Advances in Mathematics of Communications doi:10.3934/amc.2020071

SECURE AND EFFICIENT MULTIPARTY PRIVATE SET INTERSECTION CARDINALITY

Sumit Kumar Debnath*

Department of Mathematics
National Institute of Technology Jamshedpur
Jamshedpur-831014, India

Pantelimon Stănică

Department of Applied Mathematics
Naval Postgraduate School, Monterey, CA 93943, USA

Nibedita Kundu

Department of Mathematics
The LNM Institute of Information Technology, Jaipur-302031, India

Tanmay Choudhury

Department of Mathematics
National Institute of Technology Jamshedpur, Jamshedpur-831014, India

(Communicated by Sihem Mesnager)

Abstract. In the field of privacy preserving protocols, Private Set Intersection (PSI) plays an important role. In most of the cases, PSI allows two parties to securely determine the intersection of their private input sets, and no other information. In this paper, employing a Bloom filter, we propose a Multiparty Private Set Intersection Cardinality (MPSI-CA), where the number of participants in PSI is not limited to two. The security of our scheme is achieved in the standard model under the Decisional Diffie-Hellman (DDH) assumption against semi-honest adversaries. Our scheme is flexible in the sense that set size of one participant is independent from that of the others. We consider the number of modular exponentiations in order to determine computational complexity. In our construction, communication and computation overheads of each participant is $O(v_{\mathsf{max}}k)$ except that the complexity of the designated party is $O(v_1)$, where v_{max} is the maximum set size, v_1 denotes the set size of the designated party and k is a security parameter. Particularly, our MSPI-CA is the *first* that incurs *linear* complexity in terms of set size, namely $O(nv_{\mathsf{max}}k)$, where n is the number of participants. Further, we extend our MPSI-CA to MPSI retaining all the security attributes and other properties. As far as we are aware of, there is no other MPSI so far where individual computational cost of each participant is independent of the number of participants. Unlike MPSI-CA, our MPSI does not require any kind of broadcast channel as it uses star network topology in the sense that a designated party communicates with everyone else.

2010 *Mathematics Subject Classification:* Primary: 94A60, 68M12; Secondary: 68P30.
Key words and phrases: MPSI-CA, MPSI, semi-honest adversary, flexibility, Bloom filter.
* Corresponding author: sdebnath.math@nitjsr.ac.in.

1. Introduction

The widespread use of Internet greatly facilitates the distribution and exchange of information. Immediate access to content with low cost delivery is one of the main benefits Internet based distribution brings and has the potential to open up new markets. However, these raises privacy issues regarding intellectual property and copyright due to the vulnerability nature of digital contents for unauthorized distribution and use.

With the advent of Internet and distributed computing, the necessity of privacy preserving data sharing increases rapidly. In this field, one interesting problem arises when the participants wish to learn the intersection of their data sets secretly, but not more than that. PSI is ideal to solve this problem. It is mostly executed between two parties, but it can be extended to a multiparty environment in the context of PSI. This multiparty private set intersection is referred as MPSI, and has several application. For instance, a central investigative agency (e.g., CBI) wants to compare its list of suspects with the lists of local investigative agencies (e.g., local police, military, BSF, etc.). In this case, neither of the agencies will reveal their whole list of suspects to the other.

Privacy and correctness are two most important properties for an MPSI, where privacy ensures that none of the parties learn beyond the intersection and correctness means that each of the participants learn the correct output. Apart from privacy and correctness, flexibility is another desirable feature in the context of MPSI. If an MPSI is flexible that implies that the choice of input set of a party is independent from the others.

In several practical scenarios, the participants want to jointly determine the cardinality of the intersection rather than the contents. For example, suppose $n(\geq 2)$ different health organizations are doing a survey on a particular disease in a village and they wish to determine the number of common villagers who are suffering from that disease. However, none of them will disclose their list of suspects to other. Note that revealing the name of the suspects may create an impact on patient's mind. In such scenarios, we need the cardinality version of the MPSI, known as MPSI-CA. Designing efficient and flexible MPSI-CA is a challenging task.

1.1. Related works.

- **Two-party Private Set Intersection.** We now give an overview of prior works on two-party PSI protocols by classifying them in four groups based on the constructions as follows:
  (i) Oblivious Polynomial Evaluation (OPE) Based PSI: The concept of PSI relying on OPE was introduced by Freedman et al. [32], where the basic idea is to represent a set as a polynomial. Utilizing OPE and and *additively homomorphic encryption* (AHE), Kissner and Song [46] designed a PSI protocol. Following this work, Camenisch and Zaverucha [7] proposed a PSI based on OPE, where the inputs need to be certified by a trusted party. The work of [32] was further improved by Hazay and Nissim [38]. While the constructions of [32, 38] are one-way in the sense that at the end of the protocol only one of the participants learns the intersection, the constructions of [46, 7] are two-way, meaning that at the end both parties receive the intersection. None of the constructions from [32, 38, 46, 7] achieve linear computation complexity. Recently, Dong et al. [25] employed an OPE technique to construct the first fair two-way PSI protocol

in the standard model against malicious entities with the help of a semi-trusted third party. Fairness ensures that either both the involved parties receive or none of them receive the intersection of their private input sets at the completion of the protocol.

(ii) Pseudorandom Function (PRF) Based PSI: Hazay and Lindell [37] demonstrated how to obtain a PSI relying on *Oblivious Pseudorandom Function* (OPRF) which is a two party protocol that enables a sender with private key k and a receiver with private input x to securely compute a *pseudorandom function* (PRF) $f_k(x)$. Later, Jarecki and Liu [42] adopted AHE to extend the work of [37] in the standard model against malicious adversaries. In the following year, Jarecki and Liu [43] introduced the idea of the *unpredictable function* (UPF) based PSI protocol, where UPF works similar to OPRF. Recently, Hazay [36] gave a construction of an efficient PSI based on algebraic PRF. All these constructions [37, 42, 43, 36] are one-way achieving linear complexity. More recently, the authors of [23] proposed a two-way fair PSI protocol relying on two-way OPRF with linear complexity over composite order group.

(iii) Decisional Diffie-Hellman (DDH) Based PSI: A sequence of one-way PSI protocols [16, 15, 17] was proposed by De Cristofaro et al. using random hash functions and zero-knowledge proofs. All these constructions attain linear complexity. The work of Huang et al. [41] showed how to employ garbled circuit (GC) in designing a PSI protocol. The scheme is secure under the Decisional Diffie-Hellman (DDH) assumption in the ROM against semi-honest adversaries and achieves linear communication and $\Theta(v \log v)$ as computational complexity. Recently, Debnath and Dutta [21] designed a fair optimistic two-way PSI over prime order group. The scheme is optimistic in the sense that it uses an off-line semi-trusted third party. The security of this scheme is achieved in malicious environment without random oracles.

(iv) Bloom Filter (BF) Based PSI: A Bloom filter [2] is a data structure that represents a set by an array with entries 0 or 1. It exhibits itself as an useful tool to scale large data sets. The first Bloom filter based protocol was proposed by Many et al. [48], where the participants jointly execute AND of their Bloom filter to get the intersection. However, this protocol does not remain secure as it reveals information about the other party's set. Following [48], Kerschbaum [44] gave a construction of Bloom filter based PSI by incorporating Goldwasser-Micali encryption [35]. The security of this protocol is achieved in the semi-honest environment with linear complexity. Later, Dong et al. [26] combined an oblivious transfer together with a Bloom filter to construct two PSI protocols. One of the constructions of [26] is secure in the semi-honest adversarial model, while the other one is secure in malicious adversarial model under the Computational Diffie-Hellman (CDH) assumption. In the subsequent year, Debnath and Dutta proposed a sequence of PSI protocols in [18, 19, 20] employing a Bloom filter retaining linear complexity. In [45], Kiss et al. transformed four existing PSI protocols into the precomputation form such that in the setup phase the communication is linear only in the size

of the larger input set, while in the online phase the communication is linear in the size of the smaller input set.

(v) Other Paradigm Based PSI: Utilizing fully homomorphic encryption, Chen et al. [9] build a PSI in the honest-but-curious setting. Later, Rindal and Rosulek [50] proposed a PSI employing dual execution. In the following, the concept of Reactive PSI was introduced by Cerulli et al. [8]. In [11], Ciampi and Orlandi presented PSI protocol based on special purpose oblivious transfer (OT). Later, Falk et al. [29] came up with the an improved hashing-based generic PSI in semi-honest environment.

- **Multiparty Private Set Intersection.** In the last few years, although there has been a lot of research works in the direction of two-party PSI, there are only a few constructions of MPSI in the existing literature. Kissner and Song [46] designed the first secure MPSI protocol employing OPE and AHE. Their construction achieve quadratic complexity. Later, Sang and Shen [51] implemented a new MPSI protocol incurring quadratic overhead in the size of the input sets. Following that, some work on MSPI was presented in [52] in the honest majority setting, and they used bilinear groups in their construction. These constructions were further improved by Cheon et al. [10], where the dependency on the input sets is reduced from quadratic to quasi-linear. However, the communication and computation overhead per player grow quadratically with the number of participants. In [12], Dachman-Soled et al. build a multivariate polynomials based MPSI protocol. Their construction attains $O\left(n \cdot v_{\mathsf{max}} + v_{\mathsf{max}} \cdot \log^2 v_{\mathsf{max}}\right)$ and $O\left(n \cdot v_{\mathsf{max}}^2\right)$ as communication and computation complexity respectively, where n is the number of participants and v_{max} is the maximum over all input set sizes. Later, a Bloom filter based approach in MPSI was proposed by Miyaji and Nishida [49], where the security is achieved in a semi-honest environment. Their construction attains $O(n \cdot v_{\mathsf{max}})$ and $O\left(n \cdot v_{\mathsf{max}}\right)$ as communication and computation overhead complexities for the designated party. Hazay and Venkitasubramaniam [39] proposed an MPSI protocol utilizing the two-party PSI protocol of Freedman et al. [32], and very recently, Kolesnikov et al. [47] presented a new paradigm for MPSI in a semi-honest setting from symmetric key techniques.
- **Private Set Intersection Cardinality.** Agrawal et al. [1] introduced the concept of two-party PSI-CA in a semi-honest setting under the DDH assumption. Utilizing OPE, Hohenberger and Weis [40] constructed an efficient two-party PSI-CA that offers better performance over the PSI-CA obtained by extending the two-party PSI scheme of Freedman et al. [32]. Later, Kissner and Song [46] came up with the construction of MPSI-CA relying on OPE. Following this work, Camenisch and Zaverucha [7] constructed a fair two-party PSI-CA protocol for certified sets based on OPE. De Cristofaro et al. [14] designed a two-party PSI-CA with linear complexity. A sequence of two-party PSI-CA [18, 19, 21, 22] are presented by Debnath and Dutta all having linear complexity. Recently, Freedman et al. [31] modified their work of [32] to construct a two-party PSI-CA achieving security in semi-honest environment without random oracles. This scheme also have linear complexity. Employing quantum computation [53], Shi et al. [53] designed a two-party PSI-CA protocol attaining linear complexity. More recently, Dong and Loukides [27] developed an approximate PSI-CA protocol based on the Flajolet-Martin (FM) sketch [30] with logarithmic complexity.

1.2. OUR CONTRIBUTION. In this paper, our main focus is to design efficient MPSI-CA and extend it to MPSI.

- We first give a construction of MPSI-CA employing a space-efficient probabilistic data structure (Bloom filter) along with ElGamal encryption and threshold ElGamal encryption. The security of our MPSI-CA is achieved in semi-honest environment without random oracles under the Decisional Diffie-Hellman (DDH) assumption. The communication complexity of our protocol is linear in the input sizes i.e. $O\left(\sum_{i=1}^{n} v_i k\right)$, k being a security parameter. While the computation cost of each participant is $O(v_{\mathsf{max}} k)$ except for the designated party, for which the cost is $O(v_1)$. Here v_{max} is the maximum set size of the participants and v_1 denotes the set size of the designated party. Our scheme is flexible as each party's input size is independent from the others. To the best of our knowledge, the only other existing MPSI-CA is due to Kissner and Song [46]. In [46], the authors proposed an MPSI-CA with $O(n^2 v_{\mathsf{max}})$ and $O(n^2 v_{\mathsf{max}}^2)$ as communication and computation overheads. Compared to [46], our MPSI-CA is more efficient in terms of both the communication and computation complexity. In particular, our MPSI-CA is the *first* to achieve *linear* complexity in the input set sizes.
- We next extend our MPSI-CA to an MPSI protocol without changing the security attributes. Similar to [39], we use a star network topology instead of point-to-point fully connected network. In this setting, a single designated party, communicates individually with every other party via a variant of the two-party PSI of [13]. The crucial point of this topology is that all parties need not be online at the same time. Our MPSI does not require any broadcast channel during its execution as all the communication is performed only between the designated party and each other party at a point-to-point level. In contrast to [51, 52, 10, 12, 46], communication complexity of our protocol is linear in the input sizes i.e. $O\left(\sum_{i=1}^{n} v_i k\right)$. Computation cost of each participant is $O(v_{\mathsf{max}} k)$ except the designated party, for which the cost is $O(v_1)$. Unlike the existing protocols [47, 39, 49, 51, 52, 10, 12, 46], individual computation complexity of each participant does not depend on the number of participants n in our scheme. Similar to [49], our scheme is flexible as each party's input set size is independent from the others.

1.3. ORGANIZATION. The rest of our paper is organized as follows. In Section 2, we give preliminaries. The constructions of our MPSI-CA and MPSI are described in Section 3. Security proofs and efficiency analysis of our designs are given in Section 4 and Section 5, respectively. Finally, we conclude the paper in Section 6.

2. PRELIMINARIES

Throughout the paper, the notations κ, $\perp$, $x \leftarrowtail X$, $a \leftarrow A$ and $\{\mathcal{X}_t\}_{t\in\mathbb{N}} \equiv^c \{\mathcal{Y}_t\}_{t\in\mathbb{N}}$ are, respectively, used to represent "security parameter", "null string", "variable x is chosen uniformly at random from set X", "a is output of the procedure A" and "the distribution ensemble $\{\mathcal{X}_t\}_{t\in\mathbb{N}}$ is computationally indistinguishable from the distribution ensemble $\{\mathcal{Y}_t\}_{t\in\mathbb{N}}$". Informally, $\{\mathcal{X}_t\}_{t\in\mathbb{N}} \equiv^c \{\mathcal{Y}_t\}_{t\in\mathbb{N}}$ means for all probabilistic polynomial time (PPT) distinguisher $\mathcal{Z}$, there exists a negligible function $\epsilon(t)$ such that $|Prob_{x\leftarrow\mathcal{X}_t}[\mathcal{Z}(x)=1] - Prob_{x\leftarrow\mathcal{Y}_t}[\mathcal{Z}(x)=1]| \leq \epsilon(t)$. Recall that a function $\epsilon : \mathbb{N} \to \mathbb{R}$ 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.

• **Decisional Diffie-Hellman (DDH) Assumption** [3]**:** An algorithm $\mathcal{A}$ for solving the DDH problem takes as input $\langle g^a, g^b, g^{ab}, g^c\rangle$ and decides whether $g^c = g^{ab}$, where $\mathbb{G} = \langle g\rangle$ is a cyclic group of order n and $a, b, c \leftarrow\!\!\!\!\!\!\rightharpoonup \mathbb{Z}_n$. The advantage of $\mathcal{A}$ in solving the DDH problem is denoted by $\mathsf{Adv}_{\mathcal{A}}^{DDH}$ and is defined as

$$\mathsf{Adv}_{\mathcal{A}}^{DDH} = |Prob[\mathcal{A}(g, g^a, g^b, g^{ab}) = 1] - Prob[\mathcal{A}(g, g^a, g^b, g^c) = 1]|.$$

Definition 2.1. The DDH problem is (κ, t)-hard in $\mathbb{G}$ if for every PPT algorithm $\mathcal{A}$ running in time t, $\mathsf{Adv}_{\mathcal{A}}^{DDH}$ is a negligible function of κ.

2.1. Additively homomorphic encryption [5]. We describe below additively homomorphic encryption schemes: the ElGamal encryption [28] and the threshold ElGamal encryption [24] which are semantically secure provided DDH problem is hard in the underlying group.

ElGamal encryption: The ElGamal encryption is an additively homomorphic encryption $\mathcal{EL} = (\mathcal{EL}.\mathsf{Setup}, \mathcal{EL}.\mathsf{KGen}, \mathcal{EL}.\mathsf{Enc}, \mathcal{EL}.\mathsf{Dec})$, defined as follows:

- $\mathcal{EL}.\mathsf{Setup}(1^\kappa) \to (\mathsf{par})$. On input 1^κ, a trusted authority outputs a public parameter $\mathsf{par}=(p, q, g)$, where p, q are primes such that q divides $p-1$ and g is a generator of the unique cyclic subgroup $\mathbb{G}$ of $\mathbb{Z}_p^*$ of order q.
- $\mathcal{EL}.\mathsf{KGen}(\mathsf{par}, P_i) \to (epk_{P_i}, esk_{P_i})$. User P_i chooses $a_i \leftarrow\!\!\!\!\!\!\rightharpoonup \mathbb{Z}_q$, computes $y_{P_i} = g^{a_i}$, reveals $epk_{P_i} = y_{P_i}$ as his public key and keeps $esk_{P_i} = a_i$ secret to himself.
- $\mathcal{EL}.\mathsf{Enc}(\mathsf{m}, epk_{P_i}, \mathsf{par}, r) \to (\mathsf{eE}_{epk_{P_i}}(\mathsf{m}))$. The encryptor encrypts a message $\mathsf{m} \in \mathbb{Z}_q$ using the public key $epk_{P_i} = y_{P_i}$ by computing the ciphertext tuple $\mathsf{eE}_{epk_{P_i}}(\mathsf{m}) = (\alpha, \beta) = (g^r, g^{\mathsf{m}} y_{P_i}^r)$, where $r \leftarrow\!\!\!\!\!\!\rightharpoonup \mathbb{Z}_q$.
- $\mathcal{EL}.\mathsf{Dec}(\mathsf{eE}_{epk_{P_i}}(\mathsf{m}), esk_{P_i}) \to (\mathsf{m})$. On receiving the ciphertext tuple $\mathsf{eE}_{epk_{P_i}}(\mathsf{m}) = (\alpha, \beta) = (g^r, g^{\mathsf{m}} y_{P_i}^r)$, the decryptor P_i decrypts it using the secret key $esk_{P_i} = a_i$ by computing $\frac{\beta}{(\alpha)^{a_i}} = \frac{g^{\mathsf{m}}(g^{a_i})^r}{(g^r)^{a_i}} = g^{\mathsf{m}}$ and then finding m by running an exhaustive search.

The threshold ElGamal encryption $\mathcal{TEL} = (\mathcal{TEL}.\mathsf{Setup}, \mathcal{TEL}.\mathsf{KGen}, \mathcal{DEL}.\mathsf{Enc}, \mathcal{TEL}.\mathsf{Dec})$ is executed among $P_1, \ldots, P_n$ as follows:

- $\mathcal{TEL}.\mathsf{Setup}(1^\kappa) \to (\mathsf{par})$. It is the same as $\mathcal{EL}.\mathsf{Setup}$.
- $\mathcal{TEL}.\mathsf{KGen}(\mathsf{par}) \to (pk, sk)$. Each participant $P_i, i = 1, \ldots, n$ selects $a_i \leftarrow\!\!\!\!\!\!\rightharpoonup \mathbb{Z}_q$ and publishes $y_{P_i} = g^{a_i}$. The public key of $\mathcal{TEL}$ is set to be $pk = h = \prod_{i=1}^{n} y_{P_i} = g^{\sum_{i=1}^{n} a_i}$. This implicitly sets the secret key of $\mathcal{TEL}$ as $sk = \sum_{i=1}^{n} a_i$. Note that sk is not known to anyone under the hardness of DLP in $\mathbb{G}$.
- $\mathcal{TEL}.\mathsf{Enc}(\mathsf{m}, pk, \mathsf{par}, r) \to (\mathcal{TEL}.\mathsf{Enc}_{pk}(\mathsf{m}))$. The encryptor encrypts a message $\mathsf{m} \in \mathbb{Z}_q$ using the public key $pk = h = g^{\sum_{i=1}^{n} a_i}$ and computes the ciphertext tuple $\mathcal{TEL}.\mathsf{Enc}_{pk}(\mathsf{m}) = (\alpha, \beta) = (g^r, g^{\mathsf{m}} h^r)$, where $r \leftarrow\!\!\!\!\!\!\rightharpoonup \mathbb{Z}_q$.
- $\mathcal{TEL}.\mathsf{Dec}(\mathcal{TEL}.\mathsf{Enc}_{pk}(\mathsf{m}), \{a_i\}_{i=1}^n) \to (\mathsf{m} \vee \perp)$. Given a ciphertext $\mathcal{TEL}.\mathsf{Enc}_{pk}(\mathsf{m}) = (\alpha, \beta) = (g^r, g^{\mathsf{m}} h^r)$, each participant P_i shares $\alpha_i = \alpha^{a_i}$. Then they recover g^{m} as $\frac{\beta}{\prod_{i=1}^{n} \alpha_i} = \frac{\beta}{(\alpha)^{\sum_{i=1}^{n}(a_i)}} = \frac{g^{\mathsf{m}} h^r}{g^{r(\sum_{i=1}^{n}(a_i))}} = \frac{g^{\mathsf{m}} h^r}{h^r} = g^{\mathsf{m}}$; otherwise outputs $\perp$. By running an exhaustive search, the message m can be extracted from g^{m}.

Remark 1. Note that if the message m is 0 then $g^{\mathsf{m}} = 1$. Thus, in order to check that whether a chipertext $\mathcal{TEL}.\mathsf{Enc}_{pk}(\mathsf{m})$ or $\mathcal{EL}.\mathsf{Enc}_{pk}(\mathsf{m})$ decrypts to 0, the decryptor computes g^{m} and checks it is 1.

2.2. BLOOM FILTER [2]. A Bloom filter (BF) is a data structure that represents a set $X = \{x_1, \ldots, x_v\}$ of v elements by an array of m bits and uses k independent uniform hash functions $H_{\mathsf{Bloom}} = \{h_1, \ldots, h_k\}$ with $h_i : \{0,1\}^* \to \{1, \ldots, m\}$ for $i = 1, \ldots, k$ to insert elements or check the presence of elements in that array. Let $\mathsf{BF}_X \in \{0,1\}^m$ represent a Bloom filter for the set X and $\mathsf{BF}_X[i]$ denotes its i-th bit, $i = 1, \ldots, m$. We describe below a variant of a Bloom filter [2] that performs three operations- Initialization, Add and Check.

- *Initialization*: Set 1 to all the bits of an m-bit array, which is an empty Bloom filter.
- *Add(x)*: To add an element $x \in X \subseteq \{0,1\}^*$ into a Bloom filter, x is hashed with the k hash functions in $H_{\mathsf{Bloom}} = \{h_1, \ldots, h_k\}$ to get k indices $h_1(x), \ldots, h_k(x)$. Set 0 to the bit positions of the Bloom filter having indices $h_1(x), \ldots, h_k(x)$. Repeat the process for each $x \in X$ to get $\mathsf{BF}_X \in \{0,1\}^m$ – the Bloom filter for the set X.
- *Check($\hat{x}$)*: Given BF_X, to check whether an element $\hat{x}$ belongs to X without knowing X, $\hat{x}$ is hashed with the k hash functions in $H_{\mathsf{Bloom}} = \{h_1, \ldots, h_k\}$ to get k indices $h_1(\hat{x}), \ldots, h_k(\hat{x})$. 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.

Bloom filter parameters (optimal): A Bloom yields false positive i.e., an element $y \notin X$ may pass the membership test. This is due to the fact that each of $\mathsf{BF}_X[h_j(y)]$ could be 1 for $j = 1, \ldots, k$ even if $y \notin X$. The probability that a certain bit is not set to 0 by a certain hash function during insertion of an element is $1 - \frac{1}{m}$. Since there are k independent uniform hash functions, the probability that a certain bit is not set to 0 by any of the hash functions is $(1 - \frac{1}{m})^k$. If we insert all the v elements to the Bloom filter then the probability that a certain bit is still 1 is $(1 - \frac{1}{m})^{vk}$. Thus the probability that a certain bit in the Bloom filter BF_X is set to 0 is $z = 1 - (1 - \frac{1}{m})^{vk}$. If ϵ is the false positive rate of the Bloom filter BF_X then according to [4], $\epsilon \leq z^k \times (1 + O(\frac{k}{z}\sqrt{\frac{\ln m - \ln z}{m}}))$ which is negligible function in k. In practice, during the construction of Bloom filter for a set of v elements, we choose the values of k and m such that ϵ is capped at a specific low value (e.g. 2^{-80}). According to [26], performance optimality of Bloom filter is attained if $k = \frac{m}{v} \ln 2$ and $m \geq n \ln_2 e \cdot \ln_2 \frac{1}{\epsilon}$, where e is the as usual base of natural logarithm. Thus, by minimizing m i.e., by choosing optimal $m = v \ln_2 e \cdot \ln_2 \frac{1}{\epsilon}$, the optimal value of k is obtained as $k = \ln_2 \frac{1}{\epsilon}$. In the rest of the paper, we will assume that the optimal parameters are chosen.

3. PROTOCOL

In this section, we describe the construction of MPSI-CA followed by MPSI.

3.1. MULTIPARTY PRIVATE SET INTERSECTION CARDINALITY (MPSI-CA). The MPSI-CA protocol is executed among n parties $P_1, \ldots, P_n$ with the private input sets $X_1, \ldots, X_n$, respectively, with $|X_i| = v_i$ for $i = 1, \ldots, n$, where one party, say, P_1 is designated to determine the intersection of its private set with the others' private sets. The protocol completes in two phases : (I) **Setup** and (II) **Set Intersection Cardinality**. In the **Setup** phase, the parties jointly generate a public key pk for a threshold additively homomorphic encryption such as ElGamal encryption scheme and Bloom filter parameters $(m, H_{\mathsf{Bloom}} = \{h_1, \ldots, h_k\})$ with

v_{max} = maximum of $\{v_1, \ldots, v_n\}$. The **Set Intersection Cardinality** phase determines $|\bigcap_{i=1}^{n} X_i|$ and by invoking three algorithms: MPSI-CA.request, MPSI-CA.response, and MPSI-CA.computation. On MPSI-CA.request, each party $P_i (i = 2, \ldots, n)$ generates a Bloom filter $\mathsf{BF}_{X_i} \in \{0,1\}^m$ of its private set X_i, encrypts BF_{X_i} using pk and sends the encrypted Bloom filter $E_{pk}(\mathsf{BF}_{X_i})$ to P_1. The party P_1 then invokes MPSI-CA.response, where for each $x_l \in X_1$, $l = 1, \ldots, v_1$, the party P_1 extracts k ciphertexts corresponding to $h_1(x_l), \ldots, h_k(x_l)$ from $E_{pk}(\mathsf{BF}_{X_i})$ that contains m ciphertexts, for each $i = 2, \ldots, n$ and multiplies all these $k(n-1)$ ciphertexts. This yields a resulting ciphertext C_l corresponding to $x_l \in X_1$, for $l = 1, \ldots, v_1$ which decrypts to zero if the corresponding x_l is in $\bigcap_{i=1}^{n} X_i$. The party P_1 then publishes all these v_1 resulting ciphertexts $C_1, \ldots, C_{v_1}$. We initialize, and for $l = 1, \ldots, v_1$, we let $\mathsf{CT}_l^{(1)} := C_l$. Now for $i = 1, \ldots, n-1$, the party P_{i+1} randomizes the set $\{\mathsf{CT}_1^{(i)}, \ldots, \mathsf{CT}_{v_1}^{(i)}\}$ using a random permutation ϕ_i, keeps the permutation secret to itself and broadcasts the resulting set of ciphertexts $\{\mathsf{CT}_1^{(i+1)}, \ldots, \mathsf{CT}_{v_1}^{(i+1)}\}$. Thus, $\mathsf{CT}_l^{(i+1)} = \mathsf{CT}_{\phi_i^{-1}(l)}^{(i)} \cdot \mathcal{TEL}.\mathsf{Enc}(0, pk, \mathsf{par}, \sigma_l^{(i)})$ with $\sigma_l^{(i)} \leftarrow\!\!\!\!\!\!\rightharpoonup \mathbb{Z}_q$ and $l = 1, \ldots, v_1$. Finally, MPSI-CA.computation is called, whereby all the n participants involved in the threshold decryption to decrypt v_1, resulting in the ciphertexts $\{\mathsf{CT}_1^{(n)}, \ldots, \mathsf{CT}_{v_1}^{(n)}\}$ for P_1. The party P_1 then concludes that $\left|\bigcap_{i=1}^{n} X_i\right|$ equals the number of resulting ciphertexts $\{\mathsf{CT}_l^{(n)}\}_{l=1}^{v_1}$ decrypting to 0. We define MPSI-CA functionality as $\mathcal{F}_{MPSI-CA} : (X_1, \ldots, X_n) \to (|X_1 \bigcap \cdots \bigcap X_n|, \perp, \ldots, \perp)$. The **Setup** phase of our MPSI-CA is depicted in FIGURE 1.

Setup(1^κ) – We use ElGamal encryption $\mathcal{EL}$ and threshold ElGamal encryption $\mathcal{TEL}$ as described in the Section 2.1.

- A trusted authority generates $\mathsf{par} = (p, q, g) \leftarrow \mathcal{EL}.\mathsf{Setup}(1^\kappa)$, where $\mathsf{par} = (p, q, g)$, selects optimal Bloom filter parameters $(m, H_{\mathsf{Bloom}} = \{h_1, \ldots, h_k\})$ with $m = \lceil \frac{kv_{\mathsf{max}}}{\ln 2} \rceil$.
- For $i = 1, \ldots, n$, the party P_i generates $(pk_i, sk_i) \leftarrow \mathcal{EL}.\mathsf{KGen}(\mathsf{par})$, makes pk_i public and keeps sk_i secret, where $pk_i = g^{a_i}$ and $sk_i = a_i$.
- Let $pk = h = \prod_{i=1}^{n}(pk_i) = g^{\sum_{i=1}^{n} a_i}$ and $sk = \sum_{i=1}^{n}(a_i)$. Then (pk, sk) pair serves as the public-secret key pair for $\mathcal{TEL}$. Note that the secret key sk for $\mathcal{TEL}$ is not known to anyone. However, the public key pk for $\mathcal{TEL}$ is publicly computable from $pk_1, \ldots, pk_n$.

FIGURE 1. Setup algorithm of our MPSI-CA

Set Intersection Cardinality – This phase is executed between the parties P_i with the private input set X_i for $i = 1, \ldots, n$, where $|X_i| = v_i$ and $X_1 = \{x_1, \ldots, x_{v_1}\}$. It consists of three algorithms MPSI-CA.request, MPSI-CA.response, and MPSI-CA.computation which we discuss below:

- MPSI-CA.request: For $i = 2, \ldots, n$, the party P_i

(i) generates a Bloom filter BF_{X_i};
(ii) encrypts each entries of BF_{X_i} using the public key pk to get $E_{pk}(\mathsf{BF}_{X_i}) = (C_1^{(i)}, \ldots, C_m^{(i)})$, where $C_j^{(i)} = \mathcal{TEL}.\mathsf{Enc}_{pk}(\mathsf{BF}_{X_i}[j]) = \left(g^{r_{ij}}, g^{\mathsf{BF}_{X_i}[j]} h^{r_{ij}}\right)$ and $r_{ij} \longleftarrow \mathbb{Z}_q$ for $j = 1, \ldots, m$;
(iii) sends $E_{pk}(\mathsf{BF}_{X_i})$ to P_1.

We refer to FIGURE 2 for the interaction among the parties in MPSI-CA.request.

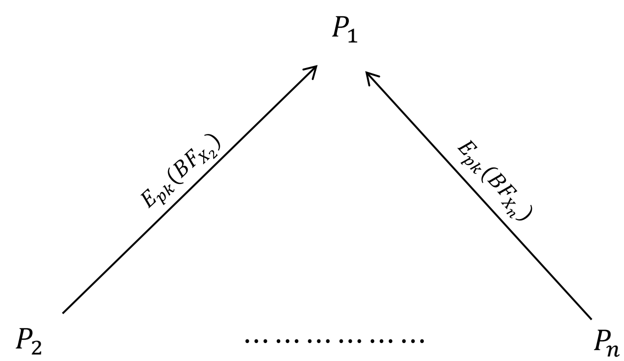

Figure 2. Interaction among parties in MPSI-CA.request

- MPSI-CA.response: The party P_1, on receiving $E_{pk}(\mathsf{BF}_{X_i}) = (C_1^{(i)}, \ldots, C_m^{(i)})$ from each P_i $(i = 2, \ldots, n)$, executes the following steps for each $x_l \in X_1$ $(l = 1, \ldots, v_1)$:
  (i) evaluates the hash values $\mathcal{J} = \{h_1(x_l), \ldots, h_k(x_l)\} \subset \{1, \ldots, m\}$;
  (ii) extracts $C_{h_1(x_l)}^{(i)}, \ldots, C_{h_k(x_l)}^{(i)}$ from $E_{pk}(\mathsf{BF}_{X_i})$ for $i = 2, \ldots, n$;
  (iii) multiplies all these $k(n-1)$ ciphertexts to get a resulting ciphertext
  $$C_l = \prod_{i=2}^{n} \prod_{t=1}^{k} (C_{h_t(x_l)}^{(i)}) = \mathcal{TEL}.\mathsf{Enc}_{pk}\left(\sum_{i=2}^{n} \sum_{t=1}^{k} (\mathsf{BF}_{X_i}[h_t(x_l)])\right)$$
  $$= \left(g^{\sum\limits_{i=2}^{n} \sum\limits_{j \in \mathcal{J}} r_{ij}}, g^{\sum\limits_{i=2}^{n} \sum\limits_{j \in \mathcal{J}} \mathsf{BF}_{X_i}[j]} h^{\sum\limits_{i=2}^{n} \sum\limits_{j \in \mathcal{J}} r_{ij}} \right) = (\alpha_l, \beta_l)$$
  as $C_j^{(i)} = \mathcal{TEL}.\mathsf{Enc}_{pk}(\mathsf{BF}_{X_i}[j]) = \left(g^{r_{ij}}, g^{\mathsf{BF}_{X_i}[j]} h^{r_{ij}}\right)$ and $\mathcal{TEL}$ is additively homomorphic.

  P_1 then broadcasts the resulting ciphertexts $\{C_1, \ldots, C_{v_1}\}$. Now for $i = 1, \ldots, n-1$, the party P_{i+1} randomizes the set $\mathsf{CP}^{(i)} = \{\mathsf{CT}_1^{(i)}, \ldots, \mathsf{CT}_{v_1}^{(i)}\}$ using a random permutation ϕ_i, keeps the permutation secret to itself and broadcasts the resulting set of ciphertexts $\mathsf{CP}^{(i+1)} = \{\mathsf{CT}_1^{(i+1)}, \ldots, \mathsf{CT}_{v_1}^{(i+1)}\}$. Note that for $l = 1, \ldots, v_1$, $\mathsf{CT}_l^{(1)} = C_l$ and for $i = 1, \ldots, n-1$, $\mathsf{CT}_l^{(i+1)} = \mathsf{CT}_{\phi_i^{-1}(l)}^{(i)} \cdot \mathcal{TEL}.\mathsf{Enc}(0, pk, \mathsf{par}, \sigma_l^{(i)})$ with $\sigma_l^{(i)} \longleftarrow \mathbb{Z}_q$ and $l = 1, \ldots, v_1$. The interaction among the participants in MPSI-CA.response is displayed in FIGURE 3.
- MPSI-CA.computation: At this stage, all of the parties $P_1, \ldots, P_n$ have the combined ciphertexts $\{\mathsf{CT}_1^{(n)}, \ldots, \mathsf{CT}_{v_1}^{(n)}\}$, where $\mathsf{CT}_l^{(n)} = (\overline{\alpha}_l, \overline{\beta}_l)$, $l = 1, \ldots, v_1$. For $i = 2, \ldots, n$, each P_i computes $\overline{T}_i = \{(\overline{\alpha}_1)^{a_i}, \ldots, (\overline{\alpha}_{v_1})^{a_i}\}$ using its secret

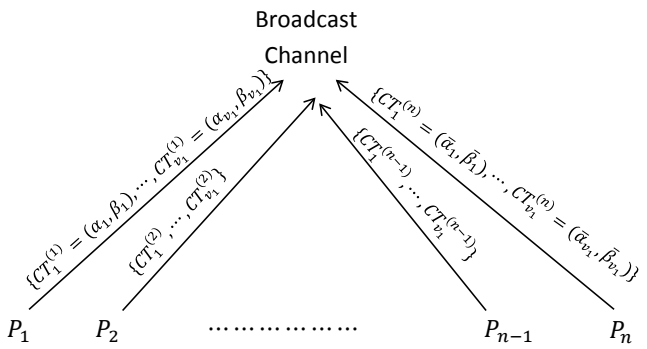

FIGURE 3. Interaction among parties in MPSI-CA.response

key a_i and sends $\overline{T}_i$ to P_1. The party P_1 then chooses a count variable card and for $l = 1, \ldots, v_1$, proceeds as follows:

(i) evaluates $(\overline{\alpha}_l)^{a_1}$ utilizing its secret key a_1;

(ii) computes $\overline{\rho}_l = \prod_{i=1}^{n} (\overline{\alpha}_l)^{a_i}$ using $\{\overline{T}_2, \ldots, \overline{T}_n\}$ and $(\overline{\alpha}_l)^{a_1}$;

(iii) evaluates $\overline{\mu}_l = \frac{\overline{\beta}_l}{\overline{\rho}_l}$ and determines that $\mathsf{CT}_l^{(n)}$ decrypts to 0 if $\overline{\mu}_l = 1$;

(iv) increases card by 1 if $\overline{\mu}_l = 1$.

Finally, the party P_1 outputs card as the cardinality of $\bigcap_{i=1}^{n} X_i$. In MPSI-CA. computation, the interaction among the parties is provided in FIGURE 4.

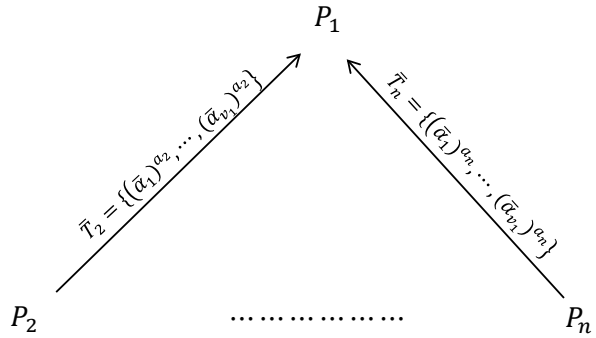

FIGURE 4. Interaction among parties in MPSI-CA.computation

Correctness of MPSI-CA: Note that the set $\{\mathsf{CT}_1^{(n)}, \ldots, \mathsf{CT}_{v_1}^{(n)}\}$ is the same as $\{C_1, \ldots, C_{v_1}\}$, in some order. We assume that $x_\lambda \in X_1$ is associated with $\mathsf{CT}_l^{(n)}$. Let $x_\lambda \in \bigcap_{i=1}^{n} X_i$. Then $x_\lambda \in X_i$ for all $i = 1, \ldots, n$. In other words, x_λ passes the check step for each of the Bloom filter BF_{X_i} $(i = 2, \ldots, n)$, i.e., $\mathsf{BF}_{X_i}[j] = 0$ for all $i = 2, \ldots, n$ and $j \in \mathcal{J} = \{h_1(x_\lambda), \ldots, h_k(x_\lambda)\}$. Thus, we have

$$\mathsf{CT}_l^{(n)} = (\overline{\alpha}_l, \overline{\beta}_l) = \left(g^{\sum_{i=2}^{n} \sum_{j \in \mathcal{J}} r_{ij} + \sigma}, g^{\sum_{i=2}^{n} \sum_{j \in \mathcal{J}} \mathsf{BF}_{X_i}[j]} h^{\sum_{i=2}^{n} \sum_{j \in \mathcal{J}} r_{ij} + \sigma} \right)$$

$$= \left(g^{\sum_{i=2}^{n} \sum_{j \in \mathcal{J}} r_{ij} + \sigma}, g^0 h^{\sum_{i=2}^{n} \sum_{j \in \mathcal{J}} r_{ij} + \sigma} \right),$$

where $\sigma = \sigma^{(2)}_{\phi_2(\lambda)} + \cdots + \sigma^{(n)}_{\phi_n(\ldots \phi_3(\phi_2(\lambda)))}$.

Further, note that

$$\begin{aligned}
\overline{\rho}_l = \prod_{t=1}^{n} (\overline{\alpha}_l)^{a_t} &= \prod_{t=1}^{n} \left(g^{\sum_{i=2}^{n} \sum_{j \in \mathcal{J}} r_{ij} + \sigma} \right)^{a_t} \\
&= g^{\left(\sum_{i=2}^{n} \sum_{j \in \mathcal{J}} r_{ij} + \sigma \right) \sum_{t=1}^{n} a_t} = \left(g^{\sum_{t=1}^{n} a_t} \right)^{\left(\sum_{i=2}^{n} \sum_{j \in \mathcal{J}} r_{ij} + \sigma \right)} \\
&= h^{\sum_{i=2}^{n} \sum_{j \in \mathcal{J}} r_{ij} + \sigma}
\end{aligned}$$

and $\overline{\mu}_l = \frac{\overline{\beta}_l}{\overline{\rho}_l} = g^0 = 1$. Therefore, $\mathsf{CT}_l^{(n)}$ decrypts to 0 if $\overline{\mu}_l = 1$, i.e., if $x_\lambda \in \bigcap_{i=1}^{n} X_i$. In other words, card is increased by 1 if $x_\lambda \in \bigcap_{i=1}^{n} X_i$.

Let us consider $x_\lambda \in X_1$ is associated with $\mathsf{CT}_l^{(n)}$ and $\mathsf{CT}_l^{(n)}$ decrypts to 0. Then $\mathsf{BF}_{X_i}[j] = 0$ for all $i = 2, \ldots, n$ and $j \in \mathcal{J} = \{h_1(x_\lambda), \ldots, h_k(x_\lambda)\}$ by the construction of $\mathsf{CT}_l^{(n)}$. In other words, $x_\lambda \in X_1$ passes the check step for each of the Bloom filter BF_{X_i} $(i = 2, \ldots, n)$. Therefore, $x_\lambda \in X_i$ for all $i = 2, \ldots, n$, except with negligible probability ϵ. This implies that $x_\lambda \in \bigcap_{i=1}^{n} X_i$, except with negligible probability ϵ. Hence, we can ensure that $x_\lambda \in \bigcap_{i=1}^{n} X_i$ if and only if $\mathsf{CT}_l^{(n)}$ decrypts to 0, i.e., card is the cardinality of $\bigcap_{i=1}^{n} X_i$, except with negligible probability ϵ.

3.2. MULTIPARTY PRIVATE SET INTERSECTION (MPSI). Similar to MPSI-CA, MPSI involve n parties $P_1, \ldots, P_n$ with their respective private input sets $X_1, \ldots, X_n$, where $|X_i| = v_i$. We assume that P_1 is the designated party that communicates with the rest of the parties $P_2, \ldots, P_n$. Let us define the functionality for MPSI as $\mathcal{F}_{MPSI} : (X_1, \ldots, X_n) \to (X_1 \bigcap \cdots \bigcap X_n, \perp, \ldots, \perp)$. The protocol completes in two phases: (I) **Setup** and (II) **Set Intersection**. The **Setup** is same as that of MPSI-CA while **Set Intersection** phase completes in 3 rounds and invokes three algorithms: MPSI.$\mathsf{request}$, MPSI.$\mathsf{response}$, and MPSI.$\mathsf{computation}$. We describe below these algorithms.

- MPSI.$\mathsf{request}$: This algorithm is exactly the same as that of MPSI-CA.$\mathsf{request}$.
- MPSI.$\mathsf{response}$: On receiving $E_{pk}(\mathsf{BF}_{X_i}) = (C_1^{(i)}, \ldots, C_m^{(i)})$ from each P_i $(i = 2, \ldots, n)$, the party P_1 does the following, for each $x_l \in X_1$ $(l = 1, \ldots, v_1)$:
  (i) computes the hash values $\mathcal{J} = \{h_1(x_l), \ldots, h_k(x_l)\} \subset \{1, \ldots, m\}$;
  (ii) extracts $C^{(i)}_{h_1(x_l)}, \ldots, C^{(i)}_{h_k(x_l)}$ from $E_{pk}(\mathsf{BF}_{X_i})$ for $i = 2, \ldots, n$;
  (iii) multiplies all these $k(n-1)$ ciphertexts to get a combined ciphertext

$$C_l = \prod_{i=2}^{n} \prod_{t=1}^{k} (C^{(i)}_{h_t(x_l)}) = \mathcal{TEL}.\mathsf{Enc}_{pk} \left(\sum_{i=2}^{n} \sum_{t=1}^{k} (\mathsf{BF}_{X_i}[h_t(x_l)]) \right)$$

$$= \left(g^{\sum_{i=2}^{n} \sum_{j \in \mathcal{J}} r_{ij}}, g^{\sum_{i=2}^{n} \sum_{j \in \mathcal{J}} \mathsf{BF}_{X_i}[j]} h^{\sum_{i=2}^{n} \sum_{j \in \mathcal{J}} r_{ij}} \right) = (\alpha_l, \beta_l),$$

as $C_j^{(i)} = \mathcal{TEL}.\mathsf{Enc}_{pk}(\mathsf{BF}_{X_i}[j]) = \left(g^{r_{ij}}, g^{\mathsf{BF}_{X_i}[j]} h^{r_{ij}}\right)$ and $\mathcal{TEL}$ is additively homomorphic.

Finally, P_1 sends $\{C_1, \ldots, C_{v_1}\}$ to all the other participants $P_2, \ldots, P_n$. In MPSI.response, the interaction among the participants is displayed in FIGURE 5.

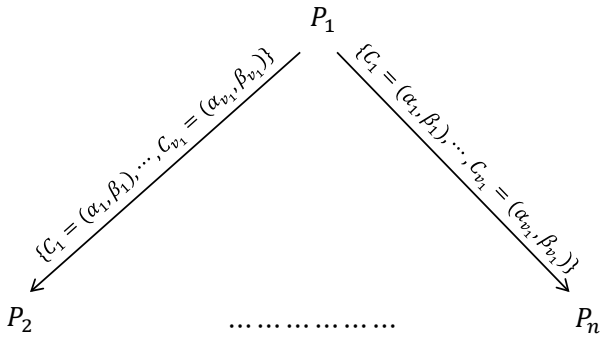

FIGURE 5. Interaction among parties in MPSI.response

- MPSI.computation: At this stage, all the participants $P_1, \ldots, P_n$ have the combined ciphertexts $\{C_1, \ldots, C_{v_1}\}$, where $C_l = (\alpha_l, \beta_l)$, $l = 1, \ldots, v_1$. Now, they involve in $\mathcal{TEL}.\mathsf{Dec}$ as follows in order to decrypt each of $\{C_1, \ldots, C_{v_1}\}$ for P_1 who concludes with the intersection $\bigcap_{i=1}^{n} X_i$:
  (i) For $i = 2, \ldots, n$, each P_i computes $T_i = \{(\alpha_1)^{a_i}, \ldots, (\alpha_{v_1})^{a_i}\}$ using its secret key a_i and sends T_i to P_1.
  (ii) The party P_1 then chooses an empty set W and executes the following steps for $l = 1, \ldots, v_1$:
    (a) evaluates $(\alpha_l)^{a_1}$ utilizing its secret key a_1;
    (b) computes $\rho_l = \prod_{i=1}^{n} (\alpha_l)^{a_i}$ using $\{T_2, \ldots, T_n\}$ and $(\alpha_l)^{a_1}$;
    (c) evaluates $\mu_l = \frac{\beta_l}{\rho_l}$ and determines that C_l decrypts to 0 if $\mu_l = 1$;
    (d) inserts x_l in W if $\mu_l = 1$.

  Finally, the party P_1 outputs W as $\bigcap_{i=1}^{n} X_i$. We refer to FIGURE 6 for the interaction among the parties in MPSI.computation.

Correctness of MPSI: Let us assume that $x_l \in \bigcap_{i=1}^{n} X_i$. Then $x_l \in X_i$ for all $i = 1, \ldots, n$. In other words, x_l passes the check step for each of the Bloom filter BF_{X_i} $(i = 2, \ldots, n)$ i.e., $\mathsf{BF}_{X_i}[j] = 0$ for all $i = 2, \ldots, n$ and $j \in \mathcal{J} = \{h_1(x_l), \ldots, h_k(x_l)\}$.

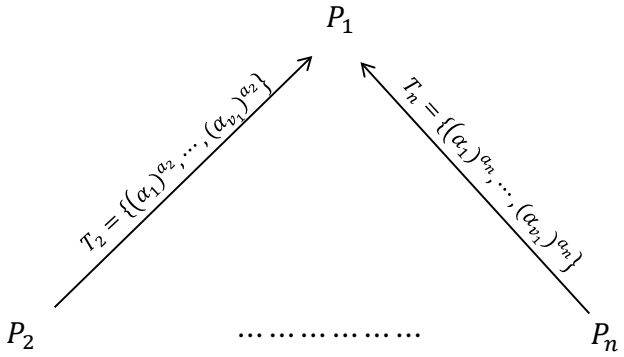

FIGURE 6. Interaction among parties in MPSI.computation

Thus, we have

$$C_l = (\alpha_l, \beta_l)$$
$$= \left(g^{\sum_{i=2}^{n} \sum_{j \in \mathcal{J}} r_{ij}}, g^{\sum_{i=2}^{n} \sum_{j \in \mathcal{J}} \mathsf{BF}_{X_i}[j]} h^{\sum_{i=2}^{n} \sum_{j \in \mathcal{J}} r_{ij}} \right) = \left(g^{\sum_{i=2}^{n} \sum_{j \in \mathcal{J}} r_{ij}}, g^0 h^{\sum_{i=2}^{n} \sum_{j \in \mathcal{J}} r_{ij}} \right).$$

Further, note that

$$\rho_l = \prod_{t=1}^{n} (\alpha_l)^{a_t} = \prod_{t=1}^{n} \left(g^{\sum_{i=2}^{n} \sum_{j \in \mathcal{J}} r_{ij}} \right)^{a_t} = g^{\left(\sum_{i=2}^{n} \sum_{j \in \mathcal{J}} r_{ij} \right) \sum_{t=1}^{n} a_t} = \left(g^{\sum_{t=1}^{n} a_t} \right)^{\sum_{i=2}^{n} \sum_{j \in \mathcal{J}} r_{ij}}$$

$= h^{\sum_{i=2}^{n} \sum_{j \in \mathcal{J}} r_{ij}}$ and $\mu_l = \frac{\beta_l}{\rho_l} = g^0 = 1$. Therefore, C_l decrypts to 0 if $\mu_l = 1$ i.e., if $x_l \in \bigcap_{i=1}^{n} X_i$.

On the other hand, if C_l decrypts to 0 then $\mathsf{BF}_{X_i}[j] = 0$ for all $i = 2, \ldots, n$ and $j \in \mathcal{J} = \{h_1(x_l), \ldots, h_k(x_l)\}$ by the construction of C_l. In other words, $x_l \in X_1$ passes the check step for each of the Bloom filter BF_{X_i} $(i = 2, \ldots, n)$. Therefore, $x_l \in X_i$ for all $i = 2, \ldots, n$ except with negligible probability ϵ. This implies that $x_l \in \bigcap_{i=1}^{n} X_i$ except with negligible probability ϵ. Hence, we can ensure that $x_l \in \bigcap_{i=1}^{n} X_i$ if and only if C_l decrypts to 0 i.e., the set W is $\bigcap_{i=1}^{n} X_i$ except with negligible probability ϵ.

4. SECURITY ANALYSIS

Theorem 4.1. *If the encryption schemes $\mathcal{EL}$ and $\mathcal{TEL}$ are semantically secure and the associated permutations are random, then our proposed MPSI-CA presented in Section 3.1 is a secure computation protocol in standard model against semi-honest adversaries except with negligible probability ϵ, where ϵ is the false positive rate of the Bloom filter.*

Proof. We prove the security of the MPSI-CA by considering two cases:

- Case I: a strict subset $\mathcal{I}_1$ of $\{P_1, \ldots, P_n\}$ is corrupted, and $P_1 \in \mathcal{I}_1$.
- Case II: a strict subset $\mathcal{I}_2$ of $\{P_1, \ldots, P_n\}$ is corrupted, and $P_1 \notin \mathcal{I}_2$.

In each of the cases, we will show that a simulator $\mathcal{SIM}$ can be constructed who simulates the MPSI-CA protocol, the simulator having access to the corrupted party's input and output such that the simulated view is computationally indistinguishable from the real world view. Here, the view of an entity consists of input message of the entity, the outcome of the entity's internal coin tosses and the messages received by the entity during the protocol execution.

Case I (a subset $\mathcal{I}_1$ of $\{P_1, \ldots, P_n\}$ is corrupted, and $P_1 \in \mathcal{I}_1$). Let the simulator $\mathcal{SIM}$ be given access to the corrupted parties' input sets $\{X_i\}_{i\in\mathcal{I}_1}$ and output $|\bigcap_{i=1}^{n} X_i|$. Then $\mathcal{SIM}$ does the following:

- generates $(pk, sk) \leftarrow \mathcal{TEL}.\mathsf{KGen}(1^\kappa)$ and uniformly chooses its random coins R;
- plays the role of the honest parties' by choosing random sets $\{\widetilde{X}_i\}_{i\notin\mathcal{I}_1}$ with $|\widetilde{X}_i| = v_i$, constructing Bloom filters $\{\mathsf{BF}_{\widetilde{X}_i}\}_{i\notin\mathcal{I}_1}$ and encrypting $\{\mathsf{BF}_{\widetilde{X}_i}\}_{i\notin\mathcal{I}_1}$ using pk to get $\{E_{pk}(\mathsf{BF}_{\widetilde{X}_i})\}_{i\notin\mathcal{I}_1}$;
- generates $\mathsf{card} = |\bigcap_{i=1}^{n} X_i|$ many ciphertexts $\{\widehat{C}_1, \ldots, \widehat{C}_{\mathsf{card}}\}$ of the form $\mathcal{TEL}.\mathsf{Enc}_{pk}(0)$ and $v_1 - \mathsf{card}$ many ciphertexts $\{\widehat{C}_{\mathsf{card}+1}, \ldots, \widehat{C}_{v_1}\}$ of the form $\mathcal{TEL}.\mathsf{Enc}_{pk}(r_l)$, where r_l is uniformly chosen from $\mathbb{Z}_q$. Shuffles the set $\zeta = \{\widehat{C}_1, \ldots, \widehat{C}_{v_1}\}$ using a random permutation ϕ over $\{1, \ldots, v_1\}$ in order to get $\chi = \{\widetilde{C}_1, \ldots, \widetilde{C}_{v_1}\}$, where $\widetilde{C}_l = \widehat{C}_{\phi^{-1}(l)} \cdot \mathcal{TEL}.\mathsf{Enc}(0, pk, \mathsf{par}, \hat{\sigma}_l^{(i)})$ with $\hat{\sigma}_l^{(i)} \leftarrow\!\!\!\!- \mathbb{Z}_q$ for $l = 1, .., v_1$. Let us consider ξ as the collection of $\{\tilde{r}_1, \ldots, \tilde{r}_{v_1}\}$, where $\tilde{r}_l$ is 0 for $l = \phi(1), \ldots, \phi(\mathsf{card})$ and $r_l \in \mathbb{Z}_q$ for $l = \phi(\mathsf{card}+1), \ldots, \phi(v_1)$;
- invokes the simulator $\mathcal{SIM}_1^{\mathsf{Dec}}$ that simulates the view of corrupted parties including P_1 in $\mathcal{TEL}.\mathsf{Dec}$ as $(\chi; \xi)$;
- outputs the simulated view as $(\{X_i\}_{i\in\mathcal{I}_1}; R; \{E_{pk}(\mathsf{BF}_{\widetilde{X}_i})\}_{i\notin\mathcal{I}_1}, \Gamma, \mathcal{SIM}_1^{\mathsf{Dec}}(\chi; \xi))$, where Γ is the collection of $|n - \mathcal{I}_1|$ many permuted form of ζ and $\chi \in \Gamma$ if P_n is not corrupted, otherwise $\chi \notin \Gamma$.

The view in the real protocol execution consists of the input sets $\{X_i\}_{i\in\mathcal{I}_1}$, the random coins $\overline{R}$, the ciphertexts $\{E_{pk}(\mathsf{BF}_{X_i}\}_{i\notin\mathcal{I}_1})$, $\{\mathsf{CP}^{(i)}\}_{i\notin\mathcal{I}_1}$ and the messages in $\mathcal{TEL}.\mathsf{Dec}$. In the simulated view, the input sets $\{X_i\}_{i\in\mathcal{I}_1}$ are the same as the view in the real execution, and the outcome of the internal random coins $\overline{R}$ is uniformly random, thus the distribution is the same as in the real execution. Since the threshold encryption scheme $\mathcal{TEL}$ is semantically secure, $(\{E_{pk}(\mathsf{BF}_{X_i})\}_{i\notin\mathcal{I}_1}, \{\mathsf{CP}^{(i)}\}_{i\notin\mathcal{I}_1})$ and $(\{E_{pk}(\mathsf{BF}_{\widetilde{X}_i})\}_{i\notin\mathcal{I}_1}, \Gamma)$ are indistinguishable. Moreover, the distribution of the view $(\chi; \xi)$ produced by $\mathcal{SIM}_1^{\mathsf{Dec}}$ should be indistinguishable from the view in the real execution of $\mathcal{TEL}.\mathsf{Dec}$ by the semantic security of $\mathcal{TEL}$. As a consequence, the simulated view is indistinguishable from the real view.

Case II (a subset $\mathcal{I}_2$ of $\{P_1, \ldots, P_n\}$ is corrupted, and $P_1 \in \mathcal{I}_2$). Let the simulator $\mathcal{SIM}$ be given access to the corrupted parties' input sets $\{X_i\}_{i\in\mathcal{I}_2}$ and output $\perp$. The simulator $\mathcal{SIM}$ then proceeds as follows:

- generates key pair $(pk, sk) \leftarrow \mathcal{TEL}.\mathsf{KGen}(1^\kappa)$ and uniformly chooses its random coins R';
- chooses random sets $\{\widetilde{X}_i\}_{i\notin\mathcal{I}_2}$ with $|\widetilde{X}_i| = v_i$, constructs Bloom filters $\{\mathsf{BF}_{\widetilde{X}_i}\}_{i\notin\mathcal{I}_2}$ and encrypts $\{\mathsf{BF}_{\widetilde{X}_i}\}_{i\notin\mathcal{I}_2}$ using pk as $\{E_{pk}(\mathsf{BF}_{\widetilde{X}_i})\}_{i\notin\mathcal{I}_2}$ in order to play the role of the honest parties;
- generates $n - |\mathcal{I}_2|$ many set of v_1 random ciphertexts as $\chi^{(i)} = \{\widetilde{C}_1^{(i)}, \ldots, \widetilde{C}_{v_1}^{(i)}\}$ for $i \notin \mathcal{I}_2$;

- invokes the simulator $\mathcal{SIM}_2^{\mathsf{Dec}}$ that simulates the view of corrupted parties excluding P_1 in the threshold decryption $\mathcal{TEL}.\mathsf{Dec}$ as $(\chi^{(1)}; \perp)$, where $\chi^{(1)} = \chi^{(n)}$ if P_n is not corrupted;
- outputs the simulated view as $\big(\{X_i\}_{i\in\mathcal{I}_2}; R'; \{E_{pk}(\mathsf{BF}_{\widetilde{X}_i}), \chi^{(i)}\}_{i\notin\mathcal{I}_2}, \mathcal{SIM}_2^{\mathsf{Dec}}(\chi^{(1)}; \perp)\big)$.

The view in the real protocol execution contains the input sets $\{X_i\}_{i\in\mathcal{I}_2}$, the random coins $\widehat{R}$, the sets of ciphertexts $\{\mathsf{CP}^{(i)}\}_{i\notin\mathcal{I}_2}$ and the messages in $\mathcal{TEL}.\mathsf{Dec}$. In the simulated view, the input sets $\{X_i\}_{i\in\mathcal{I}_2}$ and internal random coins $\widehat{R}$ are indistinguishable form the counter parts in the view of the real execution. Since the threshold encryption scheme $\mathcal{TEL}$ is semantically secure, $\{\mathsf{CP}^{(i)}\}_{i\notin\mathcal{I}_2}$ and $\{\chi^{(i)}\}_{i\notin\mathcal{I}_2}$ are indistinguishable. Consequently, the distribution of the view $(\chi^{(1)}; \perp)$ produced by $\mathcal{SIM}_2^{\mathsf{Dec}}$ is indistinguishable from the view in a real execution of $\mathcal{TEL}.\mathsf{Dec}$. Hence, the simulated view is indistinguishable from the real world view. □

Theorem 4.2. *If the encryption schemes $\mathcal{EL}$ and $\mathcal{TEL}$ are semantically secure, then our proposed MPSI presented in Section 3.2 is a secure computation protocol in the standard model against semi-honest adversaries except with negligible probability ϵ, where ϵ is the false positive rate of Bloom filter.*

Proof. In order to prove the security of the MPSI, we consider the following two cases:

- Case I: a strict subset $\mathcal{I}_1$ of $\{P_1, \ldots, P_n\}$ is corrupted, and $P_1 \in \mathcal{I}_1$.
- Case II: a strict subset $\mathcal{I}_2$ of $\{P_1, \ldots, P_n\}$ is corrupted, and $P_1 \notin \mathcal{I}_2$.

In each of the cases, we will construct a simulator $\mathcal{SIM}$ who simulates the MPSI protocol, and the simulator is given access to the corrupted party's input and output such that the simulated view is computationally indistinguishable from the real world view. Here, the view of an entity consists of input message of the entity, the outcome of the entity's internal coin tosses and the messages received by the entity during the protocol execution.

Case I (a subset $\mathcal{I}_1$ of $\{P_1, \ldots, P_n\}$ is corrupted, and $P_1 \in \mathcal{I}_1$). Let the simulator $\mathcal{SIM}$ be given access to the corrupted parties' input sets $\{X_i\}_{i\in\mathcal{I}_1}$ and output $\bigcap_{i=1}^n X_i$. Then $\mathcal{SIM}$ does the following:

- generates $(pk, sk) \leftarrow \mathcal{TEL}.\mathsf{KGen}(1^\kappa)$ and uniformly chooses its random coins R;
- plays the role of the honest parties by choosing random sets $\{\widetilde{X}_i\}_{i\notin\mathcal{I}_1}$ with $|\widetilde{X}_i| = v_i$, constructing Bloom filters $\{\mathsf{BF}_{\widetilde{X}_i}\}_{i\notin\mathcal{I}_1}$ and encrypting $\{\mathsf{BF}_{\widetilde{X}_i}\}_{i\notin\mathcal{I}_1}$ using pk to get $\{E_{pk}(\mathsf{BF}_{\widetilde{X}_i})\}_{i\notin\mathcal{I}_1}$;
- generates the ciphertext $\widehat{C}_l$ of the form $\mathcal{TEL}.\mathsf{Enc}_{pk}(0)$ for each $x_l \in \bigcap_{i=1}^n X_i$ and the ciphertext $\{\widehat{C}_l\}$ of the form $\mathcal{TEL}.\mathsf{Enc}_{pk}(r_l)$ for each $x_l \notin \bigcap_{i=1}^n X_i$, where r_l is uniformly chosen from $\mathbb{Z}_q$ and $X_1 = \{x_1, \ldots, x_{v_1}\}$. Let us consider $\chi = \{\widehat{C}_1, \ldots, \widehat{C}_{v_1}\}$ and ξ as the collection of $\{r_1, \ldots, r_{v_1}\}$, where r_l is set as 0 if $x_l \in \bigcap_{i=1}^n X_i$, otherwise $r_l \in \mathbb{Z}_q$;
- invokes the simulator $\mathcal{SIM}_1^{\mathsf{Dec}}$ that simulates the view of corrupted parties including P_1 in $\mathcal{TEL}.\mathsf{Dec}$ as $(\chi; \xi)$;
- outputs the simulated view as $\Big(\{X_i\}_{i\in\mathcal{I}_1}; R; \{E_{pk}(\mathsf{BF}_{\widetilde{X}_i})\}_{i\notin\mathcal{I}_1}, \mathcal{SIM}_1^{\mathsf{Dec}}(\chi; \xi)\Big)$.

The view in the real protocol execution consists of the input sets $\{X_i\}_{i\in\mathcal{I}_1}$, the random coins $\overline{R}$, the ciphertexts $\{E_{pk}(\mathsf{BF}_{X_i}\}_{i\notin\mathcal{I}_1})$, and the messages in $\mathcal{TEL}.\mathsf{Dec}$.

In the simulated view, the input sets $\{X_i\}_{i\in\mathcal{I}_1}$ are the same as the view in the real execution, the outcome of the internal random coins $\overline{R}$ is uniformly random, thus the distribution is the same as in the real execution. Since the threshold encryption scheme $\mathcal{TEL}$ is semantically secure, $\{E_{pk}(\mathsf{BF}_{X_i})\}_{i\notin\mathcal{I}_1}$ and $\{E_{pk}(\mathsf{BF}_{\widetilde{X}_i})\}_{i\notin\mathcal{I}_1}$ are indistinguishable. Moreover, the distribution of the view $(\chi;\xi)$ produced by $\mathcal{SIM}_1^{\mathsf{Dec}}$ should be indistinguishable from the view in the real execution of $\mathcal{TEL}.\mathsf{Dec}$ by the semantic security of $\mathcal{TEL}$. As a consequence, the simulated view is indistinguishable from the real view.

Case II (a subset $\mathcal{I}_2$ of $\{P_1,\ldots,P_n\}$ is corrupted, and $P_1\notin\mathcal{I}_2$). Let the simulator $\mathcal{SIM}$ be given access to the corrupted parties' input sets $\{X_i\}_{i\in\mathcal{I}_2}$ and output $\perp$. The simulator $\mathcal{SIM}$ then proceeds as follows:

- generates key pair $(pk,sk)\leftarrow\mathcal{TEL}.\mathsf{KGen}(1^\kappa)$ and uniformly chooses its random coins R';
- chooses random sets $\{\widetilde{X}_i\}_{i\notin\mathcal{I}_2}$ with $|\widetilde{X}_i|=v_i$, constructs Bloom filters $\{\mathsf{BF}_{\widetilde{X}_i}\}_{i\notin\mathcal{I}_2}$ and encrypts $\{\mathsf{BF}_{\widetilde{X}_i}\}_{i\notin\mathcal{I}_2}$ using pk as $\{E_{pk}(\mathsf{BF}_{\widetilde{X}_i})\}_{i\notin\mathcal{I}_2}$ in order to play the role of the honest parties;
- generates v_1 random ciphertexts as $\chi=\{\widetilde{C}_1,\ldots,\widetilde{C}_{v_1}\}$;
- invokes the simulator $\mathcal{SIM}_2^{\mathsf{Dec}}$ that simulates the view of corrupted parties excluding P_1 in threshold decryption $\mathcal{TEL}.\mathsf{Dec}$ as $(\chi;\perp)$;
- outputs the simulated view as $\Big(\{X_i\}_{i\in\mathcal{I}_2};R';\chi,\mathcal{SIM}_2^{\mathsf{Dec}}(\chi;\perp)\Big)$.

The view in the real protocol execution contains the input sets $\{X_i\}_{i\in\mathcal{I}_2}$, the random coins $\widehat{R}$, the ciphertexts $\{C_1,\ldots,C_{v_1}\}$, and the messages in $\mathcal{TEL}.\mathsf{Dec}$. In the simulated view, the input sets $\{X_i\}_{i\in\mathcal{I}_2}$ and internal random coins $\widehat{R}$ are indistinguishable form the counter parts in the view of the real execution. Since the threshold encryption scheme $\mathcal{TEL}$ is semantically secure, $\{C_1,\ldots,C_{v_1}\}$ and $\chi=\{\widetilde{C}_1,\ldots,\widetilde{C}_{v_1}\}$ are indistinguishable. Consequently, the distribution of the view $(\chi;\perp)$ produced by $\mathcal{SIM}_2^{\mathsf{Dec}}$ is indistinguishable from the view in a real execution of $\mathcal{TEL}.\mathsf{Dec}$. Hence, the simulated view is indistinguishable from the real world view. □

Remark 2. Both the schemes MPSI and MPSI-CA are secure in the semi-honest environment. However, both the schemes can be proven to be secure when the designated party P_1 is semi-honest and the remaining participants $P_2,\ldots,P_n$ are malicious by employing zero-knowledge proofs for discrete logarithm [6] and zero-knowledge argument for shuffle [33].

5. Efficiency

The computation cost of our constructions is measured by counting the number of modular exponentiations (Exp), hash function evaluations (Hash) and modular inversions (Inv). On the other hand, the number of group elements transmitted publicly by an user incurs communication overhead. We refer to TABLE 1 for the complexity of our protocols. Note that, our MPSI does not use any kind of broadcast channel in contrast to our MPSI-CA. In TABLE 2 and TABLE 3, we give a comparative summary of our constructions with the most efficient existing protocols.

TABLE 1. Complexity of MPSI and MPSI-CA

MPSI-CA	Exp	GE	Hash	Inv
P_1	v_1	$2v_1$	kv_1	v_1
$P_i, i \neq 1$	$2m + 3v_1$	$2m + 3v_1$	kv_i	

MPSI	Exp	GE	Hash	Inv
P_1	v_1	$2(n-1)v_1$	kv_1	v_1
$P_i, i \neq 1$	$2m + v_1$	$2m + v_1$	kv_i	

v_i = set size of P_i, $m = \lceil \frac{kv_{\max}}{\ln 2} \rceil$ = optimal size of Bloom filter, $v_{\max}$ = maximum of $\{v_1, \ldots, v_n\}$

TABLE 2. Comparative summary of MPSI protocols

Protocol	Adv. model	Security assumption	Comm.	Comp.	Based on
[46]	Mal	AHE	$O(n^2 v_{\max}^2)$	$O(n^2 \log nv_{\max}^2)$	OPE
[12]	Mal	DCR	$O((nv_{\max} + 10v_{\max} \log^2 v_{\max}))$	$O(nv_{\max}^2)$	MP
[10]	Mal	AHE	$O(nv_{\max}^2)$	$O(nv_{\max}^2)$	OPE
[51]	Mal	DCR	$O(n^2 \log nv_{\max}^2)$	$O(\log nv_{\max}^2)$	OPE
[52]	Mal	SD	$O(nv_{\max}^2)$	$O(nv_{\max}^2)$	BG
[49]	SH	DDH	$O(nv_{\max})$	$D : O(nv_{\max}); P_i : O(v_{\max})$	BF
Sch. 1[39]	SH	DDH	$O(nv_{\max}\kappa)$	$D : O(nv_{\max}^2\kappa); P_i : O(v_{\max}\kappa)$	OPE
Sch. 2[39]	Mal	DDH	$O((n^2 + nv_{\max} + nw \log v_{\max})\kappa)$	$D : O(nv_{\max}^2\kappa); P_i : O(v_{\max}\kappa)$	OPE
[47]	SH		$O(nv_{\max}\kappa)$	$D : O(n\lambda); P_i : O(t\lambda)$	OPRF
Our	SH	DDH	$O(nv_{\max}k)$	$D : O(v_1); P_i : O(v_{\max}k)$	BF

OPE= Oblivious Polynomial Evaluation, MP= Multivariate Polynomials, BF= Bloom Filter, SD= Subgroup Decision, BG= Bilinear Group, Mal= Malicious, AHE= Additively Homomorphic Encryption, DDH=Decisional Diffie-Hellman, DCR=Decisional Quadratic Residuosity, SH=Semi-honest, OPRF= Oblivious Pseudorandom Function, D= designated party, P_i= participants other than designated party, n= number of participants, v_1= set size of the designated party D, t= number of dishonestly colluding participants, $v_{\max}$= maximum set size, κ, k, λ= security parameters.

TABLE 3. Comparative summary of MPSI-CA protocols

Protocol	Adv. model	Security assumption	Comm.	Comp.	Based on
[46]	SH	AHE	$O(n^2 v_{\max})$	$O(n^2 v_{\max}^2)$	OPE
Our	SH	DDH	$O(nv_{\max}k)$	$D : O(v_1); P_i : O(v_{\max}k)$	BF

OPE= Oblivious Polynomial Evaluation, BF= Bloom Filter, SH=Semi-honest, AHE= Additively Homomorphic Encryption, DDH=Decisional Diffie-Hellman, D= designated party, P_i= participants other than designated party, n= number of participants, v_1= set size of the designated party D, $v_{\max}$= maximum set size.

6. CONCLUSION

In this paper, we have constructed an MPSI-CA protocol employing a Bloom filter in semi-honest environment without random oracles. Its communication and computation overheads are linear in the input set sizes. Our MPSI-CA is more efficient than the only other existing MPSI-CA of [46]. We then extended our MPSI-CA to MPSI retaining all its security attributes. In contrast to the existing MPSI protocols, the computation complexity of each party in our construction does not depend upon the total number of participants. However, our MPSI is less efficient than that of [47] in terms of set sizes.

ACKNOWLEDGMENTS

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

Appendix A

A1. Security Model. The basic security requirements of any multiparty protocol are the following:

(a) *Correctness.* At the end of the protocol, an honest party should receive the correct output.
(b) *Privacy.* After completion of the protocol, no party should learn more than its prescribed output.
(c) *Fairness.* A dishonest party receives its output if and only if the honest party also receives its output.

Security Model for Semi-honest Adversary [34]: A two-party protocol, Π is a random process that computes a function f from a pair of inputs (one per party) to a pair of outputs, i.e.,

$$f = (f_1, f_2) : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^*.$$

Let $x, y \in \{0,1\}^*$ be the inputs of parties P_1, P_2, respectively. Then the outputs of the parties P_1, P_2 are $f_1(x,y), f_2(x,y)$ respectively. A protocol Π is said to be secure in a semi-honest model if whatever can be computed by a party after participating in the protocol, it could obtain from its input and output only. This is formalized using the simulation paradigm. At the input pair (x,y), the view of the party P_i during an execution of Π is denoted by $\mathsf{View}_i^{\Pi}(x,y) = (w; r^{(i)}; m_1^{(i)}, \ldots, m_t^{(i)})$, 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 received by P_i during the execution of Π.

Definition A.1. Let $f = (f_1, f_2)$ be a deterministic function. Then we say that the protocol Π securely computes f if there exists probabilistic polynomial-time adversaries, denoted by S_1 and S_2, controlling P_1 and P_2, respectively, such that

$$\{S_1(x, f_1(x,y))\}_{x,y\in\{0,1\}^*} \equiv^c \mathsf{View}_1^{\Pi}(x,y)_{x,y\in\{0,1\}^*},$$
$$\{S_2(y, f_2(x,y))\}_{x,y\in\{0,1\}^*} \equiv^c \mathsf{View}_2^{\Pi}(x,y)_{x,y\in\{0,1\}^*}$$

In the case of a multiparty setting, the associated functionality is

$$f = (f_1, \ldots, f_n) : \{0,1\}^* \times \cdots \times \{0,1\}^* \to \{0,1\}^* \times \cdots \times \{0,1\}^*.$$

Let $X_i \in \{0,1\}^*$ be the input of party P_i, for $i = 1, \ldots, n$. Then the output of the party P_i is $f_i(X_1, \ldots, X_n)$ for $i = 1, \ldots, n$. Let the adversary $\mathcal{A}$ corrupt $\{P_i : i \in \mathcal{I} \subset \{1, \ldots, n\}\}$, a proper subset of $\{P_1, \ldots, P_n\}$. Then, similar to the two-party protocol, we say that the multiparty protocol is secure if $\mathcal{A}$'s view in the real world is indistinguishable from the simulated view. Here, the view of $\mathcal{A}$ is defined as $(W; r; M)$, $W = \{X_i : i \in \mathcal{I}\}$, r is the collection of the outcome of P_i's internal coin tosses for $i \in \mathcal{I}$ and M is the collection of messages which has been received by $\{P_i : i \in \mathcal{I}\}$ during the protocol execution.

A2. Toy Example. **Toy example for MPSI:** Let there be three participants P_1, P_2, P_3 with respective private sets $X_1 = \{alice, bob, thomas\}$, $X_2 = \{bob, harry, alice\}$, $X_3 = \{jack, alice, thomas, bob\}$ and $G =< 2 >= \{1, 2, 3, 4, 6, 8, 9, 12, 13, 16, 18\}$ be the subgroup of $\mathbb{Z}_{23}^*$ of order 11, i.e., $p = 23$ and $q = 11$. Also let the secret keys of P_1, P_2, P_3 be $2, 3, 2$, respectively, and $H_{\mathsf{Bloom}} = \{h_1, h_2\}$, i.e., $k = 2$.

Then the public key for threshold ElGamal is $(2^{2+3+2} \bmod 23) = 13$ and optimal $m = kv_{\mathsf{max}} \ln 2 = 16$ since $v_{\mathsf{max}} = 4$. Let us assume that

$$h_1(alice) = 4, h_1(bob) = 2, h_1(thomas) = 14, h_1(harry) = 14, h_1(jack) = 1,$$
$$h_2(alice) = 11, h_2(bob) = 4, h_2(thomas) = 7, h_2(harry) = 2, h_2(jack) = 6.$$

Then the steps of MPSI are described below:

: MPSI.request:

1. P_2 computes $\mathsf{BF}_{X_2} = 1010111111011011$ and sends $E(\mathsf{BF}_{X_2}) = \{E(1), E(0), E(1), \quad E(0), E(1), E(1), E(1), E(1), E(1), E(1), E(0), E(1), E(1), E(0), E(1), E(1)\} = \{C_1^{(2)}, \ldots, C_{16}^{(2)}\}$ to P_1, where E is the encryption function.
2. P_3 computes $\mathsf{BF}_{X_3} = 0010100111011011$ and sends $E(\mathsf{BF}_{X_3}) = \{E(0), E(0), E(1), \quad E(0), E(1), E(0), E(0)E(1), E(1), E(1), E(0), E(1), E(1), E(0), E(1), E(1)\} = \{C_1^{(3)}, \ldots, C_{16}^{(3)}\}$ to P_1.

: MPSI.response: P_1 does the following:

1. for $x_1 = alice$, computes $h_1(alice) = 4$, $h_2(alice) = 11$ and derives $C_1 = C_4^{(2)} C_{11}^{(2)} C_4^{(3)} C_{11}^{(3)} = E(0)E(0)E(0)E(0) = (2^{r_{2,4}}, 2^0 13^{r_{2,4}}) \cdot (2^{r_{2,11}}, 2^0 13^{r_{2,11}}) \cdot (2^{r_{3,4}}, 2^0 13^{r_{3,4}}) \cdot (2^{r_{3,11}}, 2^0 13^{r_{3,11}}) = (2^{r_1}, 2^0 13^{r_1})$, where $r_1 = r_{2,4} + r_{2,11} + r_{3,4} + r_{3,11}$;
2. for $x_2 = bob$, computes $h_1(bob) = 2$, $h_2(bob) = 4$ and derives $C_2 = C_2^{(2)} C_4^{(2)} C_2^{(3)} C_4^{(3)} = E(0)E(0)E(0)E(0) = (2^{r_{2,2}}, 2^0 13^{r_{2,2}}) \cdot (2^{r_{2,4}}, 2^0 13^{r_{2,4}}) \cdot (2^{r_{3,2}}, 2^0 13^{r_{3,2}}) \cdot (2^{r_{3,4}}, 2^0 13^{r_{3,4}}) = (2^{r_2}, 2^0 13^{r_2})$, where $r_2 = r_{2,2} + r_{2,4} + r_{3,2} + r_{3,4}$;
3. for $x_3 = thomas$, computes $h_1(thomas) = 14$, $h_2(thomas) = 7$ and derives $C_2 = C_{14}^{(2)} C_7^{(2)} C_{14}^{(3)} C_7^{(3)} = E(0)E(1)E(0)E(0) = (2^{r_{2,14}}, 2^0 13^{r_{2,14}}) \cdot (2^{r_{2,7}}, 2^1 13^{r_{2,7}}) \cdot (2^{r_{3,14}}, 2^0 13^{r_{3,14}}) \cdot (2^{r_{3,7}}, 2^0 13^{r_{3,7}}) = (2^{r_3}, 2^1 13^{r_3})$, where $r_3 = r_{2,14} + r_{2,7} + r_{3,14} + r_{3,7}$.

Finally, P_1 sends $(2^{r_1}, 2^0 13^{r_1})$, $(2^{r_2}, 2^0 13^{r_2})$ and $(2^{r_3}, 2^1 13^{r_3})$ to both P_2 and P_3.

: MPSI.computation: The party P_2 computes $\{(2^{r_1})^3, (2^{r_2})^3, (2^{r_3})^3\}$ using its secret 3 and sends this to P_1. The party P_3 computes $\{(2^{r_1})^2, (2^{r_2})^2, (2^{r_3})^2\}$ using its secret 2 and sends this to P_1. The party P_1 also computes $\{(2^{r_1})^2, (2^{r_2})^2, (2^{r_3})^2\}$ using secret 2 and does the following:

1. evaluates $\rho_1 = (2^{r_1})^{2+3+2} = (2^7)^{r_1} = 13^{r_1}$, $\rho_2 = (2^{r_2})^{2+3+2} = (2^7)^{r_2} = 13^{r_2}$, and $\rho_3 = (2^{r_3})^{2+3+2} = (2^7)^{r_3} = 13^{r_3}$;
2. computes $\mu_1 = \frac{2^0 13^{r_1}}{13^{r_1}} = 1$, $\mu_2 = \frac{2^0 13^{r_2}}{13^{r_2}} = 1$ and $\mu_3 = \frac{2^1 13^{r_3}}{13^{r_3}} = 2$;
3. outputs $\{x_1, x_2\} = \{alice, bob\}$ as the intersection $\cap_{i=1}^3 X_i$.

Toy example for MPSI-CA: Choose the same parameters as for MPSI.

: MPSI.request: It is similar to MPSI.

: MPSI.response: It is similar to MPSI up to the computation of $S_1 = \{(2^{r_1}, 2^0 13^{r_1}), (2^{r_2}, 2^0 13^{r_2}), (2^{r_3}, 2^1 13^{r_3})\}$. Then P_2 does the following:

1. randomly chooses 3 ciphertexts of the form $E(0)$, say $A_1 = \{(2^{\sigma_1}, 2^0 13^{\sigma_1}), (2^{\sigma_2}, 2^0 13^{\sigma_2}), (2^{\sigma_3}, 2^0 13^{\sigma_3})\}$;
2. multiplies the i-th ciphertext of S_1 with the i-th ciphertext of A_1 for $i = 1, 2, 3$ to get $A_2 = \{(2^{r_1+\sigma_1}, 2^0 13^{r_1+\sigma_1}), (2^{r_2+\sigma_2}, 2^0 13^{r_2+\sigma_2}), (2^{r_3+\sigma_3}, 2^1 13^{r_3+\sigma_3})\}$;
3. permutes the elements of A_2. Let the permuted version of A_2 be $A_3 = \{(2^{r_2+\sigma_2}, 2^0 13^{r_2+\sigma_2}), (2^{r_1+\sigma_1}, 2^0 13^{r_1+\sigma_1}), (2^{r_3+\sigma_3}, 2^1 13^{r_3+\sigma_3})\}$;

4. broadcasts A_3.

Then P_3 does the following:

1. randomly chooses 3 ciphertexts of the form $E(0)$, say $B_1 = \{(2^{\delta_1}, 2^0 13^{\delta_1}), (2^{\delta_2}, 2^0 13^{\delta_2}), (2^{\delta_3}, 2^0 13^{\delta_3})\}$;
2. multiplies the i-th ciphertext of A_3 with the i-th ciphertext of B_1 for $i = 1, 2, 3$ to get $B_2 = \{(2^{r_2+\sigma_2+\delta_1}, 2^0 13^{r_2+\sigma_2+\delta_1}), (2^{r_1+\sigma_1+\delta_2}, 2^0 13^{r_1+\sigma_1+\delta_2}), (2^{r_3+\sigma_3+\delta_3}, 2^1 13^{r_3+\sigma_3+\delta_3})\}$;
3. permutes the elements of B_2. Let the permuted version of B_2 be $B_3 = \{(2^{r_3+\sigma_3+\delta_3}, 2^1 13^{r_3+\sigma_3+\delta_3}), (2^{r_2+\sigma_2+\delta_1}, 2^0 13^{r_2+\sigma_2+\delta_1}), (2^{r_1+\sigma_1+\delta_2}, 2^0 13^{r_1+\sigma_1+\delta_2})\}$;
4. broadcasts B_3.

: MPSI-CA.computation: Let $r_3+\sigma_3+\delta_3 = \gamma_1$, $r_2+\sigma_2+\delta_1 = \gamma_1$ and $r_1+\sigma_1+\delta_2 = \gamma_3$. Then $B_3 = \{(2^{\gamma_1}, 2^1 13^{\gamma_1}), (2^{\gamma_2}, 2^0 13^{\gamma_2}), (2^{\gamma_3}, 2^0 13^{\gamma_3})\}$. The party P_2 computes $\{(2^{\gamma_1})^3, (2^{\gamma_2})^3, (2^{\gamma_3})^3\}$ using its secret 3 and sends this to P_1. The party P_3 computes $\{(2^{\gamma_1})^2, (2^{\gamma_2})^2, (2^{\gamma_3})^2\}$ using its secret 2 and sends this to P_1. The party P_1 also computes $\{(2^{\gamma_1})^2, (2^{\gamma_2})^2, (2^{\gamma_3})^2\}$ using secret 2 and does the following:

1. evaluates $\overline{\rho}_1 = (2^{\gamma_1})^{2+3+2} = (2^7)^{\gamma_1} = 13^{\gamma_1}$, $\overline{\rho}_2 = (2^{\gamma_2})^{2+3+2} = (2^7)^{\gamma_2} = 13^{\gamma_2}$, and $\overline{\rho}_3 = (2^{\gamma_3})^{2+3+2} = (2^7)^{\gamma_3} = 13^{\gamma_3}$;
2. computes $\overline{\mu}_1 = \frac{2^1 13^{\gamma_1}}{13^{\gamma_1}} = 2$, $\overline{\mu}_2 = \frac{2^0 13^{\gamma_2}}{13^{\gamma_2}} = 1$ and $\overline{\mu}_3 = \frac{2^0 13^{\gamma_3}}{13^{\gamma_3}} = 1$;
3. outputs 2 as the cardinality of the the intersection $\cap_{i=1}^3 X_i$ since there are only two $\overline{\mu}$'s with 1 value.

References

[1] R. Agrawal, A. Evfimievski and R. Srikant, Information sharing across private databases, *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, ACM*, (2003), 86–97.

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

[3] D. Boneh, The decision Diffie-Hellman problem, *Algorithmic Number Theory, Springer, Lecture Notes in Comput. Sci., Springer, Berlin*, **1423** (1998), 48–63.

[4] P. Bose, H. Guo, E. Kranakis, A. Maheshwari, P. Morin, J. Morrison, M. Smid and Y. H. Tang, On the false-positive rate of Bloom filters, *Inform. Proc. Lett.*, **108** (2008), 210–213.

[5] J. Camenisch and V. Shoup, Practical verifiable encryption and decryption of discrete logarithms, *Advances in Cryptology—CRYPTO 2003, Lecture Notes in Comput. Sci., Springer, Berlin*, **2729** (2003), 126–144.

[6] J. Camenisch and M. Stadler, *Proof Systems for General Statements about Discrete Logarithms*, 1997. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.56.1208.

[7] J. Camenisch and G. M. Zaverucha. Private intersection of certified sets, *Financial Cryptography and Data Security, Springer*, (2009), 108–127.

[8] A. Cerulli, E. De Cristofaro and C. Soriente, Nothing refreshes like a RePSI: Reactive private set intersection, *Applied Cryptography and Network Security, Lecture Notes in Comput. Sci., Springer, Cham*, **10892** (2018), 280–300.

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

[10] J. H. Cheon, S. Jarecki and J. H. Seo, Multi-party privacy-preserving set intersection with quasi-linear complexity, *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, **95** (2012), 1366–1378.

[11] M. Ciampi and C. Orlandi, Combining private set-intersection with secure two-party computation, *Security and Cryptography for Networks, Lecture Notes in Comput. Sci., Springer, Cham*, **11035** (2018), 464–482.

[12] D. Dachman-Soled, T. Malkin, M. Raykova and M. Yung, Secure efficient multiparty computing of multivariate polynomials and applications, *Applied Cryptography and Network Security*, (2011), 130–146.
[13] A. Davidson and C. Cid, An efficient toolkit for computing private set operations, *Information Security and Privacy - ACISP*, (2017), 261–278.
[14] E. De Cristofaro, P. Gasti and G. Tsudik, Fast and private computation of cardinality of set intersection and union, *Cryptology and Network Security, Lecture Notes in Comput. Sci., Springer, Heidelberg*, **7712** (2012), 218–231.
[15] E. De Cristofaro, J. Kim and G. Tsudik, Linear-complexity private set intersection protocols secure in malicious model, *Adv. in Cryptology - ASIACRYPT, Springer*, (2010), 213–231.
[16] E. De Cristofaro and G. Tsudik, Practical private set intersection protocols with linear complexity, *Financial Cryptography and Data Security*, (2010), 143–159.
[17] E. De Cristofaro and G. Tsudik, Experimenting with fast private set intersection, *Trust and Trustworthy Computing*, (2012), 55–73.
[18] S. K. Debnath and R. Dutta, Efficient private set intersection cardinality in the presence of malicious adversaries, *Provable Security, Lecture Notes in Comput. Sci., Springer, Cham*, **9451** (2015), 326–339.
[19] S. K. Debnath and R. Dutta, Secure and efficient private set intersection cardinality using Bloom filter, *International Information Security Conference, Springer*, (2015), 209–226.
[20] S. K. Debnath and R. Dutta, How to meet big data when private set intersection realizes constant communication complexity, *Information and Communications Security, Springer*, (2016), 445–454.
[21] S. K. Debnath and R. Dutta, New realizations of efficient and secure private set intersection protocols preserving fairness, *Information Security and Cryptology—ICISC 2016, Lecture Notes in Comput. Sci., Springer, Cham*, **10157** (2017), 254–284.
[22] S. K. Debnath and R. Dutta, Provably secure fair mutual private set intersection cardinality utilizing Bloom filter, *Information Security and Cryptology, Lecture Notes in Comput. Sci., Springer, Cham*, **10143** (2017), 505–525.
[23] S. K. Debnath and R. Dutta, Towards fair mutual private set intersection with linear complexity, *Security Comm. Networks*, **9** (2016), 1589–1612.
[24] Y. Desmedt and Y. Frankel, Threshold cryptosystems, *Adv. in Cryptology - CRYPTO 89, Springer*, (1990), 307–315.
[25] C. Y. Dong, L. Q. Chen, J. Camenisch and G. Russello, Fair private set intersection with a semi-trusted arbiter, *Data and Applications Security and Privacy XXVII, Springer*, (2013), 128–144.
[26] C. Y. Dong, L. Q. Chen and Z. K. Wen, When private set intersection meets big data: An efficient and scalable protocol, *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, ACM*, (2013), 789–800.
[27] C. Y. Dong and G. Loukides, Approximating private set union/intersection cardinality with logarithmic complexity, *IEEE Transactions on Information Forensics and Security*, **12** (2017), 2792–2806.
[28] T. ElGamal, A public key cryptosystem and a signature scheme based on discrete logarithms, *IEEE Trans. Inform. Theory*, **31** (1985), 469–472.
[29] B. H. Falk, D. Noble and R. Ostrovsky, Private set intersection with linear communication from general assumptions, (2018).
[30] P. Flajolet and G. N. Martin. Probabilistic counting algorithms for data base applications, *Journal of Computer and System Sciences*, **31** (1985), 182–209.
[31] M. J. Freedman, C. Hazay, K. Nissim and B. Pinkas, Efficient set intersection with simulation-based security, *Journal of Cryptology*, **29** (2016), 115–155.
[32] M. J. Freedman, K. Nissim and B. Pinkas, Efficient private matching and set intersection, *Advances in Cryptology—EUROCRYPT 2004, Lecture Notes in Comput. Sci., Springer, Berlin*, **3027** (2004), 1–19.
[33] J. Furukawa, Efficient and verifiable shuffling and shuffle-decryption, *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, **88** (2005), 172–188.
[34] O. Goldreich, *Foundations of Cryptography: Volume 2, Basic Applications*, Cambridge University Press, 2009.
[35] S. Goldwasser and S. Micali, Probabilistic encryption, *Journal of Computer and System Sciences*, **28** (1984), 270–299.

[36] C. Hazay, Oblivious polynomial evaluation and secure set-intersection from algebraic PRFs, *Theory of Cryptography, Part II, Lecture Notes in Comput. Sci., Springer, Heidelberg*, **9015** (2015), 90–120.
[37] C. Hazay and Y. Lindell, Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries, *Theory of Cryptography, Lecture Notes in Comput. Sci., Springer, Berlin*, **4948** (2008), 155–175.
[38] C. Hazay and K. Nissim, Efficient set operations in the presence of malicious adversaries, *Public Key Cryptography - PKC, Lecture Notes in Comput. Sci., Springer, Berlin*, **6056** (2010), 312–331.
[39] C. Hazay and M. Venkitasubramaniam, Scalable multi-party private set-intersection, *Public-Key Cryptography—PKC 2017, Part I, Lecture Notes in Comput. Sci., Springer, Berlin*, **10174** (2017), 175–203.
[40] S. Hohenberger and S. A. Weis, Honest-verifier private disjointness testing without random oracles, *Privacy Enhancing Technologies, Springer*, (2006), 277–294.
[41] Y. Huang, D. Evans and J. Katz, Private set intersection: Are garbled circuits better than custom protocols, *Network and Distributed System Security Symposium (NDSS), The Internet Society*, (2012).
[42] S. Jarecki and X. M. Liu, Efficient oblivious pseudorandom function with applications to adaptive OT and secure computation of set intersection, *Theory of Cryptography, Lecture Notes in Comput. Sci., Springer, Berlin*, **5444** (2009), 577–594.
[43] S. Jarecki and X. M. Liu, Fast secure computation of set intersection, *Security and Cryptography for Networks, Springer*, (2010), 418–435.
[44] F. Kerschbaum, Outsourced private set intersection using homomorphic encryption, *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security, ACM*, (2012), 85–86.
[45] 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** (2017), 177–197.
[46] L. Kissner and D. Song, Privacy-preserving set operations, *Adv. in Cryptology - CRYPTO 2005, Lecture Notes in Comput. Sci., Springer, Berlin*, **3621** (2005), 241–257.
[47] V. Kolesnikov, N. Matania, B. Pinkas, M. Rosulek and N. Trieu, Practical multi-party private set intersection from symmetric-key techniques, *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, ACM*, (2017), 1257–1272.
[48] D. Many, M. Burkhart and X. Dimitropoulos, Fast private set operations with sepia, *Technical Report 345, Mar, Tech. Rep.*, (2012).
[49] A. Miyaji and S. Nishida, A scalable multiparty private set intersection, *International Conference on Network and System Security, Springer*, (2015), 376–385.
[50] P. Rindal and M. Rosulek, Malicious-secure private set intersection via dual execution, *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, ACM*, (2017), 1229–1242.
[51] Y. P. Sang and H. Shen, Privacy preserving set intersection protocol secure against malicious behaviors, *Parallel and Distributed Computing, Applications and Technologies (PDCAT), IEEE International Conference on*, (2007), 461–468.
[52] Y. Sang and H. Shen, Privacy preserving set intersection based on bilinear groups, *Proceedings of the Thirty-First Australasian Conference on Computer Science, Australian Computer Society, Inc.*, **74** (2008), 47–54.
[53] R.-H. Shi, Y. Mu, H. Zhong, S. Zhang and J. Cui, Quantum private set intersection cardinality and its application to anonymous authentication, *Information Sciences*, *370/371* (2016), 147–158.

Received August 2019; revised February 2020.

E-mail address: sdebnath.math@nitjsr.ac.in
E-mail address: pstanica@nps.edu
E-mail address: nknkundu@gmail.com
E-mail address: tc499180022@gmail.com