IET Journals
The Institution of Engineering and Technology

Post-Quantum Protocol for Computing Set Intersection Cardinality with Linear Complexity

ISSN 1751-8644
doi: 0000000000
www.ietdl.org

Sumit Kumar Debnath[1*], *Pantelimon Stănică*[2], *Nibedita Kundu*[3], *Tanmay Choudhury*[4]

[1] *Department of Mathematics, National Institute of Technology Jamshedpur, Jamshedpu-831014, India;* sdebnath.math@nitjsr.ac.in
[2] *Department of Applied Mathematics, Naval Postgraduate School, Monterey, CA 93943, USA;* pstanica@nps.edu
[3] *Department of Mathematics, The LNM Institute of Information Technology, Jaipur -302031, India;* nknkundu@gmail.com
[4] *Department of Mathematics, National Institute of Technology Jamshedpur, Jamshedpur-831014, India;* tc499180022@gmail.com
** E-mail: sdebnath.math@nitjsr.ac.in*

Abstract: Nowadays, the necessity of electronic information increases rapidly. As a consequence, often, that information needs to be shared among mutually distrustful parties. In this area, Private Set Intersection (PSI) and its variants play an important role when the participants wish to do secret operations on their input sets. Unlike the most modern public key cryptosystems relying on number theoretic problems, lattice based cryptographic constructions provide security in the presence of a quantum computer. Consequently, developing Private Set Intersection and its variants using lattice based cryptosystem becomes an interesting direction for research. This paper presents the *first size-hiding post quantum* Private Set Intersection Cardinality (PSI-CA) protocol whose complexity is *linear* in the size of the sets of the participants. We use space-efficient probabilistic data structure (Bloom filter) as its building block. Further, we extend our PSI-CA to its authorized version, that is, Authorized PSI-CA (APSI-CA). Security for both of them is achieved in the standard model based on hardness of the decisional learning with errors (DLWE) problem.

Keywords: PSI-CA · APSI-CA · Bloom filter · post quantum cryptography · lattice based cryptosystem

1 Introduction

We are living in an era where sharing of digital contents among mutually distrustful parties becomes more and more important. As a consequence, we need some strong cryptographic technique that allows sharing of digital contents. Private Set Intersection (PSI) [22] is one of the most promising options in the field of secret sharing. It enables the participants only to determine the intersection of their respective secret datasets, not beyond that. PSI can be classified into two categories: (i) *one-way PSI* [28] and (ii) *two-way or mutual PSI* (mPSI) [35]. In the first case, only one learns the intersection and the other learns nothing, while in the later case, both of them get the intersection.

Consider the scenario of DNA matching [6, 14], where two participants may want to evaluate private computation of the Hamming Distance between two strings on an arbitrarily large alphabet. The cardinality variant of PSI, i.e., PSI-CA [1] is ideal for this case as it enables the participants to evaluate the cardinality of the intersection, rather than any content. The protocol has two variants: one-way PSI-CA [30] and mutual PSI-CA (mPSI-CA) [35].

A PSI-CA protocol proven secure in malicious model [26] cannot prevent the malicious client from submitting arbitrary input sets. This problem cannot be overcome without some technique to authorize client's inputs [15]. The authorized version of PSI-CA is known as Authorized PSI-CA (APSI-CA) [8], where at the beginning of the PSI-CA protocol, a mutually trusted third party authorizes client's private set.

In PSI-CA, sometimes it becomes necessary to hide the client's input set size from the server since it may represent sensitive information [2]. For example, if multiple interactions between the two same participants are given then fluctuations in input size may be sensitive. An additional reason is that the amount of computation imposed on the server may cause privacy of input set size. In a PSI-CA, if the server does not have knowledge about the size of client's set then that is known as size-hiding PSI [17]. Whereas the cardinality variant is known as size-hiding PSI-CA [17].

PSI-CA has wide applications in privacy preserving data mining [38], on-line gaming [7], collaborative botnet detection [41], social networks [37], location-based services [42], etc.

1.1 Related Works

PSI. The concept of PSI based on commutative encryption was introduced by Agrawal et al. [1]. In the following, Freedman et al. [22] proposed an oblivious polynomial evaluation (OPE) based PSI. Later, Hazay and Nissim [28] improved the work of [22]. Unpredictable function (UPF) based PSI was introduced by Jarecki and Liu [32]. Utilizing zero-knowledge proofs and random hash functions, De Cristofaro et al. [13] constructed a PSI protocol. The garbled circuit (GC) based PSI was introduced by Huang et al. [31]. In the following, two PSI protocols were designed by Dong et al. [18] using oblivious transfer and Bloom filter. Later, the existing solutions for set-intersection in the semi-honest setting were examined and compared by Pinkas et al. [46]. In the following, Freedman et al. [23] modified their work of [22] to construct a two-party PSI achieving security in semi-honest environment without random oracles. A cheat-sensitive quantum scheme for PSI was developed by Shi et al. [50]. Chen et al. [10] proposed a PSI using fully homomorphic encryption scheme. In the following, a PSI employing dual execution was developed by Rindal and Rosulek [48]. Later, Hazay and Venkitasubramaniam [29] constructed a PSI protocol in multi-party setting using the PSI technique of Freedman et al. [22]. A new paradigm for PSI in multi-party setting was presented by Kolesnikov et al. [36] using symmetric key techniques. The authors of [34] proposed PSI compatible with with unequal set sizes. The concept of of Reactive PSI was introduced by Cerulli et al. [9]. In the following, Ciampi and Orlandi [11] constructed an oblivious transfer (OT) based PSI protocol. An improved hashing-based generic PSI was designed by Falk et al. [20]. Later, Ghosh and Simkin [25] proposed a threshold PSI with logarithmic communication complexity. In the following, Groce et al. [27] developed a PSI using differentially private leakage. Pinkas et al. [45] presented the first circuit-based PSI attaining linear communication cost. Using the primitive oblivious linear function

evaluation, Ghosh and Nilges [24] gave a new approach for PSI. Recently, Pinkas et al. [43] presented a novel PSI technique using spare OT extension More recently, Pinkas et al. [44] developed an efficient PSI utilizing probe-and-XOR of strings (PaXoS).

PSI-CA. Agrawal et al. [1] presented the first PSI-CA in semi-honest model. Later, an efficient Oblivious Polynomial Evaluation (OPE) based PSI-CA was designed by Hohenberger and Weis [30]. Later, a PSI-CA with linear communication and computation complexity was constructed by De Cristofaro et al. [14]. Subsequently, Debnath and Dutta [16, 17] presented other PSI-CA protocols incurring linear communication and computation complexity. The work of [22] was modified by Freedman et al. [23] to develop another PSI-CA attaining linear communication and computation complexity. Later, a PSI-CA protocol was presented by Shi et al. [51] employing quantum computation. Recently, Dong and Loukides [19] employed Flajolet-Martin (FM) sketch [21] to design an approximate PSI-CA protocol with logarithmic complexity. More recently, Lv et al. [39] proposed an unbalanced PSI-CA with linear communication and computation complexity.

The first mPSI-CA relying on OPE was presented by Kissner and Song [35], where more than two players may be involved. However, they did not consider fairness. Later, an OPE based fair mPSI-CA protocol for certified sets was developed by Camenisch and Zaverucha [8].

APSI and APSI-CA: Camenisch and Zaverucha [8] introduced the concept of APSI. Their scheme achieves quadratic computation and linear communication complexity in malicious environment without ROM. Following this work, De Cristofaro et al. [12] presented an APSI protocol attaining linear communication and computation complexity in the ROM. Later, Stefanov et al. [53] proposed an APSI in malicious environment. Further, Kerschbaum [33] employed a Bloom filter to construct an APSI with linear communication and computation complexity, and Debnath and Dutta [17] developed a size-hiding APSI in malicious model.

The first APSI-CA was proposed by Camenisch and Zaverucha [8]. Security of their design is achieved in malicious environments without random oracles. Later, an APSI-CA was constructed by De Cristofaro et al. [14] with linear communication and computation complexity in the ROM. Also, Debnath and Dutta [17] proposed a size-hiding APSI-CA that achieves linear communication and computation complexity.

Gap Analysis: In the literature, most of the existing works on PSI-CA [1, 8, 14, 16, 17, 19, 23, 30, 39] are not quantum-resistant, as they are relying on number theoretic problems, such as factorization problem and discrete logarithm problem, which are vulnerable to attack in a quantum environment due to Shor's [52] algorithm. Thus, the design of a quantum computer resistant PSI-CA becomes an interesting direction of research.

In addition to the apparent resistance to quantum attacks, lattice based cryptography offers other useful features such as security under worst-case intractability assumptions, efficient parallel computations and homomorphic computations. As a consequence, lattice based cryptographic constructions are potential candidates for the post quantum era. Developing lattice based PSI-CA remains an interesting task as a quantum computer is not known to be able to break its security. As far as we are aware, currently there is no post quantum PSI-CA protocol.

1.2 Our Results

In this work, we tackle the issue of constructing a *post quantum* PSI-CA by integrating the lattice based public-key encryption of [47]. We employ a space-efficient probabilistic data structure (Bloom filter) as a building block of our construction. The security of the scheme is proven in semi-honest model without random oracles based on hardness of the decisional learning with errors (DLWE) problem. Communication and computation complexity of this scheme are linear in the size of the sets of the participants. In particular, our PSI-CA is the *first* post quantum PSI-CA. We refer to the appendix for basic information on the various security models.

In order to prevent the arbitrary inclusion by the client in its input set, we extend our PSI-CA to APSI-CA, where a trusted third party authorizes client's input set. Our APSI-CA attains all the properties as of PSI-CA.

More interestingly, our designs are *size-hiding* in the sense that the client need not to reveal the size of its private set to the server. Rather, only the upper bound is revealed. The server's computational effort is independent of this upper bound since it is different from the security parameter.

2 Organization

The rest of this paper is organized as follows. In Section 3, we provide preliminaries. Our proposed protocols PSI-CA and APSI-CA are described with their security analysis in Section 4. Efficiency analysis of our designs is given in Section 5, and we conclude the paper in Section 6. The appendix contains some known security models, for completeness.

3 Preliminaries

Throughout the paper, $a \leftarrow A$ denotes "a is the output of the procedure A", κ represents "security parameter", $x \in_R X$ stands for "variable x is chosen uniformly at random from set X", $\{\mathcal{X}_t\}_{t\in\mathcal{N}} \equiv^c \{\mathcal{Y}_t\}_{t\in\mathcal{N}}$ is used to denote "the distribution ensemble $\{\mathcal{X}_t\}_{t\in\mathcal{N}}$ is computationally indistinguishable from the distribution ensemble $\{\mathcal{Y}_t\}_{t\in\mathcal{N}}$" and $\mathbb{Z}_q$ represents "the ring of integers modulo q". Generally, $y \in \mathbb{Z}_q$ is associated with the value $x \in (-q/2, q/2] \cap \mathbb{Z}$ such that $x \equiv y (\mathrm{mod} q)$. A function $\epsilon : \mathbb{N} \rightarrow \mathbb{R}$ is said to be a negligible function of κ if for each constant $c > 0$, we have $\epsilon(\kappa) = o(\kappa^{-c})$ for all sufficiently large κ.

Definition 1. Functionality: *A* functionality $\mathcal{F}_\Pi$, *computed by two parties A and B with inputs X_A and X_B, respectively, by running a protocol Π, is defined as $\mathcal{F}_\Pi : X_A \times X_B \rightarrow Y_A \times Y_B$, where Y_A and Y_B are the outputs of A and B, respectively, on completion of the protocol Π between A and B.*

Definition 2. Decisional learning with errors (DLWE) assumption [5, 40, 47]: *A learning with errors (LWE) instance $LWE_{n,l,q,\chi}$ is parameterized by dimension $n \in \mathbb{N}$, modulus $q = q(n) \in$ Primes, $l = O(n \log q)$ and $\chi = \chi(n)$ (noise distribution) is a probability distribution over $\mathbb{Z}_q$. Let $A \in_R \mathbb{Z}_q^{n\times l}, \mathbf{s} \in_R \mathbb{Z}_q^n, \mathbf{e} \in_R \chi^l$ and $\mathbf{b}^T \in_R \mathbb{Z}_q^l$. Then the DLWE assumption states that no PPT (probabilistic polynomial-time) algorithm $\mathcal{A}$ can distinguish the distribution $(A, \mathbf{s}^T A + \mathbf{e}^T)$ from $(A, \mathbf{b})$, i.e., $|Pr[\mathcal{A}(1^n, (A, \mathbf{s}^T A + \mathbf{e}^T)) = 1] - Pr[\mathcal{A}(1^n, (A, \mathbf{b})) = 1]|$ is a negligible function of n.*

3.1 Bloom Filter [4]

A Bloom filter is a data structure representing a set $X = \{x_1, \ldots, x_v\}$ of v elements using an array of size m. To add elements or to check the existence of elements in that array, it uses k independent collision resistant hash functions $H_{\mathsf{Bloom}} = \{h_1, \ldots, h_k\}$ with $h_i : \{0,1\}^* \rightarrow \{1, \ldots, m\}$ for $i = 1, \ldots, k$. Let the Bloom filter for the set X be denoted by $\mathsf{BF}_X \in \{0,1\}^m$ with the i-th entry, $\mathsf{BF}_X[i]$. A variant of a Bloom filter [4] performing three operations is described below –

– *Initialization*: An empty Bloom filter with no elements is formed by putting 1 to all the entries of an array of size m.
– *Add(x)*: In this step, an element $x \in X$ is added to the Bloom filter by computing $h_1(x), \ldots, h_k(x)$ and setting 0 to all the entries of the Bloom filter with indices $h_1(x), \ldots, h_k(x)$. To get the Bloom filter BF_X for the set X, repeat the process for each $x \in X$.
– *Check($\hat{x}$)*: In this step, presence of an element $\hat{x}$ in X can be checked by computing $h_1(\hat{x}), \ldots, h_k(\hat{x})$ without knowing X. Now, if at least one of $\mathsf{BF}_X[h_1(\hat{x})], \ldots, \mathsf{BF}_X[h_k(\hat{x})]$ is 1, then $\hat{x}$ is not in X, otherwise $\hat{x}$ is *probably* in X.

A Bloom filter allows *false positives*, however it never yields false negatives.

Theorem 3. [18] *Given the number v of elements to be added and a desired maximum false positive rate of $\frac{1}{2^k}$, the optimal size m of the Bloom filter is $m = \left\lceil \frac{vk}{\ln 2} \right\rceil$.*

3.2 Learning with Errors (LWE)-Based Public Key Encryption [47]

We describe a public-key encryption scheme PKE based on LWE, due to [47]. The encryption scheme is parameterized by an LWE instance $\mathsf{LWE}_{n,l,q,\chi}$ with dimension n, modulus $q = q(n)$, $l = O(n \log q)$ and probability distribution $\chi = \chi(n)$ over $\mathbb{Z}_q$. The distribution χ is known as noise or error distribution. In practice, χ is a discrete Gaussian distribution over $\mathbb{Z}$. The encryption scheme consists of the following three algorithms:

- $(pk, sk) \longleftarrow \mathcal{PKE}.\mathsf{KGen}(1^n)$. On input $n \in \mathbb{N}$, the user generates a secret key and public key as $sk = \mathbf{s} \in \mathbb{Z}_q^n$ and $pk = (A, \mathbf{s}^T A + \mathbf{e}^T) = (A, d^T)$, respectively, where $A \in_R \mathbb{Z}_q^{n \times l}$, $\mathbf{e} \in_R \chi^l$, $q = q(n)$ is a large prime, $l = O(n \log q)$ and $\chi = \chi(n)$ is noise distribution over $\mathbb{Z}_q$ and it is B bounded, i.e., $|e| \leq B$ with high probability for $e \in \chi$.
- $c \longleftarrow \mathcal{PKE}.\mathsf{Enc}(\mu, pk)$. Given a message $\mu \in \{0,1\}$ and the public key $pk = (A, \mathbf{s}^T A + \mathbf{e}^T) = (A, d^T)$, the encryptor picks $\mathbf{r} \in_R \{0,1\}^l$ and outputs the corresponding ciphertext as $c = \mathsf{Enc}_{pk}(\mu) = (A\mathbf{r}, d^T\mathbf{r} + \mu\lceil q/2 \rceil) = (u, v)$
- $\mu' \longleftarrow \mathcal{PKE}.\mathsf{Dec}(c, sk)$. Given a ciphertext c and the secret key $sk = \mathbf{s}$, the decryptor computes $|v - \mathbf{s}^T u| = |\mathbf{e}^T\mathbf{r} + \mu\lceil q/2 \rceil|$. It then outputs the message μ' as 0 if $|v - \mathbf{s}^T u| < q/4$; otherwise, outputs μ' as 1.

Correctness: If (u, v) is a correctly formed ciphertext then $v - \mathbf{s}^T u = \mathbf{e}^T\mathbf{r} + \mu\lceil q/2 \rceil$. Note that $|\mathbf{e}^T\mathbf{r}| = |\sum_{i=1}^{l} e_i r_i| \leq \sum_{i=1}^{l} |e_i| \leq lB$ with $\mathbf{e} = (e_1, \ldots, e_l) \in \chi^l, \mathbf{r} = (r_1, \ldots, r_l) \in \{0,1\}^l$, since $|e| \leq B$ for $e \in \chi$. Now, from the definition of the decryption algorithm, it is clear that correctness holds if $|\mathbf{e}^T\mathbf{r}| < q/4$ which will happen with high probability due to choice of $q > 4lB$. However, in our constructions, the sum of $k+1$ ciphertexts $\mathsf{Enc}_{pk}(\mu_1), \ldots, \mathsf{Enc}_{pk}(\mu_{k+1})$ multiplied by t, i.e., $t\sum_{j=1}^{k+1} \mathsf{Enc}_{pk}(\mu_j)$ should decrypt to 0 only if all of $\mu_1, \ldots, \mu_{k+1}$ are 0; otherwise, it should decrypt to 1, where $t \in \mathbb{Z}_\rho$ for some security parameter ρ. As a consequence, we need the form of the ciphertext as $c = \mathsf{Enc}_{pk}(\mu) = (A\mathbf{r}, d^T\mathbf{r} + \mu\lceil q/2(k+1)\rho \rceil) = (u, v)$. In this case, decryption of c will give 0 if $|v - \mathbf{s}^T u| < q/4(k+1)\rho$; 1 otherwise. Therefore, the correctness holds if $|t\mathbf{e}^T\mathbf{r}| < q/4(k+1)\rho$, which happens with high probability due to choice of $q > 4(k+1)^2\rho^2 lB$. Semantic security of the encryption scheme is archived under the decisional learning with errors (DLWE) assumption.

3.3 Learning with Errors (LWE)-Based Digital Signature Scheme [3]

In this section, the digital signature of [3] is described briefly. We will use LWE instance $\mathsf{LWE}_{n,l,q,\chi}$ with dimension n, modulus $q = q(n), l = O(n \log q)$ and probability distribution $\chi = \chi(n)$ over $\mathbb{Z}_q$. The distribution χ is known as noise or error distribution. We denote this digital signature scheme by DSig which consists of the following three algorithms:

- $(pk_{DSig}, sk_{DSig}) \longleftarrow \mathsf{DSig.KGen}(n, l, t, q, \sigma_H, \sigma_E)$. On input $n, l, t, q, \sigma_H, \sigma_E$, the user generates a signing key and a verification key as $sk_{DSig} = \mathbf{H} \in D_S^{n \times t}$ and $pk_{DSig} = (\mathbf{G}, \mathbf{GH} + \mathbf{E}) = (\mathbf{G}, \mathbf{R})$ respectively, where $\mathbf{G} \in_R \mathbb{Z}_q^{l \times n}$, $\mathbf{E} \in_R \chi^{l \times t}$ with entries $|E_{i,j}| \leq 7\sigma_E$, D_S is distribution for secret over $\mathbb{Z}_q$ with standard deviation σ_S, $\chi = \chi(n)$ is noise distribution over $\mathbb{Z}_q$ with standard deviation σ_E and χ is B bounded, i.e., $|e| \leq B$ with high probability for $e \in \chi$. Note that $n, t \in \mathbb{N}$, $q = q(n)$, $l = O(n \log q)$.
- $(\mathbf{y}, c) \longleftarrow \mathsf{DSig.Sign}(msg, sk_{DSig})$. Given a message $msg \in \{0,1\}^*$ and the signing key $sk_{DSig} = \mathbf{H}$, signer outputs $(\mathbf{z} + \mathbf{Hc}, c) = (\mathbf{y}, c) = DSig(msg)$ as the signature, where $\mathbf{z} \in_R D_z^n$ for some uniform distribution D_z over $\mathbb{Z}_q$, $\mathbf{c} = F(c)$ for some function F from $\{0,1\}^\kappa$ to the set of t length vectors of weight wg with coefficients $\{-1, 0, 1\}$, $c = \overline{h}(\lfloor \nu \rceil_\pi, msg)$ for some hash function $\overline{h} : \{0,1\}^* \to \{0,1\}^\kappa, \nu = \mathbf{Gz}$ and $\lfloor x \rceil_\pi$ represents the integer x with its π significant bits removed for some security parameter $\pi \in \mathbb{N}$. Here the signer utilizes rejection sampling in order to make sure that the distribution of $\mathbf{y}$ is independent of the secret.
- $(accept \text{ or } reject) \longleftarrow \mathsf{DSig.Ver}(msg, DSig(msg), pk_{DSig})$. Given message-signature pair $(msg, DSig(msg) = (\mathbf{y}, c))$ and the verification key $pk_{DSig} = (\mathbf{G}, \mathbf{R})$, the verifier computes $\varpi = \mathbf{Gy} - \mathbf{Rc} \bmod q$ and $c' = \overline{h}(\lfloor \varpi \rceil_\pi, msg)$, where $\mathbf{c} = F(c)$. If $c' = c$ and $||\mathbf{y}||_\iota \leq B$ for $\iota \in \{2, \infty\}$ then the verifier outputs $accept$, otherwise outputs $reject$.

4 Protocols

Protocol Requirements: Each of our designs involves a client C and a server S with secret input sets $X = \{x_1, \ldots, x_w\}$ and $Y = \{y_1, \ldots, y_v\}$ respectively, where $w \leq v$. In the rest of the paper, the number of hash functions for Bloom filter is denoted by k, the set of k hash functions $\{h_1, \ldots, h_k\}$ is represented by H with $h_i : \{0,1\}^* \to \{1, \ldots, m\}$ for $i = 1, \ldots, k$, the variable m denotes the optimal size of a Bloom filter, the key pair for PKE encryption is denoted by (pk, sk). Moreover, $\mathsf{Enc}_{pk}/\mathsf{Dec}_{sk}$ represents the Encryption/Decryption for PKE. In our designs, we require the ciphertext of the form $c = \mathsf{Enc}_{pk}(\mu) = (A\mathbf{r}, d^T\mathbf{r} + \mu\lceil q/2(k+1)\rho \rceil) = (u, v)$ and $q > 4(k+1)^2\rho^2 lB$ for some security parameter ρ. We will use the fact that $t\sum_{j=1}^{k+1} \mathsf{Enc}_{pk}(\mu_j)$ decrypts to 0 if and only if all of $\mu_1, \ldots, \mu_{k+1}$ are 0; 1 otherwise, where

$$t\sum_{j=1}^{k+1} \mathsf{Enc}_{pk}(\mu_j) = \left(tA\sum_{j=1}^{k+1} r_j, td^T\sum_{j=1}^{k+1} r_j + t\sum_{j=1}^{k+1} \mu_j\lceil q/2(k+1)\rho \rceil \right),$$

with $t \in \mathbb{Z}_\rho$. κ, ρ (security parameters), v (the maximum set size), m, H and k (the optimal Bloom filter parameters) are considered as auxiliary inputs.

4.1 The PSI-CA

Two parties are involved in the execution of PSI-CA – a client C and a server S with their input sets $X = \{x_1, \ldots, x_w\} \subseteq \{0,1\}^*$ and $Y = \{y_1, \ldots, y_v\} \subseteq \{0,1\}^*$, respectively. On a high level, the client C first generates (pk, sk) for $\mathcal{PKE}$ and encrypts each entries of the Bloom filter BF_X. It then sends the encrypted Bloom filter $\overline{X}$ along with the public key pk to S, who in turn generates a set $\overline{Y}$ of ciphertexts using Y and $\overline{X}$. The server S then sends $\overline{Y}$ to C. Finally, C decrypts each member of $\overline{Y}$ and outputs the cardinality of $X \cap Y$ as the number of elements of $\overline{Y}$ decrypting 0. The design of our PSI-CA is depicted in Fig. 1.

Correctness: We have the following relation $\mathsf{Enc}_{pk}(\bar{y}_i) = t_i\left(\sum_{j=1}^{k}(b_{h_j(y_i)} + \vartheta_i)\right) = \big(t_i A(\sum_{j=1}^{k} \mathbf{r}_{h_j(y_i)} + \nu_i),$
$t_i d^T(\sum_{j=1}^{k} \mathbf{r}_{h_j(y_i)} + \nu_i)$
$+t_i \sum_{j=1}^{k} \mathsf{BF}_X[h_j(y_i)]\lceil q/2(k+1)\rho \rceil\big)$.
Note that $\mathsf{Enc}_{pk}(\bar{y}_i)$ decrypts to 0, i.e., $\bar{y}_i = 0$ if and only if

1. C generates $(pk, sk) \longleftarrow \mathcal{PKE}.\mathsf{KGen}(1^n)$, where $pk = (A, \mathbf{s}^T A + \mathbf{e}^T) = (A, d^T)$ and $sk = \mathbf{s}$ for PKE. It then executes the following:
(a) generates a Bloom filter $\mathsf{BF}_X = (\mathsf{BF}_X[1], \ldots, \mathsf{BF}_X[m]) \in \{0,1\}^m$ of the set $X = \{x_1, \ldots, x_w\} \subseteq \{0,1\}^*$,
(b) computes $b_i = \mathsf{Enc}_{pk}(\mathsf{BF}_X[i]) = (A\mathbf{r}_i, d^T\mathbf{r}_i + \mathsf{BF}_X[i]\lceil q/2(k+1)\rho \rceil) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ with $r_i \in_R \{0,1\}^l$ for $i = 1, \ldots, m$
(c) sets $\overline{X} = \{b_1, \ldots, b_m\}$ and sends $(\overline{X}, pk)$ to S.
2. On receiving $\overline{X} = \{b_1, \ldots, b_m\}, pk = (A, d^T)$ from C, S performs the following for $i = 1, \ldots, v$,
(a) determines $h_1(y_i), \ldots, h_k(y_i) \in \{1, \ldots, m\}$ and $\vartheta_i = \mathsf{Enc}_{pk}(0) = (A\nu_i, d^T\nu_i)$, $\nu_i \in_R \{0,1\}^l$;
(b) extracts $b_{h_1(y_i)}, \ldots, b_{h_k(y_i)}$ from $\overline{X}$;
(c) evaluates $\sum_{j=1}^k (b_{h_j(y_i)}) = \left(A\sum_{j=1}^k \mathbf{r}_{h_j(y_i)}, d^T\sum_{j=1}^k \mathbf{r}_{h_j(y_i)} + \sum_{j=1}^k \mathsf{BF}_X[h_j(y_i)]\lceil q/2(k+1)\rho \rceil\right)$; and
(d) sets $\mathsf{Enc}_{pk}(\bar{y}_i) = t_i(\sum_{j=1}^k b_{h_j(y_i)} + \vartheta_i) = (U, V)$, where $U = t_i A(\sum_{j=1}^k \mathbf{r}_{h_j(y_i)} + \nu_i)$ and $V = t_i d^T(\sum_{j=1}^k \mathbf{r}_{h_j(y_i)} + \nu_i) + t_i \sum_{j=1}^k \mathsf{BF}_X[h_j(y_i)]\lceil q/2(k+1)\rho \rceil$ with $t_i \in_R \mathbb{Z}_\rho$.
Finally S sends $\overline{Y} = \{\mathsf{Enc}_{pk}(\bar{y}_1), \ldots, \mathsf{Enc}_{pk}(\bar{y}_v)\}$ to C.
3. On receiving $\overline{Y} = \{\mathsf{Enc}_{pk}(\bar{y}_1), \ldots, \mathsf{Enc}_{pk}(\bar{y}_v)\}$ from S, the client C chooses a variable $card$ and sets $card = 0$. To compute $|X \cap Y|$, C then performs the following steps for $i = 1, \ldots, v$:
(a) decrypts $\mathsf{Enc}_{pk}(\bar{y}_i)$ to get $\bar{y}_i \in \{0,1\}$; and
(b) sets $card = card + 1$ if $\bar{y}_i$ is zero.
Finally, C outputs $card$ as $|X \cap Y|$.

Fig. 1: Description of our PSI-CA

$\mathsf{BF}_X[h_j(y_i)] = 0$, for each $j = 1, \ldots, k$. In other words, $\bar{y}_i = 0$ if and only if $y_i \in Y$ passes the set membership test for $\mathsf{BF}_X \in \{0,1\}^m$. Hence, $\bar{y}_i = 0$ if and only if $y_i \in X \cap Y$, except with negligible probability $\frac{1}{2^k}$. As the variable $card$ is set as $card + 1$ only when $\bar{y}_i$ is 0, therefore it gives $|X \cap Y|$.

4.1.1 Security of PSI-CA Protocol: Our PSI-CA trivially enjoy fairness since only client receives the output. We describe below the security of this scheme.

Theorem 4. *The proposed PSI-CA securely computes the functionality $\mathcal{F}_{PSI-CA} : (X, Y) \longrightarrow (|X \cap Y|, \perp)$ in the presence of semi-honest adversaries under the DLWE assumption.*

Proof. Let Ad be the adversary that breaks the security of our PSI-CA protocol between C and S. Then we will construct a simulator $\mathcal{SIM}$ by presenting games $\mathbf{Game_0}, \mathbf{Game_1}$, where $\mathbf{Game_1}$ slightly modifies $\mathbf{Game_0}$. Let $\mathcal{ZZ}$ be a distinguisher that controls Ad, feeds the input of the honest party, and also sees the output of the honest party. We consider $Pr[\mathbf{Game_i}]$ as the probability that $\mathcal{ZZ}$ distinguishes the view of $\mathbf{Game_i}$ from the view of real protocol for $i = 0, 1$.

Case I (Ad **corrupts** S**):** Suppose that $\mathcal{SIM}$ is given access to S's input set Y and output $\perp$.

$\mathbf{Game_0}$: This game corresponds to the real game, where the simulator $\mathcal{SIM}$ simulates C and interacts with Ad. Therefore, $Pr[\mathbf{Game_0}] = Pr[REAL_{PSI-CA}]$, where $REAL_{PSI-CA}$ denotes the real game.

$\mathbf{Game_1}$: $\mathbf{Game_1}$ is same as $\mathbf{Game_0}$ except that the simulator $\mathcal{SIM}$ randomly chooses $z_1, \ldots, z_m \in_R \mathbb{Z}_q^n \times \mathbb{Z}_q$ and replaces $\{b_1, \ldots, b_m\}$ by random $\{z_1, \ldots, z_m\}$. Note that $|Pr[\mathbf{Game_1}] - Pr[\mathbf{Game_0}]|$ is non-negligible means Ad can be used to distinguish the distribution $\{b_1, \ldots, b_m\}$ from $\{z_1, \ldots, z_m\}$. However, this is not possible since the underlying encryption scheme PKE is IND-CPA secure under the DLWE assumption. Thus, $|Pr[\mathbf{Game_1}] - Pr[\mathbf{Game_0}]| \leq \epsilon_1(\kappa)$, where $\epsilon_1(\kappa)$ is a negligible function in security parameter κ.

Now we have,

$$\begin{aligned} &|Pr[\mathbf{Game_1}] - Pr[REAL_{PSI-CA}]| \\ =&|Pr[\mathbf{Game_1}] - Pr[\mathbf{Game_0}]| \leq \epsilon_1(\kappa). \end{aligned}$$

Moreover, the the outcome of internal random coins t^S is uniformly random, thus the distribution is the same as in a real view. Therefore the simulated view $(Y; t^S; \{z_1, \ldots, z_m\})$ in the game $\mathbf{Game_1}$ is indistinguishable from the view in the real game.

Case II (Ad **corrupts** C**):** We assume that $\mathcal{SIM}$ is given access to C's input X and output $|X \cap Y|$.

$\mathbf{Game_0}$: $\mathbf{Game_0}$ is same as the real game, where the simulator $\mathcal{SIM}$ simulates S and interacts with Ad. Thus, $Pr[\mathbf{Game_0}] = Pr[REAL_{PSI-CA}]$, where $REAL_{PSI-CA}$ denotes the real game.

$\mathbf{Game_1}$: This game is similar to $\mathbf{Game_0}$ except that the simulator $\mathcal{SIM}$ replaces $\overline{Y}$ by a randomly chosen set $\{a_1, \ldots, a_v\}$ of ciphertexts, where $|X \cap Y|$ number of ciphertexts are encryption of 0 and $v - |X \cap Y|$ number of ciphertexts are encryption of 1. Since the set $\overline{Y}$ contains $|X \cap Y|$ many ciphertexts encrypting 0 and $v - |X \cap Y|$ many ciphertexts encrypting 1, therefore $|Pr[\mathbf{Game_1}] - Pr[\mathbf{Game_0}]| \leq \epsilon_2(\kappa)$, where $\epsilon_2(\kappa)$ is a negligible function in κ.

Therefore, we have,

$$\begin{aligned} &|Pr[\mathbf{Game_1}] - Pr[REAL_{PSI-CA}]| \\ =&|Pr[\mathbf{Game_1}] - Pr[\mathbf{Game_0}]| \leq \epsilon_2(\kappa). \end{aligned}$$

Further, the the outcome of internal random coins t^S is uniformly random, thus the distribution is the same as in a real view. Hence the simulated view $(X; r^C; \{a_1, \ldots, a_v\})$ in $\mathbf{Game_1}$ is indistinguishable from the real view.

4.2 The APSI-CA

Three parties are involved in the execution of APSI-CA – a client C, a server S with their input sets $X = \{x_1, \ldots, x_w\}$, $Y = \{y_1, \ldots, y_v\}$ respectively, and a mutually trusted certifying authority CA. Our APSI-CA consists of an off-line phase and an online phase, which are depicted in Fig. 2 and Fig. 3, respectively. The

1. C generates $(pk, sk) \longleftarrow \mathcal{PKE}.\mathsf{KGen}(1^n)$ and sends (X, pk) to CA, where $pk = (A, \mathbf{s}^T A + \mathbf{e}^T) = (A, d^T)$ and $sk = \mathbf{s}$ for PKE.
2. CA, on receiving (X, pk) from C, computes BF_X, generates (pk_{DSig}, sk_{DSig}) for digital signature scheme $DSig$ and proceeds as follows for $i = 1, \ldots, m$:
(a) sets $b_i = \mathsf{Enc}_{pk}(\mathsf{BF}_X[i]) = (A\mathbf{r}_i, d^T\mathbf{r}_i + \mathsf{BF}_X[i]\lceil q/2(k+1)\rho \rceil) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$,
(b) using the signing key sk_{DSig} generates a signature over $msg = b_1 || \ldots ||b_m$ as $DSig(msg)$.
CA then sends $(\overline{X} = \{b_1, \ldots, b_m\}, DSig(msg), pk_{DSig})$ to the client C and pk_{DSig} to the server S.

Fig. 2: Description of off-line phase of our APSI-CA

1. C sends $(\overline{X} = \{b_1, \ldots, b_m\}, DSig(b_1 || \ldots ||b_m)$ together with $pk = (A, d^T)$ to the server S.
2. On receiving $(\overline{X} = \{b_1, \ldots, b_m\}, DSig(msg), pk = (A, d^T))$ from C, the server S uses pk_{DSig} in order to verify the correctness of signature $DSig(msg)$. S aborts if the verification fails; otherwise, proceeds as follows for each $i = 1, 2, \ldots, v$:
(a) computes $h_1(y_i), \ldots, h_k(y_i) \in \{1, \ldots, m\}$ and $\vartheta_i = \mathsf{Enc}_{pk}(0) = (A\nu_i, d^T\nu_i)$, $\nu_i \in_R \{0,1\}^l$;
(b) extracts $b_{h_1(y_i)}, \ldots, b_{h_k(y_i)}$ from $\overline{X}$;
(c) evaluates $\sum_{j=1}^{k}(b_{h_j(y_i)}) = \left(A\sum_{j=1}^{k}\mathbf{r}_{h_j(y_i)}, d^T\sum_{j=1}^{k}\mathbf{r}_{h_j(y_i)} + \sum_{j=1}^{k}\mathsf{BF}_X[h_j(y_i)]\lceil q/2(k+1)\rho \rceil\right)$; and
(d) sets $\mathsf{Enc}_{pk}(\bar{y}_i) = t_i(\sum_{j=1}^{k} b_{h_j(y_i)} + \vartheta_i) = (U, V)$, where $U = t_i A(\sum_{j=1}^{k}\mathbf{r}_{h_j(y_i)} + \nu_i)$ and $V = t_i d^T(\sum_{j=1}^{k}\mathbf{r}_{h_j(y_i)} + \nu_i) + t_i\sum_{j=1}^{k}\mathsf{BF}_X[h_j(y_i)]\lceil q/2(k+1)\rho \rceil$ with $t_i \in_R \mathbb{Z}_\rho$.
Finally S sends $\overline{Y} = \{\mathsf{Enc}_{pk}(\bar{y}_1), \ldots, \mathsf{Enc}_{pk}(\bar{y}_v)\}$ to C.
3. On receiving $\overline{Y} = \{\mathsf{Enc}_{pk}(\bar{y}_1), \ldots, \mathsf{Enc}_{pk}(\bar{y}_v)\}$ from S, the client C chooses a variable $card$ and sets $card = 0$. To compute $|X \cap Y|$, C then performs the following steps for $i = 1, \ldots, v$:
(a) decrypts $\mathsf{Enc}_{pk}(\bar{y}_i)$ to get $\bar{y}_i \in \{0, 1\}$; and
(b) sets $card = card + 1$ if $\bar{y}_i$ is zero.
The client C then outputs $card$ as $|X \cap Y|$.

Fig. 3: Description of online phase of our APSI-CA

correctness can be achieved in the similar manner as described for PSI-CA in Section 4.1. In order to control the malicious behavior of C in APSI-CA, CA applies a signature $DSig$ over $msg = b_1 || \cdots ||b_m$ to get $DSig(msg)$ in the off-line phase and this signature is verified by the server S in the online phase. Any post-quantum secure digital signature can be employed in our construction. For instance, we considered the signature of [3] during the calculation of computation and communication costs of the proposed APSI-CA.

4.2.1 *Security of APSI-CA Protocol:*

Theorem 5. *The proposed APSI-CA securely computes the functionality $\mathcal{F}_{APSI-CA} : (X, Y) \longrightarrow (|X \cap Y|, \perp)$ in the presence of a malicious client and semi-honest server under the DLWE assumption.*

Proof. Let Ad be the adversary that breaks the security of our PSI-CA protocol between C and S.

Case I (Ad **corrupts** S)**:** It is similar to the case I of Theorem 4.

Case II (Ad **corrupts** C)**:** In real word, S is honest and C is corrupted by Ad. While for the case of ideal world, an incorruptible trusted third party T is also involved apart from C and S for computing the functionality $\mathcal{F}_{APSI-CA}$. Let $\mathcal{ZZ}$ be a distinguisher that controls Ad, feeds the input of the honest party S, and also sees the output of the honest party S. We are going to show that $\mathcal{ZZ}$'s view in the real world (Ad's view +S's output) is indistinguishable from its view in the idea world (Ad's view +S's output) by presenting a series of games $\mathbf{Game_0}, \mathbf{Game_1}, \mathbf{Game_2}$, where $\mathbf{Game_i}$ slightly modifies $\mathbf{Game_{i-1}}$ for $i = 1, 2$. We argue that in consecutive games, the views of $\mathcal{ZZ}$ are indistinguishable. Let us consider $Pr[\mathbf{Game_i}]$ as the probability that $\mathcal{ZZ}$ distinguishes the view of $\mathbf{Game_i}$ from the view of real protocol for $i = 0, 1, 2$. The simulator in $\mathbf{Game_i}$ is denoted by $\mathcal{S}_i$.

$\mathbf{Game_0}$: This game corresponds to the real game, where the simulator S_0 simulates S, CA and interacts with Ad. Therefore, $Pr[\mathbf{Game_0}] = Pr[REAL_{APSI\text{-}CA}]$, where $REAL_{APSI\text{-}CA}$ denotes the real game.

$\mathbf{Game_1}$: $\mathbf{Game_0}$ is same as $\mathbf{Game_1}$ except that the simulator S_1 plays the role of CA by generating a key pair (pk_{DSig}, sk_{DSig}), encrypting each $\mathsf{BF}_X[i]$ as $b_i = \mathsf{Enc}_{pk}(\mathsf{BF}_X[i])$ and generating signature $DSig(b_1 || \ldots ||b_m)$ using sk_{DSig}. Note that S_1 gets the knowledge of X and $|Pr[\mathbf{Game_1}] - Pr[\mathbf{Game_0}]| \leq \epsilon_1(\kappa)$, where $\epsilon_1(\kappa)$ is a negligible function in κ.

$\mathbf{Game_2}$: In this game, the simulator S_2 has the knowledge of X and the input Y of S. This game is similar to $\mathbf{Game_1}$ except that if the signature $DSig(b_1 || \ldots ||b_m)$ is correct then the simulator S_2 does the following:
1. computes $|X \cap Y|$;
2. constructs a set $\widehat{Y} = \{\hat{y}_1, \ldots, \hat{y}_v\}$ such that $|X \cap Y|$ many members of $\widehat{Y}$ are chosen as 0 and $v - |X \cap Y|$ many members of $\widehat{Y}$ are chosen as 1;
3. encrypts each $\hat{y}_i \in \{0, 1\}$ to determine $\mathsf{Enc}_{pk}(\hat{y}_i) = \bar{y}_i$ for $i = 1, \ldots, v$;
4. sends $\{\mathsf{Enc}_{pk}(\hat{y}_1), \ldots, \mathsf{Enc}_{pk}(\hat{y}_v)\}$ as $\overline{Y}$ to $\mathcal{A}$.
Since the set $\overline{Y}$ contains $|X \cap Y|$ many ciphertexts encrypting 0 and $v - |X \cap Y|$ many ciphertexts encrypting 1, therefore there exists negligible function $\epsilon_2(\kappa)$ such that $|Pr[\mathbf{Game_2}] - Pr[\mathbf{Game_1}]| \leq \epsilon_2(\kappa)$.

We now construct the ideal world adversary $\mathcal{SIM}$ that has oracle access to Ad, simulates the honest parties S, CA, controls C and incorporates all steps from $\mathbf{Game_2}$.

1. $\mathcal{SIM}$ invokes $\mathcal{A}$ with input X and generates a key pair (pk_{DSig}, sk_{DSig}).
2. In order to play the role of CA, the simulator $\mathcal{SIM}$, on receiving (X, pk) from $\mathcal{A}$, constructs BF_X, encrypts each $\mathsf{BF}_X[i]$ as $b_i = \mathsf{Enc}_{pk}(\mathsf{BF}_X[i])$ and generates signature $DSig(b_1||\ldots||b_m)$ using sk_{DSig}. $\mathcal{SIM}$ then sends the tuple $(\overline{X} = \{b_1, \ldots, b_m\}$, $DSig(b_1||\ldots||b_m), pk_{DSig})$ to $\mathcal{A}$.
3. Now the role of the real world server is played by $\mathcal{SIM}$. On receiving $(\overline{X} = \{b_1, \ldots, b_m\}, DSig(b_1||\ldots||b_m), pk)$ from $\mathcal{A}$, the simulator $\mathcal{SIM}$ checks the correctness of the signature $DSig(b_1||\ldots||b_m)$ with the help of the public key pk_{DSig}. $\mathcal{SIM}$ aborts if the verification fails, else sends the set X to T in order to play the role of the ideal world client, while the ideal world server sends $Y = \{y_1, \ldots, y_w\}$ to T. The incorruptible trusted party T, in turn computes the ideal functionality $\mathcal{F}_{APSI-CA}$ on the inputs X, Y and sends $X \cap Y$ to $\mathcal{SIM}$.
4. The simulator $\mathcal{SIM}$ constructs a set $\widehat{Y} = \{\hat{y}_1, \ldots, \hat{y}_v\}$ whose elements are 0 and 1. Note that $|X \cap Y|$ many members of $\widehat{Y}$ are chosen as 0 and $v - |X \cap Y|$ many members of $\widehat{Y}$ are chosen as 1. It encrypts each $\hat{y}_i \in \{0, 1\}$ for $i = 1, \ldots, v$ to determine $\mathsf{Enc}_{pk}(\hat{y}_i) = \bar{y}_i$ and sends $\overline{Y} = \{\mathsf{Enc}_{pk}(\hat{y}_1), \ldots, \mathsf{Enc}_{pk}(\hat{y}_v)\}$ to $\mathcal{A}$. It then outputs whatever $\mathcal{A}$ outputs and terminates.

Thus $\mathcal{SIM}$ provides Ad the same simulation as the simulator S_2 in $\mathbf{Game_2}$. This implies $Pr[IDEAL_{APSI-CA}] = Prob[\mathbf{Game_2}]$, where $IDEAL_{APSI-CA}$ is the ideal world game. Therefore, we have

$$\begin{aligned}
&|Pr[IDEAL_{APSI-CA}] - Pr[REAL_{APSI-CA}]| \\
&= |Prob[\mathbf{Game_2}] - Prob[\mathbf{Game_0}]| \\
&\leq |Prob[\mathbf{Game_2}] - Prob[\mathbf{Game_1}]| \\
&+ |Prob[\mathbf{Game_1}] - Prob[\mathbf{Game_0}]| \\
&\leq \epsilon_1(\kappa) + \epsilon_2(\kappa) = \epsilon_3(\kappa),
\end{aligned}$$

where $\epsilon_3(\kappa)$ is a negligible function. Thus we can conclude that the joint output of $C, S,$ Ad in the real game is indistinguishable from the joint output of $C, S, \mathcal{SIM}$ in the ideal world game.

Table 1 Complexity of PSI-CA

Party	MUL	GE	Hash
Client	$m(nl+l+1)+nv+nl$	$m(n+1)+nl+l$	kw
Server	$v(nl+l+1)+v(n+1)$	$v(n+1)$	kv
Total	$v\left(\frac{(nl+l+1)(k+\ln 2)}{\ln 2}+2n+1\right)+nl$	$v(n+1)\frac{(k+\ln 2)}{\ln 2}+nl+l$	$k(v+w)$

Hash=Number of hash query, $m = 2\frac{vk}{\ln 2}$, MUL=Number of multiplications, GE= Number of group elements.

5 Efficiency

The computation complexity of our PSI-CA is determined by the hash function evaluations and modular multiplications. The computation complexity of our APSI-CA is determined by the hash function evaluations, modular multiplications and signature computation. On the other hand, the communication overheads of the designs are evaluated by counting the number of publicly transmitted group elements.

Complexity of encryption scheme $\mathcal{PKE}$: The key generation algorithm $\mathcal{PKE}$.KGen requires nl modular multiplications. The encryption algorithm $\mathcal{PKE}$.Enc needs $nl + l + 1$ modular multiplications and the decryption algorithm $\mathcal{PKE}$.Dec requires n modular multiplications.

Complexity of digital signature scheme DSig**:** The key generation algorithm DSig.KGen requires lnt modular multiplications.

Table 2 Complexity of APSI-CA without digital signature

Party	MUL	GE	Hash
CA	$m(nl+l+1)$	$m(n+1)$	kw
Client	$nv+nl$	$m(n+1)+2(nl+l)+w$	
Server	$v(nl+l+1)+v(n+1)$	$v(n+1)$	kv
Total	$v\left(\frac{(nl+l+1)(k+\ln 2)}{\ln 2}+2n+1\right)+nl$	$v(n+1)\frac{(k+\ln 2)}{\ln 2}$ $+w+2(nl+l)$	$k(v+w)$

Hash=Number of hash query, $m = 2\frac{vk}{\ln 2}$, MUL=Number of multiplications, GE= Number of group elements.

Table 3 Complexity of APSI-CA with digital signature of [3]

Party	MUL	GE	Hash
CA	$m(nl+l+1)+nt+2nl+lt$	$m(n+1)+n+\kappa$	$kw+2$
Client	$nv+nl$	$m(n+1)+2(nl+l)+w+n+\kappa$	
Server	$v(nl+l+1)+v(n+1)$ $+ln+lt$	$v(n+1)$	$kv+2$
Total	$v\left(\frac{(nl+l+1)(k+\ln 2)}{\ln 2}+2n+1\right)$ $+4nl+nt+2lt$	$v(n+1)\frac{(k+\ln 2)}{\ln 2}+w$ $+2(nl+l)+2n+2\kappa$	$k(v+w)+4$

Hash=Number of hash query, $m = 2\frac{vk}{\ln 2}$, MUL=Number of multiplications, GE= Number of group elements.

The signature algorithm DSig.Sign needs $nt + 2nl + lt$ modular multiplications and 2 hash function evaluations. The verification algorithm DSig.Ver requires $ln + lt$ modular multiplications and 2 hash function evaluations.

We refer Table 1 and Table 2, for the complexity of our schemes. In our PSI-CA, the client C requires nl modular multiplications to generate the public key pk, $m(nl + l + 1)$ modular multiplications to compute $\overline{X} = \{b_1, \ldots, b_m\}$ and nv modular multiplications to decrypt $\overline{Y} = \{\mathsf{Enc}_{pk}(\bar{y}_1), \ldots, \mathsf{Enc}_{pk}(\bar{y}_v)\}$. On the other hand, the server S requires $v(nl + l + 1)$ modular multiplications for computing $\{\vartheta_1, \ldots, \vartheta_v\}$ and $v(n + 1)$ modular multiplications to compute $\overline{Y} = \{\mathsf{Enc}_{pk}(\bar{y}_1), \ldots, \mathsf{Enc}_{pk}(\bar{y}_v)\}$. Moreover, in PSI-CA, the client C needs to transmit $m(n + 1) + nl + l$ group elements $(\overline{X}, pk)$ and the server S requires to send $v(n + 1)$ group elements $\overline{Y}$. Thus the communication and computation complexity of PSI-CA are linear in v and w, i.e., $O(v + w)$, since n, l, k are security parameters.

Similarly, the complexity overheads of our APSI-CA are linear in the size of the sets of the participants. In our APSI-CA, the additional group elements and the computation cost due to the digital signature can easily be determined. For example, if we consider the digital signature of [3], then the computation cost will be increased by $nt + 2nl + lt$ modular multiplications and 2 hash function evaluations due to CA, and $ln + lt$ modular multiplications and 2 hash function evaluations due to the server S. Moreover, the communication cost will be increased by $n + \kappa$ group elements due to CA and $n + \kappa$ group elements due to C. See Table 3 for details.

In Table 4, we provide a comparison summary of our schemes with the schemes of [17] in terms of communication, computation and post-quantum security. Note that our scheme achieves post-quantum security unlike [17]. Thus, our scheme is better than [17] from the security point of view. Moreover, our operations are over the group $\mathbb{Z}_q$, while for [17] the underlying group is $\mathbb{Z}_{pq}$. Therefore, our designs are comparable with the constructions of [17], from the communication and computation complexity point of views.

The authors of [39] proposed a PSI-CA and claimed low communication cost. However, it seems to us that the PSI-CA of [39] has the following drawback: note that the public key pk_s of the sender is public, so the computation of $Enc_{pk_r}(Enc_{pk_s}(Y))$ and then shuffling is consequently unnecessary. This is because the receiver itself can compute $Enc_{pk_s}(y)$ for any y in plaintext space using the

Table 4 Comparison of our protocols with [17] and [39]

Protocol	MUL	GE	Hash	LC	Post-quantum security	drawback: $X \cap Y$ is revealed
PSI-CA of [17]	$2v(\frac{k}{\ln 2}+k)$	$\frac{vk}{\ln 2}+vk+2$	$k(w+v)$	kv	×	no
PSI-CA of [39]	$2(\log q-1)(v+2w)$	$v+2w$	$k(w+v)$	no	×	yes
Our PSI-CA	$v\left(\frac{(nl+l+1)(k+\ln 2)}{\ln 2}+2n+1\right)$ $+nl$	$v(n+1)\frac{(k+\ln 2)}{\ln 2}$ $+nl+l$	$k(v+w)$	no	√	no
APSI-CA of [17] with DSig RSA [49]	$2v(\frac{k}{\ln 2}+k)$ $+2(\log q-1)$	$2\frac{vk}{\ln 2}+kv$ $+w+6$	$k(w+v)+2$	kv	×	no
Our APSI-CA with DSig of [3]	$v\left(\frac{(nl+l+1)(k+\ln 2)}{\ln 2}+2n+1\right)$ $+4nl+nt+2lt$	$v(n+1)\frac{(k+\ln 2)}{\ln 2}+w$ $+2(nl+l)+2n+2\kappa$	$k(v+w)+4$	no	√	no

Hash=Number of hash query, $m = 2\frac{vk}{\ln 2}$, MUL=Number of multiplications, LC=Number of Legendre symbol computations, GE= Number of group elements.

Table 5 Comparison summary of PSI-CA protocols

Protocol	Security model	Adv. model	Security assumption	Comm.	Comp.	Based on	Size hiding	drawback: $X \cap Y$ is revealed
Sch. 1 of [14]	Random oracle	S-H	DDH and GOMDH	$O(w+v)$	$O(w+v)$		no	no
Sch. 2 of [14]	Random oracle	MS, SHC	GOMDH	$O(w+v)$	$O(w+v)$		no	no
[16]	Std	Mali	DDH	$O(w+v)$	$O(w+v)$		no	no
[17]	Std	S-H	QR	$O(w+v)$	$O(w+v)$	BF	yes	no
[51]	Std	S-H		$O(1)$	$O(w+v)$	QC	yes	no
[19]	Std	S-H		$O(\log \eta)$	$O(\log \eta)$	FM	no	no
[39]	Std	S-H		$O(v+w)$	$O(\log_2 q(v+w))$	CE	no	yes
Our work	Std	S-H	DLWE	$O(w+v)$	$O(w+v)$	BF	yes	no

BF= Bloom Filter, GOMDH=Gap-One-More-Diffie-Hellman, Std=Standard, S-H=Semi-honest, QR= Quadratic Residuosity, DDH=Decisional Diffie-Hellman, QC= Quantum Computation, SHC=Semi-honest Client, Mali= Malicious, DLWE= Decisional Learning With Errors, FM= Flajolet-Martin sketch, CE=Commutative Encryption, v= Server's set size, w= Client's set size, η= Intersection size.

Table 6 Comparison summary of APSI-CA protocols

Protocol	Security model	Adv. model	Security assumption	Comm.	Comp.	Based on	Size hiding
[8]	Std	Mali	Strong RSA	$O(wv)$	$O(wv)$	OPE	no
[14]	Random oracle	S-H	GOMDH	$O(w+v)$	$O(w+v)$		no
[17]	Std	MC,SHS	QR	$O(w+v)$	$O(w+v)$	BF	yes
Our work	Std	MC,SHS	DLWE	$O(w+v)$	$O(w+v)$	BF	yes

OPE=Oblivious Polynomial Evaluation,BF= Bloom Filter, GOMDH=Gap-One-More-Diffie-Hellman, Std=Standard, S-H=Semi-honest, QR= Quadratic Residuosity, SHS=Semi-honest Server, MC=Malicious Client, Mali= Malicious, DLWE= Decisional Learning With Errors, v= Server's set size, w= Client's set size, η= Intersection size.

public key pk_s. In other words, the receiver needs not to communicate with the server to evaluate $Enc_{pk_s}(y)$. This will enable the receiver to extract the $X \cap Y$ and even more, which is not desirable in PSI-CA. Moreover, [39] does not achieve post-quantum security and its computation complexity depends on the modular exponentiations, unlike our scheme. Note that in our designs, we require only modular multiplications, which makes them efficient as modular exponential is more expensive than modular multiplication. In particular, if one uses square and multiply method then 1 modular exponentiation is equivalent to approximately $2(\log_2 q - 1)$ modular multiplications over $\mathbb{Z}_q$. Following the argument given in the efficiency section of [17], we can claim that extending the efficient PSI protocols [18, 34] to PSI-CA seems to be a non-trivial task.

In Tables 5 and 6, we give a comparison summary of this research with existing works from an asymptotic point of view.

6 Conclusion

In this paper, we developed an efficient post quantum PSI-CA protocol that determines the cardinality of set intersection. We employed lattice based pubic key encryption PKE of [47] along with a Bloom filter which is a space-efficient probabilistic data structure. As far as we know, our PSI-CA is the *first* post quantum protocol of its kind. Its security is achieved against semi-honest adversaries based on the hardness of the DLWE problem. The complexity overheads of PSI-CA are linear in the size of the sets of the participants. Our scheme requires only modular multiplications which makes it quite efficient. Moreover, to prevent malicious client from arbitrary inclusion in its private set, we extend our PSI-CA to APSI-CA retaining (almost) all the properties. Particularly, our APSI-CA is the *first* that employs lattice based encryption. Extending our work from one-way to two-way is an interesting direction of future work.

Acknowledgements. The authors express their deep appreciation to the editor for promptly handling our paper, as well as to the anonymous referees, whose thorough reading and constructive comments have improved the paper, significantly.

7 References

1 R. Agrawal, A. Evfimievski, and R. Srikant. Information sharing across private databases. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 86–97. ACM, 2003.

2 G. Ateniese, E. De Cristofaro, and G. Tsudik. (If) size matters: Size-hiding private set intersection. In *International Workshop on Public Key Cryptography*, pages 156–173. Springer, 2011.

3 S. Bai and S. D. Galbraith. An improved compression technique for signatures based on learning with errors. In *Cryptographers Track at the RSA Conference*, pages. 28–47. Springer, 2014.

4 B. H Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

5 Z. Brakerski and R. Perlman. Lattice-based fully dynamic multi-key fhe with short ciphertexts. In *Annual International Cryptology Conference*, pages 190–213. Springer, 2016.

6 F. Bruekers, S. Katzenbeisser, K. Kursawe, and P. Tuyls. Privacy-preserving matching of dna profiles. *IACR Cryptology ePrint Archive*, 2008:203, 2008.

7 E. Bursztein, M. Hamburg, J. Lagarenne, and D. Boneh. Openconflict: Preventing real time map hacks in online games. In *2011 IEEE Symposium on Security and Privacy*, pages 506–520. IEEE, 2011.

8 J. Camenisch and G. M Zaverucha. Private intersection of certified sets. In *Financial Cryptography and Data Security*, pages 108–127. Springer, 2009.

9 A. Cerulli, E. De Cristofaro, and C. Soriente. Nothing refreshes like a repsi: Reactive private set intersection. In *Lecture Notes in Computer Science*, volume 16. Springer Verlag, 2018.

10 H. Chen, K. Laine, and P. Rindal. Fast private set intersection from homomorphic encryption. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1243–1255. ACM, 2017.

11 M. Ciampi and C. Orlandi. Combining private set-intersection with secure two-party computation. *IACR ePrint*, 105:2018, 2018.

12 E. De Cristofaro, J. Kim, and G. Tsudik. Linear-complexity private set intersection protocols secure in malicious model. In *Advances in Cryptology-ASIACRYPT 2010*, pages 213–231. Springer, 2010.

13 E. De Cristofaro and G. Tsudik. Experimenting with fast private set intersection. In *Trust and Trustworthy Computing*, pages 55–73. Springer, 2012.

14 E. De Cristofaro, P. Gasti, and G. Tsudik. Fast and private computation of cardinality of set intersection and union. In *Cryptology and Network Security*, pages 218–231. Springer, 2012.

15 E. De Cristofaro and G. Tsudik. Practical private set intersection protocols with linear complexity. In *International Conference on Financial Cryptography and Data Security*, pages 143–159. Springer, 2010.

16 S. K. Debnath and R. Dutta. Efficient private set intersection cardinality in the presence of malicious adversaries. In *Provable Security*, pages 326–339. Springer, 2015.

17 S. K. Debnath and R. Dutta. Secure and efficient private set intersection cardinality using bloom filter. In *International Information Security Conference*, pages 209–226. Springer, 2015.

18 C. Dong, L. Chen, and Z. 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. ACM, 2013.

19 C. Dong and G. Loukides. Approximating private set union/intersection cardinality with logarithmic complexity. *IEEE Transactions on Information Forensics and Security*, 12(11):2792–2806, 2017.

20 B. H. Falk, D. Noble, and R. Ostrovsky. Private set intersection with linear communication from general assumptions. 2018.

21 P. Flajolet and G N. Martin. Probabilistic counting algorithms for data base applications. *Journal of computer and system sciences*, 31(2):182–209, 1985.

22 M. J Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Advances in Cryptology – EUROCRYPT 2004*, pages 1–19. Springer, 2004.

23 M. J Freedman, C. Hazay, K. Nissim, and B. Pinkas. Efficient set intersection with simulation-based security. *Journal of Cryptology*, 29(1):115–155, 2016.

24 S. Ghosh and T. Nilges. An algebraic approach to maliciously secure private set intersection. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques – EUROCRYPT 2019*, pages 154–185. Springer, 2019.

25 S. Ghosh and M. Simkin. The communication complexity of threshold private set intersection. In *Annual International Cryptology Conference – CRYPTO 2019*, pages 3–29. Springer, 2019.

26 O. Goldreich. *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.

27 A. Groce, P. Rindal, and M. Rosulek. Cheaper private set intersection via differentially private leakage. *IACR ePrint*, 239:2019, 2019.

28 C. Hazay and K. Nissim. Efficient set operations in the presence of malicious adversaries. In *Public Key Cryptography–PKC 2010*, pages 312–331. Springer, 2010.

29 C. Hazay and M. Venkitasubramaniam. Scalable multi-party private set-intersection. In *IACR International Workshop on Public Key Cryptography*, pages 175–203. Springer, 2017.

30 S. Hohenberger and S. A Weis. Honest-verifier private disjointness testing without random oracles. In *Privacy Enhancing Technologies*, pages 277–294. Springer, 2006.

31 Y. Huang, D. Evans, and J. Katz. Private set intersection: Are garbled circuits better than custom protocols. In *Network and Distributed System Security Symposium (NDSS). The Internet Society*, 2012.

32 S. Jarecki and X. Liu. Fast secure computation of set intersection. In *Security and Cryptography for Networks*, pages 418–435. Springer, 2010.

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

34 Á. Kiss, J. Liu, T. Schneider, N Asokan, and B. Pinkas. Private set intersection for unequal set sizes with mobile applications. *Proceedings on Privacy Enhancing Technologies*, 2017(4):177–197, 2017.

35 L. Kissner and D. Song. Privacy-preserving set operations. In *Advances in Cryptology–CRYPTO 2005*, pages 241–257. Springer, 2005.

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

37 M. Li, N. Cao, S. Yu, and W. Lou. Findu: Privacy-preserving personal profile matching in mobile social networks. In *2011 Proceedings IEEE INFOCOM*, pages 2435–2443. IEEE, 2011.

38 Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Annual International Cryptology Conference*, pages 36–54. Springer, 2000.

39 S. Lv, J. Ye, S. Yin, X. Cheng, C. Feng, X. Liu, R. Li, Zhaohui Li, Zheli Liu, and Li Zhou. Unbalanced private set intersection cardinality protocol with low communication cost. *Future Generation Computer Systems*, 102:1054–1061, 2020.

40 P. Mukherjee and D. Wichs. Two round multiparty computation via multi-key fhe. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 735–763. Springer, 2016.

41 S. Nagaraja, P. Mittal, C.-Y. Hong, M. Caesar, and N. Borisov. Botgrep: Finding p2p bots with structured graph analysis. In *USENIX Security Symposium*, volume 10, pages 95–110, 2010.

42 A. Narayanan, N. Thiagarajan, M. Lakhani, M. Hamburg, D. Boneh, et al. Location privacy via private proximity testing. In *NDSS*, volume 11, 2011.

43 B. Pinkas, M. Rosulek, N. Trieu, and A. Yanai. Spot-light: Lightweight private set intersection from sparse ot extension. In *Annual International Cryptology Conference-CRYPTO 2019*, pages 401–431. Springer, 2019.

44 B. Pinkas, M. Rosulek, N. Trieu, and A. Yanai. Psi from paxos: Fast, malicious private set intersection. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques-EUROCRYPT 2020*. Springer, 2020.

45 B. Pinkas, T. Schneider, O. Tkachenko, and A. Yanai. Efficient circuit-based psi with linear communication. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques-EUROCRYPT 2019*, pages 122–153. Springer, 2019.

46 B. Pinkas, T. Schneider, and M. Zohner. Faster private set intersection based on ot extension. In *USENIX Security*, volume 14, pages 797–812, 2014.

47 O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):34, 2009.

48 P. Rindal and M. 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. ACM, 2017.

49 R. L Rivest, A. Shamir, and L. Adleman A method for obtaining digital signatures and public-key cryptosystems *Communications of the ACM*, 21(2):120–126, 1978.

50 R.-hua Shi, Y. Mu, H. Zhong, J. Cui, and S. Zhang. An efficient quantum scheme for private set intersection. *Quantum Information Processing*, 15(1):363–371, 2016.

51 R.-hua Shi, Y. Mu, H. Zhong, S. Zhang, and J. Cui. Quantum private set intersection cardinality and its application to anonymous authentication. *Information Sciences*, 370:147–158, 2016.

52 P. W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

53 E. Stefanov, E. Shi, and D. Song. Policy-enhanced private set intersection: sharing information while enforcing privacy policies. In *International Workshop on Public Key Cryptography*, pages 413–430. Springer, 2012.

Appendix

We will display here a few known security models, for completeness.

1. Security Model. A multi-party protocol should satisfy the following basic security requirements: *Correctness* (this ensures that an honest party should receive the correct output at the end of the protocol); *Privacy* (this ensures that no party should learn more than its prescribed output on completion of the protocol); *Fairness* (this ensures that a dishonest party receives its output if and only if the honest party also receives its output).

2. Security Model for Semi-honest Adversary [16, 17, 26] A two-party protocol Π is a random process that computes a function $f = (f_1, f_2)$ from a pair of inputs (one per party) to pair of outputs, that is, $f_i : \{0,1\}^* \rightarrow \{0,1\}^*$, $i = 1, 2$. Let P_1, P_2 be two parties with inputs, x, and y, respectively, both in $\{0,1\}^*$, and outputs, $f_1(x, y), f_2(x, y)$, respectively. A protocol Π is said to be *secure in the semi-honest model* if whatever can be computed by a party after participating in the protocol, it could find

from its input and output only. This can be formalized in the following way, via the simulation paradigm. On the input (x, y), let $\mathsf{View}_i^{\Pi}(x, y) = (w, r^{(i)}, m_1^{(i)}, \ldots, m_t^{(i)})$ be the view of the party P_i during an execution of Π, where $w \in \{x, y\}$ represents the input of the party P_i, $r^{(i)}$ is the outcome of P_i's internal coin tosses, and $m_j^{(i)}$ $(j = 1, 2, \ldots, t)$ represents the j-th message which has been received by P_i during the execution of Π.

Given a deterministic function f, we say that the protocol Π *securely computes* f if there exist probabilistic polynomial-time (PPT) adversaries, denoted by S_1 and S_2, controlling P_1 and P_2 respectively, such that

$$\{S_i(x, f_i(x, y))\}_{x,y\in\{0,1\}^*} \equiv^c \mathsf{View}_i^{\Pi}(x, y)_{x,y\in\{0,1\}^*}, i = 1, 2.$$

3. Security Model for Malicious Adversary [16, 17, 26] The formal security framework of a two-party protocol in malicious model is described below.

The real world: In the real world, a protocol Π is executed and an honest party follows the instructions of Π, but an adversary $\mathcal{A}_i$, controlling the party P_i, can behave arbitrarily. Let the party P_1 have the private input X, the party P_2 have the private input Y and the adversary $\mathcal{A}_i$ have auxiliary input Z. At the end of the execution an honest party outputs whatever is prescribed in the protocol, a corrupted party outputs nothing and an adversary outputs its view which consists of the transcripts available to the adversary. The joint output in the real world is denoted by $\mathsf{REAL}_{\Pi,\mathcal{A}_i(Z)}(X, Y)$.

The ideal process: Let $\mathcal{SIM}_i$ be the ideal process adversary that corrupts a party $P_i, i \in \{1, 2\}$. The ideal process involves an incorruptible trusted third party.

Input: Let X and Y be the inputs of parties P_1 and P_2, respectively, and assume that $\mathcal{SIM}_i$ gets P_i's input and an auxiliary input Z.

Sending inputs to the trusted party: An honest party always sends his original input to the trusted party whereas a corrupted party may send "abort" or an arbitrary input. Let the trusted party receive $(\widetilde{X}, \widetilde{Y})$ (these could be different from X, Y). If any of $\widetilde{X}, \widetilde{Y}$ is "abort", then the trusted party sends $\perp$ to both parties.

The trusted party answers the adversary: The trusted party computes $\mathcal{F} : (\widetilde{X}, \widetilde{Y}) \to (F_1(\widetilde{X}, \widetilde{Y}), F_2(\widetilde{X}, \widetilde{Y}))$ and sends $F_i(\widetilde{X}, \widetilde{Y})$ to $\mathcal{SIM}_i$. Then $\mathcal{SIM}_i$ sends "abort" or "continue" to the trusted party.

The trusted party answers the honest party: If the trusted party receives "continue" from $\mathcal{SIM}_i$, then the trusted party sends $F_j(\widetilde{X}, \widetilde{Y})$ to the honest party $P_j, j \in \{1, 2\} \setminus \{i\}$. Otherwise, the trusted party sends $\perp$ to the honest party.

Output: An honest party always outputs the output value it obtained from the trusted party. The corrupted party outputs nothing. The adversary outputs his view. The joint output of the ideal process is denoted by $\mathsf{IDEAL}_{\mathcal{F},\mathcal{SIM}_i(Z)}(X, Y)$.

Simulatability: Let Π be a protocol and $\mathcal{F}$ be a functionality. Then protocol Π is said to *securely compute* $\mathcal{F}$ in the malicious model if for every PPT adversary $\mathcal{A}_i$ in the real world, there exists a PPT adversary $\mathcal{SIM}_i$ in the ideal model, such that for every $i \in \{1, 2\}$, $\mathsf{IDEAL}_{\mathcal{F},\mathcal{SIM}_i(Z)}(X, Y) \equiv^c \mathsf{REAL}_{\Pi,\mathcal{A}_i(Z)}(X, Y)$.