_i y^i) = Enc(p(y))$$

It is relatively straightforward to prove that the BFV-based OPE protocol is secure in the semi-honest setting, assuming the RLWE ciphertext is semantically secure. However, the direct method to compute OPE using this approach requires approximately $\Omega(N)$ homomorphic multiplications, and the multiplication depth is around $\Omega(\log N)$. As previously mentioned, BFV in leveled homomorphic encryption mode can only support limited multiplication depth, making the straightforward approach impractical for large values of N.

A more practical method proposed in [CLR17] involves the receiver preparing $\lceil \log N_s \rceil$ ciphertexts, denoted as $Enc(y^{2^i})$ for $i \in [\lceil \log N_s \rceil]$, and sending these $logN$ ciphertexts to the sender. The sender then employs a homomorphic multiplication algorithm to retrieve $Enc(y^i)$ for all $i \in [N_s]$. It's important to note that this alternative method requires only $logN_s$ homomorphic multiplications and consumes a relatively low multiplication depth of approximately $\Omega(\log \log N_s)$. However, the trade-off for this efficiency improvement is a significant increase in communication complexity, specifically the number of ciphertexts sent from the receiver to the sender, which goes from approximately $\Omega(1)$ to $\Omega(\log N_s)$.

3.6 Bloom Filter

Bloom filter [Blo70] is a data structure designed to efficiently determine if an element belongs to a set, but they do not provide a way to retrieve the actual object. It stores membership information in a compact manner.

A Bloom filter is represented as a bit array, denoted as $\mathbf{BF}[\cdot]$ (initialized as an all-zero vector). Bloom filter supports two elementary operations: `AppendObject` and `TestMembership`.

`AppendObject` adds the target object to the Bloom filter array and follows these steps:

1. Select k independent hash functions denoted as $(Hash_1(\cdot), \cdots, Hash_k(\cdot))$. These hash functions do not necessarily need to be cryptographically secure.
2. Compute the hash value of the target object s for each hash function, denoted as $\text{hash}_i \leftarrow Hash_i(s)$ for all i.

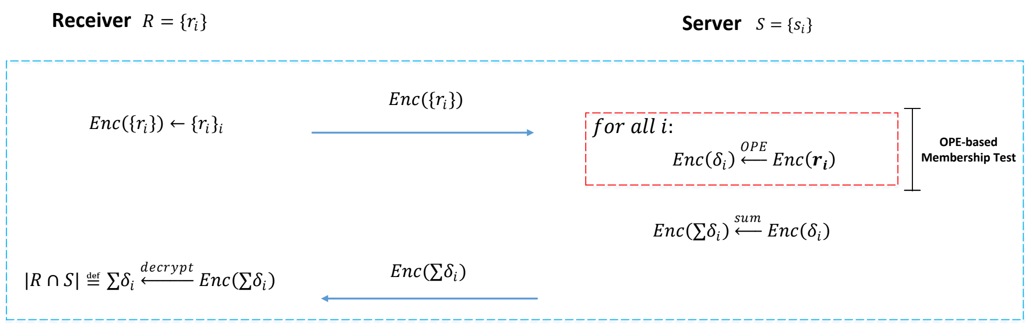
Receiver $R = \{r_i\}$
Server $S = \{s_i\}$
$Enc(\{r_i\}) \leftarrow \{r_i\}_i$
$Enc(\{r_i\})$
for all i:
$Enc(\delta_i) \xleftarrow{OPE} Enc(\boldsymbol{r_i})$
OPE-based Membership Test
$Enc(\sum\delta_i) \xleftarrow{sum} Enc(\delta_i)$
$|R \cap S| \overset{\text{def}}{=} \sum\delta_i \xleftarrow{decrypt} Enc(\sum\delta_i)$
$Enc(\sum\delta_i)$

Figure 1: Private set intersection cardinality `PSI-cardinaity` protocol

3. Treat each hash value, hash_i, as an index in the Bloom filter array and set $\mathbf{BF}[\text{hash}_i] = 1$ for all i.

`TestMembership` checks whether a target object has been logged in the Bloom filter and follows these steps:

1. Use the same set of hash functions mentioned in the `AppendObject` operation to compute a set of hash values, $\{\text{hash}_i\}_{i\in[k]}$, for the given target object.
2. Treat each hash value hash_i as an index in the Bloom filter array and check the value of $\mathbf{BF}[\text{hash}_i]$. If $\mathbf{BF}[\text{hash}_i] = 1$ for all hash_i, the target object is considered to be in the Bloom filter. Otherwise, the object is not present.

The false positive rate (p_{FPR}) is an important performance measure for Bloom filters. It represents the probability that the membership test operation incorrectly asserts that the target object is in the Bloom filter, which is estimated using the formula [BGK+08]:

$$p_{\text{FPR}} = \left(1 - e^{\frac{-k\cdot|S|}{|\mathbf{BF}|}}\right)^k$$

where k is the number of hash functions used, $|\mathbf{BF}|$ is the bit length of the Bloom filter array, and $|S|$ is the maximum size of the set that the Bloom filter needs to record.

For a fixed set size $|S|$, a fixed number of hash functions k, and a desired false positive rate p_{FPR}, the size of a Bloom filter, namely the bit length of $\mathbf{BF}$ can be computed as:

$$|\mathbf{BF}| = \frac{k\cdot|S|}{\ln(1 - p_{\text{FPR}}^{\frac{1}{k}})^{-1}}$$

In other words, when the Bloom filter system parameters p_{FPR}, k are preconfigured, the size of the Bloom filter is consequently determined and should be proportional to the size of the set S, given that $|\mathbf{BF}| = \mathcal{O}(|S|)$. For instance, setting the false positive rate to 2^{-20} and the number of hash functions to $k = 10$ leads to a Bloom filter size of $|\mathbf{BF}| \approx 34.8 \times |S|$.

4 The proposed Private Set Cardinality Protocol

In this section, we introduce our novel protocol for private set cardinality, also known as `PSI-cardinality`. This protocol enables the secure computation of intersection cardinality without disclosing any information about the sets involved, utilizing the BFV Fully Homomorphic Encryption scheme.

The `PSI-cardinality` protocol can be roughly divided into two distinct phases, as illustrated in Fig. 1:

- **Membership Test (First Phase):** During this phase, the receiver/client encrypts their set elements, denoted as $Enc(r_i)$ for all i, and transmits them to the sender/server. Subsequently, the sender/server conducts homomorphic tests on each r_i and returns results denoted as δ_i. Here, $\delta_i = 1$ signifies that r_i exists in set S, while $\delta_i = 0$ indicates that r_i is not present in set S.

- **Homomorphic Sum (Second Phase):** the sender/server homomorphically sums the results of the membership tests to obtain the cardinality as $\sum_i \delta_i$.

It's worth noting that the correctness of the `PSI-cardinality` protocol is self-evident, as it accurately computes the intersection cardinality while preserving the confidentiality of the sets involved.

4.1 Challenges in `PSI-cardinality`

In this subsection, we highlight the primary challenges encountered in the `PSI-cardinality` protocol and present our innovative methods for addressing these challenges.

Challenge 1: Homomorphic Computation of Membership Indicator (δ): The first challenge revolves around the homomorphic computation of an indicator δ for the membership test. In essence, the sender, which possesses the set S, must homomorphically determine whether an element r_i from the receiver's set R belongs to S or not. This can be formally defined as:

$$\delta_i \overset{def}{=} \delta(r_i, S) = \begin{cases} 1 & \text{if } r_i \in S \\ 0 & \text{Otherwise} \end{cases}$$

The complexity lies in the need to convert this problem into an Oblivious Polynomial Evaluation (OPE) while maintaining a low multiplicative depth.

Challenge 2: Homomorphic Summation of δ_i: The second challenge involves homomorphically summing the δ_i values for all r_i, resulting in the intersection size $|R \cap S| = \sum_{i \text{ s.t. } r_i \in R} \delta(s_i, S)$. This task becomes intricate due to the utilization of Single Instruction, Multiple Data (SIMD) coding in Fully Homomorphic Encryption (FHE). Specifically, it requires manipulation of the elements within the encrypted vector, i.e., the summation of all elements in the SIMD slot.

To tackle these challenges, our `PSI-cardinality` protocol introduces innovative methods that enable efficient homomorphic computation while maintaining data privacy and security.

4.2 Using OPE for Membership Test

In this subsection, we address the first critical task in our proposed PSI protocol, which is the implementation of the primitive δ in the membership test. We assume that each element in the sets R and S is represented as a σ_{max}-bit integer. To achieve this, we define a super set $S^{(sup)} \overset{def}{=} \{1, 2, \cdots, 2^{\sigma_{max}} - 1\}$ that encompasses the sender's set S. This super set satisfies two important properties:

1. $N_{s^{sup}} \overset{def}{=} |S^{(sup)}| = 2^{\sigma_{max}} - 1$
2. $S \subset S^{(sup)}$

Notably, the element 0 is deliberately excluded from $S^{(sup)}$ and reserved as a dummy item.

To build the polynomial $P_\delta(X)$, we opt for Newton interpolation due to its ability to incorporate new points without the need for a complete reevaluation of all coefficients.

This advantageous feature substantially reduces the offline processing workload on the sender's side, especially when the sender's set experiences frequent updates.

The Newton interpolation polynomial for $P_\delta(X)$ is constructed as:

$$P_\delta(X) = [y_0] + [y_0, y_1](x - x_0) + \cdots + [y_0, \cdots, y_{k-1}](x - x_0)(x - x_1)\cdots(x - x_{k-2})$$

where $[y_0, \cdots, y_j]$ is the notation for divided differences. The set S uses evaluation points (x_i, y_i) for $x_i \in S \cap S^{(sup)}, y_i = 1$ and for $x_i \in S^{(sup)} - S, y_i = 0$. It is important to note that the degree of $P_\delta(X)$ is precisely $2^{\sigma_{max}} - 1$, aligning with the properties of the super set $S^{(sup)}$.

It's essential to emphasize that the membership test described above must be executed in a homomorphic fashion. Here's a step-by-step overview of the process:

1. The receiver encrypts each element r_i in their set R and transmits the resulting ciphertexts, denoted as $Enc(r_i)$, to the sender/server.
2. The sender/server applies Oblivious Polynomial Evaluation (OPE) on $Enc(r_i)$ and $P_\delta(X)$ to compute $Enc(P_\delta(r_i))$.

The computation of $Enc(P_\delta(c_i))$ using a straightforward method, which computes $Enc(r_i)^k$ for all k from $Enc(r_i)$, would typically require $\log N_{s^{sup}}$ multiplicative depth. However, it's important to note that BFV supports a very limited multiplicative depth, with a typical value being around 8. Consequently, the straightforward approach is not feasible within the constraints of BFV's limitations. This challenge highlights the need for innovative methods to efficiently compute $Enc(P_\delta(c_i))$ while adhering to the restrictions imposed by BFV's limited multiplicative depth.

Reduce the multiplicative depth To address the challenge of limited multiplicative depth in BFV, we can employ a technique that balances communication complexity and multiplicative depth. Here are the key components of this approach:

- Increasing Communication Complexity: The receiver prepares a set of $N_{s^{sup}}$ ciphertexts for each r_i. These ciphertexts include $Enc(r_i), Enc(r_i^2), \ldots, Enc(r_i^{N_{s^{sup}}})$ for each element r_i. All of these ciphertexts are then sent to the sender/server.
- Computing $Enc(P_\delta(X))$ in Depth One: With the receiver's preparation of these $N_{s^{sup}}$ ciphertexts, the sender/server can efficiently compute $Enc(P_\delta(X))$ with a multiplicative depth of just one.

However, this approach significantly increases the communication complexity by a factor of $N_{s^{sup}}$. To strike a balance between efficiency and communication complexity, we can implement a method called "windowing" proposed in [CLR17], which proceeds as follows:

- Balanced Approach - Windowing: In this method, the receiver prepares a more balanced set of ciphertexts. Specifically, they generate $\lfloor \log_2 N_{s^{sup}} \rfloor + 1$ ciphertexts for each c_i. These ciphertexts include encrypted powers-of-two $Enc(r_i), Enc(r_i^2), \ldots, Enc(r_i^{2^{\lfloor \log_2 N_{s^{sup}} \rfloor}})$.
- Improved Depth: With this setup, the sender can compute $Enc(P_\delta(X))$ with an improved depth of $\lceil \log(\lfloor \log N_{s^{sup}} \rfloor + 1) \rceil$, which is roughly equivalent to $\log \log N_{s^{sup}}$.

Improve computational complexity It's important to highlight the computational differences between ciphertext-ciphertext multiplication and ciphertext-plaintext multiplication within the context of homomorphic encryption:

- Ciphertext-Plaintext Multiplication: This operation involves just two multiplications over the ring R_q. It is relatively efficient and consumes minimal computational resources.

- Ciphertext-Ciphertext Multiplication: In contrast, this operation is significantly more complex. It typically requires about $4 + \Omega(\log_2 q)$ multiplications over the ring R_q. As a result, ciphertext-ciphertext multiplication is generally at least two orders of magnitude slower than ciphertext-plaintext multiplication.

Given this substantial difference in computational cost, it becomes evident that the bottleneck in the computation of Oblivious Polynomial Evaluation (OPE) lies in the number of ciphertext-ciphertext homomorphic multiplications used to compute all powers of X. Therefore, it is reasonable to estimate the computation overhead of the membership test based on the number of homomorphic multiplications involved in the OPE protocol.

As mentioned earlier, the windowing method is employed to strike a balance between multiplicative depth and communication overhead. In this method, the number of ciphertext-ciphertext multiplications is upper-bounded by approximately $N_{s^{sup}} - \lfloor \log_2 N_{s^{sup}} \rfloor$, which is roughly equivalent to $N_{s^{sup}}$.

To enhance computation efficiency while maintaining a low multiplicative depth, we employ a better method than windowing called the Paterson-Stockmeyer algorithm [CMdG+21] in the Oblivious Polynomial Evaluation (OPE) protocol. In this context, we rewrite $P_\delta(Enc(X))$ as follows:

$$P_\delta(Enc(X)) = \sum_{i=0}^{B} P_{\delta,i} \cdot (Enc(X))^i = \sum_{i=0}^{H-1} \left(\sum_{j=0}^{L-1} P_{\delta,iL+j} \cdot Enc(X)^j \right) \cdot Enc(X)^{iL}$$

Here, we ensure that $H \cdot L \geq B + 1$ and $H \approx L = \Omega(\sqrt{N_{s^{sup}}})$. The sender's precomputation involves calculating $Enc(X)^j$ for all $j \in [L]$ and $Enc(X)^{iL}$ for all $i \in [H]$ using values sent by the receiver. This process consumes exactly $L + H - \lceil \log_2 L \rceil - \lceil \log_2 H \rceil - 2$ ciphertext-ciphertext multiplications. Additionally, $H-1$ ciphertext-ciphertext multiplications are required to evaluate $P_\delta(Enc(X))$.

In summary, the OPE protocol's total consumption of ciphertext-ciphertext multiplications is approximately $L + 2H - \lceil \log_2 H \rceil - \lceil \log_2 L \rceil - 3 \approx 3\sqrt{N_{s^{sup}}}$. This approach significantly improves efficiency by striking a better balance between computational complexity and multiplicative depth.

Process items with arbitary bit length In the preceding discussion, we make the assumption that each item s_i in the sender's set S is at most σ_{max} bits long. However, due to the limited system parameters supported by leveled FHE, the value of σ_{max} cannot be set too large, typically staying below 32. Nevertheless, this relatively small value for σ_{max} might prove inadequate for real-world applications, as the string length used to represent an item in the set could be arbitrary. We propose here an alternative solution involves compressing the bit length of s_i through Bloom Filter (BF) encoding.

BF encoding transforms each $s_i \in S$ into an array of indices $(idx_{s_i,1}, \cdots, idx_{s_i,k})$, where $idx_{s_i,j} \leftarrow Hash_j(s_i)$, and sets $\mathbf{BF}_S[idx_{s_i,j}] = 1$. Since the size of the BF array is proportional to the set size $|S|$, *i.e.*, $|\mathbf{BF}_S| = \mathcal{O}(|S|)$, the upper bound for $idx_{s_i,j}$ is $\mathcal{O}(|S|)$, and thus $\mathcal{O}(\log |S|)$ bits are sufficient to represent it, regardless of the bit length of the string s_i.

We provide a more detailed description of the BF-based method in Algorithm 1 for handling arbitrary-long strings in the membership test. In line 1, a BF array $\mathbf{BF}_S$ associated with S is constructed, where $\mathbf{BF}_S[idx_{s_i,j}] = 1$ and $idx_{s_i,j} \leftarrow Hash_j(s_i)$ for all i, j. In line 2, an equivalent polynomial $f_{\mathbf{BF}_S}(X)$ associated with $\mathbf{BF}_S$ is constructed, such that $f_{\mathbf{BF}_S}(idx_{s_i,j}) = 1$ and otherwise 0 for all i, j. Lines 3-4 encrypt the BF encoding indices associated with the receiver's set element $c_i \in C$. The for loop in lines 5-6 exploits the OPE primitive to obliviously compute $Enc(f_{\mathbf{BF}_S}(idx_{c_i,j}))$ for all j, where $f_{\mathbf{BF}_S}(idx_{c_i,j})$ is either 1 or 0. Line 7 performs $k-1$ homomorphic multiplications to obtain $Enc(\delta_i)$.

It is worth noting that $Enc(f_{\mathbf{BF}_S}(idx_{c_i,j}))$ computed in line 6 might be too noisy to continue further homomorphic multiplication required in line 7. However, this issue can

```
Input: The sender holding a set $S$ and the reciever holding an element $c_i$ in his
       own set $C$
Output: an encryption of $\delta_i$ where $\delta_i = 1$ if $c_i \in S$ and $\delta_i = 0$ otherwise
1 Sender encodes the sender's set $S$ into a Bloom filter array as $\mathbf{BF}_S$
2 Sender encodes $\mathbf{BF}_S$ into a polynomial $f_{\mathbf{BF}_S}(X)$
3 Client encodes an element $c_i$ in the receiver's set $C$ as BF indices
  $(idx_{c_i,1}, \cdots, idx_{c_i,k})$ where $idx_{c_i,j} \leftarrow Hash_j(c_i)$ for all $j$
4 Client encrypts BF indices assoicated to $c_i$ as $(Enc(idx_{c_i,1}), \cdots, Enc(idx_{c_i,k}))$
5 for $j \leftarrow 1$ up to $k$ do
6     Sender obliviously evaluates $f_{\mathbf{BF}_S}(X)$ at $Enc(idx_{c_i,j})$ to obtain
       $Enc(f_{\mathbf{BF}_S}(idx_{c_i,j}))$
7 Sender obliviously computes $Enc(\prod_{j=1}^{k} f_{\mathbf{BF}_S}(idx_{c_i,j})) \leftarrow \prod_{j=1}^{k} Enc(f_{\mathbf{BF}_S}(idx_{c_i,j}))$
8 return $Enc(\delta_i)$ where $\delta_i \overset{def}{=} \prod_{j=1}^{k} f_{\mathbf{BF}_S}(idx_{c_i,j})$
```

Algorithm 1: Use Bloom filter (BF) encoding to compress long items in the set for computing δ_i in the OPE-based membership test

be easily resolved using the randomize-decrypt-re-encrypt paradigm. In this approach, the sender randomizes $Enc(f_{\mathbf{BF}_S}(idx_{c_i,j}))$ as $Enc(f_{\mathbf{BF}_S}(idx_{c_i,j}) + r)$, where r is a random number in the field $\mathbb{F}_t$, and sends it to the receiver for decryption. Once the receiver decrypts $Enc(f\mathbf{BF}_S(idx_{c_i,j}) + r)$ as $f_{\mathbf{BF}_S}(idx_{c_i,j}) + r$, he re-encrypts $f_{\mathbf{BF}_S}(idx_{c_i,j}) + r$ to send a fresh ciphertext $Enc(f_{\mathbf{BF}_S}(idx_{c_i,j}) + r)$ with minimal noise to the sender. The sender finally recovers $Enc(f_{\mathbf{BF}_S}(idx_{c_i,j})) \leftarrow Enc(f_{\mathbf{BF}_S}(idx_{c_i,j}) + r) - r$.

4.3 Use hashing to accelerate OPE

In this subsection, we discuss the hashing techniques designed to enable SIMD-style computations for the OPE-based membership test in the context of the PSI-cardinality protocol. These techniques are used to reduce computational overhead.

Permutation-based Hashing: Assuming both the receiver and sender each hold a set of size m, they agree to use permutation-based hashing to compress their sets, thereby reducing computation overhead. Specifically, consider an element x represented as a σ_{max}-bit string in the receiver's set, with $x = x_L || x_R$ denoting its left and right parts. To insert (x_L, j) into the $\texttt{Loc}_j(x)$-th bin, the location function $\texttt{Loc}_j(x)$ is defined as follows:

$$\texttt{Loc}_j(x) = H_j(x_L) \oplus x_R$$

Here, $H_j(\cdot)$ hashes the input to an integer in the range of $[0, \log_2 m - 1]$, and three independent hash functions H_0, H_1, H_2 are employed. This hashing technique effectively reduces the string presentation for x from a σ_{max}-bit representation down to $\sigma_{max} - \log_2 m + 2$ bits.

Cuckoo Hashing for Receiver: The receiver employs Cuckoo hashing on his side to insert his set elements into m bins, ensuring that each bin can hold at most one element.

Simple Hashing for Sender: On the sender's side, simple hashing is utilized. Each element $s_i \in S^{sup}$ is hashed to three different locations $\texttt{Loc}_j(s_i)$ for $j = 0, 1, 2$.

To further decrease the computational complexity of evaluating $P_\delta(X)$ for all $r_i \in R$, the classic hash-to-bins technique is employed. Assuming that each element in sets R and S is represented as a σ_{max}-bit integer, and reasonably setting $|S^{(sup)}| = N_{S^{(sup)}} = 2^{\sigma_{max}} - 1 > N_s$ and $|R| = N_r$, the receiver invokes Cuckoo Hashing (with three hash functions h_0, h_1 and h_2) to hash his set elements $\{r_i\}_i$ into approximately $\Omega(N_r)$ bins (with $1.5N_r$ bins prepared in practice to achieve a negligible hash collision probability of 2^{-30}), with each bin having at most one element. The sender invokes simple hashing (with the same hash functions h_0, h_1 and h_2) to hash his set elements $\{s_i^{(sup)}\}_i$ into approximately $\Omega(N_r)$ bins (with the number of bins also set to $1.5N_r$ in practice). Each bin contains up to $\frac{N^{(sup)}}{N_r} + \mathcal{O}(\sqrt{\frac{N^{(sup)} \cdot \log N_r}{N_r}})$ elements with high probability.

As a result, the sender and receiver only need to perform bin-wise PSI for each bin, where the polynomial associated with each sender's bin has a reduced degree of $\frac{N_{s^{(sup)}}}{N_r} + \mathcal{O}(\sqrt{\frac{N_{s^{(sup)}} \cdot \log N_r}{N_r}})$.

An illustrative toy example is provided in Fig. 2, where $|S^{(sup)}| = N_{S^{(sup)}} = 5$, $|C| = n = 5$, and the number of bins is set to 5. The sender uses simple hashing to hash his set elements $x_1, \cdots, x_5$ to 5 bins, and the receiver uses Cuckoo hashing to hash his set elements $y_1, \cdots, y_5$ to 5 bins without hash collision (with a high probability). After hashing, the sender and receiver perform bin-wise comparisons to determine whether y_i belongs to set S.

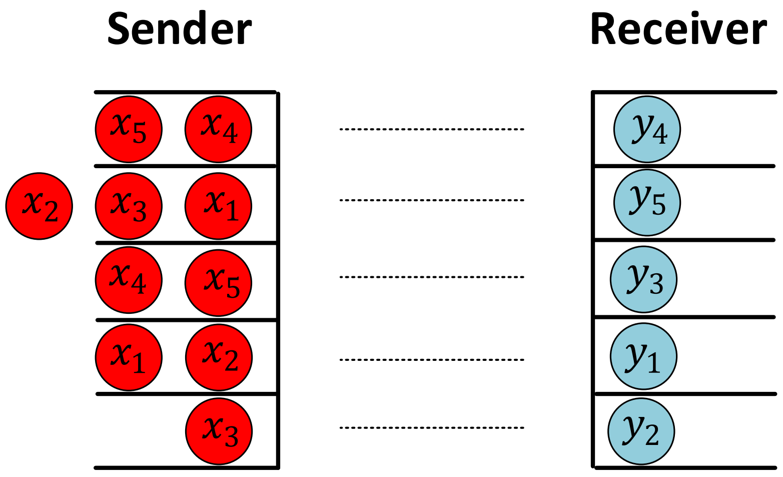

Figure 2: An illustrative example for hash-to-bins technqiue used in the OPE-based memberhip test

4.4 Batching the computation of OPE

To further enhance the computational efficiency while keeping the multiplicative depth low, we propose a method that leverages SIMD encoding to parallelize the computation of OPE. In practice, the elements in the sender's bins are represented by membership test polynomials denoted as $P_{\delta_i}(X)$ for bin No.$i, i \in [N]$.

How this batching optimization works is as follows:

- Receiver Batching: The receiver batches N_r items from N distinct bins into one ciphertext, which is then sent to the sender.
- Sender Batching: The sender batches N coefficients from his membership test polynomials $\{P_{\delta_i}(X)\}_i$, taking one coefficient from each membership test polynomial, into one plaintext. Consequently, $\frac{N_{s^{(sup)}}}{N} + \mathcal{O}(\sqrt{\frac{N_{s^{(sup)}} \cdot \log N}{N}})$ such plaintexts are sufficient for representing the coefficients of N distinct membership test polynomials.
- OPE Computation: The sender performs OPE on the ciphertext sent by the receiver and the $\frac{N_{s^{(sup)}}}{N} + \mathcal{O}(\sqrt{\frac{N_{s^{(sup)}} \cdot \log N}{N}})$ plaintexts from his own side to obtain an encryption of N membership test results.

An illustrative toy example for batching OPE is shown in Fig.3 with the same configuration as in Fig.2. For security reasons, the degree of each membership test polynomial $P_{\delta_i}(X)$ associated with bin No.i on the sender's side must be equal. To achieve this, these bins are inserted with dummy items. For instance, sender's bin No.1 has x_5 and x_4, and a direct Newton interpolation yields a polynomial $P_{\delta_1}(X)$ with degree 2. To equalize the degree, the sender inserts a dummy item, i.e., $X = 0$, and makes $P_{\delta_1}(X) = X \cdot P_{\delta_i}(X)$ with degree 3. This interpolation polynomial construction process is repeated for all bins to create $P_{\delta_i}(X) = \sum_{j=0}^{3} P_{\delta_i,j} X^j$ for all $i \in \{1, 2, 3, 4, 5\}$.

Then, the sender batches $P_{\delta_i,j}$ for all i, i.e., the j-th coefficient of the polynomial $P_{\delta_i}(X)$ for all i, into one plaintext denoted as $pt^{(j)}(X)$. It is evident that 3 such plaintexts are sufficient to batch all the coefficients $P_{\delta_i,j}$ for all i and all j. Finally, the sender combines the ciphertext $Enc(\mathbf{y})$ that encrypts $\mathbf{y} = (y_4, y_5, y_3, y_1, y_2)$ sent by the receiver with his own plaintexts $pt^{(j)}(X)$ for $j \in 1, 2, 3$ to perform an OPE protocol, effectively batching the computation of $\mathbf{P}_\delta(\mathbf{y}) = \{P_{\delta_1}(y_4), \cdots, P_{\delta_5}(y_2)\}$.

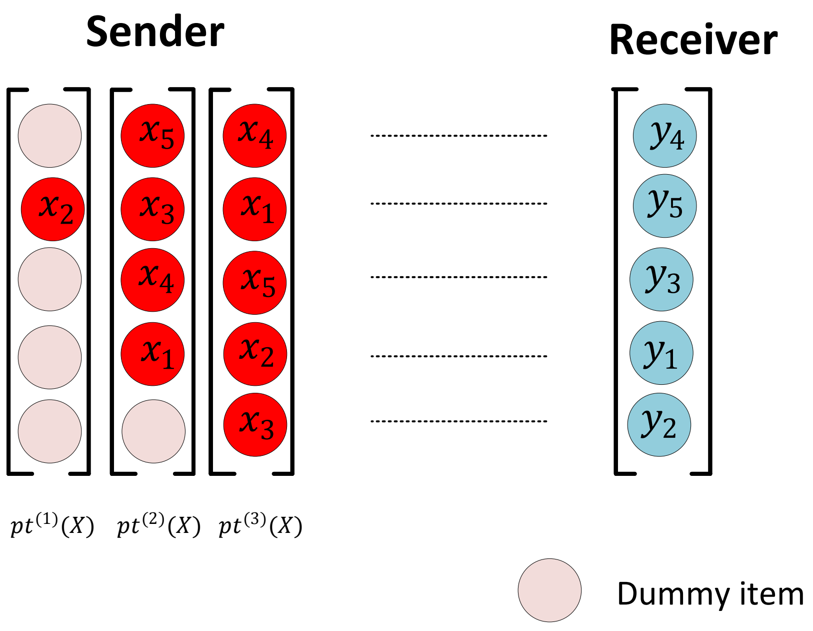

Figure 3: Batching the computation of $P_\delta(X)$ together with the hash-to-bins technique

Finally, we present a quantitative performance analysis of the OPE-based membership test, taking into account all the optimizations we have introduced.

Computation Overhead: To align with FHE parameters, we set the size of the hash table (the number of bins used) equal to the FHE parameter N. In this setup, we utilize three hash functions for simple hashing on the sender's side and implement the Cuckoo hashing scheme for the receiver. Consequently, the maximum load in a bin for the sender/server's set is approximately $B \approx 3 \cdot |S^{(sup)}|/N$ when $|S^{(sup)}|$ is sufficiently large, while for the receiver, each bin holds a maximum of one element. Given that we employ batching, a single OPE operation is sufficient to complete the membership test. The OPE operation is implemented using the Paterson-Stockmeyer algorithm, which precisely requires $L + 2H - \lceil \log_2 L \rceil - \lceil \log_2 H \rceil - 3$ ciphertext-ciphertext (ctx-ctx) multiplications and $B - 1$ plaintext-ciphertext (ptx-ctx) multiplications. To simplify implementations, we set $L = 65$ and $H = \lceil B/L \rceil \approx \frac{3}{65} \cdot \frac{|S^{(sup)}|}{N}$, which suggests the computation overhead only depends on the ratio of $|S^{(sup)}|$ and N.

Noise Growth: In the OPE-based membership test, the noise growth primarily occurs in two stages. The first stage involves preprocessing for $Enc(x^j)$ with $j \in [L]$ and $Enc(x^{iL})$ with $i \in [H]$, derived from $Enc(x), Enc(x^2), \ldots, Enc(x^{2^{\log_2 L}})$ and $Enc(x^L), Enc(x^{2L}), \ldots, Enc(x^{2^{\log_2 H} L})$. This stage consumes $\lceil \log_2 \lceil \log_2(\max(H, L)) \rceil \rceil$ layers of multiplicative depth.

The second stage pertains to the Paterson-Stockmeyer method, where one layer of plaintext-ciphertext (ptx-ctx) multiplication and one layer of ciphertext-ciphertext (ctx-ctx) multiplication are utilized. In summary, the total multiplicative depth employed in the OPE protocol is bounded by $\lceil \log_2 \lceil \log_2(\max(H, L)) \rceil \rceil + 2$.

4.5 Homomorphic Sum

In the previous subsection, we computed a single ciphertext denoted as $Enc(\mathbf{v}) \overset{def}{=} Enc(v_0, \cdots, v_{N-1})$, where each $v_i \in 0, 1$ represents the memership test result for an individual element. However, to obtain the encryption of the sum of these elements, denoted as $Enc(sum(\mathbf{v})) = Enc(\sum_i v_i)$, a homomorphic summation procedure is required.

Method-I with optimal communication complexity There are two methods to implement the homomorphic sum, and we will first introduce the first method, which utilizes the homomorphic cyclic rotation to accumulate all the elements in the encrypted vector $(v_0, \cdots, v_{n-1})$. This method is illustrated in the total sum algorithm (see Alg. 2). It eventually returns $Enc(\sum_{i=0}^{N-1} v_i)$, where $\sum_{i=0}^{N-1} v_i = |R \cap S|$.

```
   Input: RLWE ciphertext $\mathbf{ct}$ which encrypts a vector $\mathbf{v} = (v_0, v_1, \cdots, v_{N-1})$
   Output: RLWE ciphertext $\mathbf{ct}'$ which encrypts a vector $\mathbf{v}' = (\sum_i v_i, \cdots, \sum_i v_i)$
1  $\mathbf{ct}' \leftarrow \mathbf{ct}, e \leftarrow 1$
2  for $j \leftarrow numBits(N) - 2$ down to $0$ do
3      $\mathbf{ct}' \leftarrow \mathbf{ct}' + LeftRotate(\mathbf{ct}', -e)$
4      $e \leftarrow 2 \cdot e$
5      if $bit_j(N) == 1$ then
6          $\mathbf{ct}' \leftarrow \mathbf{ct} + LeftRotate(\mathbf{ct}', -1)$
7          $e \leftarrow e + 1$
8  return $\mathbf{ct}'$
```

Algorithm 2: `Total_Sum` algorithm which accumulates every element in the encrypted vector

The function $numBits(\cdot)$ in step 2 of Alg.2 denotes the number of bits used to represent the input integer. This algorithm is highly efficient in terms of computational complexity. Assuming that N is $2^n - 1$ for some n, which is exactly the worst case since the if condition (step 10 to step 12) is always performed. In this case, it is readily seen that the number of homomorphic rotations required in Alg.2 is $2(\log N - 1) = \Omega(\log N)$. This means that the algorithm has a computational complexity proportional to the logarithm of the input size, which is a favorable characteristic for efficient execution.

However, one challenge with the `Total_Sum` algorithm described in Alg. 2 is that it can require a large amount of storage for rotation keys. Specifically, it needs $log_2 N$ rotation keys to perform each rotation mentioned in step 3.

To address this storage issue, a more storage-efficient version of `Total_Sum` is presented in Alg. 3. The main idea behind this improvement is to factor the right rotation by e slots into a combination of at most k types of right rotations. In this approach, we choose k rotation keys, corresponding to right rotations by 2^{ℓ_i} slots for $i \in [k]$, in order to save storage for rotation keys. A natural choice for the values of $\{\ell_i\}_{i \in [k]}$ is to evenly divide the interval $[0, 1, \cdots, \log_2 N]$.

For the purpose of analysis, let's assume that N is $2^n - 1$ for some n, which represents the worst-case scenario, as e starts with 1_2 and grows to $11_2, 111_2$, and so on. For e falling in the interval i with $[2^{i\cdot \log N/k}, 2^{(i+1)\cdot \log N/k} - 1]$ and $i \in [k]$, the number of cyclic rotations is given by:

$$(2^{\frac{logN}{k}} - 1)i + \sum_{j=1}^{\frac{logN}{k}} (2^j - 1) = (2^{\frac{logN}{k}} - 1)i + 2^{\frac{logN}{k}+1} - \frac{logN}{k} - 2$$

Therefore, the total number of cyclic rotations in the worst case is:

$$(2^{\frac{logN}{k}-1} \cdot \frac{k-1}{2} \cdot k) + 2^{\frac{logN}{k}+1} \cdot k - (\frac{logN}{k} + 2)k = \Omega(k^2 2^{\frac{logN}{k}})$$

This indicates that the total number of rotations required is quadratic in k and exponential in $\log N/k$. The storage-efficient `Total_Sum` algorithm significantly reduces the number of required rotation keys while still achieving the desired functionality.

```
   Input: RLWE ciphertext $\mathbf{ct}$ which encrypts a vector $\mathbf{v} = (v_0, v_1, \cdots, v_{N-1})$
   Output: RLWE ciphertext $\mathbf{ct}'$ which encrypts a vector $\mathbf{v}' = (\sum_i v_i, \cdots, \sum_i v_i)$
1  $\mathbf{ct}' \leftarrow \mathbf{ct}, e \leftarrow 1$
2  for $j \leftarrow numBits(N) - 2$ down to $0$ do
3      rewrite $e$ w.r.t the base $\{2^{\ell_1}, \cdots, 2^{\ell_k}\}$ such that $e = \sum_i e_i \cdot 2^{\ell_i}$
4      $\mathbf{ct}'' \leftarrow \mathbf{ct}'$
5      for $i \leftarrow k$ down to $1$ do
6          for $j \leftarrow e_i - 1$ down to $0$ do
7              $\mathbf{ct}'' \leftarrow LeftRotate(\mathbf{ct}'', -2^{\ell_i})$
8      $\mathbf{ct}' \leftarrow \mathbf{ct}' + \mathbf{ct}''$
9      $e \leftarrow 2 \cdot e$
10     if $bit_j(N) == 1$ then
11         $\mathbf{ct}' \leftarrow \mathbf{ct} + LeftRotate(\mathbf{ct}', -1)$
12         $e \leftarrow e + 1$
13 return $\mathbf{ct}'$
```

Algorithm 3: `Total_Sum` algorithm (storage efficient version) which accumulates every element in the encrypted vector

Method-II with balanced performance In the previous section, we introduced Method-I for homomorphic accumulation of the encrypted vector, which extensively utilizes cyclic rotations but lacks computational efficiency due to its heavy reliance on rotations. Method-I also avoids interaction between the sender and receiver entirely. However, to address this computational inefficiency, we introduce a more practical Method-II, as illustrated in Fig. 4, which reduces computational overhead through a single round of light communication between the sender and receiver.

In Fig. 4, the sender randomizes the encryption of a sensitive and secret vector $\mathbf{v} \overset{def}{=} (v_0, \cdots, v_{N-1})$ by adding a random vector $\mathbf{r}$ to it, creating a new ciphertext $Enc(\mathbf{v}+\mathbf{r})$. It's important to note that the noise term of the input ciphertext $Enc(\mathbf{v})$ may contain additional sensitive information from the sender's set. There exists two approaches to handle this privacy issue, either by OPRF preprocessing [CMdG+21] or noise flooding [AJLA+12]. In this work, we choose to use noise flooding to obscure this undesirable leakage before sending it to the receiver since this technqiue is slightly more computationally efficient. Upon receiving $Enc(\mathbf{v}+\mathbf{r})$, the receiver locally decrypts it to $\mathbf{v}+\mathbf{r}$ and calculates the sum of all elements in the vector $\mathbf{v}+\mathbf{r}$ to obtain $sum(\mathbf{v}+\mathbf{r})$. The receiver then re-encrypts

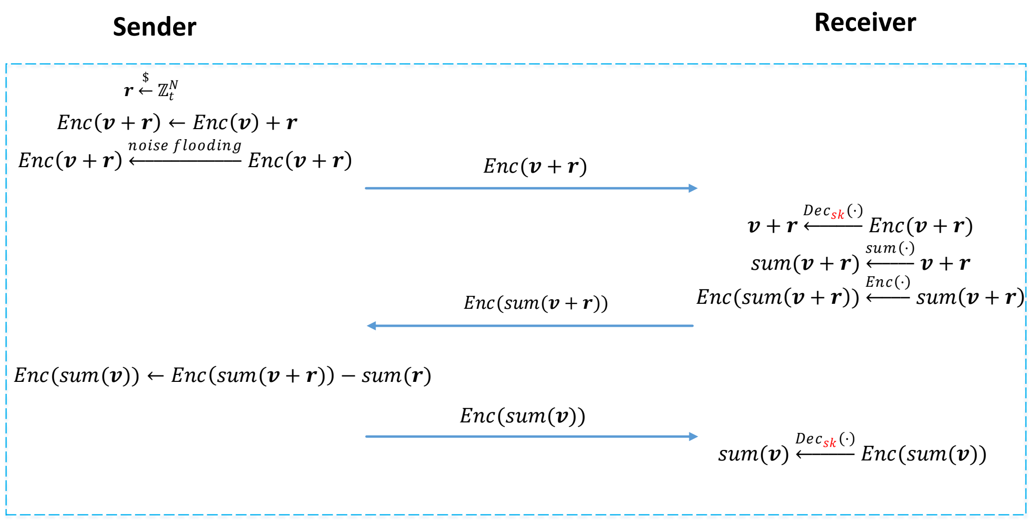
Sender
Receiver
$r \xleftarrow{\$} \mathbb{Z}_t^N$
$Enc(v+r) \leftarrow Enc(v) + r$
$Enc(v+r) \xleftarrow{noise\ flooding} Enc(v+r)$
$Enc(v+r)$
$v + r \xleftarrow{Dec_{sk}(\cdot)} Enc(v+r)$
$sum(v+r) \xleftarrow{sum(\cdot)} v + r$
$Enc(sum(v+r)) \xleftarrow{Enc(\cdot)} sum(v+r)$
$Enc(sum(v+r))$
$Enc(sum(v)) \leftarrow Enc(sum(v+r)) - sum(r)$
$Enc(sum(v))$
$sum(v) \xleftarrow{Dec_{sk}(\cdot)} Enc(sum(v))$

Figure 4: A fast homomorphic accumulation for all elements in an encrypted vector by interaction

$sum(\mathbf{v}+\mathbf{r})$ and sends this ciphertext back to the sender. Finally, the sender de-randomizes $Enc(sum(\mathbf{v}+\mathbf{r}))$ by subtracting $\mathbf{r}$ to obtain $Enc(sum(\mathbf{v}))$ without gaining knowledge of the exact $sum(\mathbf{v})$.

In terms of performance, the sender performs two homomorphic additions and one noise flooding operation, while the receiver executes one encryption, one decryption, and one sum of all elements in the plaintext. The computational overhead for these operations is almost negligible when compared to the homomorphic cyclic rotations used in Method-I. The cost of Method-II is an additional round of communication, including two BFV ciphertexts, which is approximately a few megabytes in size. This trade-off improves computational efficiency while maintaining security and privacy.

4.6 Performance analysis

In summary, the proposed `PSI-cardinality` protocol, with all the optimizations incorporated, exhibits reasonable computational and communication overheads. The computational cost is primarily determined by the number of ciphertext-ciphertext multiplications used in the membership test, which is approximately $L + 2H - \lceil \log_2 L \rceil - \lceil \log_2 H \rceil - 3$ with specific values of $L = 65$ and $H \approx \lceil \frac{3|S^{(sup)}|}{L \cdot N} \rceil$. On the other hand, the communication overhead is mainly influenced by the number of ciphertexts exchanged during the protocol. The membership test requires $\lceil \log_2 L \rceil + \lceil \log_2 H \rceil$ ciphertexts, while the homomorphic sum involves 3 ciphertexts for interaction. Consequently, the overall communication overhead is on the order of $(\lceil \log_2 L \rceil + \lceil \log_2 H \rceil + 3) \cdot 2 \cdot \log_q \cdot N$ bits, which exhibits logarithmic scaling with respect to the size of the universe set $|S^{sup}|$.

4.7 Security analysis

We use the following theorem to argue that our new `PSI-cardinality` protocol is secure against semi-honest adversaries:

Theorem 1. *Assuming the existence of CPA-secure fully homomorphic encryption scheme, semi-honest secure private set intersection cardinality protocol and semi-honest seucre standard PSI protocol; then the protocol in Fig. 1 is secure in the presence of semi-honest adversaries.*

Proof. Assume that P_1 who plays as the sender/server is corrupted. We construct the simulator $\mathcal{S}_1$ to simulate P_1's view in both OPE-based membership test and homomorphic accumulation as follows. In the OPE-based membership, P_1's view includes $Enc(\{r_i\}_i)$ for all i and thus can be simulated by a random message whose length is as the same as $Enc(\{r_i\}_i)$. In the homomorphic accumulation, P_1 sees $Enc(\mathbf{v}+\mathbf{r})$ which is obviously indistinguishable from a random message generated by $\mathcal{S}_1$. To conclude, it is possible for the simulator $\mathcal{S}_1$ to generate a view which is computationally indistinguishable from P_1's view.

Assume that P_2 who plays as the receiver/client is corrupted. We construct the simulator $\mathcal{S}_2$ to simulate P_2's view in both OPE-based membership test and homomorphic accumulation as follows. In the OPE-based membership, P_2 sees nothing and therefore no need to simulate. In the homomorphic accumulation, P_2 sees $Enc(\mathbf{v}+\mathbf{r})$, $\mathbf{v}+\mathbf{r}$ and $Enc(sum(\mathbf{v}))$. $Enc(\mathbf{v}+\mathbf{r})$ can be simulated by an encryption of a random message since the CPA security of FHE holds and $\mathbf{v}+\mathbf{r}$ is a random vector. $\mathcal{S}_2$ knows the final result $sum(\mathbf{v})$ and therefore it is natural for him to simulate $Enc(sum(\mathbf{v}))$ by encrypting $sum(\mathbf{v})$. In both cases, the simulators are capable of generating views that are computationally indistinguishable from the actual views of the corrupted players, ensuring semi-honest security of the protocol. □

Discussion In the proof, the circuit privacy property of fully homomorphic encryption schemes must be fulfiled. In the `secretTokenGen` subprotocol, the sender/server may interact with the receiver/client for reducing the noise in the ciphertexts. However, the noise term is related to the function that the FHE computes and the receiver/client can extract the noise term for learning information leakage in the ciphertext by decrypting it locally. To fix this circuit privacy issue, we apply the noise flooding technique, where the sender re-randomizes the output ciphertext $\mathbf{c}$ by homomorphically adding an encryption of zero with a large noise term [Gen09]. Specifically, if the error component in the output ciphertext is bounded by B, denoted as $|e| < B$, then the noise term in $Enc(0)$ is bounded by $\Omega(2^{\lambda}B)$ [AJLA+12]. By adding this large noise term, denoted as $\mathbf{c}' = \mathbf{c} + Enc(0)$, a statistical distance of $2^{-\lambda}$ (negligible) is achieved between $\mathbf{c}$ and $\mathbf{c}'$.

5 The proposed Private Set Sum with Cardinality Protocol

In this section, we present our new protocol for private set sum with cardinality, denoted as `PSI-sum-cardinality`. This protocol securely computes the sum of the payload values associated with the intersected set elements and also computes the size of the set intersection using BFV Fully Homomorphic Encryption. Unlike the previous `PSI-cardinality` protocol, which only computes the intersection cardinality, `PSI-sum-cardinality` extends its functionality to sum the payload values associated with the intersected elements.

5.1 Tweaking `PSI-cardinality` for `PSI-sum-cardinality`

We build upon the foundation of `PSI-cardinality` described in the previous section. In `PSI-sum-cardinality`, we approach the Newton interpolation in the membership test from a different angle. Each set element $s_i \in S^{sup} \cap S$ is now associated with a default payload value of '1' and '0' for elements that are not in the intersection. To compute the sum of all payload values v_i associated with the intersection elements s_i (i.e., $s_i \in R \cap S$), we modify the default payload value '1' to v_i when constructing the membership polynomial $P_\delta(X)$. Now, when we execute the `PSI-cardinality`, it securely computes the sum of the payload values.

Protocol Execution The `PSI-sum-cardinality` protocol can be divided into two executions, as depicted in Fig. 5:

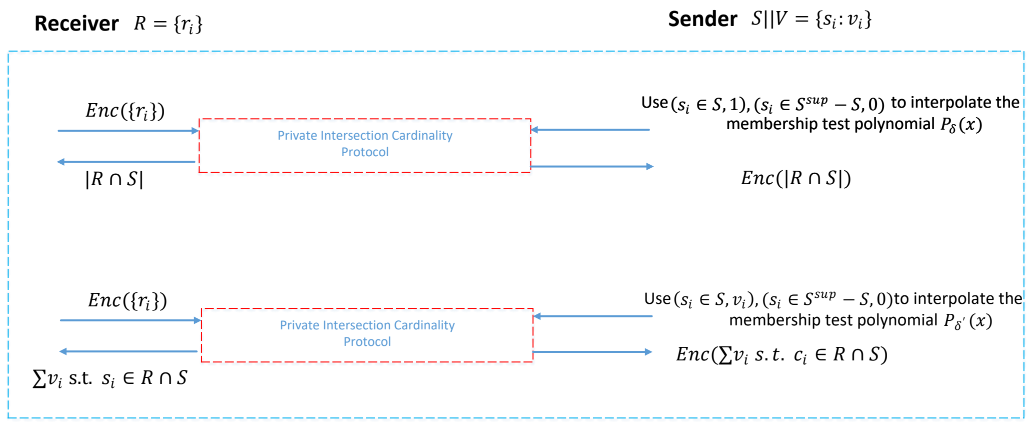
Receiver $R = \{r_i\}$
Sender $S||V = \{s_i : v_i\}$
$Enc(\{r_i\})$
Private Intersection Cardinality Protocol
Use $(s_i \in S, 1), (s_i \in S^{sup} - S, 0)$ to interpolate the membership test polynomial $P_\delta(x)$
$|R \cap S|$
$Enc(|R \cap S|)$
$Enc(\{r_i\})$
Private Intersection Cardinality Protocol
Use $(s_i \in S, v_i), (s_i \in S^{sup} - S, 0)$ to interpolate the membership test polynomial $P_{\delta'}(x)$
$Enc(\sum v_i \; s.t. \; c_i \in R \cap S)$
$\sum v_i$ s.t. $s_i \in R \cap S$

Figure 5: Private set intersection sum with cardinality (`PSI-sum-cardinaity`) protocol

1. In the first execution, the protocol computes the intersection cardinality using the `PSI-cardinality` protocol, as described in the previous section.
2. In the second execution, the membership test polynomial $P_{\delta'}(x)$ is modified such that $P_{\delta'}(s_i \in R \cap S) = v_i$ for elements in the intersection and $P_{\delta'}(s_i \notin R \cap S) = 0$ for others. This modified polynomial is then used to securely compute the sum $\sum_i v_i$ of all payload values associated with the intersection elements.

5.2 Performance Analysis

The computational overhead in `PSI-sum-cardinality` is roughly doubled compared to `PSI-cardinality`, as it involves two executions of the protocol. However, it's important to note that the computational complexity remains asymptotically proportional to the square root of $|S^{sup}|$. On the other hand, the communication overhead remains largely unchanged ($\lceil \log_2(L) \rceil + \lceil \log_2(H) \rceil + 6$ BFV ciphertexts in total, with $L = 25, H = \lceil \frac{3|S^{(sup)}|}{N \cdot L} \rceil$).

In summary, `PSI-sum-cardinality` extends the functionality of `PSI-cardinality` by securely computing the sum of payload values associated with intersected elements. While this enhanced functionality comes with an increased computational overhead, it remains affordable, especially when considering the security and privacy benefits it provides. Importantly, the communication overhead in `PSI-sum-cardinality` remains at a similar level to that of `PSI-cardinality`, making it a practical choice for secure set intersection and sum operations.

5.3 Security analysis

We use the following theorem to argue that our new `PSI-sum-cardinality` protocol is secure against semi-honest adversaries:

Theorem 2. *Assuming the existence of semi-honest secure private set intersection cardinality protocol, then the protocol in Fig. 5 is secure in the presence of semi-honest adversaries.*

Proof. The proof is obvious. The first execution of `PSI-cardinality` protocol guarantees the receiver/client learns nothing but the intersection size $|R \cap S|$. The second execution of `PSI-cardinality` protocol guarantees the receiver/client learns nothing but the sum of payload values associated with the intersection, *i.e.*, $\sum_i v_i$ such that $s_i \in R \cap S$. □

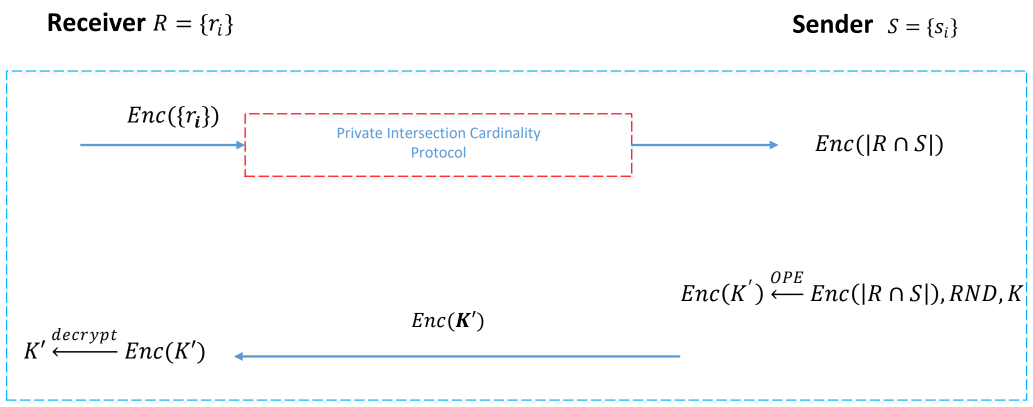
Receiver $R = \{r_i\}$
Sender $S = \{s_i\}$
$Enc(\{r_i\})$
Private Intersection Cardinality Protocol
$Enc(|R \cap S|)$
$Enc(K') \overset{OPE}{\longleftarrow} Enc(|R \cap S|), RND, K$
$Enc(\boldsymbol{K'})$
$K' \overset{decrypt}{\longleftarrow} Enc(K')$

Figure 6: Private set token with threshold on intersection cardinality(`PSI-token-threshold`) protocol

6 The Proposed Private Set Token with a Threshold on Intersection Cardinality

In this section, we introduce our new protocol for private set token with a threshold on cardinality, referred to as `PSI-token-threshold`. This protocol securely determines whether the intersection cardinality satisfies a given condition without revealing any information about the sets involved, using BFV Fully Homomorphic Encryption.

`PSI-token-threshold` can be built upon the `PSI-cardinality` protocol, as illustrated in Fig 6. An additional homomorphic binary evaluation step is required to indicate whether the intersection size meets the specified condition. To be more specific, the challenge here is how to homomorphically evaluate the following threshold function:

$$f_{th}(|R \cap S|) = \begin{cases} K_0 & \text{if } |R \cap S| \text{ satisfies threshold} \\ K_1 & \text{Otherwise} \end{cases}$$

We achieve this by using Newton interpolation to construct $f_{th}(X)$, where the interpolation points are $(x \in \text{threshold}, K_0)$ and $(x \notin \text{threshold}, K_1)$. We then homomorphically evaluate the Newton polynomial, which returns an encryption of K' with either $K' = K_0$ or $K' = K_1$. The challenge lies in the fact that the initial noise in the ciphertext $Enc(|R \cap S|)$ is too large to perform a meaningful evaluation of $f_{th}(X)$. While bootstrapping is a theoretical approach to reduce the noise level, in practice, we aim to avoid using this method.

6.1 Homomorphic Comparison by Reusing OPE

We propose a method to efficiently reduce the noise in the ciphertext $Enc(|R \cap S|)$ while preserving the security of the protocol. The basic idea involves a receiver/client-assisted bootstrapping process in which the sender/server resends the ciphertext $Enc(|R \cap S|)$ to the receiver/client. The receiver/client can then decrypt it and re-encrypt $|R \cap S|$ to create a new ciphertext with minimal noise. However, the receiver/client would learn the exact value of $|R \cap S|$ during this process, which is not permissible for security reasons. To address this, the sender/server randomizes the ciphertext before sending it back to the receiver/client. This ensures that the receiver/client receives a randomized message that leaks no information. Concretely, the sender performs $Enc(|R \cap S| + r') = Enc(|R \cap S|) + (0, r')$ and sends it to the receiver. The receiver re-encrypts it to create a refreshed ciphertext $Enc(|R \cap S| + r')$. Finally, the sender de-randomizes $Enc(|R \cap S| + r')$ by performing $Enc(|R \cap S|) = Enc(|R \cap S| + r') - (0, r')$.

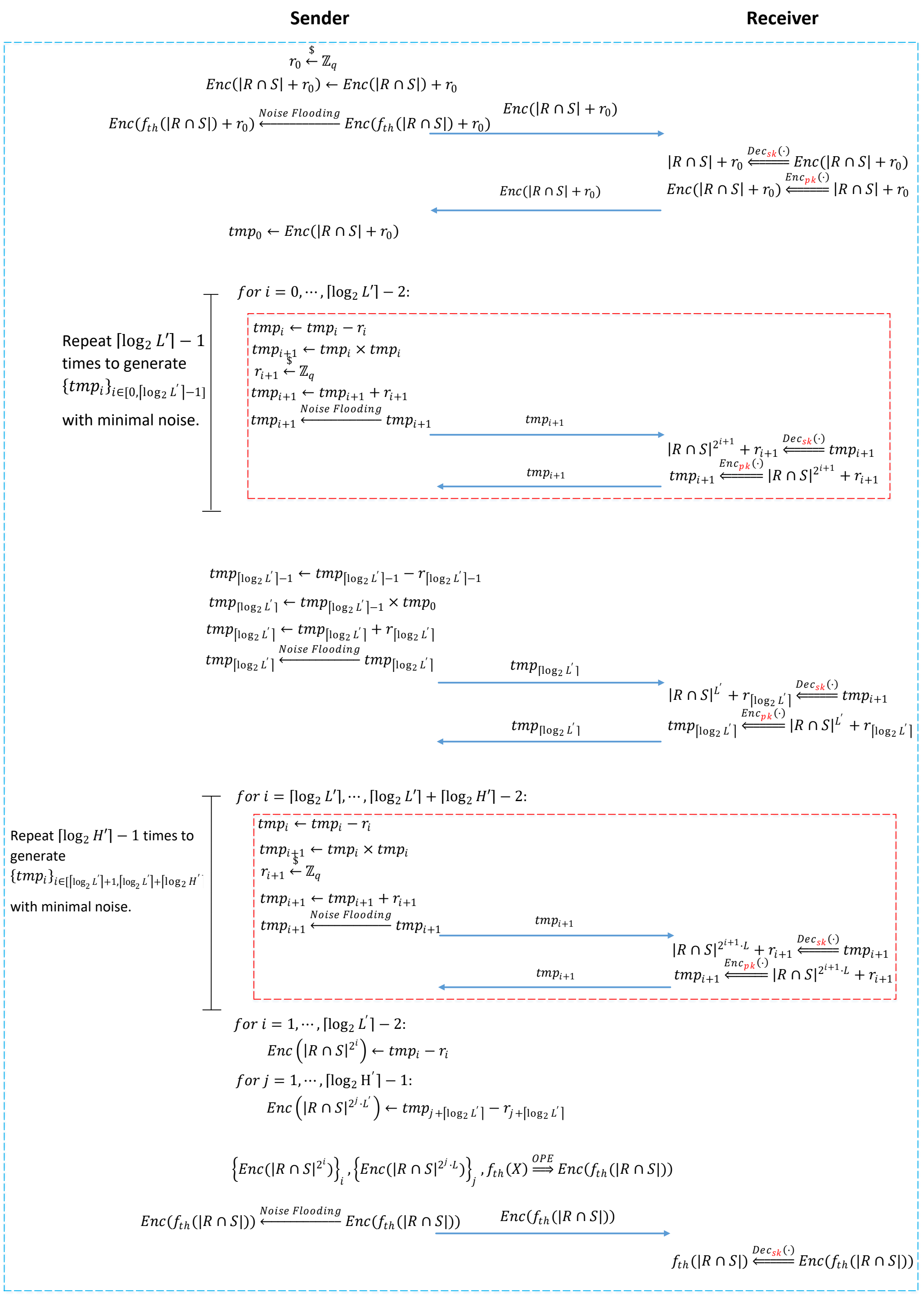
Sender
Receiver
Repeat ⌈log₂ L′⌉ − 1 times to generate {tmp_i} with minimal noise.
Repeat ⌈log₂ H′⌉ − 1 times to generate {tmp_i} with minimal noise.
Noise Flooding
OPE

Figure 7: Homomorphic comparison used in the `PSI-cardinaity-threshold` protocol

To ensure that the noise reduction is effective, the server must refresh not only $Enc(|R \cap S|)$ but also $Enc(|R \cap S|)^{2^m}$ for $m \in [\lceil \log_2 L' \rceil]$ and $Enc(|R \cap S|)^{2^n L'}$ for $n \in$

$[\lceil \log_2 H' \rceil]$. This is necessary for compatibility with Paterson's algorithm and to minimize the multiplicative depth. Fig. 7 provides a detailed description of the homomorphic comparison sub-protocol, where the server inputs an RLWE encryption of the intersection cardinality, i.e., $Enc(|R \cap S|)$, and outputs an RLWE encryption of $f_{th}(|R \cap S|)$ after $\lceil \log_2 L' \rceil + \lceil \log_2 H' \rceil$ rounds of interactions.

In summary, this client-assisted bootstrapping technique allows for noise reduction in the ciphertext $Enc(|R \cap S|)$, and it is essential for the efficient execution of the PSI-caPrdinality-threshold protocol while preserving security.

6.2 Performance analysis

We analyze quntatively the timing performance and the comunication cost of the proposed `PSI-token-threshold` protocol. Compared with `PSI-cardinality` protocol, the extra computational overhead mainly comes from the number of ciphertext-ciphertext multiplications used in the OPE-based comparsion, which is $L' + 2H' - 4$ with $L' = 65, H' = \lceil \frac{|R|}{L'} \rceil$. Therefore, the total computational overhead is $L + 2H - \lceil \log_2 L \rceil - \lceil \log_2 H \rceil + L' + 2H' - 7$ with $L = 65, H = \lceil \frac{3|S^{sup}|}{N \cdot L} \rceil$. This amount is essentially aymptotically logrithmic in $\sqrt{|S^{(sup)}|/N} + \sqrt{|R|}$.

As for the communication cost, the OPE-based comparison introduces $2\lceil \log_2 L' \rceil + 2\lceil \log_2 H' \rceil$ ciphertexts. Therefore, the total communication overhead is $(\lceil \log_2 L \rceil + \lceil \log_2 H \rceil + 2\lceil \log_2 L' \rceil + 2\lceil \log_2 H' \rceil + 4) \cdot 2 \cdot \log_2 q \cdot N$ bits.

6.3 Security analysis

We use the following theorem to argue that our new `PSI-token-threshold` protocol is secure against semi-honest adversaries:

Theorem 3. *Assuming the existence of CPA-secure fully homomorphic encryption scheme, semi-honst secure private set intersection cardinality protocol and semi-honest seucre standard PSI protocol; then the protocol in Fig. 6 is secure in the presence of semi-honest adversaries.*

Proof. We first prove that the OPE-based homomorphic comparsion as shown in Fig. 7 is secure and later prove `PSI-token-threshold` is secure based on it.

Assume that P_1 plays the role of sender/server and is corrupted. The simulator $\mathcal{S}_1$ simulating the view of P_1 is constructed as follows: $\mathcal{S}_1$ randomly generates $\lceil \log_2 L' \rceil + \lceil \log_2 H' \rceil$ messages with the same length as FHE ciphertext. These messages are indistinguishable from tmp_i for $i = 0, \cdots, \lceil \log_2 L' \rceil + \lceil \log_2 H' \rceil - 1$ in P_1's view.

Assume that P_2 plays the role of receiver/client and is corrupted. The simulator $\mathcal{S}_2$ simulating the view of P_2 is constructed as follows: $\mathcal{S}_2$ randomly generates $\lceil \log_2 L' \rceil + \lceil \log_2 H' \rceil$ messages with the same length as FHE ciphertext. Decryption of these random messages are random integers indistinguishable from the masked $|R \cap S|^{2^i}$ and $|R \cap S|^{2^j \cdot L}$ computed on P_1. $\mathcal{S}_2$ certainly knows the final result $f_{th}(|R \cap S|)$ and therefore the encryption of $f_{th}(|R \cap S|)$ is indistinguishable from $Enc(f_{th}(|R \cap S|))$ sent by the sender. This completes the security proof for the OPE-based homomorphic comparsion as shown in Fig. 7.

Next we proceed to prove that the main protocol `PSI-token-threshold` is secure. Assume that P_1 plays the role of sender/server and is corrupted. Since `PSI-cardinality` and the OPE-based homomorphic comparison are secure, the sender's view can be fully regenerated by $\mathcal{S}_1$. The same argument goes to the corrupted P_2 who plays as the receiver/client. □

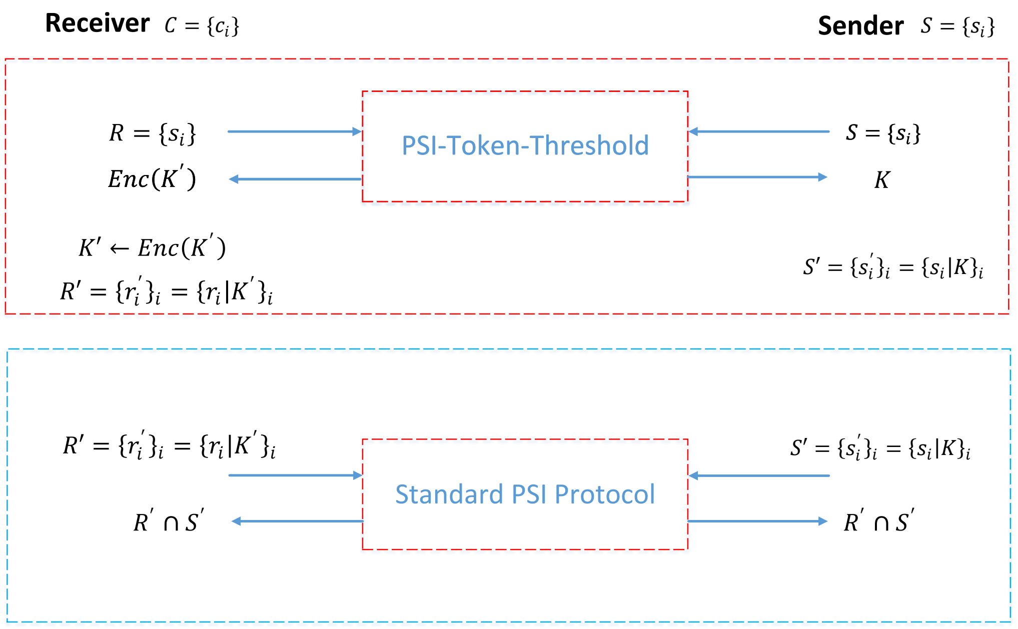

Figure 8: An abstract framework for our threshold PSI protocol

7 The Proposed Threshold Private Set Intersection Protocol

In this section, we introduce our construction for threshold PSI, denoted as `tPSI`, at an abstract level. Threshold PSI encompasses different types of thresholds: below-threshold (returning the set intersection if the cardinality is below a certain value), above-threshold (returning the intersection if the cardinality exceeds a threshold), and in-between-threshold (returning the intersection if the cardinality falls within a specified range). The setup involves a receiver/client with a set R and a sender/server with a set S. The `tPSI` protocol is built on two subprotocols: the private set intersection token with threshold and the standard PSI protocol as shown in Fig. 8.

In the private set intersection token with threshold on cardinality subprotocol, known as `PSI-token-threshold`, two secret tokens, K' and K, are generated by the sender/server based on the encrypted intersection cardinality, $Enc(|R \cap S|)$. The crucial point here is that $K' = K$ if the intersection cardinality satisfies the threshold criterion; otherwise, K' is a random token. The receiver/client decrypts the encryption of K' and appends it to every item in their set R, resulting in the updated set R'. Simultaneously, the sender/server updates their set S to S', with each item augmented by K.

In the standard PSI protocol, the receiver/client inputs their updated set R', and the sender/server inputs their updated set S'. The protocol then proceeds like a typical PSI protocol. At the protocol's conclusion, the receiver/client learns no more than the intersection $R \cap S$ if the threshold criterion is met. Otherwise, the receiver/client, with overwhelming probability, receives an empty set.

7.1 Challenges in the proposed threshold PSI

One of the main challenges in the proposed threshold PSI (tPSI) protocol is the need to homomorphically compare the intersection size $|R \cap S|$ with a given threshold t and generate a secret token based on this comparison result. We denote this variant of

`PSI-token-threshold` as `SecretTokenGen`. Specifically, the challenge is to compute the function:

$$f_{th}(|R \cap S|) = \begin{cases} K & \text{if } |R \cap S| \text{ satisfies threshold} \\ RND & \text{Otherwise} \end{cases}$$

Here, K is generated by the sender/server to update their own set, while RND is a completely random token. The difficulty lies in ensuring low multiplicative depth for the underlying OPE-based comparison. This is made even more challenging because the input, i.e., the encryption of $|R \cap S|$, contains a relatively high level of noise. The noise budget available for computing f_{th} is limited, adding to the complexity of the problem.

7.2 Homomorphic Secret Token Generation

In the `SecretTokenGen` subprotocol, we apply OPE to perform a specific homomorphic comparison that aims to generate a secret token based on whether the input satisfies a given threshold. This is different from the normal comparison used in the PSI-token-threshold protocol. Here, we want to determine if a value is smaller or larger than a threshold and produce a corresponding secret token.

For instance, in the above-threshold criteria, where we output a valid secret token K if and only if the cardinality is above a certain threshold t, the interpolation polynomial is constructed in the following form:

$$f_{th}(X) = r \cdot \prod_{i=t}^{|R|} (X - i) + K$$

Here, r is a random number, and K is the target secret token. The multiplicative depth for computing $f_{th}(X)$ is $\log\log(N - t + 1)$.

To make this functionality more general for any discrete threshold criteria, where it outputs a valid secret token K if and only if the cardinality satisfies some given threshold t, the polynomial can be constructed as follows:

$$f_{th}(X) = r \cdot \prod_{i \in \textbf{threshold}} (X - i) + K$$

Analogous to the method used in PSI-token-threshold, the receiver/client and the sender/server interact to generate the encryption of $|R \cap S|^m$ for $m \in [\log_2 L]$ and $|R \cap S|^{2^n L}$ for $n \in [\log_2 H]$.

7.3 Performance analysis

The computation overhead for the tPSI protocol, which includes homomorphic secret token generation, is primarily determined by the number of ciphertext-ciphertext homomorphic multiplications used. If we assume that the degree of the polynomial $f_t h(X)$ is $\varepsilon|R|$, where $0 < \varepsilon < 1$, and we apply a combination of the windowing method and the Paterson-Stockmeyer algorithm, then the number of ctx-ctx multiplications is reduced to $L' + 2H' - 3$ where $L' = 65, H' = \lceil \frac{\varepsilon|R|}{L'} \rceil$.

Assuming we instantiate the standard PSI protocol from FHE-based constructions, the overall computational overhead for the `tPSI` protocol is: $L + 2H - \lceil \log_2 L \rceil - \lceil \log_2 H \rceil - 3 + L' + 2H' - 3 + L'' + 2H'' - \lceil \log_2 L'' \rceil - \lceil \log_2 H'' \rceil - 3$ where $L = 65, H = \lceil \frac{3|S^{sup}|}{NL} \rceil, L'' = 65, H'' = \lceil \frac{3|S|}{NL} \rceil$. This computational overhead is asymptotically on the order of $\sqrt{|S^{sup}|} + \sqrt{|S|} + \sqrt{\varepsilon|R|}$.

For the communication overhead, the homomorphic secret token generation requires $\lceil\log_2(L)\rceil+\lceil\log_2(H)\rceil+2\lceil\log_2(L')\rceil+2\lceil\log_2(H')\rceil+3$ ciphertexts. The standard PSI requires $\lceil\log_2(L'')\rceil+\lceil\log_2(H'')\rceil+1$ ciphertexts. Therefore, the total communication overhead is $(\lceil\log_2(L)\rceil+\lceil\log_2(H)\rceil+2\lceil\log_2(L')\rceil+2\lceil\log_2(H')\rceil+\lceil\log_2(L'')\rceil+\lceil\log_2(H'')\rceil+4)\cdot 2\cdot\log_2 q\cdot N$ bits.

7.4 Security analysis

We use the following theorem to argue that our new threshold PSI protocol is secure against semi-honest adversaries:

Theorem 4. *Assuming the existence of CPA-secure fully homomorphic encryption scheme, semi-honst secure private set intersection cardinality with threshold protocol and semi-honest seucre standard PSI protocol; then the protocol in Fig.8 is secure in the presence of semi-honest adversaries.*

Proof. Let P_1 play as the receiver/client, and P_2 play as the sender/server. Since we use a standard PSI protocol secure against semi-honest adversaries, it suffices to prove that the threshold token generation protocol is semi-honest secure. First, **we consider the case that P_2 is corrupted**. Thus, we merely need to show how to simulate the view of the incoming messages received by P_2 by constructing a simulator called $\mathcal{S}_2$. This part is easy because P_2's view contains only FHE encryptions of the receiver/client's set, *i.e.*, $Enc(\{r_i\}_i)\overset{def}{=}Enc(r_i)$ for all $r_i\in R$. $\mathcal{S}_2$ chooses a uniformly distributed message whose length equals to that of the ciphertext $Enc(\{r_i\}_i)$. This simulated message cannot be distinguished from $Enc(\{r_i\}_i)$ in the real model by a probabilistic polynomial-time-bounded adversaries due to the CPA-security of FHE ciphertexts.

Next, **we proceed to the case that P_1 is corrupted**, and construct a simulator $\mathcal{S}_1$. The view of P_1 merely contains an FHE ciphertext $Enc(K')$ sent by P_2. $Enc(K')$ is the output of the `PSI-token-threshold` protocol performed on P_2, and thus is still CPA secure. Note that in this case, P_1 outputs the secret token K' which is either the actual secret token K or a random value RND depending on whether or not the threshold condition is fulfilled. Either way, $\mathcal{S}_1$ uses P_1's FHE secret key to encrypt K' and thus this new FHE ciphertext is indistinguishable from the incoming $Enc(K')$ from P_2 in the real model. □

8 Experimental Results and Comparisons

We have implemented the `PSI-cardinality`, `PSI-token-threshold`, and `tPSI` protocols in C++, utilizing the SEAL [Res21b] and APSI [Res21a] libraries. Initially, we evaluated the performance of the `tPSI` protocol by varying the size of the sender's set while keeping the receiver's set relatively small as shown in Table. 1. The results demonstrate that the online runtime has been significantly reduced, ranging from 10.2 to 98.1 seconds, which appears to be a favorable outcome for practical real-world applications. However, it's noteworthy that the offline phase experiences notable growth due to our utilization of Newton interpolation for constructing the membership test polynomial, a process with a time complexity of $\Omega(|S^{sup}|^2)$. It's important to emphasize that this offline phase is a one-time computation performed on the sender's side, and its amortized cost could be negligible when the `tPSI` protocol is executed across multiple clients and scenarios. Moreover, we observed that the communication overhead exhibited gradual growth, ranging from 16 MB to 22 MB, as the size of the sender's set increased. This trend underscores the advantage of our FHE-based design, which keeps communication costs relatively low even with larger sets.

Table 1: Details of Computation and communication cost of our `tPSI` implementation

$\|S^{sup}\|$	$\|S\|$	$\|R\|$	sender offline			sender online					comm.(MB)	
			total	`interpolate_set`	`cmp_poly`	total	`membership_test`	`homo_sum`	`homo_cmp`	`standard_PSI`	$R \to S$	$S \to R$
2^{16}	2^{12}	2^{12}	5.4s	0.3s	5.1s	10.2s	0.5s	0.3s	6.4s	0.7s	8.77	7.40
2^{20}	2^{16}		6.8s	1.6s	5.2s	13.3s	1.7s	0.3s	6.5s	2.0s	9.77	7.40
2^{24}	2^{20}		43.3s	40.0s	3.3s	22.0s	7.1s	0.4s	6.7s	4.7s	12.70	8.33
2^{25}	2^{24}		80.2s	79.0s	1.2s	98.1s	14.1s	1.5s	7.2s	71.1s	13.7	8.33

Table 2: Comparisons between our computations-over-PSI implementation and others

$\|S\|$	$\|R\|$	Protocol	sender offline	sender online	comm.(MB)
2^{12}	2^{12}	ours, `PSI-cardinality`	2.8s	1.6s	4.62
		ours,`PSI-token-threshold`	8.3s	9.5s	12.95
		ours, `tPSI`	8.3s	10.2s	15.73
		[JTKA22], `PSI`	—	7.396s	141
2^{20}	11041	ours, `tPSI`	52s	35.4s	20.36
		[CLR17], `PSI`	43s	4.47s	14.34
		[CMdG+21], `PSI`	29s	4.23s	8.94
2^{20}	1024	ours, `PSI-cardinality`	73.1s	9.1s	6.47
		ours, `PSI-token-threshold`	73.4s	12.1s	16.66
$\approx 2^{22}$	1120	[TSS+20], `PSI-cardinality`	—	35.2s	126.7
2^{20}	2^{8}	[LPR+21], `circuit-PSI`, Construction 1	137	0.62	7
		[LPR+21], `circuit-PSI`, Construction 2	0	3026	29
2^{20}	2^{8}	[SJ23], `circuit-PSI`, Construction 1	7.64	4.96	4.8
		[SJ23], `circuit-PSI`, Construction 2	3.8	2.59	12.2
2^{20}	2^{20}	[IKN+20], `PSI-sum-cardinality`	—	776.4s	84
2^{20}	2^{20}	[MPR+20], `PSI-sum-cardinality`	—	35583s	436.7
2^{20}	2^{20}	[PSWW18], `PSI-token-threshold`	—	86.6s	6950.6

In Table 2, we present a comparative analysis of our work with existing literature on computation-over-PSI protocols. Our contributions include `tPSI` for securely computing the intersection based on a threshold condition, `PSI-cardinality` for determining the size of the set intersection, `PSI-sum-cardinality` for securely summing payload values associated with intersected elements and computing the intersection size, and `PSI-token-threshold` for generating an encryption of a valid token when the intersection size exceeds a threshold value. To evaluate our work, we first compare it with FHE-based standard PSI protocols such as [CLR17, CMdG+21]. Our protocols exhibit acceptable computation and communication overhead compared to these reference works.

Another recent approach presented in [JTKA22] employs homomorphic evaluation of a branching program for constructing PSI and its variants. While their approach has advantages in certain communication cost scenarios, our work offers competitive performance, even for parameter setups involving relatively small set sizes.

In contrast, [IKN+20] and [MPR+20] focus on the private intersection-sum with cardinality (`PSI-sum-cardinality`) protocol, utilizing Diffie-Hellman Protocol + Paillier and distributed oblivious pseudorandom function (D-OPRF), respectively. Our `PSI-cardinality` design can be easily adapted to implement `PSI-sum-cardinality` with only a modest increase in runtime and communication overhead, making it a more favorable option for asymmetric scenarios where the sender's set is substantially larger than the receiver's.

[TSS+20] introduces a `PSI-cardinality` protocol based on keyword PIR, which is particularly suited for asymmetric sets. Although their work provides performance estimates rather than direct implementations, our approach exhibits better runtime efficiency and lower communication cost.

[PSWW18] proposes a circuit-based construction for PSI protocols, optimizing performance efficiency for symmetric sets. In contrast, our work focuses on asymmetric sets, where the sender's set is large and the receiver's set is small, resulting in superior communication complexity for such scenarios.

Lastly, [LPR$^+$21, SJ23] propose to use FHE to construct the circuit-based PSI. Generally, the performance of these state-of-the-art circuit PSI protocols surpasses that of our work. However, it is essential to highlight that the functionality of our threshold PSI protocols is quite different from the circuit protocol. Circuit-PSI must additionally employ generic MPC protocols to construct `SecretTokenGen` and other threshold-related functions. Our primary focus in this study is to explore the feasibility of employing FHE technique for a direct construction of a PSI-with-computations protocol. In comparison, our approach achieves comparable performance, and our protocols maintain conceptual simplicity and modularity. This simplicity streamlines both security analysis and implementations.

9 Conclusions

In this study, we address the challenge of constructing threshold-related functionalities for private set intersection (PSI) from fully homomorphic encryptions (FHE). Existing PSI protocols with computations primarily target symmetric sets, where the sender and the receiver possess sets of equal size. However, these protocols face challenges in achieving optimal communication overhead when transitioning to asymmetric sets, where the sender's set is substantially larger than the receiver's. Leveraging the advantages of FHE primitives in communication complexity, we tackle this issue while maintaining computational efficiency by introducing various optimization techniques.

To be more specific, we initially present a variant of the FHE-based Oblivious Polynomial Evaluation (OPE) primitive to build the `PSI-cardinality` protocol. This protocol securely computes the cardinality of the set intersection. We then modify this FHE-based OPE primitive to develop the `PSI-cardinality-sum` protocol, which securely computes both the cardinality of the set intersection and the sum of payload values associated with the intersected elements. Augmenting the `PSI-cardinality-sum` protocol with a novel proposal for an OPE-based comparison primitive, we establish the `PSI-token-threshold` protocol, which securely computes a token if and only if the intersection size satisfies a presribed requirement. Finally, by combining `PSI-token-threshold` with a standard PSI, we construct a `tPSI` protocol. This protocol securely computes the set intersection if and only if the intersection size satisfies a prescribed requirement.

Our findings indicate a close connection between the constructions of threshold-related PSI variants, demonstrating the technical feasibility of developing efficient PSI-with-computation protocols over asymmetric sets exclusively from FHE. We hope that our work stimulates interest among researchers in the field, encouraging further exploration of practical solutions for FHE-based PSI-with-computation, especially in scenarios involving disparate set sizes.

References

[ACT11] Giuseppe Ateniese, Emiliano De Cristofaro, and Gene Tsudik. (if) size matters: size-hiding private set intersection. In *International Workshop on Public Key Cryptography*, pages 156–173. Springer, 2011.

[AJLA$^+$12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel 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*, pages 483–501. Springer, 2012.

[BGK$^+$08] Prosenjit Bose, Hua Guo, Evangelos Kranakis, Anil Maheshwari, Pat Morin, Jason Morrison, Michiel H. M. Smid, and Yihui Tang. On the false-positive rate of bloom filters. *Inf. Process. Lett.*, 108(4):210–213, 2008.

[Blo70] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, 1970.

[BMRR21] Saikrishna Badrinarayanan, Peihan Miao, Srinivasan Raghuraman, and Peter Rindal. Multi-party threshold private set intersection with sublinear communication. In *IACR International Conference on Public-Key Cryptography*, pages 349–379. Springer, 2021.

[CGS21] Nishanth Chandran, Divya Gupta, and Akash Shah. Circuit-psi with linear complexity via relaxed batch opprf. Cryptology ePrint Archive, Paper 2021/034, 2021. https://eprint.iacr.org/2021/034.

[CKT10] Emiliano De Cristofaro, Jihye Kim, and Gene Tsudik. Linear-complexity private set intersection protocols secure in malicious model. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 213–231. Springer, 2010.

[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.

[CM20] Melissa Chase and Peihan Miao. Private set intersection in the internet setting from lightweight oblivious prf. In *Annual International Cryptology Conference*, pages 34–63. Springer, 2020.

[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.

[DCW13] Changyu Dong, Liqun Chen, and Zikai Wen. When private set intersection meets big data: an efficient and scalable protocol. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 789–800, 2013.

[DD15] Sumit Kumar Debnath and Ratna Dutta. Secure and efficient private set intersection cardinality using bloom filter. In *International Conference on Information Security*, pages 209–226. Springer, 2015.

[DSMRY09] Dana Dachman-Soled, Tal Malkin, Mariana Raykova, and Moti Yung. Efficient robust private set intersection. In *International Conference on Applied Cryptography and Network Security*, pages 125–142. Springer, 2009.

[EFG$^+$15] Rolf Egert, Marc Fischlin, David Gens, Sven Jacob, Matthias Senker, and Jörn Tillmanns. Privately computing set-union and set-intersection cardinality via bloom filters. In *Australasian Conference on Information Security and Privacy*, pages 413–430. Springer, 2015.

[FHNP16] Michael J Freedman, Carmit Hazay, Kobbi Nissim, and Benny Pinkas. Efficient set intersection with simulation-based security. *Journal of Cryptology*, 29(1):115–155, 2016.

[FNP04] Michael J Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *International conference on the theory and applications of cryptographic techniques*, pages 1–19. Springer, 2004.

[Gen09] Craig Gentry. *A fully homomorphic encryption scheme*. Stanford university, 2009.

[GN19] Satrajit Ghosh and Tobias Nilges. An algebraic approach to maliciously secure private set intersection. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 154–185. Springer, 2019.

[GPR+21] Gayathri Garimella, Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. Oblivious key-value stores and amplification for private set intersection. In *Annual International Cryptology Conference*, pages 395–425. Springer, 2021.

[GS19] Satrajit Ghosh and Mark Simkin. The communication complexity of threshold private set intersection. In *Annual International Cryptology Conference*, pages 3–29. Springer, 2019.

[Haz18] Carmit Hazay. Oblivious polynomial evaluation and secure set-intersection from algebraic prfs. *Journal of Cryptology*, 31(2):537–586, 2018.

[HEK12] Yan Huang, David Evans, and Jonathan Katz. Private set intersection: Are garbled circuits better than custom protocols? In *NDSS*, 2012.

[HFH99] Bernardo A Huberman, Matt Franklin, and Tad Hogg. Enhancing privacy and trust in electronic communities. In *Proceedings of the 1st ACM conference on Electronic commerce*, pages 78–86, 1999.

[HN10] Carmit Hazay and Kobbi Nissim. Efficient set operations in the presence of malicious adversaries. In *International Workshop on Public Key Cryptography*, pages 312–331. Springer, 2010.

[HOS17] Per Hallgren, Claudio Orlandi, and Andrei Sabelfeld. Privatepool: Privacy-preserving ridesharing. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 276–291. IEEE, 2017.

[HV17] Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. Scalable multi-party private set-intersection. In *IACR international workshop on public key cryptography*, pages 175–203. Springer, 2017.

[HW06] Susan Hohenberger and Stephen A Weis. Honest-verifier private disjointness testing without random oracles. In *International Workshop on Privacy Enhancing Technologies*, pages 277–294. Springer, 2006.

[IKN+20] Mihaela Ion, Ben Kreuter, Ahmet Erhan Nergiz, Sarvar Patel, Shobhit Saxena, Karn Seth, Mariana Raykova, David Shanahan, and Moti Yung. On deploying secure computing: Private intersection-sum-with-cardinality. In *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 370–389. IEEE, 2020.

[JL10] Stanisław Jarecki and Xiaomin Liu. Fast secure computation of set intersection. In *International Conference on Security and Cryptography for Networks*, pages 418–435. Springer, 2010.

[JTKA22] Jonas Janneck, Anselme Tueno, Jörn Kußmaul, and Matthew Akram. Private computation on set intersection with sublinear communication. *Cryptology ePrint Archive*, 2022.

[KKRT16] Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. Efficient batched oblivious prf with applications to private set intersection. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 818–829, 2016.

[KS05] Lea Kissner and Dawn Song. Privacy-preserving set operations. In *Annual International Cryptology Conference*, pages 241–257. Springer, 2005.

[LPR$^+$21] Tancrede Lepoint, Sarvar Patel, Mariana Raykova, Karn Seth, and Ni Trieu. Private join and compute from pir with default. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 605–634. Springer, 2021.

[Mea86] Catherine 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*, pages 134–134. IEEE, 1986.

[MPR$^+$20] Peihan Miao, Sarvar Patel, Mariana Raykova, Karn Seth, and Moti Yung. Two-sided malicious security for private intersection-sum with cardinality. In *Annual International Cryptology Conference*, pages 3–33. Springer, 2020.

[OOS17] Michele Orrù, Emmanuela Orsini, and Peter Scholl. Actively secure 1-out-of-n ot extension with application to private set intersection. In *Cryptographers' Track at the RSA Conference*, pages 381–396. Springer, 2017.

[PRTY19] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. Spot-light: Lightweight private set intersection from sparse ot extension. In *Annual International Cryptology Conference*, pages 401–431. Springer, 2019.

[PSSZ15] Benny Pinkas, Thomas Schneider, Gil Segev, and Michael Zohner. Phasing: Private set intersection using permutation-based hashing. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 515–530, 2015.

[PSTY19] Benny Pinkas, Thomas Schneider, Oleksandr Tkachenko, and Avishay 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*, pages 122–153. Springer, 2019.

[PSWW18] Benny Pinkas, Thomas Schneider, Christian Weinert, and Udi Wieder. Efficient circuit-based psi via cuckoo hashing. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 125–157. Springer, 2018.

[PSZ14] Benny Pinkas, Thomas Schneider, and Michael Zohner. Faster private set intersection based on {OT} extension. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 797–812, 2014.

[PSZ18] Benny Pinkas, Thomas Schneider, and Michael Zohner. Scalable private set intersection based on ot extension. *ACM Transactions on Privacy and Security (TOPS)*, 21(2):1–35, 2018.

[Res21a] Microsoft Research. APSI: C++ library for Asymmetric PSI. https://github.com/microsoft/APSI, 2021. [Online; accessed September-2023].

[Res21b] Microsoft Research. Microsoft SEAL. https://github.com/microsoft/SEAL, 2021. [Online; accessed September-2023].

[RR17] Peter Rindal and Mike Rosulek. Malicious-secure private set intersection via dual execution. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1229–1242, 2017.

[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.

[SJ23] Yongha Son and Jinhyuck Jeong. Psi with computation or circuit-psi for unbalanced sets from homomorphic encryption. In *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security*, pages 342–356, 2023.

[TSS+20] Ni Trieu, Kareem Shehata, Prateek Saxena, Reza Shokri, and Dawn Song. Epione: Lightweight contact tracing with strong privacy. *arXiv preprint arXiv:2004.13293*, 2020.

[ZC18] Yongjun Zhao and Sherman SM Chow. Can you find the one for me? In *Proceedings of the 2018 Workshop on Privacy in the Electronic Society*, pages 54–65, 2018.