Fair mPSI and mPSI-CA: Efficient Constructions in Prime Order Groups with Security in the Standard Model against Malicious Adversary

Sumit Kumar Debnath*, Ratna Dutta
Department of Mathematics
Indian Institute of Technology Kharagpur
Kharagpur -721302, India

Abstract

In this paper, we propose a construction of fair and efficient mutual Private Set Intersection (mPSI) with linear communication and computation complexities, where the underlying group is of prime order. The main tools in our approach include: (i) ElGamal and Distributed ElGamal Cryptosystems as multiplicatively Homomorphic encryptions, (ii) Cramer-Shoup Cryptosystem as Verifiable encryption. Our mPSI is secure in standard model against malicious parties under Decisional Diffie-Hellman (DDH) assumption. Fairness is achieved using an off-line semi-trusted arbiter. Further, we extend our mPSI to mutual Private Set Intersection Cardinality (mPSI-CA) retaining all the security properties of mPSI. More interestingly, our mPSI-CA is the *first fair* mPSI-CA with *linear complexity*.

Keywords: mPSI, mPSI-CA, malicious adversary, fairness, semi-trusted arbiter.

1 Introduction

In everyday life, dependence on the availability of electronic information increases rapidly. As a consequence, there is a strong need for efficient cryptographic techniques that allow secret sharing of information. Among these, *Private Set Intersection* (PSI) has received considerable attention to the recent research community due to its importance and wide applications. PSI enables two parties to compute secretly the intersection of their respective private input sets. At the end of the protocol, one or both parties must obtain the intersection, but not more than that.

The PSI protocol that enables both the parties to learn the intersection, is known as *mutual* PSI (mPSI). If only one of the parties learns the intersection, then the protocol is known as *one-way* PSI. PSI protocols have found several practical applications, particularly in location-based

*Corresponding author. E-mail: sd.iitkgp@gmail.com

services, privacy preserving data mining, social networks, testing of fully sequenced human genomes, collaborative botnet detection, on-line gaming etc. For instance, suppose two real estate companies want to detect the customers (e.g., homeowners) who are double dealing, i.e. have signed exclusive contracts with both the companies to assist them in selling their house. mPSI is a proper choice for this situation.

There are several existing works on mPSI. Two main challenges in this area – apart from establishing security in standard model against malicious adversaries to obtain *efficient* mPSI construction and to obtain *fairness* for such construction. Efficiency is measured by communication and computation complexities while fairness means if one party gets the intersection then the other party should also get the intersection. An mPSI can be obtained by two instantiations of an one-way PSI protocol [12]. However, this approach does not prevent a player from unfairly aborting the protocol, thereby unable to maintain fairness. Most of the fair cryptographic protocols achieve fairness in the optimistic way i.e., they use an off-line trusted third party, called *arbiter* who involves in the protocol to recover the output for the honest party only if a corrupted player prematurely aborts the protocol. However, in real life it is practically infeasible to find such a fully trusted third party. Achieving optimistic fairness in PSI protocol is not an easy task mainly due to – (a) lack of generic construction for optimistic fair protocols, (b) use of fully trusted arbiter who gets access to some private information.

Consider the scenario, where the parties may wish to learn cardinality rather than the contents of the intersection of their respective private sets. *Private Set Intersection Cardinality* (PSI-CA) is an appropriate choice for this scenario. PSI-CA has emerged as an object of fundamental interest for many real life applications such as privately compare equal-size low-entropy vectors, genomic operations, affiliation-hiding authentication, social networks, location sharing, role-based association mining, etc.

Mutual private set intersection cardinality (mPSI-CA) is another flavor of PSI-CA, where both the parties obtain the cardinality of the intersection. For instance, suppose two different health organizations want to know the number of common villagers who are suffering from a particular disease in a village. None of the organizations will disclose their list of suspects but they may learn the number of common suspects by running an mPSI-CA.

Our Contribution: In this paper, we restrict ourselves to mPSI and its cardinality variant mPSI-CA. ElGamal encryption [17], distributed ElGamal encryption [4] and Cramer-Shoup cryptosystem [8] are important ingredient of our constructions. We provide a novel and unique approach for unification of the above ingredients to realize practical mPSI and mPSI-CA protocols. The importance of this work is two fold:

• In the first phase we design a *fair* mPSI protocol with *linear complexity* over *prime* order group. Our mPSI is secure against both the malicious parties under the DDH assumption.

Fairness is arguably hardest part to achieve for mPSI. We achieve fairness in the optimistic way by using an off-line arbiter. While most of the prior works use fully trusted arbiter, our mPSI uses only semi-trusted arbiter who does not have access to the private information of any of the parties, but follows the protocol honestly.

We emphasize that our mPSI protocol is more efficient than the existing mPSI protocols [7, 14, 15, 28, 29]. The mPSI constructions of [15], [7] and [29] attain quadratic computation complexities. The mPSI of [15] has the additional restriction that the party constructing the polynomial should have more number of inputs than the other party, whereas our protocol does not have any such restriction. The mPSI of [29] and [28] do not preserve fairness. To

the best of our knowledge, [14, 15] are the most efficient fair mPSI protocols. More precisely, [14] requires approximately $182(v + w)$ exponentiations and $119(v + w)$ group elements over *composite* order group. In contrast, our mPSI requires only $55v + 82w + 27$ exponentiations and $22v + 31w + 20$ group elements over *prime* order group. The security analysis in [14] is in hybrid model.

• The second phase of this work extends our mPSI to mPSI-CA utilizing two random permutations. Before our work, no fair mPSI-CA were known over *prime* order group with *linear* complexity. To the best of our knowledge, there are only two mPSI-CA protocols [7, 29], both of which are based on composite order group and attain quadratic computational overhead which is linear for our case. The mPSI-CA of [7] can be made to achieve fairness using an optimistic fair exchange scheme. However, this approach does not work in general cases where inputs are not certified as it is hard to force the participants to use the same inputs in two different instances. On the contrary, our mPSI-CA achieves fairness using only a semi-trusted arbiter. Particularly, our mPSI-CA is the *first fair* mPSI-CA which is proven to be secure against malicious adversaries with *linear complexity*.

Related Works: There has been a sequence of works on constructing PSI [7, 11–16, 19, 21–23, 25–30]. These works employed several existing ideas and advances such as Oblivious Polynomial Evaluations (OPE), Oblivious Pseudorandom Function (OPRF), Unpredictable function (UPF), Additively Homomorphic Encryption (AHE), Garble Circuit (GC), Bloom filter (BF) etc. While the PSI schemes [7, 11, 14–16, 21–23, 26, 27, 29] are secure in malicious model, the constructions [12, 13, 19, 25, 28, 30] are secure only in semi-honest model. In malicious model, adversaries can run any efficient strategy in order to carry out their attack and can deviate at will from the prescribed protocol. On the other hand, in semi-honest model, adversaries follow the prescribed protocol, but try to gain more information than allowed from the protocol transcript.

PSI-CA has been studied extensively and a variety of solutions are provided with improved efficiency and security level [1, 7, 10, 19, 24, 29].

Kissner and Song [29] presented the first mPSI protocol based on OPE that can support more than two players in the communication system. However, it does not guarantee fairness. Later on, another OPE-based mPSI was proposed by Camenisch and Zaverucha [7], where the inputs must be certified by a trusted party. Computation overhead of the scheme is quadratic and fairness can be achieved using an optimistic fair exchange scheme. However, it does not work in general cases where inputs are not certified. Kim et al. [28] proposed an mPSI using prime representation technique and threshold additive homomorphic encryption [9]. Fairness is not considered in their security model. Dong et al. [15] sketched the first fair optimistic mPSI protocol with quadratic computation overhead. Prior to our work, the only fair optimistic mPSI protocol [14] with linear computation and communication overhead was over composite order group. However, constructions for fair mPSI-CA with linear complexity have remained elusive.

Kissner and Song [29] designed a mPSI-CA protocol which can support more than two participants. The scheme does not preserve fairness. Camenisch and Zaverucha [7] constructed an OPE-based fair mPSI-CA protocol for certified sets with quadratic communication and computation complexity. Both the constructions [7, 29] are over composite order group. We briefly summarize the results on mPSI and mPSI-CA from prior work in Table 1. While existential results have been obtained for efficient mPSI and mPSI-CA constructions, achieving fairness

Table 1 : Comparative summary of mutual PSI and mutual PSI-CA protocols

mPSI Protocol	Adv. model	Security assumption	Comm. cost	Comp. cost	Fairness	Optimistic	Group order
[29]	Mal	AHE	$O(w+v)$	$O(wv)$	no	no	composite
[7]	Mal	Strong RSA	$O(w+v)$	$O(wv)$	yes	yes	composite
[28]	SH	AHE	$O(w+v)$	$O(w+v)$	no	no	composite
[15]	Mal	AHE,VE	$O(w+v)$	$O(wv)$	yes	yes	prime
[14]	Mal	Dq-DHI, DCR, DDH	$O(w+v)$	$O(w+v)$	yes	yes	composite
Our	Mal	DDH	$O(w+v)$	$O(w+v)$	yes	yes	prime
mPSI-CA Protocol	Adv. model	Security assumption	Comm. cost	Comp. cost	Fairness	Optimistic	Group order
[29]	Mal	AHE	$O(v)$	$O(v^2)$	no	no	composite
[7]	Mal	Strong RSA	$O(w+v)$	$O(wv)$	yes	yes	composite
Our	Mal	DDH	$O(w+v)$	$O(w+v)$	yes	yes	prime

AHE=Additively Homomorphic Encryption, VE=Verifiable Encryption, SH= Semi-honest, Dq-DHI=Decisional q-Diffie-Hellman Inversion, DCR=Decisional Composite Residuosity, DDH=Decisional Diffie-Hellman, Mal=Malicious, v, w are the sizes of input sets.

have turned out to be much harder.

1.1 Organization

The rest of the paper is organized as follows. In Section 2 we provide preliminaries. We describe constructions and security proofs of our mPSI and mPSI-CA in Section 3 and 4 respectively. The efficiency is given in Section 5. We conclude the paper in Section 6.

2 Preliminaries

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

Definition 2.1. Negligible Function: *A function $\epsilon : \mathbb{N} \rightarrow \mathbb{R}$ is said to be negligible function of κ if for each constant $c > 0$, we have $\epsilon(\kappa) = o(\kappa^{-c})$ for all sufficiently large κ.*

Definition 2.2. *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 after completion of the protocol Π between A and B.*

Definition 2.3. Decisional Diffie-Hellman (DDH) Assumption [3]: *Let the algorithm $gGen$ generates a modulus n and a generator g of a multiplicative group $\mathbb{G}$ of order n on the input 1^κ. Suppose $a, b, c \leftarrowtail \mathbb{Z}_n$. Then the DDH assumption states that no PPT algorithm $\mathcal{A}$ can distinguish between the two distributions $\langle g^a, g^b, g^{ab}\rangle$ and $\langle g^a, g^b, g^c\rangle$ i.e., $|Prob[\mathcal{A}(g, g^a, g^b, g^{ab}) = 1] - Prob[\mathcal{A}(g, g^a, g^b, g^c) = 1]|$ is negligible function of κ.*

2.1 Security Model

Informally, the basic security requirements of any multi-party protocol are

(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 prescribe output.

(c) *Fairness.* A dishonest party should receive its output if and only if the honest party also receives its output.

In this work, we focus on the *malicious* model where the adversary can behave arbitrarily. The security is formalized by an ideal process that involves an incorruptible trusted third party, who receives the inputs of the involved parties, computes the functionality on the receiving inputs and sends outputs back to the parties. A protocol is said to be secure if any adversary in the real protocol can be simulated by an adversary in the ideal world. The security framework of mPSI is formally described below following [15].

The real world : The protocol has three participants – party A, party B and an arbiter Ar. All the participants have access to the public parameters of the protocol including the functionality $\mathcal{F}_{\mathsf{mPSI}} : (X, Y) \rightarrow (X \cap Y, X \cap Y)$, the security parameter κ, Ar's public key pk_{Ar} and other cryptographic parameters to be used. Party A has a private input X, party B has a private input Y and Ar has an input $\in \{\circ, \perp\}$. The adversary $\mathcal{C}$ can corrupt upto two parties in the protocol and can behave arbitrarily. At the end of the execution, an honest party outputs whatever 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 of $A, B, Ar, \mathcal{C}$ in the real world is denoted by $\mathsf{REAL}_{\mathsf{mPSI},\mathcal{C}}(X, Y)$.

The ideal process : In the ideal process, there is an incorruptible trusted party T who can compute the ideal functionality $\mathcal{F}_{\mathsf{mPSI}}$, and parties $\bar{A}, \bar{B}$ and $\bar{Ar}$. Party $\bar{A}$ has input X, $\bar{B}$ has input Y and $\bar{Ar}$ has an input $\in \{\circ, \perp\}$. The interaction is as follows:

(i) $\bar{A}$ sends $\overline{X}$ or $\perp$ to T, following it $\bar{B}$ sends $\overline{Y}$ or $\perp$ to T; and then $\bar{Ar}$ sends two messages $b_A \in \{\circ, \perp\} \cup X_A$ and $b_B \in \{\circ, \perp\} \cup Y_B$ to T, where X_A and Y_B are two arbitrary sets. The inputs $\overline{X}$ and $\overline{Y}$ may be different from X and Y respectively if the party is malicious.

(ii) T sends private delayed output to $\bar{A}$ and $\bar{B}$. T's reply to $\bar{A}(\bar{B})$ depends on $\bar{A}$ and $\bar{B}$'s messages and $b_A(b_B)$. Response of T to $\bar{A}(\bar{B})$ is as follows:

  (a) If $b_A(b_B) = \circ$, and T has received $\overline{X} \neq \perp$ from $\bar{A}$ and $\overline{Y} \neq \perp$ from $\bar{B}$, then T sends $\overline{X} \cap \overline{Y}$ to $\bar{A}(\bar{B})$.

  (b) Else if $b_A(b_B) = \circ$, but T has received $\perp$ from either $\bar{A}$ or $\bar{B}$, then T sends $\perp$ to $\bar{A}(\bar{B})$.

  (c) Else $b_A(b_B) \neq \circ$, then T sends $b_A(b_B)$ to $\bar{A}(\bar{B})$.

In the ideal process, if $\bar{A}$, $\bar{B}$ and $\bar{A}r$ are honest then they behave as follows: $\bar{A}$ and $\bar{B}$ send their inputs to T and $\bar{A}r$ sends $b_A = \circ$ and $b_B = \circ$. The ideal process adversary $\mathcal{SIM}$ gets the inputs of the corrupted parties and may replace them and gets T's response to corrupted parties. The joint output of $\bar{A}, \bar{B}, \bar{A}r, \mathcal{SIM}$ in the ideal process is denoted by $\mathsf{IDEAL}_{\mathcal{F}_{\mathsf{mPSI}},\mathcal{SIM}}(X, Y)$. The security definition in terms of simulatability is

Definition 2.4. Simulatability: *Let $\mathcal{F}_{\mathsf{mPSI}} : (X, Y) \to (X \cap Y, X \cap Y)$ be the functionality for* mPSI *protocol. Then the protocol* mPSI *is said to securely compute $\mathcal{F}_{\mathsf{mPSI}}$ in malicious model if for every real world adversary $\mathcal{C}$, there exists an ideal world adversary $\mathcal{SIM}$ such that the joint distribution of all outputs of the ideal world is computationally indistinguishable from the outputs in the real world, i.e.,* $\mathsf{IDEAL}_{\mathcal{F}_{\mathsf{mPSI}},\mathcal{SIM}}(X, Y) \equiv^c \mathsf{REAL}_{\mathsf{mPSI},\mathcal{C}}(X, Y)$.

Note that the security framework for mPSI-CA is same as the security framework of mPSI except that each $X \cap Y$ will be changed to $|X \cap Y|$.

2.2 Homomorphic Encryption [5]

We describe below *multiplicatively* homomorphic encryption schemes the ElGamal encryption [17] and the distributed ElGamal encryption [4] which are semantically secure provided DDH problem is hard in underlying group.

ElGamal encryption: The ElGamal encryption [17] is a multiplicatively homomorphic encryption $\mathcal{EL} = (\mathcal{EL}.\mathsf{Setup}, \mathcal{EL}.\mathsf{KGen}, \mathcal{EL}.\mathsf{Enc}, \mathcal{EL}.\mathsf{Dec})$ which works as follows:

- $\mathcal{EL}.\mathsf{Setup}(1^\kappa)$ – On input 1^κ, a trusted third party 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.
- $\mathcal{EL}.\mathsf{KGen}(\mathsf{par})$ – User A_i chooses $a_i \leftarrowtail \mathbb{Z}_q$, computes $y_{A_i} = g^{a_i}$, reveals $epk_{A_i} = y_{A_i}$ as his public key and keeps $esk_{A_i} = a_i$ secret to himself.
- $\mathcal{EL}.\mathsf{Enc}(m, epk_{A_i}, \mathsf{par}, r)$ – Encryptor encrypts a message $m \in \mathbb{G}$ using the public key $epk_{A_i} = y_{A_i}$ by computing ciphertext tuple $\mathsf{eE}_{epk_{A_i}}(m) = (\alpha, \beta) = (g^r, my_{A_i}^r)$, where $r \leftarrowtail \mathbb{Z}_q$.
- $\mathcal{EL}.\mathsf{Dec}(\mathsf{eE}_{epk_{A_i}}(m), esk_{A_i})$ – On receiving ciphertext tuple $\mathsf{eE}_{epk_{A_i}}(m) = (\alpha, \beta) = (g^r, my_{A_i}^r)$, decryptor A_i decrypts it using the secret key $esk_{A_i} = a_i$ by computing $\frac{\beta}{(\alpha)^{a_i}} = \frac{m(g^{a_i})^r}{(g^r)^{a_i}} = m$.

Distributed ElGamal encryption [4]: The distributed ElGamal encryption $\mathcal{DEL} = (\mathcal{DEL}.\mathsf{Setup}, \mathcal{DEL}.\mathsf{KGen}, \mathcal{DEL}.\mathsf{Enc}, \mathcal{DEL}.\mathsf{Dec})$ is executed between two parties A_1 and A_2 as follows:

- $\mathcal{DEL}.\mathsf{Setup}(1^\kappa)$ – Same as the ElGamal encryption.
- $\mathcal{DEL}.\mathsf{KGen}(\mathsf{par})$ – Each participant $A_i, i = 1, 2$ selects $a_i \leftarrowtail \mathbb{Z}_q$, publishes $y_{A_i} = g^{a_i}$ along with a zero-knowledge proof $\mathsf{PoK}\{a_i | y_{A_i} = g^{a_i}\}$. Then, each of A_1, A_2 publishes the public key for the $\mathcal{DEL}$ as $pk = h = g^{a_1+a_2}$, while the secret key for $\mathcal{DEL}$ is $sk = a_1+a_2$. Note that sk is not known to anyone under the hardness of DLP in $\mathbb{G}$.

$\mathcal{DEL}.\mathsf{Enc}(m, pk, \mathsf{par}, r)$ – Encryptor encrypts a message $m \in \mathbb{G}$ using public key $pk = h = g^{a_1+a_2}$ and computes the ciphertext tuple $\mathsf{dE}_{pk}(m) = (\alpha, \beta) = (g^r, mh^r)$, where $r \leftarrow\!\!\!\!\!\!\sim\ \mathbb{Z}_q$.

$\mathcal{DEL}.\mathsf{Dec}(\mathsf{dE}_{pk}(m), a_1, a_2)$ – Given a ciphertext $\mathsf{dE}_{pk}(m) = (\alpha, \beta) = (g^r, mh^r)$, each participant A_i publishes $\alpha_i = \alpha^{a_i}$ and proves the correctness of the proof $\mathsf{PoK}\{a_i | y_{A_i} = g^{a_i} \wedge \alpha_i = \alpha^{a_i}\}$ to A_j, where $i, j \in \{1, 2\}$ and $i \neq j$. If proofs are valid, then each of A_1, A_2 recovers the message m as $\frac{\beta}{\alpha_1\alpha_2} = \frac{\beta}{(\alpha)^{(a_1+a_2)}} = \frac{mh^r}{g^{r(a_1+a_2)}} = \frac{mh^r}{h^r} = m$.

2.3 Verifiable Encryption [5]

We describe below a CCA2-secure verifiable encryption scheme $\mathcal{VE}$= ($\mathcal{VE}.\mathsf{Setup}$, $\mathcal{VE}.\mathsf{KGen}, \mathcal{VE}.\mathsf{Enc}, \mathcal{VE}.\mathsf{Dec}$) which is a variant of Cramer-Shoup cryptosystem [8] over prime order group [15].

$\mathcal{VE}.\mathsf{Setup}(1^\kappa)$ – On input 1^κ, a trusted third party outputs a public parameter $\mathsf{gpar}=(\mathsf{par}, \widehat{g}, \mathcal{H})$, where $\mathsf{par} = (p, q, g)$, p, q are primes such that q divides $p - 1$ and $g, \widehat{g}$ are generators of the unique cyclic subgroup $\mathbb{G}$ of $\mathbb{Z}_p^*$ of order q, $\mathcal{H} : \{0, 1\}^* \rightarrow \mathbb{Z}_q$ is an one-way hash function.

$\mathcal{VE}.\mathsf{KGen}(\mathsf{par}, \widehat{g})$ – User U chooses $u_1, u_2, v_1, v_2, w_1 \leftarrow\!\!\!\!\!\!\sim\ \mathbb{Z}_q$, computes $a = g^{u_1}\widehat{g}^{u_2}, b = g^{v_1}\widehat{g}^{v_2}$, $c = g^{w_1}$, publishes $vpk_U = (a, b, c)$ as his public key and keeps $vsk_U = (u_1, u_2, v_1, v_2, w_1)$ secret to himself.

$\mathcal{VE}.\mathsf{Enc}(m, vpk_U, \mathsf{gpar}, z, L, \mathcal{H})$ – To encrypt a message $m \in \mathbb{G}$ using public key $vpk_U = (a, b, c)$, encryptor picks $z \leftarrow\!\!\!\!\!\!\sim\ \mathbb{Z}_q$ and sets $e_1 = g^z, e_2 = \widehat{g}^z, e_3 = c^z m$, constructs a label $L \in \{0, 1\}^*$ using information that are available to both encryptor and decryptor, computes $\rho = \mathcal{H}(e_1, e_2, e_3, L)$, sets $e_4 = a^z b^{z\rho}$, and computes the ciphertext $\mathsf{vE}_{vpk_U}(m) = (e_1, e_2, e_3, e_4)$.

$\mathcal{VE}.\mathsf{Dec}(\mathsf{vE}_{vpk_U}(m), vsk_U, L, \mathcal{H})$ – Decryptor U, on receiving ciphertext $\mathsf{vE}_{vpk_U}(m) = (e_1, e_2, e_3, e_4)$, computes $\rho = \mathcal{H}(e_1, e_2, e_3, L)$ and then verifies $e_1^{u_1} e_2^{u_2} (e_1^{v_1} e_2^{v_2})^\rho = e_4$ using secret key $vsk_U = (u_1, u_2, v_1, v_2, w_1)$. If the verification succeeds, then he recovers the message m by computing $e_3/(e_1)^{w_1} = c^z m/g^{zw_1} = g^{zw_1} m/g^{zw_1} = m$.

2.4 Zero-Knowledge Proof of Knowledge [2]

Zero-Knowledge proof [2] π is a two-party protocol, where prover (P) wants to convince the verifier (V) about the truth of the claim that he knows some secret values, and the verifier wants to check that the claim is true. The prover can prove to the verifier that the claim is true, without conveying any additional information apart from the fact that the claim is indeed true. A zero-knowledge proof protocol π for relation R should satisfy following three properties:

(a) *Completeness* – If P and V follow on input x and secret value w (witness) to P where $(x, w) \in R$, then V always accepts.

(b) *Soundness* – Given any input x and any pair of accepting transcripts $(v, c, r), (v, c', r')$ on x with $c \neq c'$, there exists a polynomial-time algorithm $\mathcal{B}$ that outputs w such that $(x, w) \in R$.

(c) *Zero-knowledge* – There exists a PPT algorithm $\mathcal{A}$ such that

$$\left\{\left\langle P(x,w), V(x,c)\right\rangle\right\}_{(x,w)\in R, c\in\{0,1\}^*} \equiv \{\mathcal{A}(x,c)\}_{x\in L_R, c\in\{0,1\}^*},$$

where L_R is the language of relation R, $\mathcal{A}(x, c)$ denotes the output of $\mathcal{A}$ upon input x and c, and $\left\langle P(x, w), V(x, c)\right\rangle$ denotes the output transcript of an execution between P and V, where P has input (x, w), V has input x, and V's random tape (determining its query) equals c.

Zero-Knowledge Proof for Discrete Logarithm:

We follow the notations introduced by [6] for the various zero-knowledge proofs of knowledge of discrete logarithms and proofs of validity of statements about discrete logarithms. We describe below a general construction of interactive zero-knowledge proofs of knowledge, denoted by

$$\mathsf{PoK}\{(\alpha_1, ..., \alpha_l) \mid \wedge_{i=1}^{m} X_i = f_i(\alpha_1, ..., \alpha_l)\}, \tag{2.1}$$

where the prover wants to prove the knowledge of $(\alpha_1, ..., \alpha_l)$ to the verifier by sending the commitments $X_i = f_i(\alpha_1, ..., \alpha_l), i = 1, ..., m$ such that extracting $(\alpha_1, ..., \alpha_l)$ from $X_1, ..., X_m$ is infeasible for anyone. For each $i = 1, ..., m$, f_i is publicly computable linear function from $\mathcal{X}^l$ to $\mathcal{Y}$, where $\mathcal{X}$ is *additive* set and $\mathcal{Y}$ is *multiplicative* set. The verification of the proof is done by executing the following steps:

1. The prover chooses $v_1, ..., v_l$ and sends the commitments $\overline{X}_i = f_i(v_1, ..., v_l), i = 1, ..., m$ to the verifier.
2. The verifier sends a challenge $c \in \mathcal{X}$ to the prover.
3. For each $j = 1, ..., l$, prover sets $r_j = v_j + c\alpha_j$ and sends the response $(r_1, ..., r_l)$ to the verifier.
4. The verifier checks whether the relations $f_i(r_1, ..., r_l) = \overline{X}_i X_i^c$, $i = 1, ..., m$ hold or not. If all of them hold then the verifier accepts it, else rejects it.

Lemma 2.1. *If EXP be the number of exponentiations and GE be the total number of group elements for verification of the proof system represented by the equation 2.1, then we have:* (a) *Exp =* $m + 2\Sigma_{i=1}^{m}$(*number of exponentiations to compute* X_i), (b) *GE =* $m + l + 1$.

Zero-Knowledge Argument for Shuffle [20]:

We briefly discuss the zero-knowledge argument for shuffle of [20] which we use in our mPSI-CA. Let p, q be two primes such that q divide $p - 1$, $\mathbb{G}$ be a subgroup of $\mathbb{Z}_p^*$ of order q, $g_0(\neq 1)$ be an element of $\mathbb{G}$, $x \leftarrow\!\!\!\!\!\!\rightarrow \mathbb{Z}_q$ be a private key and $m_0 = g_0^x \bmod p$ be a public key used for re-encryption in shuffling. The prover P chooses $\{A_{0i} \leftarrow\!\!\!\!\!\!\rightarrow \mathbb{Z}_q\}_{i=1}^{v}$ and a permutation matrix $(A_{ji})_{j,i=1,...,v}$ of order $v \times v$ corresponding to a permutation $\phi \in \Sigma_v$, where Σ_v denotes the set of

all possible permutations over the set $\{1, ..., v\}$. P shuffles v ElGamal ciphertexts $\{(g_i, m_i)\}_{i=1}^{v}$ to $\{(g'_i, m'_i)\}_{i=1}^{v}$ as

$$(g'_i, m'_i) = (\prod_{u=0}^{v} g_u^{A_{ui}}, \prod_{u=0}^{v} m_u^{A_{ui}}) = (g_0^{A_{0i}} g_{\phi^{-1}(i)}, m_0^{A_{0i}} m_{\phi^{-1}(i)}) \bmod p. \quad (2.2)$$

The zero-knowledge argument of [20] for the correctness of a shuffle is denoted by

$$\hat{\pi} = \mathsf{PoK}\{(\phi \in \Sigma_v, A_{01}, ..., A_{0v} \in \mathbb{Z}_q) | \{(g'_i, m'_i) = (g_0^{A_{0i}} g_{\phi^{-1}(i)}, m_0^{A_{0i}} m_{\phi^{-1}(i)})\}_{i=1}^{v}\}. \quad (2.3)$$

The prover P wants to prove the knowledge of the permutation $\phi \in \Sigma_v$ and randomness $\{A_{0i} \in \mathbb{Z}_q\}_{i=1}^{v}$ to the verifier V such that equation 2.2 holds for each $i = 1, ..., v$. Note that decryption of the ciphertexts (g'_i, m'_i) and $(g_{\phi^{-1}(i)}, m_{\phi^{-1}(i)})$ give same message. The prover P and the verifier V executes the following steps for the verification of $\hat{\pi}$:

Commitment: P chooses $\{A_{u0}, A'_u \leftarrow \mathbb{Z}_q\}_{u=-4}^{v}$, $\{A_{-1i} \leftarrow \mathbb{Z}_q\}_{i=1}^{v}$ and computes the followings:

$$A_{-2i} = \sum_{j=1}^{v} 3A_{j0}^2 A_{ji} \bmod q, i = 1, ..., v;$$

$$A_{-3i} = \sum_{j=1}^{v} 3A_{j0} A_{ji} \bmod q, i = 1, ..., v;$$

$$A_{-4i} = \sum_{j=1}^{v} 2A_{j0} A_{ji} \bmod q, i = 1, ..., v;$$

$$f'_k = \prod_{u=-4}^{v} f_u^{A_{uk}} \bmod p, k = 0, ..., v;$$

$$\tilde{f}'_0 = \prod_{u=-4}^{v} f_u^{A'_u} \bmod p;$$

$$g'_0 = \prod_{u=0}^{v} g_u^{A_{u0}} \bmod p;$$

$$m'_0 = \prod_{u=0}^{v} m_u^{A_{u0}} \bmod p;$$

$$w = \sum_{j=1}^{v} A_{j0}^3 - A_{-20} - A'_{-3} \bmod q;$$

$$\hat{w} = \sum_{j=1}^{v} A_{j0}^2 - A_{-40} \bmod q.$$

Finally P sends $g'_0, m'_0, \tilde{f}'_0, \{f'_k\}_{k=0}^{v}, w, \hat{w}$ as a commitment to V.

Challenge: V chooses $\{c_i \leftarrow \mathbb{Z}_q\}_{i=1}^{v}$ and sends it as challenge to P.

Response: P sets $c_0 = 1 \bmod q$ and for each $u = -4, ..., v$ computes the following:

$$r_u = \sum_{k=0}^{v} A_{uk} c_k \bmod q;$$
$$r'_u = \sum_{i=1}^{v} A_{ui} c_i^2 + A'_u \bmod q$$

P then sends the responses $\{r_u, r'_u\}_{u=-4}^{v}$ to V.

Verification: V accepts the shuffle if each of the following equations holds for an $\alpha \leftarrow\!\!\!\!\!\!\rightharpoonup \mathbb{Z}_q$:

$$\prod_{u=-4}^{v} f_u^{r_u + \alpha r'_u} \equiv f'_0 \tilde{f}_0'^{\alpha} \prod_{i=1}^{v} f_i'^{c_i + \alpha c_i^2} \bmod p;$$
$$\prod_{u=0}^{v} g_u^{r_u} \equiv \prod_{u=0}^{v} g_u'^{c_u} \bmod p;$$
$$\prod_{u=0}^{v} m_u^{r_u} \equiv \prod_{u=0}^{v} m_u'^{c_u} \bmod p;$$
$$\sum_{i=1}^{v} (r_i^3 - c_i^3) \equiv r_{-2} + r'_{-3} + w \bmod q;$$
$$\sum_{i=1}^{v} (r_i^2 - c_i^2) \equiv r_{-4} + \hat{w} \bmod q.$$

This proof system satisfies soundness property under the hardness of DDH assumption. Total number of exponentiations and group elements for verification of the proof system are respectively $15v + 22$ and $4v + 16$. For the distributed ElGamal encryption $\mathcal{DEL}$ presented in the section 2.2, the zero-knowledge argument for shuffle will be of the form

$$\mathsf{PoK}\big\{(\phi \in \Sigma_v, \rho_1, ..., \rho_v \in \mathbb{Z}_q) | \{C'_i = C_{\phi^{-1}(i)} \mathcal{DEL}.\mathsf{Enc}(1, pk, \mathsf{par}, \rho_i)\}_{i=1}^{v}\big\},$$

where ciphertexts $\{C_i = (g_i, m_i)\}_{i=1}^{v}$ and $\{C'_i = (g_{i'}, m_{i'})\}_{i=1}^{v}$.
Note that using Fiat-Shamir method [18], the interactive proof systems represented by the equations 2.1 and 2.3 can be converted to non-interactive proof system.

3 The mPSI

3.1 Construction

Our mPSI protocol consists of

– a **Setup** algorithm to generate global parameter by a trusted third party, public/private key generation of participants A, B and an arbiter Ar,

– an $\mathbf{mPSI}$ **Protocol** executed between two parties A, B with their private input sets X, Y respectively to compute $X \cap Y$, and

– a **Dispute Resolution Protocol** involving an off-line arbiter Ar. The arbiter Ar takes part into the Dispute Resolution protocol only when a corrupted player prematurely aborts the protocol and resolve the dispute without knowing the private information of A and B.

The $\mathtt{Setup}$ algorithm is represented by Figure 1.

Setup(1^κ) – We use the ElGamal encryption $\mathcal{EL}$, the distributed ElGamal encryption $\mathcal{DEL}$ and the verifiable encryption $\mathcal{VE}$ over prime order group as described in the sections 2.3 and 2.4.

– A trusted third party generates global parameter

$$\mathsf{gpar} = (\mathsf{par}, \widehat{g}, \mathcal{H}) \leftarrow \mathcal{VE}.\mathsf{Setup}(1^\kappa), \text{ where } \mathsf{par} = (p, q, g).$$

– Each of A, B generates

$$(epk_A = y_A = g^{a_1}, esk_A = a_1) \leftarrow \mathcal{EL}.\mathsf{KGen}(\mathsf{par}), \text{ where } a_1 \leftarrow\!\!\prec \mathbb{Z}_q,$$
$$(epk_B = y_B = g^{a_2}, esk_B = a_2) \leftarrow \mathcal{EL}.\mathsf{KGen}(\mathsf{par}), \text{ where } a_2 \leftarrow\!\!\prec \mathbb{Z}_q.$$

They publishe the public keys epk_A, epk_B through the trusted third party who acts as certifying authority in this case. Parties A, B keeps the respcetive secret keys esk_A, esk_B to themselves.

– Arbiter Ar generates

$$(vpk_{Ar} = (a = g^{u_1}\widehat{g}^{u_2}, b = g^{v_1}\widehat{g}^{v_2},\ c = g^{w_1}), vsk_{Ar} = (u_1, u_2, v_1, v_2, w_1))$$
$$\leftarrow \mathcal{VE}.\mathsf{KGen}(\mathsf{par}, \widehat{g}), \text{ where } u_1, u_2, v_1, v_2, w_1 \leftarrow\!\!\prec \mathbb{Z}_q$$

and publishes the public key vpk_{Ar} through the trusted third party who works as certifying authority in this case.

– Let $pk = h = epk_A \cdot epk_B = g^{a_1+a_2}$ and $sk = a_1 + a_2$. Then (pk, sk) pair serves as the public-secret key pair for $\mathcal{DEL}$. Note that the secret key $sk = a_1 + a_2$ for $\mathcal{DEL}$ is not known to anyone. However, the public key pk for $\mathcal{DEL}$ is publicly computable from epk_A and epk_B.

Figure 1 : Setup algorithm of our mPSI

We use the multiplicatively homomorphic property of $\mathcal{DEL}$ i.e.,

$$(\mathsf{dE}_{pk}(m_1))(\mathsf{dE}_{pk}(m_2)) = \mathsf{dE}_{pk}(m_1 m_2), (\mathsf{dE}_{pk}(m))^k = \mathsf{dE}_{pk}(m^k), \text{ where } k \in \mathbb{Z}_q.$$

mPSI Protocol: The 5 round mPSI protocol (see Figure 2) is an interactive protocol between parties A and B. In each round, an interactive proof is generated and sent by one party, which is then verified by the other party using a similar technique presented in section 2.4. Party A with private input set $X = \{x_1, ..., x_v\}$ and B with private input set $Y = \{y_1, ..., y_w\}$ engage in mPSI protocol, where $(\mathsf{gpar}, epk_A, epk_B, pk = epk_A \cdot epk_B)$ is their common input. To get the intersection $X \cap Y$, A and B proceed in 6 steps as follows:

Step 1. Party A

(i) chooses $r_{x_1}, ..., r_{x_v} \leftarrowtail \mathbb{Z}_q$ and encrypts each member $x_i \in X$ with the public key $pk = h = g^{a_1+a_2}$ to get

$$\mathsf{dE}_{pk}(x_i) = (c_{x_i} = g^{r_{x_i}}, d_{x_i} = x_i h^{r_{x_i}}) \leftarrow \mathcal{DEL}.\mathsf{Enc}(x_i, pk, \mathsf{par}, r_{x_i});$$

(ii) generates the proof

$$\pi_1 = \mathsf{PoK}\big\{(r_{x_1}, ..., r_{x_v}) | \wedge_{i=1}^{v} (c_{x_i} = g^{r_{x_i}})\big\};$$

(iii) sends $R_1 = \langle \{\mathsf{dE}_{pk}(x_i)\}_{i=1}^{v}, \pi_1 \rangle$ to B.

Step 2. On receiving $R_1 = \langle \{\mathsf{dE}_{pk}(x_i)\}_{i=1}^{v}, \pi_1 \rangle$ from A, party B verifies the validity of the proof π_1. If verification fails, then B aborts. Otherwise, B does the following:

(i) chooses $r_{y_1}, ..., r_{y_w} \leftarrowtail \mathbb{Z}_q$ and encrypts each $y_j \in Y$ with the public key $pk = h = g^{a_1+a_2}$ and generates

$$\mathsf{dE}_{pk}(y_j) = (c_{y_j} = g^{r_{y_j}}, d_{y_j} = y_j h^{r_{y_j}}) \leftarrow \mathcal{DEL}.\mathsf{Enc}(y_j, pk, \mathsf{par}, r_{y_j});$$

(ii) selects $r, r_{\hat{g}}, \alpha \leftarrowtail \mathbb{Z}_q$ and computes $\hat{g} = g^{\alpha}$,

$$\begin{aligned}
&\mathsf{dE}_{pk}(\hat{g}^r) = (c_{\hat{g}} = g^{r_{\hat{g}}}, d_{\hat{g}} = \hat{g}^r h^{r_{\hat{g}}}) \leftarrow \mathcal{DEL}.\mathsf{Enc}(\hat{g}^r, pk, \mathsf{par}, r_{\hat{g}}),\\
&\mathsf{dE}_{pk}((y_j)^r) = (\hat{c}_{y_j} = (c_{y_j})^r, \hat{d}_{y_j} = (d_{y_j})^r) \text{ for } 1 \le j \le w,\\
&\mathsf{dE}_{pk}((x_i)^r) = (\hat{c}_{x_i} = (c_{x_i})^r, \hat{d}_{x_i} = (d_{x_i})^r) \text{ for } 1 \le i \le v;
\end{aligned}$$

(iii) constructs proof

$$\begin{aligned}
\pi_2 = \mathsf{PoK}\big\{(r_{y_1}, ..., r_{y_w}, r, r_{\hat{g}}) | \wedge_{j=1}^{w} (c_{y_j} = g^{r_{y_j}})(\hat{c}_{y_j} = (c_{y_j})^r)(\hat{d}_{y_j} = (d_{y_j})^r)\\
\wedge_{i=1}^{v} (\hat{c}_{x_i} = (c_{x_i})^r)(\hat{d}_{x_i} = (d_{x_i})^r) \wedge (c_{\hat{g}} = g^{r_{\hat{g}}}) \wedge (d_{\hat{g}} = \hat{g}^r h^{r_{\hat{g}}})\}\big\};
\end{aligned}$$

(iv) sends $R_2 = \big\langle \{\mathsf{dE}_{pk}(y_j), \mathsf{dE}_{pk}((y_j)^r)\}_{j=1}^{w}, \{\mathsf{dE}_{pk}((x_i)^r)\}_{i=1}^{v}, \mathsf{dE}_{pk}(\hat{g}^r), \hat{g}, \pi_2 \big\rangle$ to A.

Step 3. Party A, on receiving $R_2 = \big\langle \{\mathsf{dE}_{pk}(y_j), \mathsf{dE}_{pk}((y_j)^r)\}_{j=1}^{w}, \{\mathsf{dE}_{pk}((x_i)^r)\}_{i=1}^{v}, \mathsf{dE}_{pk}(\hat{g}^r), \hat{g}, \pi_2 \big\rangle$ from B, checks the validity of the proof π_2. Party A aborts if the verification fails, else dose the following:

(i) selects $r', r_{\bar{g}}, \beta \leftarrowtail \mathbb{Z}_q$ and computes $\bar{g} = g^{\beta}$

$$\begin{aligned}
&\mathsf{dE}_{pk}(\bar{g}^{r'}) = (c_{\bar{g}} = g^{r_{\bar{g}}}, d_{\bar{g}} = \bar{g}^{r'} h^{r_{\bar{g}}}) \leftarrow \mathcal{DEL}.\mathsf{Enc}(\bar{g}^{r'}, pk, \mathsf{par}, r_{\bar{g}}),\\
&\mathsf{dE}_{pk}((x_i)^{rr'}) = (\bar{c}_{x_i} = (\hat{c}_{x_i})^{r'} = (c_{x_i})^{rr'}, \bar{d}_{x_i} = (\hat{d}_{x_i})^{r'} = (d_{x_i})^{rr'}), 1 \le i \le v,\\
&\mathsf{dE}_{pk}((y_j)^{rr'}) = (\bar{c}_{y_j} = (\hat{c}_{y_j})^{r'} = (c_{y_j})^{rr'}, \bar{d}_{y_j} = (\hat{d}_{y_j})^{r'} = (d_{y_j})^{rr'}), 1 \le j \le w;
\end{aligned}$$

(ii) chooses $\alpha_1, ..., \alpha_v \leftarrow\!\!\!\!\!\!\rightharpoonup \mathbb{Z}_q$ and for each $i = 1, ..., v$, computes $(C_{x_i})^{a_1} = (\bar{c}_{x_i} c_{\bar{g}} c_{\hat{g}})^{a_1}$ with his secret key $esk_A = a_1$ and encrypts $(C_{x_i})^{a_1}$ using B's public key epk_B to generate

$$\begin{aligned}\mathsf{eE}_{epk_B}((C_{x_i})^{a_1}) &= (u_{x_i} = g^{\alpha_i}, \bar{u}_{x_i} = (C_{x_i})^{a_1}(y_B)^{\alpha_i}) \\ &\leftarrow \mathcal{EL}.\mathsf{Enc}((C_{x_i})^{a_1}, epk_B = y_B, \mathsf{par}, \alpha_i);\end{aligned}$$

(iii) generates a label $L \in \{0,1\}^*$ using a session ID which has been agreed by all parities beforehand and the hash of past communication;

(iv) chooses $r_1, ..., r_w, z_1, ..., z_w \leftarrow\!\!\!\!\!\!\rightharpoonup \mathbb{Z}_q$, computes $\{\bar{u}_{y_j} = (C_{y_j})^{a_1} g^{r_j} = (\bar{c}_{y_j} c_{\bar{g}} c_{\hat{g}})^{a_1} g^{r_j}\}_{j=1}^{w}$ and for each $j = 1, ..., w$, generates

$$\begin{aligned}&\mathsf{vE}_{vpk_{Ar}}(g^{r_j}) = (t_{1j} = g^{z_j}, t_{2j} = \widehat{g}^{z_j}, t_{3j} = c^{z_j} g^{r_j}, t_{4j} = a^{z_j} b^{z_j \rho_j}) \\ &\quad \leftarrow \mathcal{VE}.\mathsf{Enc}(g^{r_j}, vpk_{Ar}, \mathsf{gpar}, z_j, L, \mathcal{H}), \\ &\text{where } vpk_{Ar} = (a, b, c), \rho_j = \mathcal{H}(t_{1j}, t_{2j}, t_{3j}, L);\end{aligned}$$

(v) constructs proof

$$\begin{aligned}\pi_3 = \mathsf{PoK}\big\{&(a_1, r', r_1, ..., r_w, z_1, ..., z_w, \alpha_1, ..., \alpha_v, r_{\bar{g}}) | (y_A = g^{a_1}) \\ &\wedge_{j=1}^{w} (\bar{c}_{y_j} = (\hat{c}_{y_j})^{r'})(\bar{d}_{y_j} = (\hat{d}_{y_j})^{r'})(\bar{u}_{y_j} = (C_{y_j})^{a_1} \cdot g^{r_j}) \wedge (d_{\bar{g}} = \bar{g}^r h^{r_{\bar{g}}}) \\ &\wedge_{j=1}^{w} (t_{1j} = g^{z_j})(t_{2j} = \widehat{g}^{z_j})(t_{3j} = c^{z_j} g^{r_j})(t_{4j} = a^{z_j} b^{z_j \rho_j}) \wedge (c_{\bar{g}} = g^{r_{\bar{g}}}) \\ &\wedge_{i=1}^{v} (\bar{c}_{x_i} = (\hat{c}_{x_i})^{r'})(\bar{d}_{x_i} = (\hat{d}_{x_i})^{r'})(u_{x_i} = g^{\alpha_i})(\bar{u}_{x_i} = (C_{x_i})^{a_1}(y_B)^{\alpha_i})\big\};\end{aligned}$$

(vi) sends $R_3 = \big\langle \{\mathsf{dE}_{pk}((x_i)^{rr'}), \mathsf{eE}_{epk_B}((C_{x_i})^{a_1})\}_{i=1}^{v}, \{\mathsf{dE}_{pk}((y_j)^{rr'}), \mathsf{vE}_{vpk_{Ar}}(g^{r_j}), \bar{u}_{y_j}\}_{j=1}^{w}, \mathsf{dE}_{pk}(\bar{g}^{r'}), \bar{g}, \pi_3 \big\rangle$ to B.

Step 4. On receiving $R_3 = \big\langle \{\mathsf{dE}_{pk}((x_i)^{rr'}), \mathsf{eE}_{epk_B}((C_{x_i})^{a_1})\}_{i=1}^{v}, \{\mathsf{dE}_{pk}((y_j)^{rr'}), \mathsf{vE}_{vpk_{Ar}}(g^{r_j}), \bar{u}_{y_j}\}_{j=1}^{w}, \mathsf{dE}_{pk}(\bar{g}^{r'}), \bar{g}, \pi_3 \big\rangle$, party B verifies the validity of the proof π_3. If the verification fails, then B aborts. Otherwise, B proceeds as follows:

(i) extracts $\{\bar{c}_{x_i}\}_{i=1}^{v}, \{\bar{c}_{y_j}\}_{j=1}^{w}, c_{\bar{g}}$ from $\{\mathsf{dE}_{pk}((x_i)^{rr'})\}_{i=1}^{v}, \{\mathsf{dE}_{pk}((y_j)^{rr'})\}_{j=1}^{w}, \mathsf{dE}_{pk}(\bar{g}^{r'})$ respectively in R_3 and computes $\{s_{x_i} = (C_{x_i})^{a_2} = (\bar{c}_{x_i} c_{\bar{g}} c_{\hat{g}})^{a_2}\}_{i=1}^{v}, \{s_{y_j} = (C_{y_j})^{a_2} = (\bar{c}_{y_j} c_{\bar{g}} c_{\hat{g}})^{a_2}\}_{j=1}^{w}$ using his secret key $esk_B = a_2$ and $c_{\hat{g}}$ computed in *Step* 2;

(ii) constructs the proof

$$\pi_4 = \mathsf{PoK}\big\{(a_2) | (y_B = g^{a_2}) \wedge_{i=1}^{v} (s_{x_i} = (C_{x_i})^{a_2}) \wedge_{j=1}^{w} (s_{y_j} = (C_{y_j})^{a_2})\big\};$$

(iii) sends $R_4 = \big\langle \{s_{x_i}\}_{i=1}^{v}, \{s_{y_j}\}_{j=1}^{w}, \pi_4 \big\rangle$ to A.

Step 5. Party A, on receiving $R_4 = \big\langle \{s_{x_i} = (C_{x_i})^{a_2}\}_{i=1}^{v}, \{s_{y_j} = (C_{y_j})^{a_2}\}_{j=1}^{w}, \pi_4 \big\rangle$ from B, checks the validity of the proof π_4. Party A aborts if the verification does not succeed, else extracts $d_{\hat{g}}$ from $\mathsf{dE}_{pk}(\hat{g}^r)$ in R_2, does the following using his secret key $esk_A = a_1$ and $\{C_{x_i}, \bar{d}_{x_i}\}_{i=1}^{v}, \{C_{y_j}, \bar{d}_{y_j}\}_{j=1}^{w}, d_{\bar{g}}$ computed in *Step* 3:

(i) computes

$$\frac{\bar{d}_{x_i} d_{\hat{g}} d_{\bar{g}}}{(C_{x_i})^{a_1} s_{x_i}} = \frac{\bar{d}_{x_i} d_{\hat{g}} d_{\bar{g}}}{(\bar{c}_{x_i} c_{\hat{g}} c_{\bar{g}})^{(a_1+a_2)}} = \frac{(d_{x_i})^{rr'} d_{\hat{g}} d_{\bar{g}}}{((c_{x_i})^{rr'} c_{\hat{g}} c_{\bar{g}})^{a_1+a_2}}$$

Common input: $\mathsf{gpar}, epk_A, epk_B, pk = epk_A \cdot epk_B, vpk_{Ar}$

A's private input:		B's private input:
$X = \{x_1, ..., x_v\}, esk_A = a_1$		$Y = \{y_1, ..., y_w\}, esk_B = a_2$
$r_{x_1}, ..., r_{x_v} \leftarrow\!\!\!\!\!\!\rightharpoonup \mathbb{Z}_q$ for $1 \le i \le v$, $\mathsf{dE}_{pk}(x_i) = (c_{x_i} = g^{r_{x_i}}, d_{x_i} = x_i h^{r_{x_i}})$ $\leftarrow \mathcal{DEL}.\mathsf{Enc}(x_i, pk, \mathsf{par}, r_{x_i})$ $R_1 = \langle \{\mathsf{dE}_{pk}(x_i)\}_{i=1}^{v}, \pi_1 \rangle$	$\xrightarrow{R_1}$	$r_{y_1}, ..., r_{y_w}, r, r_{\hat{g}}, \alpha \leftarrow\!\!\!\!\!\!\rightharpoonup \mathbb{Z}_q, \hat{g} = g^{\alpha}$, $\mathsf{dE}_{pk}(\hat{g}^r) = (c_{\hat{g}} = g^{r_{\hat{g}}}, d_{\hat{g}} = \hat{g}^r h^{r_{\hat{g}}})$ $\leftarrow \mathcal{DEL}.\mathsf{Enc}(\hat{g}^r, pk, \mathsf{par}, r_{\hat{g}})$ for $1 \le j \le w$, $\mathsf{dE}_{pk}(y_j) = (c_{y_j} = g^{r_{y_j}}, d_{y_j} = y_j h^{r_{y_j}})$ $\leftarrow \mathcal{DEL}.\mathsf{Enc}(y_j, pk, \mathsf{par}, r_{y_j})$, $\mathsf{dE}_{pk}((y_j)^r) = (\hat{c}_{y_j} = (c_{y_j})^r, \hat{d}_{y_j} = (d_{y_j})^r)$
	$\xleftarrow{R_2}$	for $1 \le i \le v$, $\mathsf{dE}_{pk}((x_i)^r) = (\hat{c}_{x_i} = (c_{x_i})^r, \hat{d}_{x_i} = (d_{x_i})^r)$ $R_2 = \big\langle \{\mathsf{dE}_{pk}(y_j), \mathsf{dE}_{pk}((y_j)^r)\}_{j=1}^{w}, \{\mathsf{dE}_{pk}((x_i)^r)\}_{i=1}^{v}, \mathsf{dE}_{pk}(\hat{g}^r), \hat{g}, \pi_2 \big\rangle$
$\alpha_1, ..., \alpha_v, r_1, ..., r_w, z_1, ..., z_w, r', r_{\bar{g}}, \beta \leftarrow\!\!\!\!\!\!\rightharpoonup \mathbb{Z}_q$ $\bar{g} = g^{\beta}$ and $\mathsf{dE}_{pk}(\bar{g}^{r'}) = (c_{\bar{g}} = g^{r_{\bar{g}}}, d_{\bar{g}} = \bar{g}^{r'} h^{r_{\bar{g}}})$ $\leftarrow \mathcal{DEL}.\mathsf{Enc}(\bar{g}^{r'}, pk, \mathsf{par}, r_{\bar{g}})$, for $1 \le i \le v$, $\mathsf{dE}_{pk}((x_i)^{rr'}) = (\bar{c}_{x_i} = (\hat{c}_{x_i})^{r'}, \bar{d}_{x_i} = (\hat{d}_{x_i})^{r'})$, $\mathsf{eE}_{epk_B}((C_{x_i})^{a_1} = (\bar{c}_{x_i} c_{\bar{g}} c_{\hat{g}})^{a_1})$ $\leftarrow \mathcal{EL}.\mathsf{Enc}((C_{x_i})^{a_1}, epk_B, \mathsf{par}, \alpha_i)$ for $1 \le j \le w$, $\mathsf{dE}_{pk}((y_j)^{rr'}) = (\bar{c}_{y_j} = (\hat{c}_{y_j})^{r'}, \bar{d}_{y_j} = (\hat{d}_{y_j})^{r'})$, $\bar{u}_{y_j} = (C_{y_j})^{a_1} g^{r_j} = (\bar{c}_{y_j} c_{\bar{g}} c_{\hat{g}})^{a_1} g^{r_j}$, $\mathsf{vE}_{vpk_{Ar}}(g^{r_j}) \leftarrow \mathcal{VE}.\mathsf{Enc}(g^{r_j}, vpk_{Ar}, \mathsf{gpar}, z_j, L, \mathcal{H})$ $R_3 = \big\langle \{\mathsf{dE}_{pk}((x_i)^{rr'}), \mathsf{eE}_{epk_B}((C_{x_i})^{a_1})\}_{i=1}^{v}, \bar{g}, \{\mathsf{dE}_{pk}((y_j)^{rr'}), \mathsf{vE}_{vpk_{Ar}}(g^{r_j}), \bar{u}_{y_j}\}_{j=1}^{w}, \mathsf{dE}_{pk}(\bar{g}^{r'}), \pi_3 \big\rangle$	$\xrightarrow{R_3}$	for $1 \le i \le v, C_{x_i} = \bar{c}_{x_i} c_{\bar{g}} c_{\hat{g}}$ and $s_{x_i} = (C_{x_i})^{a_2}$ for $1 \le j \le w, C_{y_j} = \bar{c}_{y_j} c_{\bar{g}} c_{\hat{g}}$ and $s_{y_j} = (C_{y_j})^{a_2}$
	$\xleftarrow{R_4}$	$R_4 = \big\langle \{s_{x_i}\}_{i=1}^{v}, \{s_{y_j}\}_{j=1}^{w}, \pi_4 \big\rangle$
for $1 \le i \le v$, $\frac{\bar{d}_{x_i} d_{\bar{g}} d_{\hat{g}}}{(C_{x_i})^{a_1} s_{x_i}} = \bar{g}^{r'} \hat{g}^r (x_i)^{rr'}$ for $1 \le j \le w$, $\frac{\bar{d}_{y_j} d_{\bar{g}} d_{\hat{g}}}{(C_{y_j})^{a_1} s_{y_j}} = \bar{g}^{r'} \hat{g}^r (y_j)^{rr'}$ $X \cap Y = \{x_i \in X \mid \bar{g}^{r'} \hat{g}^r (x_i)^{rr'} \in \{\bar{g}^{r'} \hat{g}^r (y_1)^{rr'}, ..., \bar{g}^{r'} \hat{g}^r (y_w)^{rr'}\}\}$ $R_5 = \big\langle \{(g^{r_j})\}_{j=1}^{w}, \pi_5 \big\rangle$	$\xrightarrow{R_5}$	for $1 \le i \le v$, $(C_{x_i})^{a_1} \leftarrow \mathcal{EL}.\mathsf{Dec}(\mathsf{eE}_{epk_B}((C_{x_i})^{a_1}), esk_B)$ $\frac{\bar{d}_{x_i} d_{\bar{g}} d_{\hat{g}}}{(C_{x_i})^{a_1} s_{x_i}} = \bar{g}^{r'} \hat{g}^r (x_i)^{rr'}$ for $1 \le j \le w$, $\frac{\bar{u}_{y_j}}{g^{r_j}} = (C_{y_j})^{a_1}$, $\frac{\bar{d}_{y_j} d_{\bar{g}} d_{\hat{g}}}{(C_{y_j})^{a_1} s_{y_j}} = \bar{g}^{r'} \hat{g}^r (y_j)^{rr'}$ $X \cap Y = \{y_j \in Y \mid \bar{g}^{r'} \hat{g}^r (y_j)^{rr'} \in \{\bar{g}^{r'} \hat{g}^r (x_1)^{rr'}, ..., \bar{g}^{r'} \hat{g}^r (x_v)^{rr'}\}\}$

Figure 2 : Communication flow of our mPSI

$$= \frac{\bar{g}^{r'}\hat{g}^{r}(x_i)^{rr'}g^{(r_{x_i}rr'+r_{\bar{g}}+r_{\hat{g}})(a_1+a_2)}}{g^{(r_{x_i}rr'+r_{\bar{g}}+r_{\hat{g}})(a_1+a_2)}} = \bar{g}^{r'}\hat{g}^{r}(x_i)^{rr'}, 1 \leq i \leq v,$$

$$\text{and } \frac{\bar{d}_{y_j}d_{\hat{g}}d_{\bar{g}}}{(C_{y_j})^{a_1}s_{y_j}} = \frac{\bar{d}_{y_j}d_{\hat{g}}d_{\bar{g}}}{(\bar{c}_{y_j}c_{\hat{g}}c_{\bar{g}})^{(a_1+a_2)}} = \frac{(d_{y_j})^{rr'}d_{\hat{g}}d_{\bar{g}}}{((c_{y_j})^{rr'}c_{\hat{g}}c_{\bar{g}})^{a_1+a_2}}$$

$$= \frac{\bar{g}^{r'}\hat{g}^{r}(y_j)^{rr'}g^{(r_{y_j}rr'+r_{\bar{g}}+r_{\hat{g}})(a_1+a_2)}}{g^{(r_{y_j}rr'+r_{\bar{g}}+r_{\hat{g}})(a_1+a_2)}} = \bar{g}^{r'}\hat{g}^{r}(y_j)^{rr'}, 1 \leq j \leq w;$$

(ii) sets $X \cap Y = \{x_i \in X | \bar{g}^{r'}\hat{g}^{r}(x_i)^{rr'} \in \{\bar{g}^{r'}\hat{g}^{r}(y_1)^{rr'}, ..., \bar{g}^{r'}\hat{g}^{r}(y_w)^{rr'}\}\}$;

(iii) constructs the proof

$$\pi_5 = \mathsf{PoK}\{(z_1, ..., z_w) | \wedge_{j=1}^{w} (t_{1j} = g^{z_j})(t_{2j} = \widehat{g}^{z_j})(t_{3j} = c^{z_j}g^{r_j})(t_{4j} = a^{z_j}b^{z_j\rho_j})\};$$

(iv) sends $R_5 = \left\langle \{(g^{r_j})\}_{j=1}^{w}, \pi_5 \right\rangle$ to B.

Step 6. On receiving $R_5 = \left\langle \{(g^{r_j})\}_{j=1}^{w}, \pi_5 \right\rangle$ from A, party B verifies the validity of the proof π_5. If the verification of the proof succeeds, then B

(i) for each $i = 1, ..., v$, decrypts $\mathsf{eE}_{epk_B}((C_{x_i})^{a_1})$ received in *Step* 3 using his secret key $esk_B = a_2$ to get $(C_{x_i})^{a_1} \leftarrow \mathcal{EL}.\mathsf{Dec}(\mathsf{eE}_{epk_B}((C_{x_i})^{a_1}), esk_B)$, extracts $\bar{d}_{x_i}, d_{\bar{g}}$ from $\mathsf{dE}_{pk}((x_i)^{rr'}), \mathsf{dE}_{pk}(\bar{g}^{r'})$ respectively in R_3, uses s_{x_i} computed in *Step* 4 and $d_{\hat{g}}$ computed in *Step* 2 to generate

$$\frac{\bar{d}_{x_i}d_{\hat{g}}d_{\bar{g}}}{(C_{x_i})^{a_1}s_{x_i}} = \frac{\bar{d}_{x_i}d_{\hat{g}}d_{\bar{g}}}{(\bar{c}_{x_i}c_{\hat{g}}c_{\bar{g}})^{(a_1+a_2)}} = \frac{(d_{x_i})^{rr'}d_{\hat{g}}d_{\bar{g}}}{((c_{x_i})^{rr'}c_{\hat{g}}c_{\bar{g}})^{a_1+a_2}}$$

$$= \frac{\bar{g}^{r'}\hat{g}^{r}(x_i)^{rr'}g^{(r_{x_i}rr'+r_{\bar{g}}+r_{\hat{g}})(a_1+a_2)}}{g^{(r_{x_i}rr'+r_{\bar{g}}+r_{\hat{g}})(a_1+a_2)}} = \bar{g}^{r'}\hat{g}^{r}(x_i)^{rr'};$$

(ii) for each $j = 1, ..., w$, extracts $\bar{d}_{y_j}, d_{\bar{g}}$ from $\mathsf{dE}_{pk}((y_j)^{rr'}), \mathsf{dE}_{pk}(\bar{g}^{r'})$ from in R_3 respectively, uses $\bar{u}_{y_j}$ obtained from R_3, s_{y_j} computed in *Step* 4 and $d_{\hat{g}}$ computed in *Step* 2 to generate

$$\frac{\bar{u}_{y_j}}{g^{r_j}} = \frac{(C_{y_j})^{a_1} \cdot g^{r_j}}{g^{r_j}} = (C_{y_j})^{a_1},$$

$$\text{and } \frac{\bar{d}_{y_j}d_{\hat{g}}d_{\bar{g}}}{(C_{y_j})^{a_1}s_{y_j}} = \frac{\bar{d}_{y_j}d_{\hat{g}}d_{\bar{g}}}{(\bar{c}_{y_j}c_{\hat{g}}c_{\bar{g}})^{(a_1+a_2)}} = \frac{(d_{y_j})^{rr'}d_{\hat{g}}d_{\bar{g}}}{((c_{y_j})^{rr'}c_{\hat{g}}c_{\bar{g}})^{a_1+a_2}}$$

$$= \frac{\bar{g}^{r'}\hat{g}^{r}(y_j)^{rr'}g^{(r_{y_j}rr'+r_{\bar{g}}+r_{\hat{g}})(a_1+a_2)}}{g^{(r_{y_j}rr'+r_{\bar{g}}+r_{\hat{g}})(a_1+a_2)}} = \bar{g}^{r'}\hat{g}^{r}(y_j)^{rr'};$$

(iii) sets $X \cap Y = \{y_j \in Y | \bar{g}^{r'}\hat{g}^{r}(y_j)^{rr'} \in \{\bar{g}^{r'}\hat{g}^{r}(x_1)^{rr'}, ..., \bar{g}^{r'}\hat{g}^{r}(x_v)^{rr'}\}\}$.

If the verification of π_5 does not succeed or B does not get $R_5 = \left\langle \{(g^{r_j})\}_{j=1}^{w}, \pi_5 \right\rangle$ from A i.e., if A prematurely aborts, then B sends a dispute resolution request to the arbiter Ar.

We now describe the Dispute Resolution Protocol in Figure 3.

The arbiter Ar, on receiving a dispute resolution request from B, interacts with A and B in the following way:

Step 1. Party B sends all the messages sent and received in Step 1-3 of the mPSI protocol to the arbiter Ar. On receiving the messages, Ar verifies the consistency between messages and the label L. If the verification fails or if the transcript ends before the end of *Step* 3 of the mPSI protocol then Ar aborts so that neither party gets any advantage. Otherwise, Ar continue with the following steps.

Step 2. Similar to *Step* 4 of the mPSI protocol, B sends $R_4 = \big\langle \{s_{x_i} = (C_{x_i})^{a_2}\}_{i=1}^{v}, \{s_{y_j} = (C_{y_j})^{a_2}\}_{j=1}^{w}, \pi_4 \big\rangle$ to Ar, where π_4 is same the as π_4 of the mPSI protocol.

Step 3. The arbiter Ar, on receiving $R_4 = \big\langle \{s_{x_i} = (C_{x_i})^{a_2}\}_{i=1}^{v}, \{s_{y_j} = (C_{y_j})^{a_2}\}_{j=1}^{w}, \pi_4 \big\rangle$ from B, verifies the validity of the proof π_4. If the verification does not succeed then Ar aborts, there by neither party gets any advantage. Otherwise, Ar decrypts $\{\mathsf{vE}_{vpk_{Ar}}(g^{r_j})\}_{j=1}^{w}$ and sends $\{g^{r_j}\}_{j=1}^{w}$ to B so that B can compute $X \cap Y$ using the similar technique as described in *Step* 6 of our mPSI protocol. The arbiter Ar also forwards $\big\langle \{s_{x_i}\}_{i=1}^{v}, \{s_{y_j}\}_{j=1}^{w} \big\rangle$ to A who in turns can compute $X \cap Y$ using the similar technique as explained in *Step* 5 of our mPSI protocol.

Figure 3 : Dispute Resolution Protocol of our mPSI

Remark 3.1. *In* Step *3 of our* mPSI *protocol, A encrypts each g^{r_j} to get $\mathsf{vE}_{pk_{Ar}}(g^{r_j})$ for $1 \leq j \leq w$, using the public key pk_{Ar} of Ar and a label $L \in \{0,1\}^*$. Note that the label L used by Ar should be same as the label L used by A. Party A generates label L using the following two inputs –*
(i) a session ID which has been agreed by all parities beforehand,
(ii) the hash of past communication.
As Ar knows the session ID, after receiving all the messages from B in the Step *1 of dispute resolution protocol Ar can compute the label L. Due to the session ID, Ar can verify the identities of A, B and that the protocol execution is within a certain time window. As only B can raise a dispute resolution request to Ar, party A uses the hash of past communication as an input of L to ensure that B cannot get any advantage by modifying messages.*

Remark 3.2. *Note that sometimes the construction may reveal more information than claimed. Let us consider the simplest case where all parties faithfully follow the protocol. Party A holds set $\{1, 2, 4, 8\}$ while party B holds set $\{16, 32, 64, 128\}$. Ideally any secure protocol should reveal $\perp$ to both parties since the intersection is empty. But at the end of the construction, both parties actually learn $\{\bar{g}^{r'}\hat{g}^{r}x_i^{rr'}\}_{i=1}^{v}$ and $\{\bar{g}^{r'}\hat{g}^{r}y_j^{rr'}\}_{j=1}^{w}$. By dividing neighboring elements in these two sets respectively, the random mask $\bar{g}^{r'}\hat{g}^{r}$ cancels out, leaving only 6 elements: $2^{rr'}$. As a result, both party knows that two sets are both geometry sequence with the same scaling factor 2. This kind of attack (or similar attacks) can be prevented by adopting the following steps:*

1. *In* Step *1 of* mPSI *protocol, A has to give a random permutation ϕ to his private set*

$X = \{x_1, ..., x_v\}$ *before encrypting the elements of* X. *Due to this, again in* Step *5 of* mPSI *protocol or in* Step *3 of dispute resolution protocol,* A *has to give the inverse permutation* ϕ^{-1} *to the set* $\{\bar{g}^{r'}\hat{g}^r x_{\phi(i)}^{rr'}\}_{i=1}^v$ *to get* $\{\bar{g}^{r'}\hat{g}^r x_i^{rr'}\}_{i=1}^v$ *just before computing* $X \cap Y$.

2. *In* Step *2 of* mPSI *protocol,* B *has to give a random permutation* ψ *to his private set* $Y = \{y_1, ..., y_w\}$ *before encrypting the elements of* Y. *Due to this, again in* Step *6 of* mPSI *protocol or in* Step *3 of dispute resolution protocol,* B *has to give the inverse permutation* ψ^{-1} *to the set* $\{\bar{g}^{r'}\hat{g}^r y_{\psi(j)}^{rr'}\}_{j=1}^w$ *to get* $\{\bar{g}^{r'}\hat{g}^r y_j^{rr'}\}_{j=1}^w$ *just before computing* $X \cap Y$.

3.2 Security

We consider two cases: (case I) when the adversary corrupts two parties among the three parties and (case II) when the adversary corrupts only one party among the three parties.

Theorem 3.1. *If the encryption schemes* $\mathcal{EL}$, $\mathcal{DEL}$ *and* $\mathcal{VE}$ *are semantically secure, the associated proof protocols are zero knowledge proof, then the protocol* mPSI *presented in section 3 is a secure computation protocol for the functionality* $\mathcal{F}_{\mathsf{mPSI}} : (X, Y) \to (X \cap Y, X \cap Y)$ *in the security model described in section 2.1.*

Proof. Let us consider $\mathcal{C}$ as the real world adversary that breaks the security of our mPSI protocol among three parties A with private input set X, B with private input set Y and Ar. Also let there be an incorruptible trusted party T, parties $\bar{A}$, $\bar{B}$, $\bar{Ar}$ and simulator $\mathcal{SIM}$ in the ideal process. In real world, the global parameter $\mathsf{gpar} = (\mathsf{par}, \widehat{g}, \mathcal{H})$, where $\mathsf{par} = (p, q, g)$ is generated by a trusted party who certifies the public key pk_A, pk_B, pk_{Ar} of A, B, Ar respectively. In contrast, in ideal process simulator $\mathcal{SIM}$ does those things. We denote the joint output of $A, B, Ar, \mathcal{C}$ in the real world as $\mathsf{REAL}_{\mathsf{mPSI},\mathcal{C}}(X, Y)$ and the joint output of $\bar{A}, \bar{B}, \bar{Ar}, \mathcal{SIM}$ in the ideal process as $\mathsf{IDEAL}_{\mathcal{F}_{\mathsf{mPSI}},\mathcal{SIM}}(X, Y)$.

• **Case I (When the adversary $\mathcal{C}$ corrupts two parties).**

1. A **and** Ar **are corrupted.** Let $\mathcal{Z}$ be a distinguisher who controls $\mathcal{C}$, feeds the input of the honest party B, and also sees the output of B. Now we will present a series of games $Game_0, ..., Game_4$ to prove that $\mathcal{Z}$'s view in the real world ($\mathcal{C}$'s view +B's output) and its view in the ideal world ($\mathcal{C}$'s view + $\bar{B}$'s output) are indistinguishable. For each $i = 0, ..., 3$, $Game_{i+1}$ modifies $Game_i$ slightly such that $\mathcal{Z}$'s views in $Game_i$ and $Game_{i+1}$ remain indistinguishable. The probability that $\mathcal{Z}$ distinguishes the view of $Game_i$ from the view of real protocol, is denoted by $Pr[Game_i]$ and S_i is considered as simulator in $Game_i$.
   $Game_0$: This game is same as real world protocol, where the simulator S_0 has full knowledge of B and interacts with $\mathcal{C}$. Hence,

$$Prob[\mathsf{REAL}_{\mathsf{mPSI},\mathcal{C}}(X, Y)] = Prob[Game_0].$$

   $Game_1$: $Game_1$ is same as $Game_0$ except that if the proof π_1 is valid then the simulator S_1 runs the extractor algorithm for π_1 with $\mathcal{C}$ to extract the exponents $\{r_{x_1}, ..., r_{x_v}\}$. The

simulator S_1 then extracts $x_i = \frac{d_{x_i}}{h^{r_{x_i}}}$ by extracting $d_{x_i} = x_i h^{r_{x_i}}$ from $\mathsf{dE}_{pk}(x_i)$ in R_1, h from $pk = epk_A \cdot epk_B$ and using the exponent r_{x_i} for $1 \leq i \leq v$. In this way S_1 extracts the private input set $X = \{x_1, ..., x_v\}$ of A. $\mathcal{Z}$'s views in $Game_0$ and $Game_1$ are indistinguishable because of simulation soundness of the proof π_1. Therefore,

$$|Prob[Game_1] - Prob[Game_0]| \leq \epsilon_1(\kappa), \text{ where } \epsilon_1(\kappa) \text{ is a negligible function.}$$

$Game_2$: Note that in this game the simulator S_2 has the knowledge of extracted set $X = \{x_1, ..., x_v\}$, input set $Y = \{y_1, ..., y_w\}$ and secret key $esk_B = a_2$ of B. $Game_2$ is same as $Game_1$ except that

(a) if the verification of the proof π_5 succeeds then S_3 outputs $X \cap Y$ as the final output of B making use of the extracted X,

(b) if the verification of the proof π_5 does not succeed or $\mathcal{C}$ aborts prematurely in mPSI protocol then the following cases arise:

$\diamond$ if $\mathcal{C}$ sends $\{g_1, ..., g_w\} \subset \mathbb{G}$ to S_3 in dispute resolution protocol then S_3 does the following:
– for each $i = 1, ..., v$, decrypts $\mathsf{eE}_{epk_B}((C_{x_i})^{a_1})$ using $esk_B = a_2$ to get $(C_{x_i})^{a_1} \leftarrow \mathcal{EL}.\mathsf{Dec}(\mathsf{eE}_{epk_B}((C_{x_i})^{a_1}), esk_B)$, extracts $\bar{d}_{x_i}, \bar{c}_{x_i}$ from $\mathsf{dE}_{pk}((x_i)^{rr'})$ and $c_{\bar{g}}, d_{\bar{g}}$ from $\mathsf{dE}_{pk}(\bar{g}^{r'})$ in R_3 and uses $c_{\hat{g}}, d_{\hat{g}}$ computed in ***Step*** 2 to compute $\frac{\bar{d}_{x_i} d_{\bar{g}} d_{\hat{g}}}{(C_{x_i})^{a_1} (\bar{c}_{x_i} c_{\bar{g}} c_{\hat{g}})^{a_2}} = \bar{g}^{r'} \hat{g}^r x_i^{rr'}$;
– for each $j = 1, ..., w,$, computes $\frac{\bar{d}_{y_j} d_{\bar{g}} d_{\hat{g}}}{\frac{\bar{u}_{y_j}}{g_j} (\bar{c}_{y_j} c_{\bar{g}} c_{\hat{g}})^{a_2}} = \hat{y}_j$ by extracting $\bar{d}_{y_j}, \bar{c}_{y_j}$ from $\mathsf{dE}_{pk}((y_j)^{rr'})$ and $c_{\bar{g}}, d_{\bar{g}}$ from $\mathsf{dE}_{pk}(\bar{g}^{r'})$ in R_3, using $\bar{u}_{y_j}$ obtained from R_3 and $c_{\hat{g}}, d_{\hat{g}}$ computed in ***Step*** 2;
– outputs $\{y_j \in Y | \hat{y}_j \in \{\bar{g}^{r'} \hat{g}^r x_1^{rr'}, ..., \bar{g}^{r'} \hat{g}^r x_v^{rr'}\}\}$ as the final output of B.
$\diamond$ if $\mathcal{C}$ aborts in dispute resolution protocol then S_3 outputs $\perp$ as the final output of B.

By the simulation soundness property of the proof π_5, $\mathcal{Z}$'s views in $Game_2$ and $Game_3$ are indistinguishable. Hence,

$$|Prob[Game_2] - Prob[Game_1]| \leq \epsilon_2(\kappa), \text{ where } \epsilon_2(\kappa) \text{ is a negligible function.}$$

$Game_3$: $Game_3$ is same as $Game_2$ except that S_3 does the following after extracting $X = \{x_1, ..., x_v\}$:

(a) computes $X \cap Y$,

(b) constructs a set $\overline{Y} = \{\bar{y}_1, ..., \bar{y}_w\}$ by including all the elements of $X \cap Y$ together with $w - |X \cap Y|$ many random elements chosen from $\mathbb{G}$,

(c) chooses $r, \alpha \leftarrow\!\!\!\!- \mathbb{Z}_q$,

(d) computes $\hat{g} = g^\alpha$ and $\left\langle \{\mathsf{dE}_{pk}(\bar{y}_j), \mathsf{dE}_{pk}((\bar{y}_j)^r)\}_{j=1}^w, \{\mathsf{dE}_{pk}((x_i)^r)\}_{i=1}^v, \mathsf{dE}_{pk}(\hat{g}^r) \right\rangle$,

(e) sends the tuple $\left\langle \{\mathsf{dE}_{pk}(\bar{y}_j), \mathsf{dE}_{pk}((\bar{y}_j)^r)\}_{j=1}^w, \{(\mathsf{dE}_{pk}((x_i)^r)\}_{i=1}^v, \mathsf{dE}_{pk}(\hat{g}^r), \hat{g} \right\rangle$ as $\left\langle \{\mathsf{dE}_{pk}(y_j), \mathsf{dE}_{pk}((y_j)^r)\}_{j=1}^w, \{\mathsf{dE}_{pk}((x_i)^r)\}_{i=1}^v, \mathsf{dE}_{pk}(\hat{g}^r), \hat{g} \right\rangle$ to $\mathcal{C}$ and simulates π_2.

As the encryption scheme $\mathcal{DEL}$ is semantically secure the tuple $\langle \{\mathsf{dE}_{pk}(y_j), \mathsf{dE}_{pk}((y_j)^r)\}_{j=1}^{w}, \{\mathsf{dE}_{pk}((x_i)^r)\}_{i=1}^{v}, \mathsf{dE}_{pk}(\hat{g}^r), \hat{g}\rangle$ is identically distributed in $Game_3$ and $Game_2$. The zero-knowledge (simulatability) of π_2 and indistinguishability of the tuple $\langle \{\mathsf{dE}_{pk}(y_j), \mathsf{dE}_{pk}((y_j)^r)\}_{j=1}^{w}, \{\mathsf{dE}_{pk}((x_i)^r)\}_{i=1}^{v}, \mathsf{dE}_{pk}(\hat{g}^r), \hat{g}\rangle$ make the views of $\mathcal{Z}$'s in $Game_2$ and $Game_3$ indistinguishable. Therefore, there exists a negligible function $\epsilon_3(\kappa)$ such that

$$|Prob[Game_3] - Prob[Game_2]| \leq \epsilon_3(\kappa).$$

$Game_4$: This game is same as $Game_3$ except that during the setup phase S_4 chooses $a_2 \leftarrow\!\!\!\!\!\!\rightharpoonup \mathbb{Z}_q$ and in *Step* 4 simulates π_4, instead of proving it. By the zero-knowledge (simulatability) of π_4 the views of $\mathcal{Z}$'s in $Game_3$ and $Game_4$ are indistinguishable. Consequently,

$$|Prob[Game_4] - Prob[Game_3]| \leq \epsilon_4(\kappa), \text{ where } \epsilon_4(\kappa) \text{ is a negligible function.}$$

Let us construct the ideal world adversary $\mathcal{SIM}$ that uses $\mathcal{C}$ as subroutine, simulates the honest party B and controls $\bar{A}, \bar{A}r$ and incorporates all steps from $Game_4$.

(i) First $\mathcal{SIM}$ plays the role of trusted party by generating the global parameter $\mathsf{gpar} = (\mathsf{par}, \widehat{g}, \mathcal{H})$, where $\mathsf{par} = (p, q, g)$. $\mathcal{SIM}$ then plays the role of honest party B by choosing $\bar{a}_2 \leftarrow\!\!\!\!\!\!\rightharpoonup \mathbb{Z}_q$ and publishing $g^{\bar{a}_2}$ as the public key $epk_B = y_B$. $\mathcal{SIM}$ also acts as certifying authority to obtain respective public keys epk_A, vpk_{Ar} of A, Ar. $\mathcal{SIM}$ then invokes $\mathcal{C}$.

(ii) On receiving $R_1 = \langle \{\mathsf{dE}_{pk}(x_i)\}_{i=1}^{v}, \pi_1\rangle$ from $\mathcal{C}$, $\mathcal{SIM}$ verifies the proof π_1. If the verification does not succeed, then $\mathcal{SIM}$ instructs $\bar{A}$ to send $\perp$ to T and terminates the execution. Otherwise, $\mathcal{SIM}$ runs the extractor algorithm for π_1 with $\mathcal{C}$ to extract $\{r_{x_1}, ..., r_{x_v}\}$. Utilizing $\{r_{x_1}, ..., r_{x_v}\}$, $\mathcal{SIM}$ extracts the input set $X = \{x_1, ..., x_v\}$ by extracting $\{d_{x_i} = x_i h^{r_{x_i}}\}_{i=1}^{v}$ from $\{\mathsf{dE}_{pk}(x_i)\}_{i=1}^{v}$ in R_1 and h from $pk = epk_A \cdot epk_B$. $\mathcal{SIM}$ then instructs $\bar{A}$ to send X to T, $\bar{A}r$ to send $b_A = \circ$ to T and receives $X \cap Y$ from T.

(iii) $\mathcal{SIM}$ constructs a set $\overline{Y} = \{\bar{y}_1, ..., \bar{y}_w\}$ by including all the elements of $X \cap Y$ together with $w - |X \cap Y|$ many random elements chosen from $\mathbb{G}$. $\mathcal{SIM}$ then chooses $r, \alpha \leftarrow\!\!\!\!\!\!\rightharpoonup \mathbb{Z}_q$, computes $\hat{g} = g^{\alpha}$ and $\langle \{\mathsf{dE}_{pk}(\bar{y}_j), \mathsf{dE}_{pk}((\bar{y}_j)^r)\}_{j=1}^{w}, \{\mathsf{dE}_{pk}((x_i)^r)\}_{i=1}^{v}, \mathsf{dE}_{pk}(\hat{g}^r)\rangle$, sends the tuple $\langle \{\mathsf{dE}_{pk}(\bar{y}_j), \mathsf{dE}_{pk}((\bar{y}_j)^r)\}_{j=1}^{w}, \{(\mathsf{dE}_{pk}((x_i)^r)\}_{i=1}^{v}, \mathsf{dE}_{pk}(\hat{g}^r), \hat{g}\rangle$ as $\langle \{\mathsf{dE}_{pk}(y_j), \mathsf{dE}_{pk}((y_j)^r)\}_{j=1}^{w}, \{\mathsf{dE}_{pk}((x_i)^r)\}_{i=1}^{v}, \mathsf{dE}_{pk}(\hat{g}^r), \hat{g}\rangle$ to $\mathcal{C}$ and simulates π_2.

(iv) On receiving $R_3 = \langle \{\mathsf{dE}_{pk}((x_i)^{rr'}), \mathsf{eE}_{epk_B}((C_{x_i})^{a_1})\}_{i=1}^{v}, \{\mathsf{dE}_{pk}((\bar{y}_j)^{rr'}), \mathsf{vE}_{vpk_{Ar}}(g^{r_j}), \bar{u}_{\bar{y}_j}\}_{j=1}^{w}, \mathsf{dE}_{pk}(\bar{g}^{r'}), \bar{g}, \pi_3\rangle$ from $\mathcal{C}$, $\mathcal{SIM}$ verifies the validity of the proof π_3. If the verification fails then $\mathcal{SIM}$ instructs $\bar{A}$ to send $\perp$ to T and terminates the execution. Otherwise, $\mathcal{SIM}$ computes $\{s_{x_i} = (C_{x_i})^{\bar{a}_2}\}_{i=1}^{v}, \{s_{y_j} = (C_{\bar{y}_j})^{\bar{a}_2}\}_{j=1}^{w}$, sends it to $\mathcal{C}$ and simulates the proof π_4. $\mathcal{SIM}$ then executes following steps according to $\mathcal{C}$'s reply.

(v) If $\mathcal{C}$ instructs A to send $\{g_1, ..., g_w\} \subset \mathbb{G}$, then $\mathcal{SIM}$ verifies the validity of the proof π_5. If the verification succeeds then $\mathcal{SIM}$ instructs $\bar{A}r$ to send $b_B = \circ$. If verification fails or $\mathcal{C}$ instructs A to abort in mPSI protocol then the following cases

arise:

$\diamond$ if $\mathcal{C}$ instructs Ar to send $\{g_1, ..., g_w\} \subset \mathbb{G}$ in dispute resolution protocol, then $\mathcal{SIM}$ does the following:

– for each $i = 1, ..., v$, decrypts $\mathsf{eE}_{epk_B}((C_{x_i})^{a_1})$ using $esk_B = a_2$ to get $(C_{x_i})^{a_1} \leftarrow \mathcal{EL}.\mathsf{Dec}(\mathsf{eE}_{epk_B}((C_{x_i})^{a_1}), esk_B)$, extracts $\bar{d}_{x_i}, \bar{c}_{x_i}$ from $\mathsf{dE}_{pk}((x_i)^{rr'})$ and $c_{\bar{g}}, d_{\bar{g}}$ from $\mathsf{dE}_{pk}(\bar{g}^{r'})$ in R_3 and uses $c_{\hat{g}}, d_{\hat{g}}$ computed in ***Step*** 2 to compute $\frac{\bar{d}_{x_i} d_{\bar{g}} d_{\hat{g}}}{(C_{x_i})^{a_1} (\bar{c}_{x_i} c_{\bar{g}} c_{\hat{g}})^{\bar{a}_2}} = \bar{g}^{r'} \hat{g}^r x_i^{rr'}$;

– for each $j = 1, ..., w,$, computes $\frac{\bar{d}_{\bar{y}_j} d_{\bar{g}} d_{\hat{g}}}{\frac{\bar{u}_{\bar{y}_j}}{g_j} (\bar{c}_{\bar{y}_j} c_{\bar{g}} c_{\hat{g}})^{\bar{a}_2}} = \tilde{y}_j$ by extracting $\bar{d}_{\bar{y}_j}, \bar{c}_{\bar{y}_j}$ from $\mathsf{dE}_{pk}((\bar{y}_j)^{rr'})$ and $c_{\bar{g}}, d_{\bar{g}}$ from $\mathsf{dE}_{pk}(\bar{g}^{r'})$ in R_3, using $\bar{u}_{\bar{y}_j}$ obtained from R_3 and $c_{\hat{g}}, d_{\hat{g}}$ computed in ***Step*** 2;

– instructs $\bar{A}r$ to send $b_B = \{\bar{y}_j \in \overline{Y} | \tilde{y}_j \in \{\bar{g}^{r'} \hat{g}^r x_1^{rr'}, ..., \bar{g}^{r'} \hat{g}^r x_v^{rr'}\}\}$ to T, outputs whatever $\mathcal{C}$ outputs and terminates.

$\diamond$ if $\mathcal{C}$ instructs Ar to abort in dispute resolution protocol $\mathcal{SIM}$ instructs $\bar{A}r$ to send $b_B = \perp$ to T. Then $\mathcal{SIM}$ outputs whatever $\mathcal{C}$ outputs and terminates.

(vi) If $\mathcal{C}$ instructs both A and Ar to abort, then $\mathcal{SIM}$ instructs $\bar{A}r$ to send $b_B = \perp$ to T, outputs whatever $\mathcal{C}$ outputs and terminates.

Thus the ideal world adversary $\mathcal{SIM}$ provides $\mathcal{C}$ the same simulation as the simulator S_4 in $Game_4$. Hence $Prob[\mathsf{IDEAL}_{\mathcal{F}_{\mathsf{mPSI}},\mathcal{SIM}}(X, Y)] = Prob[Game_4]$ and

$$
\begin{aligned}
&|Prob[\mathsf{IDEAL}_{\mathcal{F}_{\mathsf{mPSI}},\mathcal{SIM}}(X, Y)] - Prob[\mathsf{REAL}_{\mathsf{mPSI},\mathcal{C}}(X, Y)]| \\
&= |Prob[Game_4] - Prob[Game_0]| \leq \Sigma_{i=1}^{4} |Prob[Game_i] - Prob[Game_{i-1}]| \\
&\leq \Sigma_{i=1}^{4} \epsilon_i(\kappa) = \rho(\kappa), \text{ where } \rho(\kappa) \text{ is a negligible function.}
\end{aligned}
$$

Therefore we have

$$\mathsf{IDEAL}_{\mathcal{F}_{\mathsf{mPSI}},\mathcal{SIM}}(X, Y) \equiv^c \mathsf{REAL}_{\mathsf{mPSI},\mathcal{C}}(X, Y).$$

2. B **and** Ar **are corrupted.** Let us consider $\mathcal{Z}$ as a distinguisher who controls $\mathcal{C}$, feeds the input of the honest party A, and also sees the output of B. Now we argue that $\mathcal{Z}$'s view in the real world ($\mathcal{C}$'s view +A's output) and its view in the ideal world ($\mathcal{C}$'s view + $\bar{A}$'s output) are indistinguishable. To prove that a series of games $Game_0, ..., Game_5$ is presented, where each $Game_{i+1}$ modifies $Game_i$ slightly such that $\mathcal{Z}$'s views in $Game_i$ and $Game_{i+1}$ remain indistinguishable, for $i = 0, .., 4$. Let us denote the probability that $\mathcal{Z}$ distinguishes the view of $Game_i$ from the view of real protocol by $Pr[Game_i]$. We consider S_i as simulator in $Game_i$.

$Game_0$: This game is same as real world protocol, where the simulator S_0 has full knowledge of A and interacts with $\mathcal{C}$. Hence,

$$Prob[\mathsf{REAL}_{\mathsf{mPSI},\mathcal{C}}(X, Y)] = Prob[Game_0].$$

$Game_1$: This game is same as $Game_0$ except that S_1 simulates π_1, instead of proving it. $\mathcal{Z}$'s views in $Game_0$ and $Game_1$ are indistinguishable because of zero-knowledge

(simulatability) of the proof π_1. Therefore, there exists a negligible function $\epsilon_1(\kappa)$ such that

$$|Prob[Game_1] - Prob[Game_0]| \leq \epsilon_1(\kappa).$$

$Game_2$: $Game_1$ is same as $Game_2$ except that if the verification of the proof π_2 succeeds then the simulator S_2 runs the extractor algorithm for π_2 with $\mathcal{C}$ to extract the exponents r and $\{r_{y_1}, ..., r_{y_w}\}$. The simulator S_2 then extracts $y_j = \frac{d_{y_j}}{h^{r_{y_j}}}$ by extracting $d_{y_j} = y_j h^{r_{y_j}}$ from $\mathsf{dE}_{pk}(y_j)$ in R_2, h from $pk = epk_A \cdot epk_B$ and using the exponent r_{y_j} for $1 \leq j \leq w$. In this way S_2 extracts the private input set $Y = \{y_1, ..., y_w\}$ of B. The simulation soundness of the proof π_2 makes $\mathcal{Z}$'s views in $Game_1$ and $Game_2$ indistinguishable. Consequently,

$$|Prob[Game_2] - Prob[Game_1]| \leq \epsilon_2(\kappa), \text{ where } \epsilon_2(\kappa) \text{ is a negligible function.}$$

$Game_3$: Note that in this game the simulator S_3 has the knowledge of input set $X = \{x_1, ..., x_v\}$, secret key $esk_A = a_1$ of A and extracted set $Y = \{y_1, ..., y_w\}$ of B. This game is same as $Game_2$ except that

(a) if the verification of the proof π_4 succeeds then S_3 outputs $X \cap Y$ as the final output of A making use of the extracted set Y,

(b) if the verification of the proof π_4 does not succeed or $\mathcal{C}$ aborts in mPSI protocol then the following cases arise:

$\diamond$ if $\mathcal{C}$ sends $\left\langle \{s_{x_i}\}_{i=1}^{v}, \{s_{y_j}\}_{j=1}^{w} \right\rangle$ to S_3 in dispute resolution protocol then S_3 does the following:

– for each $i = 1, ..., v$, computes $\frac{\bar{d}_{x_i} d_{\bar{g}} d_{\hat{g}}}{(C_{x_i})^{a_1} s_{x_i}} = \hat{x}_i$ using $esk_A = a_1$;

– for each $j = 1, ..., w$, computes $\frac{\bar{d}_{y_j} d_{\bar{g}} d_{\hat{g}}}{(C_{y_j})^{a_1} s_{y_j}} = \hat{y}_j$ using $esk_A = a_1$;

– outputs $\{x_i \in X | \hat{x}_i \in \{\hat{y}_1, ..., \hat{y}_w\}\}$ as the final output of A.

$\diamond$ if $\mathcal{C}$ aborts in dispute resolution protocol then S_3 outputs $\perp$ as the final output of A.

By the simulation soundness property of the proof π_4, $\mathcal{Z}$'s views in $Game_2$ and $Game_3$ are indistinguishable. Therefore, there exists a negligible function $\epsilon_3(\kappa)$ such that

$$|Prob[Game_3] - Prob[Game_2]| \leq \epsilon_3(\kappa).$$

$Game_4$: $Game_4$ is same as $Game_3$ except that S_4 does the following after extracting $Y = \{y_1, ...y_w\}, r$:

(a) computes $X \cap Y$,

(b) constructs a set $\overline{X} = \{\bar{x}_1, ..., \bar{x}_v\}$ by including all the elements of $X \cap Y$ together with $v - |X \cap Y|$ many random elements chosen from $\mathbb{G}$.

(c) chooses $r', r_1, ..., r_w, \beta \leftarrow\!\!\!\!\rightarrow \mathbb{Z}_q$,

(d) computes $\bar{g} = g^{\beta}, \left\langle \{\mathsf{dE}_{pk}((\bar{x}_i)^{rr'}) = (\bar{c}_{\bar{x}_i}, \bar{d}_{\bar{x}_i})\}_{i=1}^{v}, \{\mathsf{dE}_{pk}((y_j)^{rr'}) = (\bar{c}_{y_j}, \bar{d}_{y_j})\}_{j=1}^{w}, \mathsf{dE}_{pk}((\bar{g})^{r'}) = (c_{\bar{g}}, d_{\bar{g}}), \{\mathsf{eE}_{epk_B}((C_{\bar{x}_i})^{a_1})\}_{i=1}^{v} \right\rangle$, where $C_{\bar{x}_i} = \bar{c}_{\bar{x}_i} c_{\bar{g}} c_{\hat{g}}$,

(e) computes $\left\langle \{\bar{u}_{y_j} = (C_{y_j})^{a_1} \cdot g^{r_j}\}_{j=1}^{w}, \{\mathsf{vE}_{vpk_{Ar}}(g^{r_j})\}_{j=1}^{w} \right\rangle$, where $C_{y_j} = \bar{c}_{y_j} c_{\bar{g}} c_{\hat{g}}$,

(f) sends $\big\langle \{\mathsf{dE}_{pk}((\bar{x}_i)^{rr'}), \mathsf{eE}_{epk_B}((C_{\bar{x}_i})^{a_1})\}_{i=1}^{v}, \{\mathsf{dE}_{pk}((y_j)^{rr'}), \mathsf{vE}_{vpk_{Ar}}(g^{r_j}), \bar{u}_{y_j}\}_{j=1}^{w}, \mathsf{dE}_{pk}((\bar{g})^{r'}), \bar{g}\big\rangle$ as $\big\langle \{\mathsf{dE}_{pk}((x_i)^{rr'}), \mathsf{eE}_{epk_B}((C_{x_i})^{a_1})\}_{i=1}^{v}, \{\mathsf{dE}_{pk}((y_j)^{rr'}), \mathsf{vE}_{vpk_{Ar}}(g^{r_j}), \bar{u}_{y_j}\}_{j=1}^{w}, \mathsf{dE}_{pk}((\bar{g})^{r'}), \bar{g}\big\rangle$ to $\mathcal{C}$ and simulates the proofs π_3.

As the associated encryption schemes $\mathcal{DEL}$ and $\mathcal{EL}$ are semantically secure the tuple $\big\langle \{\mathsf{dE}_{pk}(x_i)^{rr'}), \mathsf{eE}_{epk_B}((C_{x_i})^{a_1})\}_{i=1}^{v}, \mathsf{dE}_{pk}((\bar{g})^{r'})\big\rangle$ is identically distributed in $Game_4$ and $Game_3$. Indistinguishability of $\big\langle \{\mathsf{dE}_{pk}((x_i)^{rr'}), \mathsf{eE}_{epk_B}((C_{x_i})^{a_1})\}_{i=1}^{v}, \mathsf{dE}_{pk}((\bar{g})^{r'})\big\rangle$ and the zero-knowledge (simulatability) of π_3 makes the views of $\mathcal{Z}$'s in $Game_3$ and $Game_4$ indistinguishable. Hence,

$$|Prob[Game_4] - Prob[Game_3]| \leq \epsilon_4(\kappa), \text{ where } \epsilon_4(\kappa) \text{ is a negligible function.}$$

$Game_5$: This game is same as $Game_4$ except that during the setup phase S_5 chooses $a_1 \leftarrow\!\!\!\!\circ\ \mathbb{Z}_q$ and in *Step* 5 simulates π_5, instead of proving it. By the zero-knowledge (simulatability) of π_5 the views of $\mathcal{Z}$'s in $Game_4$ and $Game_5$ are indistinguishable. Consequently, there exists a negligible function $\epsilon_5(\kappa)$ such that

$$|Prob[Game_5] - Prob[Game_4]| \leq \epsilon_5(\kappa).$$

Let us construct the ideal world adversary SIM that uses $\mathcal{C}$ as subroutine, simulates the honest party A and controls $\bar{B}, \bar{A}r$ and incorporates all steps from $Game_5$.

(i) $\mathcal{SIM}$ first plays the role of trusted party by generating the global parameter $\mathsf{gpar} = (\mathsf{par}, \widehat{g}, \mathcal{H})$, where $\mathsf{par} = (p, q, g)$. $\mathcal{SIM}$ then plays the role of honest party A by choosing $\bar{a}_1 \leftarrow\!\!\!\!\circ\ \mathbb{Z}_q$ and publishing $g^{\bar{a}_1}$ as the public key $epk_A = y_A$. $\mathcal{SIM}$ also acts as certifying authority to obtain public keys epk_B, vpk_{Ar} of B, Ar. $\mathcal{SIM}$ then invokes $\mathcal{C}$.

(ii) $\mathcal{SIM}$ chooses $\breve{x}_1, ..., \breve{x}_v$ randomly from $\mathbb{G}$ and sends $\{\mathsf{dE}_{pk}(\breve{x}_i)\}_{i=1}^{v}$ as $\{\mathsf{dE}_{pk}(x_i)\}_{i=1}^{v}$ to $\mathcal{C}$ and simulates the proof π_1.

(iii) On receiving $R_2 = \big\langle \{\mathsf{dE}_{pk}(y_j), \mathsf{dE}_{pk}((y_j)^r)\}_{j=1}^{w}, \{\mathsf{dE}_{pk}((x_i)^r)\}_{i=1}^{v}, \mathsf{dE}_{pk}((\hat{g})^r), \hat{g}, \pi_2\big\rangle$ from $\mathcal{C}$, $\mathcal{SIM}$ verifies the proof π_2. If the verification does not succeed, then $\mathcal{SIM}$ instructs $\bar{B}$ to send $\perp$ to T and terminates the execution. Otherwise, $\mathcal{SIM}$ runs the extractor algorithm for π_2 with $\mathcal{C}$ to extract the exponents r and $\{r_{y_1}, ..., r_{y_w}\}$. Utilizing $\{r_{y_1}, ..., r_{y_w}\}$, $\mathcal{SIM}$ extracts $Y = \{y_1, ..., y_w\}$ by extracting $\{d_{y_j} = y_j h^{r_{y_j}}\}_{j=1}^{w}$ from $\{\mathsf{dE}_{pk}(y_j)\}_{j=1}^{w}$ in R_2, h from $pk = epk_A \cdot epk_B$. $\mathcal{SIM}$ then instructs $\bar{B}$ to send Y to T, $\bar{A}r$ to send $b_B = \circ$ to T and receives $X \cap Y$ from T.

(iv) $\mathcal{SIM}$ constructs a set $\overline{X} = \{\bar{x}_1, ..., \bar{x}_v\}$ by including all the elements of $X \cap Y$ together with $v - |X \cap Y|$ many random elements chosen from $\mathbb{G}$. $\mathcal{SIM}$ then does the following:

– chooses $r', r_1, ..., r_w, \beta \leftarrow\!\!\!\!\circ\ \mathbb{Z}_q$;
– computes $\bar{g} = g^{\beta}$, $\big\langle \{\mathsf{dE}_{pk}((\bar{x}_i)^{rr'}) = (\bar{c}_{\bar{x}_i}, \bar{d}_{\bar{x}_i})\}_{i=1}^{v}, \{\mathsf{dE}_{pk}((y_j)^{rr'}) = (\bar{c}_{y_j}, \bar{d}_{y_j})\}_{j=1}^{w}, \mathsf{dE}_{pk}((\bar{g})^{r'}) = (c_{\bar{g}}, d_{\bar{g}}), \{\mathsf{eE}_{epk_B}((C_{\bar{x}_i})^{a_1})\}_{i=1}^{v}\big\rangle$;
– computes $\big\langle \{\bar{u}_{y_j} = (C_{y_j})^{a_1} \cdot g^{r_j}\}_{j=1}^{w}, \{\mathsf{vE}_{vpk_{Ar}}(g^{r_j})\}_{j=1}^{w}\big\rangle$;
– sends $\big\langle \{\mathsf{dE}_{pk}((\bar{x}_i)^{rr'}), \mathsf{eE}_{epk_B}((C_{\bar{x}_i})^{a_1})\}_{i=1}^{v}, \{\mathsf{dE}_{pk}((y_j)^{rr'}), \mathsf{vE}_{vpk_{Ar}}(g^{r_j}), \bar{u}_{y_j}\}_{j=1}^{w},$

$\mathsf{dE}_{pk}((\bar{g})^{r'}), \bar{g}\big\rangle$ as $\big\langle\{\mathsf{dE}_{pk}((x_i)^{rr'}), \mathsf{eE}_{epk_B}((C_{x_i})^{a_1})\}_{i=1}^{v}, \{\mathsf{dE}_{pk}((y_j)^{rr'}), \mathsf{vE}_{vpk_{Ar}}(g^{r_j}), \bar{u}_{y_j}\}_{j=1}^{w}, \mathsf{dE}_{pk}((\bar{g})^{r'}), \bar{g}\big\rangle$ to $\mathcal{C}$ and simulates the proofs π_3.

$\mathcal{SIM}$ executes following steps according to $\mathcal{C}$'s reply.

(v) If $\mathcal{C}$ instructs both B and Ar to abort, then $\mathcal{SIM}$ instructs $\bar{Ar}$ to send $b_A = \perp$ to T. Then outputs whatever $\mathcal{C}$ outputs and terminates.

(vi) If $\mathcal{C}$ instructs B to send $\big\langle\{s_{x_i}\}_{i=1}^{v}, \{s_{y_j}\}_{j=1}^{w}\big\rangle$, then $\mathcal{SIM}$ checks the validity of the proof π_4. If the verification succeeds then $\mathcal{SIM}$ instructs $\bar{Ar}$ to send $b_A = \circ$ to T and sends $\{g^{r_j}\}_{j=1}^{w}$ to $\mathcal{C}$ and simulates the proof π_5. If verification fails or $\mathcal{C}$ instructs B to abort in mPSI protocol then the following cases arise:

$\diamond$ if $\mathcal{C}$ instructs Ar to send $\big\langle\{s_{x_i}\}_{i=1}^{v}, \{s_{y_j}\}_{j=1}^{w}\big\rangle$ in dispute resolution protocol then $\mathcal{SIM}$ does the following:

– for each $i = 1, ..., v$, computes $\frac{\bar{d}_{\bar{x}_i} d_{\bar{g}} d_{\hat{g}}}{(C_{\bar{x}_i})^{\bar{a}_1 s_{x_i}}} = \tilde{x}_i$;

– for each $j = 1, ..., w$, computes $\frac{\bar{d}_{y_j} d_{\bar{g}} d_{\hat{g}}}{(C_{y_j})^{\bar{a}_1 s_{y_j}}} = \tilde{y}_j$;

– instructs $\bar{Ar}$ to send $b_A = \{\bar{x}_i \in \overline{X} | \tilde{x}_i \in \{\tilde{y}_1, ..., \tilde{y}_w\}\}$ to T. $\mathcal{SIM}$ then outputs whatever $\mathcal{C}$ outputs and terminates.

$\diamond$ if $\mathcal{C}$ instructs Ar to abort in dispute resolution protocol then $\mathcal{SIM}$ instructs $\bar{Ar}$ to send $b_A = \perp$ to T. $\mathcal{SIM}$ then outputs whatever $\mathcal{C}$ outputs and terminates.

Therefore, the ideal world adversary $\mathcal{SIM}$ provides $\mathcal{C}$ the same simulation as the simulator S_5 as in $Game_5$. Hence $Prob[\mathsf{IDEAL}_{\mathcal{F}_{\mathsf{mPSI}},\mathcal{SIM}}(X, Y)] = Prob[Game_5]$ and

$$
\begin{aligned}
&|Prob[\mathsf{IDEAL}_{\mathcal{F}_{\mathsf{mPSI}},\mathcal{SIM}}(X, Y)] - Prob[\mathsf{REAL}_{\mathsf{mPSI},\mathcal{C}}(X, Y)]| \\
&= |Prob[Game_5] - Prob[Game_0]| \leq \Sigma_{i=1}^{5} |Prob[Game_i] - Prob[Game_{i-1}]| \\
&\leq \Sigma_{i=1}^{5} \epsilon_i(\kappa) = \rho(\kappa), \text{ where } \rho(\kappa) \text{ is a negligible function.}
\end{aligned}
$$

Thus we have

$$\mathsf{IDEAL}_{\mathcal{F}_{\mathsf{mPSI}},\mathcal{SIM}}(X, Y) \equiv^{c} \mathsf{REAL}_{\mathsf{mPSI},\mathcal{C}}(X, Y).$$

3. A **and** B **are corrupted.** This case is trivial as $\mathcal{C}$ has full knowledge of X and Y and the encryption scheme used by Ar is semantically secure. Therefore a simulator can always be constructed.

• Case II (When the adversary $\mathcal{C}$ corrupts only one party).

If only Ar is corrupted then Ar is not involved in the protocol as A and B are honest. Thus it is trivial to construct a simulator in this case. If only A or B is corrupted then the simulator can be constructed as steps (i)-(iv) of the case when A and Ar are corrupted or steps (i)-(iv) of the case when B and Ar are corrupted. The only change is that $\bar{Ar}$ is honest and always sends $\circ$ to T in these cases.

4 The mPSI-CA

4.1 Construction

Similar to the mPSI, our mPSI-CA also consists of a $\mathbf{Setup}$ algorithm, an **mPSI-CA Protocol** and a **Dispute Resolution Protocol**.

Setup(1^κ) : Similar to the Setup algorithm of the mPSI.

mPSI-CA Protocol: Our mPSI-CA protocol is also an interactive protocol between parties A and B consisting 5 rounds. In each round, a proof is generated and sent by one party, which is then verified in the subsequent round by the other party using a similar way described in section 2.4. Two random permutations ϕ and ψ are to be used by B and A respectively. Let the parties A, B have private input sets $X = \{x_1, ..., x_v\}, Y = \{y_1, ..., y_w\}$ respectively and $(\mathsf{gpar}, epk_A, epk_B, pk = epk_A \cdot epk_B = h)$ be their common input. Then the parties A and B interacts to get the cardinality $|X \cap Y|$ of $X \cap Y$ and a high level intuitive explanation of the interaction is represented by Figure 4. The interaction between A and B is as follows:

Step 1. Party A proceeds as follows:

(i) chooses $r_{x_1}, ..., r_{x_v} \leftarrowtail \mathbb{Z}_q$ and encrypts each member $x_i \in X$ with the public key $pk = h = g^{a_1+a_2}$ to get

$$\mathsf{dE}_{pk}(x_i) = (c_{x_i} = g^{r_{x_i}}, d_{x_i} = x_i h^{r_{x_i}}) \leftarrow \mathcal{DEL}.\mathsf{Enc}(x_i, pk, \mathsf{par}, r_{x_i});$$

(ii) generates the proof

$$\pi_1 = \mathsf{PoK}\big\{(r_{x_1}, ..., r_{x_v}) | \wedge_{i=1}^{v} (c_{x_i} = g^{r_{x_i}})\big\};$$

(iii) sends $R_1 = \big\langle \{\mathsf{dE}_{pk}(x_i)\}_{i=1}^{v}, \pi_1 \big\rangle$ to B.

Step 2. Party B, on receiving $R_1 = \big\langle \{\mathsf{dE}_{pk}(x_i)\}_{i=1}^{v}, \pi_1 \big\rangle$ from A, verifies the validity of the proof π_1. If verification fails, then B aborts. Otherwise, B does the following:

(i) chooses $r_{y_1}, ..., r_{y_w} \leftarrowtail \mathbb{Z}_q$ and encrypts each $y_j \in Y$ with the public key $pk = h = g^{a_1+a_2}$ to get

$$\mathsf{dE}_{pk}(y_j) = (c_{y_j} = g^{r_{y_j}}, d_{y_j} = y_j h^{r_{y_j}}) \leftarrow \mathcal{DEL}.\mathsf{Enc}(y_j, pk, \mathsf{par}, r_{y_j});$$

(ii) selects a random permutation $\phi \in \Sigma_v$, $\alpha_1, ..., \alpha_v \leftarrowtail \mathbb{Z}_q$ and computes for each $i = 1, ..., v$,

$$\begin{aligned}\mathsf{dE}_{pk}(\bar{x}_i) &= \mathsf{dE}_{pk}(x_{\phi^{-1}(i)})\mathcal{DEL}.\mathsf{Enc}(1, pk, \mathsf{par}, \alpha_i)\\ &= (c'_{x_i} = c_{x_{\phi^{-1}(i)}} g^{\alpha_i}, d'_{x_i} = d_{x_{\phi^{-1}(i)}} h^{\alpha_i})\end{aligned}$$

(iii) chooses $r \leftarrowtail \mathbb{Z}_q$ and computes

$$\begin{aligned}&\mathsf{dE}_{pk}((gy_j)^r) = (c'_{y_j} = (c_{y_j})^r, d'_{y_j} = g^r (d_{y_j})^r) \text{ for } 1 \le j \le w,\\ &\mathsf{dE}_{pk}((g\bar{x}_i)^r) = (\hat{c}_{x_i} = (c'_{x_i})^r, \hat{d}_{x_i} = g^r (d'_{x_i})^r) \text{ for } 1 \le i \le v;\end{aligned}$$

(iv) constructs proof

$$\pi_2 = \mathsf{PoK}\big\{(r_{y_1}, ..., r_{y_w}, r) | \wedge_{i=1}^{v} (\hat{c}_{x_i} = (c'_{x_i})^r)(\hat{d}_{x_i} = g^r (d'_{x_i})^r) \\ \wedge_{j=1}^{w} (c_{y_j} = g^{r_{y_j}})(c'_{y_j} = (c_{y_j})^r)(d'_{y_j} = (d_{y_j})^r)\big\},$$

$$\hat{\pi}_2 = \mathsf{PoK}\big\{(\phi \in \Sigma_v, \alpha_1, ..., \alpha_v)| \\ \{\mathsf{dE}_{pk}(\bar{x}_i) = \mathsf{dE}_{pk}(x_{\phi^{-1}(i)})\mathcal{DEL}.\mathsf{Enc}(1, pk, \mathsf{par}, \alpha_i)\}_{i=1}^{v}\big\};$$

(v) sends $R_2 = \big\langle \{\mathsf{dE}_{pk}(y_j), \mathsf{dE}_{pk}((gy_j)^r)\}_{j=1}^{w}, \{\mathsf{dE}_{pk}(\bar{x}_i), \mathsf{dE}_{pk}((g\bar{x}_i)^r)\}_{i=1}^{v}, \pi_2, \hat{\pi}_2 \big\rangle$ to A.

Step 3. On receiving $R_2 = \big\langle \{\mathsf{dE}_{pk}(y_j), \mathsf{dE}_{pk}((gy_j)^r)\}_{j=1}^{w}, \{\mathsf{dE}_{pk}(\bar{x}_i), \mathsf{dE}_{pk}((g\bar{x}_i)^r)\}_{i=1}^{v}, \pi_2, \hat{\pi}_2 \big\rangle$ from B, party A verifies the validity of the proofs $\pi_2, \hat{\pi}_2$. If at least one of the verification fails then A aborts. Otherwise, proceeds as follows:

(i) selects a random permutation $\psi \in \Sigma_w$, $\beta_1, ..., \beta_w \longleftarrow \mathbb{Z}_q$ and computes for each $j = 1, ..., w$,

$$\mathsf{dE}_{pk}((g\bar{y}_j)^r) = \mathsf{dE}_{pk}((gy_{\psi^{-1}(j)})^r)\mathcal{DEL}.\mathsf{Enc}(1, pk, \mathsf{par}, \beta_j) \\ = (\hat{c}_{y_j} = c'_{y_{\psi^{-1}(j)}} g^{\beta_j}, \hat{d}_{y_j} = d'_{y_{\psi^{-1}(j)}} h^{\beta_j})$$

(ii) selects $r' \longleftarrow \mathbb{Z}_q$ and computes

$$\mathsf{dE}_{pk}((g\bar{x}_i)^{rr'}) = (\bar{c}_{x_i}, \bar{d}_{x_i}) \text{ for } i = 1, ..., v, \text{ where} \\ \bar{c}_{x_i} = (\hat{c}_{x_i})^{r'} = (c'_{x_i})^{rr'} = (c_{x_{\phi^{-1}(i)}} g^{\alpha_i})^{rr'}, \\ \bar{d}_{x_i} = (\hat{d}_{x_i})^{r'} = (gd'_{x_i})^{rr'} = (gd_{x_{\phi^{-1}(i)}} h^{\alpha_i})^{rr'}, \\ \text{and } \mathsf{dE}_{pk}((g\bar{y}_j)^{rr'}) = (\bar{c}_{y_j}, \bar{d}_{y_j}) \text{ for } j = 1, ..., w, \text{ where} \\ \bar{c}_{y_j} = (\hat{c}_{y_j})^{r'} = (c'_{y_{\psi^{-1}(j)}} g^{\beta_j})^{r'} = (c_{y_{\psi^{-1}(j)}} g^{\beta_j})^{rr'}, \\ \bar{d}_{y_j} = (\hat{d}_{y_j})^{r'} = (d'_{y_{\psi^{-1}(j)}} h^{\beta_j})^{r'} = (gd_{y_{\psi^{-1}(j)}} h^{\beta_j})^{rr'};$$

(iii) chooses $\sigma_1, ..., \sigma_v \longleftarrow \mathbb{Z}_q$ and for each $i = 1, ..., v$, computes $(\bar{c}_{x_i})^{a_1}$ using his secret key $esk_A = a_1$ and encrypts $(\bar{c}_{x_i})^{a_1}$ using B's public key epk_B to generate

$$\mathsf{eE}_{epk_B}((\bar{c}_{x_i})^{a_1}) = (u_{x_i} = g^{\sigma_i}, \bar{u}_{x_i} = (\bar{c}_{x_i})^{a_1}(y_B)^{\sigma_i}) \\ \leftarrow \mathcal{EL}.\mathsf{Enc}((\bar{c}_{x_i})^{a_1}, epk_B = y_B, \mathsf{par}, \sigma_i);$$

(iv) generates a label $L \in \{0, 1\}^*$ using a session ID which has been agreed by all parities beforehand and the hash of past communication;

(v) chooses $r_1, ..., r_w, z_1, ..., z_w \longleftarrow \mathbb{Z}_q$ and computes $\{\bar{u}_{y_j} = (\bar{c}_{y_j})^{a_1} \cdot g^{r_j}\}_{j=1}^{w}$, and for each $j = 1, ..., w$, generates

$$\mathsf{vE}_{vpk_{Ar}}(g^{r_j}) = (t_{1j} = g^{z_j}, t_{2j} = \widehat{g}^{z_j}, t_{3j} = c^{z_j} g^{r_j}, t_{4j} = a^{z_j} b^{z_j \rho_j}) \\ \leftarrow \mathcal{VE}.\mathsf{Enc}(g^{r_j}, vpk_{Ar}, \mathsf{gpar}, z_j, L, \mathcal{H}), \\ \text{where } vpk_{Ar} = (a, b, c), \rho_j = \mathcal{H}(t_{1j}, t_{2j}, t_{3j}, L);$$

Common input: $\mathsf{gpar}, epk_A, epk_B, pk = epk_A \cdot epk_B, vpk_{Ar}$

A's private input:		B's private input:
$X = \{x_1, ..., x_v\}, esk_A = a_1$		$Y = \{y_1, ..., y_w\}, esk_B = a_2$
$r_{x_1}, ..., r_{x_v} \leftarrow\!\!\!\!\!\!\rightharpoonup \mathbb{Z}_q$ for $1 \le i \le v$,		
$\mathsf{dE}_{pk}(x_i) = (c_{x_i} = g^{r_{x_i}}, d_{x_i} = x_i h^{r_{x_i}})$ $\leftarrow \mathcal{DEL}.\mathsf{Enc}(x_i, pk, r_{x_i})$	$\xrightarrow{R_1}$	$r, r_{y_1}, ..., r_{y_w}, \alpha_1, ..., \alpha_v \leftarrow\!\!\!\!\!\!\rightharpoonup \mathbb{Z}_q, \phi \leftarrow\!\!\!\!\!\!\rightharpoonup \Sigma_v$
$R_1 = \langle \{\mathsf{dE}_{pk}(x_i)\}_{i=1}^{v}, \pi_1 \rangle$		for $1 \le j \le w$, $\mathsf{dE}_{pk}(y_j) = (c_{y_j} = g^{r_{y_j}}, d_{y_j} = y_j h^{r_{y_j}})$ $\leftarrow \mathcal{DEL}.\mathsf{Enc}(y_j, pk, r_{y_j})$,
	$\xleftarrow{R_2}$	$\mathsf{dE}_{pk}((gy_j)^r) = (\hat{c}_{y_j} = (c_{y_j})^r, \hat{d}_{y_j} = (gd_{y_j})^r)$
$r', \sigma_1, ..., \sigma_v, r_1, ..., r_w, z_1, ..., z_w \leftarrow\!\!\!\!\!\!\rightharpoonup \mathbb{Z}_q$, $\beta_1, ..., \beta_w \leftarrow\!\!\!\!\!\!\rightharpoonup \mathbb{Z}_q, \psi \leftarrow\!\!\!\!\!\!\rightharpoonup \Sigma_w$ for $1 \le i \le v$,		for $1 \le i \le v$, $\mathsf{dE}_{pk}(\bar{x}_i) = \mathsf{dE}_{pk}(x_{\phi^{-1}(i)}) \mathcal{DEL}.\mathsf{Enc}(1, pk, \mathsf{par}, \alpha_i)$ $= (c'_{x_i} = c_{x_{\phi^{-1}(i)}} g^{\alpha_i}, d'_{x_i} = d_{x_{\phi^{-1}(i)}} h^{\alpha_i})$
$\mathsf{dE}_{pk}((g\bar{x}_i)^{rr'}) = (\bar{c}_{x_i} = (\hat{c}_{x_i})^{r'}, \bar{d}_{x_i} = (\hat{d}_{x_i})^{r'})$, $\mathsf{eE}_{epk_B}((\bar{c}_{x_i})^{a_1}) \leftarrow \mathcal{EL}.\mathsf{Enc}((\bar{c}_{x_i})^{a_1}, epk_B, \alpha_i)$ for $1 \le j \le w$,		$\mathsf{dE}_{pk}((g\bar{x}_i)^r) = (\hat{c}_{x_i} = (c'_{x_i})^r, \hat{d}_{x_i} = g^r (d'_{x_i})^r)$ $R_2 = \big\langle \{\mathsf{dE}_{pk}(y_j), \mathsf{dE}_{pk}((gy_j)^r)\}_{j=1}^{w},$ $\{\mathsf{dE}_{pk}(\bar{x}_i), \mathsf{dE}_{pk}((g\bar{x}_i)^r)\}_{i=1}^{v}, \pi_2, \hat{\pi}_2 \big\rangle$
$\mathsf{dE}_{pk}((g\bar{y}_j)^r) = (\hat{c}_{y_j}, \hat{d}_{y_j})$ $= \mathsf{dE}_{pk}((gy_{\psi^{-1}(j)}))^r \mathcal{DEL}.\mathsf{Enc}(1, pk, \mathsf{par}, \beta_j)$ $\mathsf{dE}_{pk}((g\bar{y}_j)^{rr'}) = (\bar{c}_{y_j} = (\hat{c}_{y_j})^{r'}, \bar{d}_{y_j} = (\hat{d}_{y_j})^{r'})$ $\bar{u}_{y_j} = (\bar{c}_{y_j})^{a_1} \cdot g^{r_j}$,		
$\mathsf{vE}_{vpk_{Ar}}(g^{r_j}) \leftarrow \mathcal{VE}.\mathsf{Enc}(g^{r_j}, vpk_{Ar}, z_j, L, \mathcal{H})$ $R_3 = \big\langle \{\mathsf{dE}_{pk}((g\bar{x}_i)^{rr'}), \mathsf{eE}_{epk_B}((\bar{c}_{x_i})^{a_1})\}_{i=1}^{v},$ $\{\mathsf{dE}_{pk}((g\bar{y}_j)^r), \mathsf{dE}_{pk}((g\bar{y}_j)^{rr'}),$ $\mathsf{vE}_{vpk_{Ar}}(g^{r_j}), \bar{u}_{y_j}\}_{j=1}^{w}, \pi_3, \hat{\pi}_3 \big\rangle$	$\xrightarrow{R_3}$	for $1 \le i \le v$, extract $\bar{c}_{x_i}$ from R_3, $s_{x_i} = (\bar{c}_{x_i})^{a_2}$ for $1 \le j \le w$, extract $\bar{c}_{y_j}$ from R_3, $s_{y_j} = (\bar{c}_{y_j})^{a_2}$
	$\xleftarrow{R_4}$	$R_4 = \big\langle \{s_{x_i}\}_{i=1}^{v}, \{s_{y_j}\}_{j=1}^{w}, \pi_4 \big\rangle$
for $1 \le i \le v$, $\frac{\bar{d}_{x_i}}{(\bar{c}_{x_i})^{a_1} s_{x_i}} = (gx_{\phi^{-1}(i)})^{rr'}$ for $1 \le j \le w$, $\frac{\bar{d}_{y_j}}{(\bar{c}_{y_j})^{a_1} s_{y_j}} = (gy_{\psi^{-1}(j)})^{rr'}$		
$\lvert X \cap Y \rvert = \lvert \{(gx_{\phi^{-1}(1)})^{rr'}, ..., (gx_{\phi^{-1}(v)})^{rr'}\}$ $\cap \{(gy_{\psi^{-1}(1)})^{rr'}, ..., (gy_{\psi^{-1}(w)})^{rr'}\} \rvert$ $R_5 = \big\langle \{(g^{r_j})\}_{j=1}^{w}, \pi_5 \big\rangle$	$\xrightarrow{R_5}$	for $1 \le i \le v$, extract $\bar{d}_{x_i}$ from R_3, $(\bar{c}_{x_i})^{a_1} \leftarrow \mathcal{EL}.\mathsf{Dec}(\mathsf{eE}_{epk_B}((\bar{c}_{x_i})^{a_1}), esk_B)$ $\frac{\bar{d}_{x_i}}{(\bar{c}_{x_i})^{a_1} s_{x_i}} = (gx_{\phi^{-1}(i)})^{rr'}$ for $1 \le j \le w$, extract $\bar{d}_{\bar{y}_j}, \bar{u}_{\bar{y}_j}$ from R_3, $\frac{\bar{u}_{y_j}}{g^{r_j}} = (\bar{c}_{y_j})^{a_1}, \frac{\bar{d}_{y_j}}{(\bar{c}_{y_j})^{a_1} s_{y_j}} = (gy_{\psi^{-1}(j)})^{rr'}$ $\lvert X \cap Y \rvert = \lvert \{(gx_{\phi^{-1}(1)})^{rr'}, ..., (gx_{\phi^{-1}(v)})^{rr'}\}$ $\cap \{(gy_{\psi^{-1}(1)})^{rr'}, ..., (gy_{\psi^{-1}(w)})^{rr'}\} \rvert$

Figure 4 : Communication flow of our mPSI-CA

(vi) constructs proof

$$\pi_3 = \mathsf{PoK}\big\{(a_1, r', r_1, ..., r_w, z_1, ..., z_w, \sigma_1, ..., \sigma_v)|(y_A = g^{a_1}) \\ \wedge_{j=1}^{w} (\bar{c}_{y_j} = (\hat{c}_{y_j})^{r'}) \wedge_{j=1}^{w} (\bar{d}_{y_j} = (\hat{d}_{y_j})^{r'}) \\ \wedge_{i=1}^{v} (\bar{c}_{x_i} = (\hat{c}_{x_i})^{r'})(\bar{d}_{x_i} = (\hat{d}_{x_i})^{r'})(u_{x_i} = g^{\sigma_i})(\bar{u}_{x_i} = (\bar{c}_{x_i})^{a_1}(y_B)^{\sigma_i}) \\ \wedge_{j=1}^{w} (\bar{u}_{y_j} = (\bar{c}_{y_j})^{a_1} \cdot g^{r_j})(t_{1j} = g^{z_j})(t_{2j} = \widehat{g}^{z_j})(t_{3j} = c^{z_j} g^{r_j})(t_{4j} = a^{z_j} b^{z_j \rho_j})\big\},$$

$$\hat{\pi}_3 = \mathsf{PoK}\big\{(\psi \in \Sigma_w, \beta_1, ..., \beta_v)| \\ \{\mathsf{dE}_{pk}((g\bar{y}_j)^r) = \mathsf{dE}_{pk}((gy_{\psi^{-1}(j)})^r)\mathcal{DEL}.\mathsf{Enc}(1, pk, \mathsf{par}, \beta_j)\}_{j=1}^{w}\big\};$$

(vii) sends $R_3 = \big\langle \{\mathsf{dE}_{pk}((g\bar{x}_i)^{rr'}), \mathsf{eE}_{epk_B}((\bar{c}_{x_i})^{a_1})\}_{i=1}^{v}, \{\mathsf{dE}_{pk}((g\bar{y}_j)^{rr'}), \mathsf{dE}_{pk}((g\bar{y}_j)^r), \mathsf{vE}_{vpk_{Ar}}(g^{r_j}), \bar{u}_{y_j}\}_{j=1}^{w}, \pi_3, \hat{\pi}_3\big\rangle$ to B.

Step 4. On receiving $R_3 = \big\langle \{(\mathsf{dE}_{pk}(g\bar{x}_i))^{rr'}, \mathsf{eE}_{epk_B}((\bar{c}_{x_i})^{a_1})\}_{i=1}^{v}, \{(\mathsf{dE}_{pk}(g\bar{y}_j))^{rr'}, (\mathsf{dE}_{pk}(g\bar{y}_j))^r, \mathsf{vE}_{vpk_{Ar}}(g^{r_j}), \bar{u}_{y_j}\}_{j=1}^{w}, \pi_3, \hat{\pi}_3\big\rangle$ from A, party B checks the proofs $\pi_3, \hat{\pi}_3$. If the verification of at least one of the proof fails then B aborts, else dose the following:

(i) extracts $\{\bar{c}_{x_i}\}_{i=1}^{v}, \{\bar{c}_{y_j}\}_{j=1}^{w}$ from $\{\mathsf{dE}_{pk}((g\bar{x}_i)^{rr'})\}_{i=1}^{v}, \{\mathsf{dE}_{pk}((g\bar{y}_j)^{rr'})\}_{j=1}^{w}$ respectively in R_3 and computes $\{s_{x_i} = (\bar{c}_{x_i})^{a_2}\}_{i=1}^{v}, \{s_{y_j} = (\bar{c}_{y_j})^{a_2}\}_{j=1}^{w}$ using his secret key $esk_B = a_2$;

(ii) constructs the proof

$$\pi_4 = \mathsf{PoK}\big\{(a_2)|(y_B = g^{a_2}) \wedge_{i=1}^{v} (s_{x_i} = (\bar{c}_{x_i})^{a_2})(s_{y_j} = (\bar{c}_{y_j})^{a_2})\big\};$$

(iii) sends $R_4 = \big\langle \{s_{x_i}\}_{i=1}^{v}, \{s_{y_j}\}_{j=1}^{w}, \pi_4\big\rangle$ to A.

Step 5. Party A, on receiving $R_4 = \big\langle \{s_{x_i} = (\bar{c}_{x_i})^{a_2}\}_{i=1}^{v}, \{s_{y_j} = (\bar{c}_{y_j})^{a_2}\}_{j=1}^{w}, \pi_4\big\rangle$ from B, verifies the validity of the proof π_4. Party A aborts if the verification does not succeed, else does the following using his secret key $esk_A = a_1$ and $\{\bar{c}_{x_i}, \bar{d}_{x_i}\}_{i=1}^{v}, \{\bar{c}_{y_j}, \bar{d}_{y_j}\}_{j=1}^{w}$ computed in *Step* 3:

(i) computes for $i = 1, ..., v$,

$$\frac{\bar{d}_{x_i}}{(\bar{c}_{x_i})^{a_1} s_{x_i}} = \frac{\bar{d}_{x_i}}{(\bar{c}_{x_i})^{a_1+a_2}} = \frac{(gd_{x_{\phi^{-1}(i)}} h^{\alpha_i})^{rr'}}{(c_{x_{\phi^{-1}(i)}} g^{\alpha_i})^{(a_1+a_2)rr'}}$$

$$= \frac{(gx_{\phi^{-1}(i)})^{rr'} g^{(r_{x_{\phi^{-1}(i)}} + \alpha_i)(a_1+a_2)rr'}}{g^{(r_{x_{\phi^{-1}(i)}} + \alpha_i)(a_1+a_2)rr'}} = (gx_{\phi^{-1}(i)})^{rr'},$$

and for $j = 1, ..., w$,

$$\frac{\bar{d}_{y_j}}{(\bar{c}_{y_j})^{a_1} s_{y_j}} = \frac{\bar{d}_{y_j}}{(\bar{c}_{y_j})^{a_1+a_2}} = \frac{(gd_{y_{\psi^{-1}(j)}} h^{\beta_j})^{rr'}}{(c_{y_{\psi^{-1}(j)}} g^{\beta_j})^{(a_1+a_2)rr'}}$$

$$= \frac{(gy_{\psi^{-1}(j)})^{rr'} g^{(\beta_j + r_{y_{\psi^{-1}(j)}})(a_1+a_2)rr'}}{g^{(\beta_j + r_{y_{\psi^{-1}(j)}})(a_1+a_2)rr'}} = (gy_{\psi^{-1}(j)})^{rr'};$$

(ii) sets the cardinality of $X \cap Y$ as

$$|X \cap Y| = |\{(gx_{\phi^{-1}(1)})^{rr'}, ..., (gx_{\phi^{-1}(v)})^{rr'}\} \cap \{(gy_{\psi^{-1}(1)})^{rr'}, ..., (gy_{\psi^{-1}(w)})^{rr'}\}|;$$

(iii) constructs the proof

$$\pi_5 = \mathsf{PoK}\big\{(z_1, ..., z_w) | \wedge_{j=1}^{w} (t_{1j} = g^{z_j})(t_{2j} = \widehat{g}^{z_j})(t_{3j} = c^{z_j} g^{r_j})(t_{4j} = a^{z_j} b^{z_j \rho_j})\big\};$$

(iv) sends $R_5 = \big\langle \{(g^{r_j})\}_{j=1}^{w}, \pi_5 \big\rangle$ to B.

Step 6. On receiving $R_5 = \big\langle \{(g^{r_j})\}_{j=1}^{w}, \pi_5 \big\rangle$ from A, party B verifies the validity of the proof π_5. If the verification of the proof succeeds, then B

(i) for each $i = 1, ..., v$, decrypts $\mathsf{eE}_{pk_B}(\bar{c}_{x_i}^{a_1})$ received in *Step* 3 using his secret key $sk_B = a_2$ to get $(\bar{c}_{x_i})^{a_1} \leftarrow \mathcal{EL}.\mathsf{Dec}(\mathsf{eE}_{epk_B}((\bar{c}_{x_i})^{a_1}), esk_B)$, extracts $\bar{d}_{x_i}$ from $\mathsf{dE}_{pk}((g\bar{x}_i)^{rr'})$ in R_3, uses s_{x_i} computed in *Step* 4 to generate

$$\frac{\bar{d}_{x_i}}{(\bar{c}_{x_i})^{a_1} s_{x_i}} = \frac{\bar{d}_{x_i}}{(\bar{c}_{x_i})^{a_1+a_2}} = \frac{(g d_{x_{\phi^{-1}(i)}} h^{\alpha_i})^{rr'}}{(c_{x_{\phi^{-1}(i)}} g^{\alpha_i})^{(a_1+a_2)rr'}}$$

$$= \frac{(g x_{\phi^{-1}(i)})^{rr'} g^{(r_{x_{\phi^{-1}(i)}} + \alpha_i)(a_1+a_2)rr'}}{g^{(r_{x_{\phi^{-1}(i)}} + \alpha_i)(a_1+a_2)rr'}} = (g x_{\phi^{-1}(i)})^{rr'};$$

(ii) for each $j = 1, ..., w$, extracts $\bar{d}_{y_j}$ from $\mathsf{dE}_{pk}((g\bar{y}_j)^{rr'})$ in R_3 and uses $\bar{u}_{y_j}$ obtained from R_3, s_{y_j} computed in *Step* 4 to generate

$$\frac{\bar{u}_{y_j}}{g^{r_j}} = \frac{(\bar{c}_{y_j})^{a_1} \cdot g^{r_j}}{g^{r_j}} = (\bar{c}_{y_j})^{a_1} \text{ and}$$

$$\frac{\bar{d}_{y_j}}{(\bar{c}_{y_j})^{a_1} s_{y_j}} = \frac{\bar{d}_{y_j}}{(\bar{c}_{y_j})^{a_1+a_2}} = \frac{(g d_{y_{\psi^{-1}(j)}} h^{\beta_j})^{rr'}}{(c_{y_{\psi^{-1}(j)}} g^{\beta_j})^{(a_1+a_2)rr'}}$$

$$= \frac{(g y_{\psi^{-1}(j)})^{rr'} g^{(\beta_j + r_{y_{\psi^{-1}(j)}})(a_1+a_2)rr'}}{g^{(\beta_j + r_{y_{\psi^{-1}(j)}})(a_1+a_2)rr'}} = (g y_{\psi^{-1}(j)})^{rr'};$$

(iii) sets $|X \cap Y| = |\{(g x_{\phi^{-1}(1)})^{rr'}, ..., (g x_{\phi^{-1}(v)})^{rr'}\} \cap \{(g y_{\psi^{-1}(1)})^{rr'}, ..., (g y_{\psi^{-1}(w)})^{rr'}\}|$.

If the verification of π_5 does not succeed or A does not send $R_5 = \big\langle \{(g^{r_j})\}_{j=1}^{w}, \pi_5 \big\rangle$ i.e., if A prematurely aborts, then B sends a dispute resolution request to the arbiter Ar.

Dispute Resolution Protocol: This is analogous to the Dispute Resolution Protocol of the mPSI except that each $X \cap Y$ will be replaced by $|X \cap Y|$.

4.2 Security

We consider two cases: (case I) when the adversary corrupts two parties among the three parties and (case II) when the adversary corrupts only one party among the three parties.

Theorem 4.1. *If the encryption schemes $\mathcal{EL}$, $\mathcal{DEL}$ and $\mathcal{VE}$ are semantically secure, the associated proof protocols are zero knowledge proof and the associated permutations are random, then our* mPSI-*CA presented in section 4 is a secure computation protocol for the functionality* $\mathcal{F}_{\text{mPSI-}CA} : (X, Y) \rightarrow (|X \cap Y|, |X \cap Y|)$ *in the security model described in section 2.1.*

Proof. Let us consider $\mathcal{C}$ as the real world adversary that breaks the security of our mPSI-CA protocol among three parties A with private input set X, B with private input set Y and Ar. Also let there be an incorruptible trusted party T, parties $\bar{A}, \bar{B}$, $\bar{Ar}$ and simulator $\mathcal{SIM}$ in the ideal process. In real world, the global parameter $\mathsf{gpar} = (\mathsf{par}, \widehat{g}, \mathcal{H})$, where $\mathsf{par} = (p, q, g)$ is generated by a trusted party who certifies the public key pk_A, pk_B, pk_{Ar} of A, B, Ar respectively. In contrast, in ideal process simulator $\mathcal{SIM}$ does those things. We denote the joint output of $A, B, Ar, \mathcal{C}$ in the real world as $\mathsf{REAL}_{\text{mPSI-CA},\mathcal{C}}(X, Y)$ and the joint output of $\bar{A}, \bar{B}, \bar{Ar}, \mathcal{SIM}$ in the ideal process as $\mathsf{IDEAL}_{\mathcal{F}_{\text{mPSI-CA}},\mathcal{SIM}}(X, Y)$.

• **Case I (When the adversary $\mathcal{C}$ corrupts two parties).**

1. A **and** Ar **are corrupted.** Let $\mathcal{Z}$ be a distinguisher who controls $\mathcal{C}$, feeds the input of the honest party B, and also sees the output of B. Now we will present a series of games $Game_0, ..., Game_4$ to prove that $\mathcal{Z}$'s view in the real world ($\mathcal{C}$'s view +B's output) and its view in the ideal world ($\mathcal{C}$'s view + $\bar{B}$'s output) are indistinguishable. For each $i = 0, ..., 3$, $Game_{i+1}$ modifies $Game_i$ slightly such that $\mathcal{Z}$'s views in $Game_i$ and $Game_{i+1}$ remain indistinguishable. The probability that $\mathcal{Z}$ distinguishes the view of $Game_i$ from the view of real protocol, is denoted by $Pr[Game_i]$ and S_i is considered as simulator in $Game_i$.
   $Game_0$: This game is same as real world protocol, where the simulator S_0 has full knowledge of B and interacts with $\mathcal{C}$. Hence,

$$Prob[\mathsf{REAL}_{\text{mPSI-CA},\mathcal{C}}(X, Y)] = Prob[Game_0].$$

   $Game_1$: $Game_1$ is same as $Game_0$ except that if the proof π_1 is valid then the simulator S_1 runs the extractor algorithm for π_1 with $\mathcal{C}$ to extract the exponents $\{r_{x_1}, ..., r_{x_v}\}$. The simulator S_1 then extracts $x_i = \frac{d_{x_i}}{h^{r_{x_i}}}$ by extracting $d_{x_i} = x_i h^{r_{x_i}}$ from $\mathsf{dE}_{pk}(x_i)$ in R_1, h from $pk = epk_A \cdot epk_B$ and using the exponent r_{x_i} for $1 \leq i \leq v$. In this way S_1 extracts the private input set $X = \{x_1, ..., x_v\}$ of A. $\mathcal{Z}$'s views in $Game_0$ and $Game_1$ are indistinguishable because of simulation soundness of the proof π_1. Therefore,

$$|Prob[Game_1] - Prob[Game_0]| \leq \epsilon_1(\kappa), \text{ where } \epsilon_1(\kappa) \text{ is a negligible function.}$$

   $Game_2$: Note that in this game the simulator S_2 has the knowledge of extracted set $X = \{x_1, ..., x_v\}$, input set $Y = \{y_1, ..., y_w\}$ and secret key $esk_B = a_2$ of B. $Game_2$ is same as $Game_1$ except that

   (a) if the verification of the proof π_5 succeeds then S_3 outputs $|X \cap Y|$ as the final output of B making use of the extracted X,

   (b) if the verification of the proof π_5 does not succeed or $\mathcal{C}$ aborts prematurely in mPSI-CA protocol then the following cases arise:

   ⋄ if $\mathcal{C}$ sends $\{g_1, ..., g_w\} \subset \mathbb{G}$ to S_3 in dispute resolution protocol then S_3 does the

following:
– for each $i = 1, ..., v$, decrypts $\mathsf{eE}_{epk_B}((\bar{c}_{x_i})^{a_1})$ using $esk_B = a_2$ to get $(\bar{c}_{x_i})^{a_1} \leftarrow \mathcal{EL}.\mathsf{Dec}(\mathsf{eE}_{epk_B}((\bar{c}_{x_i})^{a_1}), esk_B)$, extracts $\bar{d}_{x_i}, \bar{c}_{x_i}$ from $\mathsf{dE}_{pk}((g\bar{x}_i)^{rr'})$ in R_3, computes $\frac{\bar{d}_{x_i}}{(\bar{c}_{x_i})^{a_1}(\bar{c}_{x_i})^{a_2}} = (gx_{\phi^{-1}(i)})^{rr'}$;
– for each $j = 1, ..., w$, computes $\frac{\bar{d}_{y_j}}{\frac{\bar{u}_{y_j}}{g_j}(\bar{c}_{y_j})^{a_2}} = \hat{y}_j$ by extracting $\bar{d}_{y_j}, \bar{c}_{y_j}$ from $\mathsf{dE}_{pk}((g\bar{y}_j)^{rr'})$ in R_3 and using $\bar{u}_{y_j}$ obtained from R_3;
– outputs $|\{(gx_{\phi^{-1}(1)})^{rr'}, ..., (gx_{\phi^{-1}(v)})^{rr'}\} \cap \{\hat{y}_1, ..., \hat{y}_w\}|$ as the final output of B.
$\diamond$ if $\mathcal{C}$ aborts in dispute resolution protocol then S_3 outputs $\perp$ as the final output of B.

By the simulation soundness property of the proof π_5, $\mathcal{Z}$'s views in $Game_2$ and $Game_3$ are indistinguishable. Hence,

$$|Prob[Game_2] - Prob[Game_1]| \leq \epsilon_2(\kappa), \text{ where } \epsilon_2(\kappa) \text{ is a negligible function.}$$

$Game_3$: $Game_3$ is same as $Game_2$ except that S_3 does the following after extracting $X = \{x_1, ..., x_v\}$:

(a) computes $|X \cap Y|$,

(b) constructs a set $\overline{\overline{Y}} = \{\bar{\bar{y}}_1, ..., \bar{\bar{y}}_w\}$ by including $|X \cap Y|$ many random elements of X together with $w - |X \cap Y|$ many random elements chosen from $\mathbb{G}$,

(c) selects a random permutation $\widehat{\phi} \in \Sigma_v$, $\alpha_1, ..., \alpha_v \leftarrow\!\!\!\!\!\rightharpoonup \mathbb{Z}_q$ and computes for each $i = 1, ..., v$,
$$\mathsf{dE}_{pk}(\bar{x}_i) = \mathsf{dE}_{pk}(x_{\widehat{\phi}^{-1}(i)})\mathcal{DEL}.\mathsf{Enc}(1, pk, \mathsf{par}, \alpha_i)$$

(d) chooses $r \leftarrow\!\!\!\!\!\rightharpoonup \mathbb{Z}_q$,

(e) computes $\left\langle \{\mathsf{dE}_{pk}(\bar{\bar{y}}_j), \mathsf{dE}_{pk}((g\bar{\bar{y}}_j)^r)\}_{j=1}^w, \{\mathsf{dE}_{pk}((g\bar{x}_i)^r)\}_{i=1}^v \right\rangle$,

(f) sends the tuple $\left\langle \{\mathsf{dE}_{pk}(\bar{\bar{y}}_j), \mathsf{dE}_{pk}((g\bar{\bar{y}}_j)^r)\}_{j=1}^w, \{\mathsf{dE}_{pk}(\bar{x}_i), \mathsf{dE}_{pk}((g\bar{x}_i)^r)\}_{i=1}^v \right\rangle$ as $\left\langle \{\mathsf{dE}_{pk}(y_j), \mathsf{dE}_{pk}((gy_j)^r)\}_{j=1}^w, \{\mathsf{dE}_{pk}(\bar{x}_i), \mathsf{dE}_{pk}((g\bar{x}_i)^r)\}_{i=1}^v \right\rangle$ to $\mathcal{C}$ and simulates $\pi_2, \hat{\pi}_2$.

As the encryption scheme $\mathcal{DEL}$ is semantically secure, the tuple $\left\langle \{\mathsf{dE}_{pk}(y_j), \mathsf{dE}_{pk}((gy_j)^r)\}_{j=1}^w, \{\mathsf{dE}_{pk}(\bar{x}_i), \mathsf{dE}_{pk}((g\bar{x}_i)^r)\}_{i=1}^v \right\rangle$ is identically distributed in $Game_3$ and $Game_2$. The zero-knowledge (simulatability) of $\pi_2, \hat{\pi}_2$ and indistinguishability of the tuple make the views of $\mathcal{Z}$'s in $Game_2$ and $Game_3$ indistinguishable. Therefore, there exists a negligible function $\epsilon_3(\kappa)$ such that

$$|Prob[Game_3] - Prob[Game_2]| \leq \epsilon_3(\kappa).$$

$Game_4$: This game is same as $Game_3$ except that during the setup phase S_4 chooses $a_2 \leftarrow\!\!\!\!\!\rightharpoonup \mathbb{Z}_q$ and in *Step* 4 simulates π_4, instead of proving it. By the zero-knowledge (simulatability) of π_4 the views of $\mathcal{Z}$'s in $Game_3$ and $Game_4$ are indistinguishable. Consequently,

$$|Prob[Game_4] - Prob[Game_3]| \leq \epsilon_4(\kappa), \text{ where } \epsilon_4(\kappa) \text{ is a negligible function.}$$

Let us construct the ideal world adversary $\mathcal{SIM}$ that uses $\mathcal{C}$ as subroutine, simulates the honest party B and controls $\bar{A}, \bar{A}r$ and incorporates all steps from $Game_4$.

(i) First $\mathcal{SIM}$ plays the role of trusted party by generating the global parameter $\mathsf{gpar} = (\mathsf{par}, \widehat{g}, \mathcal{H})$, where $\mathsf{par} = (p, q, g)$. $\mathcal{SIM}$ then plays the role of honest party B by choosing $\bar{a}_2 \leftarrow\!\!\prec \mathbb{Z}_q$ and publishing $g^{\bar{a}_2}$ as the public key $epk_B = y_B$. $\mathcal{SIM}$ also acts as certifying authority to obtain respective public keys epk_A, vpk_{Ar} of A, Ar. $\mathcal{SIM}$ then invokes $\mathcal{C}$.

(ii) On receiving $R_1 = \left\langle \{\mathsf{dE}_{pk}(x_i)\}_{i=1}^{v}, \pi_1 \right\rangle$ from $\mathcal{C}$, $\mathcal{SIM}$ verifies the proof π_1. If the verification does not succeed, then $\mathcal{SIM}$ instructs $\bar{A}$ to send $\perp$ to T and terminates the execution. Otherwise, $\mathcal{SIM}$ runs the extractor algorithm for π_1 with $\mathcal{C}$ to extract $\{r_{x_1}, ..., r_{x_v}\}$. Utilizing $\{r_{x_1}, ..., r_{x_v}\}$, $\mathcal{SIM}$ extracts the input set $X = \{x_1, ..., x_v\}$ by extracting $\{d_{x_i} = x_i h^{r_{x_i}}\}_{i=1}^{v}$ from $\{\mathsf{dE}_{pk}(x_i)\}_{i=1}^{v}$ in R_1 and h from $pk = epk_A \cdot epk_B$. $\mathcal{SIM}$ then instructs $\bar{A}$ to send X to T, $\bar{A}r$ to send $b_A = \circ$ to T and receives $|X \cap Y|$ from T.

(iii) $\mathcal{SIM}$ constructs a set $\overline{\overline{Y}} = \{\bar{\bar{y}}_1, ..., \bar{\bar{y}}_w\}$ by including $|X \cap Y|$ many random elements of X together with $w - |X \cap Y|$ many random elements chosen from $\mathbb{G}$. $\mathcal{SIM}$ then

– selects a random permutation $\widehat{\phi} \in \Sigma_v$, $\alpha_1, ..., \alpha_v \leftarrow\!\!\prec \mathbb{Z}_q$ and computes for each $i = 1, ..., v$,

$$\mathsf{dE}_{pk}(\bar{x}_i) = \mathsf{dE}_{pk}(x_{\widehat{\phi}^{-1}(i)})\mathcal{DEL}.\mathsf{Enc}(1, pk, \mathsf{par}, \alpha_i)$$

– chooses $r \leftarrow\!\!\prec \mathbb{Z}_q$ and computes $\left\langle \{\mathsf{dE}_{pk}(\bar{\bar{y}}_j), \mathsf{dE}_{pk}((g\bar{\bar{y}}_j)^r)\}_{j=1}^{w}, \{\mathsf{dE}_{pk}((g\bar{x}_i)^r)\}_{i=1}^{v} \right\rangle$,

– finally, sends $\left\langle \{\mathsf{dE}_{pk}(\bar{\bar{y}}_j), \mathsf{dE}_{pk}((g\bar{\bar{y}}_j)^r)\}_{j=1}^{w}, \{\mathsf{dE}_{pk}(\bar{x}_i), \mathsf{dE}_{pk}((g\bar{x}_i)^r)\}_{i=1}^{v} \right\rangle$ as $\left\langle \{\mathsf{dE}_{pk}(y_j), \mathsf{dE}_{pk}((gy_j)^r)\}_{j=1}^{w}, \{\mathsf{dE}_{pk}(\bar{x}_i), \mathsf{dE}_{pk}((g\bar{x}_i)^r)\}_{i=1}^{v} \right\rangle$ to $\mathcal{C}$ and simulates $\pi_2, \hat{\pi}_2$.

(iv) On receiving $R_3 = \left\langle \{\mathsf{dE}_{pk}((g\bar{x}_i)^{rr'}), \mathsf{eE}_{epk_B}((\bar{c}_{x_i})^{a_1})\}_{i=1}^{v}, \{\mathsf{dE}_{pk}((g\bar{\bar{y}}_j)^{rr'}), \mathsf{dE}_{pk}((g\bar{\bar{y}}_j)^r), \mathsf{vE}_{vpk_{Ar}}(g^{r_j}), \bar{u}_{y_j}\}_{j=1}^{w}, \pi_3, \hat{\pi}_3 \right\rangle$ from $\mathcal{C}$, $\mathcal{SIM}$ verifies the validity of the proofs $\pi_3, \hat{\pi}_3$. If the verification of at least one of the proof fails then $\mathcal{SIM}$ instructs $\bar{A}$ to send $\perp$ to T and terminates the execution. Otherwise, $\mathcal{SIM}$ computes $\{s_{x_i} = (\bar{c}_{x_i})^{\bar{a}_2}\}_{i=1}^{v}$, $\{s_{y_j} = (\bar{c}_{y_j})^{\bar{a}_2}\}_{j=1}^{w}$, sends it to $\mathcal{C}$ and simulates the proof π_4. $\mathcal{SIM}$ then executes following steps according to $\mathcal{C}$'s reply.

(v) If $\mathcal{C}$ instructs A to send $\{g_1, ..., g_w\} \subset \mathbb{G}$, then $\mathcal{SIM}$ verifies the validity of the proof π_5. If the verification succeeds then $\mathcal{SIM}$ instructs $\bar{A}r$ to send $b_B = \circ$. If verification fails or $\mathcal{C}$ instructs A to abort in mPSI-CA protocol then the following cases arise:

$\diamond$ if $\mathcal{C}$ instructs Ar to send $\{g_1, ..., g_w\} \subset \mathbb{G}$ in dispute resolution protocol, then $\mathcal{SIM}$ does the following:

– for each $i = 1, ..., v$, decrypts $\mathsf{eE}_{epk_B}((\bar{c}_{x_i})^{a_1})$ using $esk_B = \bar{a}_2$ to get $(\bar{c}_{x_i})^{a_1} \leftarrow \mathcal{EL}.\mathsf{Dec}(\mathsf{eE}_{epk_B}((\bar{c}_{x_i})^{a_1}), esk_B)$, extracts $\bar{d}_{x_i}, \bar{c}_{x_i}$ from $\mathsf{dE}_{pk}((g\bar{x}_i)^{rr'})$ in R_3, computes $\frac{\bar{d}_{x_i}}{(\bar{c}_{x_i})^{a_1}(\bar{c}_{x_i})^{\bar{a}_2}} = (gx_{\hat{\phi}^{-1}(i)})^{rr'}$;

– for each $j = 1, ..., w$, computes $\frac{\bar{d}_{y_j}}{\frac{\bar{u}_{y_j}}{g_j}(\bar{c}_{y_j})^{\bar{a}_2}} = \tilde{y}_j$ by extracting $\bar{d}_{y_j}, \bar{c}_{y_j}$ from $\mathsf{dE}_{pk}((g\bar{\bar{y}}_j)^{rr'})$ in R_3 and using $\bar{u}_{y_j}$ obtained from R_3;

– instructs $\bar{A}r$ to send $b_B = |\{(gx_{\hat{\phi}^{-1}(1)})^{rr'}, ..., (gx_{\hat{\phi}^{-1}(v)})^{rr'}\} \cap \{\tilde{y}_1, ..., \tilde{y}_w\}|$ to T, outputs whatever $\mathcal{C}$ outputs and terminates.

$\diamond$ if $\mathcal{C}$ instructs Ar to abort in dispute resolution protocol $\mathcal{SIM}$ instructs $\bar{A}r$ to send

$b_B = \perp$ to T. Then $\mathcal{SIM}$ outputs whatever $\mathcal{C}$ outputs and terminates.

(vi) If $\mathcal{C}$ instructs both A and Ar to abort, then $\mathcal{SIM}$ instructs $\bar{A}r$ to send $b_B = \perp$ to T, outputs whatever $\mathcal{C}$ outputs and terminates.

Thus the ideal world adversary $\mathcal{SIM}$ provides $\mathcal{C}$ the same simulation as the simulator S_4 in $Game_4$. Hence $Prob[\mathsf{IDEAL}_{\mathcal{F}_{\text{mPSI-CA}},\mathcal{SIM}}(X,Y)] = Prob[Game_4]$ and

$$\begin{aligned}&|Prob[\mathsf{IDEAL}_{\mathcal{F}_{\text{mPSI-CA}},\mathcal{SIM}}(X,Y)] - Prob[\mathsf{REAL}_{\text{mPSI-CA},\mathcal{C}}(X,Y)]|\\ &= |Prob[Game_4] - Prob[Game_0]| \leq \Sigma_{i=1}^{4}|Prob[Game_i] - Prob[Game_{i-1}]|\\ &\leq \Sigma_{i=1}^{4}\epsilon_i(\kappa) = \rho(\kappa), \text{ where } \rho(\kappa) \text{ is a negligible function.}\end{aligned}$$

Therefore we have

$$\mathsf{IDEAL}_{\mathcal{F}_{\text{mPSI-CA}},\mathcal{SIM}}(X,Y) \equiv^c \mathsf{REAL}_{\text{mPSI-CA},\mathcal{C}}(X,Y).$$

2. B **and** Ar **are corrupted.** Let us consider $\mathcal{Z}$ as a distinguisher who controls $\mathcal{C}$, feeds the input of the honest party A, and also sees the output of B. Now we argue that $\mathcal{Z}$'s view in the real world ($\mathcal{C}$'s view +A's output) and its view in the ideal world ($\mathcal{C}$'s view + $\bar{A}$'s output) are indistinguishable. To prove that a series of games $Game_0, ..., Game_5$ is presented, where each $Game_{i+1}$ modifies $Game_i$ slightly such that $\mathcal{Z}$'s views in $Game_i$ and $Game_{i+1}$ remain indistinguishable, for $i = 0, .., 4$. Let us denote the probability that $\mathcal{Z}$ distinguishes the view of $Game_i$ from the view of real protocol by $Pr[Game_i]$. We consider S_i as simulator in $Game_i$.
$Game_0$: This game is same as real world protocol, where the simulator S_0 has full knowledge of A and interacts with $\mathcal{C}$. Hence,

$$Prob[\mathsf{REAL}_{\text{mPSI-CA},\mathcal{C}}(X,Y)] = Prob[Game_0].$$

$Game_1$: This game is same as $Game_0$ except that S_1 simulates π_1, instead of proving it. $\mathcal{Z}$'s views in $Game_0$ and $Game_1$ are indistinguishable because of zero-knowledge (simulatability) of the proof π_1. Therefore, there exists a negligible function $\epsilon_1(\kappa)$ such that

$$|Prob[Game_1] - Prob[Game_0]| \leq \epsilon_1(\kappa).$$

$Game_2$: $Game_1$ is same as $Game_2$ except that if the verification of the proof π_2 succeeds then the simulator S_2 runs the extractor algorithm for π_2 with $\mathcal{C}$ to extract the exponents r and $\{r_{y_1}, ..., r_{y_w}\}$. The simulator S_2 then extracts $y_j = \frac{d_{y_j}}{h^{r_{y_j}}}$ by extracting $d_{y_j} = y_j h^{r_{y_j}}$ from $\mathsf{dE}_{pk}(y_j)$ in R_2, h from $pk = epk_A \cdot epk_B$ and using the exponent r_{y_j} for $1 \leq j \leq w$. In this way S_2 extracts the private input set $Y = \{y_1, ..., y_w\}$ of B. The simulation soundness of the proof π_2 makes $\mathcal{Z}$'s views in $Game_1$ and $Game_2$ indistinguishable. Consequently,

$$|Prob[Game_2] - Prob[Game_1]| \leq \epsilon_2(\kappa), \text{ where } \epsilon_2(\kappa) \text{ is a negligible function.}$$

$Game_3$: Note that in this game the simulator S_3 has the knowledge of input set $X = \{x_1, ..., x_v\}$, secret key $esk_A = a_1$ of A and extracted set $Y = \{y_1, ..., y_w\}$ of B. This game is same as $Game_2$ except that

(a) if the verification of the proof π_4 succeeds then S_3 outputs $|X \cap Y|$ as the final output of A making use of the extracted set Y,

(b) if the verification of the proof π_4 does not succeed or $\mathcal{C}$ aborts in mPSI-CA protocol then the following cases arise:

$\diamond$ if $\mathcal{C}$ sends $\langle \{s_{x_i}\}_{i=1}^{v}, \{s_{y_j}\}_{j=1}^{w} \rangle$ to S_3 in dispute resolution protocol then S_3 does the following:

– for each $i = 1, ..., v$, computes $\frac{\bar{d}_{x_i}}{(\bar{c}_{x_i})^{a_1} s_{x_i}} = \hat{x}_i$ using $esk_A = a_1$;

– for each $j = 1, ..., w$, computes $\frac{\bar{d}_{y_j}}{(\bar{c}_{y_j})^{a_1} s_{y_j}} = \hat{y}_j$ using $esk_A = a_1$;

– outputs $|\{\hat{x}_1, ..., \hat{x}_v\} \in \cap \{\hat{y}_1, ..., \hat{y}_w\}|$ as the final output of A.

$\diamond$ if $\mathcal{C}$ aborts in dispute resolution protocol then S_3 outputs $\perp$ as the final output of A.

By the simulation soundness property of the proof π_4, $\mathcal{Z}$'s views in $Game_2$ and $Game_3$ are indistinguishable. Therefore, there exists a negligible function $\epsilon_3(\kappa)$ such that

$$|Prob[Game_3] - Prob[Game_2]| \leq \epsilon_3(\kappa).$$

$Game_4$: $Game_4$ is same as $Game_3$ except that S_4 does the following after extracting $Y = \{y_1, ...y_w\}, r$:

(a) computes $|X \cap Y|$,

(b) constructs a set $\overline{\overline{X}} = \{\bar{\bar{x}}_1, ..., \bar{\bar{x}}_v\}$ by including $|X \cap Y|$ many random elements of Y together with $v - |X \cap Y|$ many random elements chosen from $\mathbb{G}$.

(c) selects a random permutation $\hat{\psi} \in \Sigma_w, \beta_1, ..., \beta_w, \tau_1, ..., \tau_v \leftarrow \mathbb{Z}_q$ and computes

$$\mathsf{dE}_{pk}((g\bar{y}_j)^r) = \mathsf{dE}_{pk}((gy_{\hat{\psi}^{-1}(j)})^r)\mathcal{DEL}.\mathsf{Enc}(1, pk, \mathsf{par}, \beta_j), 1 \leq j \leq w$$

$$\text{and } (\mathsf{dE}_{pk}(g\bar{\bar{x}}_i)) \leftarrow \mathcal{DEL}.\mathsf{Enc}(g\bar{\bar{x}}_i, pk, \mathsf{par}, \tau_i), 1 \leq i \leq v,$$

(d) chooses $r', r_1, ..., r_w \leftarrow \mathbb{Z}_q$,

(e) computes $\langle \{\mathsf{dE}_{pk}((g\bar{\bar{x}}_i)^{rr'}) = (\bar{c}_{\bar{x}_i}, \bar{d}_{\bar{x}_i}), \mathsf{eE}_{epk_B}((\bar{c}_{\bar{x}_i})^{a_1})\}_{i=1}^{v}, \{\mathsf{dE}_{pk}((g\bar{y}_j)^{rr'}) = (\bar{c}_{y_j}, \bar{d}_{y_j}), \mathsf{vE}_{vpk_{Ar}}(g^{r_j})\}_{j=1}^{w}, \rangle$,

(f) computes $\{\bar{u}_{y_j} = (\bar{c}_{y_j})^{a_1} \cdot g^{r_j}\}_{j=1}^{w}$,

(g) sends $\langle \{\mathsf{dE}_{pk}((g\bar{\bar{x}}_i)^{rr'}), \mathsf{eE}_{epk_B}((\bar{c}_{\bar{x}_i})^{a_1})\}_{i=1}^{v}, \{\mathsf{dE}_{pk}((g\bar{y}_j)^{rr'}), \mathsf{dE}_{pk}((g\bar{y}_j)^{r}), \mathsf{vE}_{vpk_{Ar}}(g^{r_j}), \bar{u}_{y_j}\}_{j=1}^{w} \rangle$ as $\langle \{\mathsf{dE}_{pk}((g\bar{x}_i)^{rr'}), \mathsf{eE}_{epk_B}((\bar{c}_{x_i})^{a_1})\}_{i=1}^{v}, \{\mathsf{dE}_{pk}((g\bar{y}_j)^{rr'}), \mathsf{dE}_{pk}((g\bar{y}_j)^{r}), \mathsf{vE}_{vpk_{Ar}}(g^{r_j}), \bar{u}_{y_j}\}_{j=1}^{w} \rangle$ to $\mathcal{C}$ and simulates the proofs $\pi_3, \hat{\pi}_3$.

As the associated encryption schemes $\mathcal{DEL}$ and $\mathcal{EL}$ are semantically secure the tuple $\langle \{\mathsf{dE}_{pk}((g\bar{x}_i)^{rr'}), \mathsf{eE}_{epk_B}((\bar{c}_{x_i})^{a_1})\}_{i=1}^{v}, \{\mathsf{dE}_{pk}((g\bar{y}_j)^{rr'}), \mathsf{dE}_{pk}((g\bar{y}_j)^{r}), \mathsf{vE}_{vpk_{Ar}}(g^{r_j}), \bar{u}_{y_j}\}_{j=1}^{w} \rangle$ is identically distributed in $Game_4$ and $Game_3$. Indistinguishability of the tuple and the zero-knowledge (simulatability) of $\pi_3, \hat{\pi}_3$ makes the views of $\mathcal{Z}$'s in $Game_3$ and $Game_4$ indistinguishable. Hence,

$$|Prob[Game_4] - Prob[Game_3]| \leq \epsilon_4(\kappa), \text{ where } \epsilon_4(\kappa) \text{ is a negligible function.}$$

$Game_5$: This game is same as $Game_4$ except that during the setup phase S_5 chooses $a_1 \leftarrow\!\!\!\!\circ\ \mathbb{Z}_q$ and in *Step* 5 simulates π_5, instead of proving it. By the zero-knowledge (simulatability) of π_5 the views of $\mathcal{Z}$'s in $Game_4$ and $Game_5$ are indistinguishable. Consequently, there exists a negligible function $\epsilon_5(\kappa)$ such that

$$|Prob[Game_5] - Prob[Game_4]| \leq \epsilon_5(\kappa).$$

Let us construct the ideal world adversary $\mathcal{SIM}$ that uses $\mathcal{C}$ as subroutine, simulates the honest party A and controls $\bar{B}, \bar{A}r$ and incorporates all steps from $Game_5$.

(i) $\mathcal{SIM}$ first plays the role of trusted party by generating the global parameter $\mathsf{gpar} = (\mathsf{par}, \widehat{g}, \mathcal{H})$, where $\mathsf{par} = (p, q, g)$. $\mathcal{SIM}$ then plays the role of honest party A by choosing $\bar{a}_1 \leftarrow\!\!\!\!\circ\ \mathbb{Z}_q$ and publishing $g^{\bar{a}_1}$ as the public key $epk_A = y_A$. $\mathcal{SIM}$ also acts as certifying authority to obtain public keys epk_B, vpk_{Ar} of B, Ar. $\mathcal{SIM}$ then invokes $\mathcal{C}$.

(ii) $\mathcal{SIM}$ chooses $\breve{x}_1, ..., \breve{x}_v$ randomly from $\mathbb{G}$ and sends $\{\mathsf{dE}_{pk}(\breve{x}_i)\}_{i=1}^{v}$ as $\{\mathsf{dE}_{pk}(x_i)\}_{i=1}^{v}$ to $\mathcal{C}$ and simulates the proof π_1.

(iii) On receiving $R_2 = \big\langle \{\mathsf{dE}_{pk}(y_j), \mathsf{dE}_{pk}((gy_j)^r)\}_{j=1}^{w}, \{\mathsf{dE}_{pk}(\bar{x}_i), \mathsf{dE}_{pk}((g\bar{x}_i)^r)\}_{i=1}^{v}, \pi_2, \hat{\pi}_3 \big\rangle$ from $\mathcal{C}$, $\mathcal{SIM}$ verifies the proof π_2. If the verification does not succeed, then $\mathcal{SIM}$ instructs $\bar{B}$ to send $\perp$ to T and terminates the execution. Otherwise, $\mathcal{SIM}$ runs the extractor algorithm for π_2 with $\mathcal{C}$ to extract the exponents r and $\{r_{y_1}, ..., r_{y_w}\}$. Utilizing $\{r_{y_1}, ..., r_{y_w}\}$, $\mathcal{SIM}$ extracts $Y = \{y_1, ..., y_w\}$ by extracting $\{d_{y_j} = y_j h^{r_{y_j}}\}_{j=1}^{w}$ from $\{\mathsf{dE}_{pk}(y_j)\}_{j=1}^{w}$ in R_2, h from $pk = epk_A \cdot epk_B$. $\mathcal{SIM}$ then instructs $\bar{B}$ to send Y to T, $\bar{A}r$ to send $b_B = \circ$ to T and receives $|X \cap Y|$ from T.

(iv) $\mathcal{SIM}$ constructs a set $\overline{\overline{X}} = \{\bar{\bar{x}}_1, ..., \bar{\bar{x}}_v\}$ by including $|X \cap Y|$ many random elements of Y together with $v - |X \cap Y|$ many random elements chosen from $\mathbb{G}$. $\mathcal{SIM}$ then does the following:

– selects a random permutation $\widehat{\psi} \in \Sigma_w$, $\beta_1, ..., \beta_w, \tau_1, ..., \tau_v \leftarrow\!\!\!\!\circ\ \mathbb{Z}_q$ and computes

$$\mathsf{dE}_{pk}((g\bar{y}_j)^r) = \mathsf{dE}_{pk}((gy_{\widehat{\psi}^{-1}(j)})^r)\mathcal{DEL}.\mathsf{Enc}(1, pk, \mathsf{par}, \beta_j), 1 \leq j \leq w$$

$$\text{and } (\mathsf{dE}_{pk}(g\bar{\bar{x}}_i)) \leftarrow \mathcal{DEL}.\mathsf{Enc}(g\bar{\bar{x}}_i, pk, \mathsf{par}, \tau_i), 1 \leq i \leq v$$

– chooses $r', r_1, ..., r_w \leftarrow\!\!\!\!\circ\ \mathbb{Z}_q$,
– computes $\big\langle \{\mathsf{dE}_{pk}((g\bar{\bar{x}}_i)^{rr'}) = (\bar{c}_{\bar{x}_i}, \bar{d}_{\bar{x}_i}), \mathsf{eE}_{epk_B}((\bar{c}_{\bar{x}_i})^{a_1})\}_{i=1}^{v}, \{\mathsf{dE}_{pk}((g\bar{y}_j)^{rr'}) = (\bar{c}_{y_j}, \bar{d}_{y_j}), \mathsf{vE}_{vpk_{Ar}}(g^{r_j})\}_{j=1}^{w}, \big\rangle$,
– computes $\{\bar{u}_{y_j} = (\bar{c}_{y_j})^{a_1} \cdot g^{r_j}\}_{j=1}^{w}$,
– sends $\big\langle \{\mathsf{dE}_{pk}((g\bar{\bar{x}}_i)^{rr'}), \mathsf{eE}_{epk_B}((\bar{c}_{\bar{x}_i})^{a_1})\}_{i=1}^{v}, \{\mathsf{dE}_{pk}((g\bar{y}_j)^{rr'}), \mathsf{dE}_{pk}((g\bar{y}_j)^r), \mathsf{vE}_{vpk_{Ar}}(g^{r_j}), \bar{u}_{y_j}\}_{j=1}^{w} \big\rangle$ as $\big\langle \{\mathsf{dE}_{pk}((g\bar{x}_i)^{rr'}), \mathsf{eE}_{epk_B}((\bar{c}_{x_i})^{a_1})\}_{i=1}^{v}, \{\mathsf{dE}_{pk}((g\bar{y}_j)^{rr'}), \mathsf{dE}_{pk}((g\bar{y}_j)^r), \mathsf{vE}_{vpk_{Ar}}(g^{r_j}), \bar{u}_{y_j}\}_{j=1}^{w} \big\rangle$ to $\mathcal{C}$ and simulates the proofs $\pi_3, \hat{\pi}_3$.

$\mathcal{SIM}$ executes following steps according to $\mathcal{C}$'s reply.

(v) If $\mathcal{C}$ instructs both B and Ar to abort, then $\mathcal{SIM}$ instructs $\bar{A}r$ to send $b_A = \perp$ to T. Then outputs whatever $\mathcal{C}$ outputs and terminates.

(vi) If $\mathcal{C}$ instructs B to send $\left\langle \{s_{x_i}\}_{i=1}^{v}, \{s_{y_j}\}_{j=1}^{w} \right\rangle$, then $\mathcal{SIM}$ checks the validity of the proof π_4. If the verification succeeds then $\mathcal{SIM}$ instructs $\bar{Ar}$ to send $b_A = \circ$ to T and sends $\{g^{r_j}\}_{j=1}^{w}$ to $\mathcal{C}$ and simulates the proof π_5. If verification fails or $\mathcal{C}$ instructs B to abort in mPSI-CA protocol then the following cases arise:

$\diamond$ if $\mathcal{C}$ instructs Ar to send $\left\langle \{s_{x_i}\}_{i=1}^{v}, \{s_{y_j}\}_{j=1}^{w} \right\rangle$ in dispute resolution protocol then $\mathcal{SIM}$ does the following:

– for each $i = 1, ..., v$, computes $\frac{\bar{d}_{\bar{x}_i}}{(\bar{c}_{\bar{x}_i})^{\bar{a}_1 s_{x_i}}} = \tilde{x}_i$;

– for each $j = 1, ..., w$, computes $\frac{\bar{d}_{y_j}}{(\bar{c}_{y_j})^{\bar{a}_1 s_{y_j}}} = \tilde{y}_j$;

– instructs $\bar{Ar}$ to send $b_A = |\{\tilde{x}_1, ..., \tilde{x}_v\} \cap \{\tilde{y}_1, ..., \tilde{y}_w\}|$ to T. $\mathcal{SIM}$ then outputs whatever $\mathcal{C}$ outputs and terminates.

$\diamond$ if $\mathcal{C}$ instructs Ar to abort in dispute resolution protocol then $\mathcal{SIM}$ instructs $\bar{Ar}$ to send $b_A = \perp$ to T. $\mathcal{SIM}$ then outputs whatever $\mathcal{C}$ outputs and terminates.

Therefore, the ideal world adversary $\mathcal{SIM}$ provides $\mathcal{C}$ the same simulation as the simulator S_5 as in $Game_5$. Hence $Prob[\mathsf{IDEAL}_{\mathcal{F}_{\mathsf{mPSI\text{-}CA}},\mathcal{SIM}}(X,Y)] = Prob[Game_5]$ and

$$
\begin{aligned}
&|Prob[\mathsf{IDEAL}_{\mathcal{F}_{\mathsf{mPSI\text{-}CA}},\mathcal{SIM}}(X,Y)] - Prob[\mathsf{REAL}_{\mathsf{mPSI\text{-}CA},\mathcal{C}}(X,Y)]| \\
&= |Prob[Game_5] - Prob[Game_0]| \leq \Sigma_{i=1}^{5} |Prob[Game_i] - Prob[Game_{i-1}]| \\
&\leq \Sigma_{i=1}^{5} \epsilon_i(\kappa) = \rho(\kappa), \text{ where } \rho(\kappa) \text{ is a negligible function.}
\end{aligned}
$$

Thus we have

$$\mathsf{IDEAL}_{\mathcal{F}_{\mathsf{mPSI\text{-}CA}},\mathcal{SIM}}(X,Y) \equiv^{c} \mathsf{REAL}_{\mathsf{mPSI\text{-}CA},\mathcal{C}}(X,Y).$$

3. A **and** B **are corrupted.** This case is trivial as $\mathcal{C}$ has full knowledge of X and Y and the encryption scheme used by Ar is semantically secure. Therefore a simulator can always be constructed.

• Case II (When the adversary $\mathcal{C}$ corrupts only one party).

If only Ar is corrupted then Ar is not involved in the protocol as A and B are honest. Thus it is trivial to construct a simulator in this case. If only A or B is corrupted then the simulator can be constructed as steps (i)-(iv) of the case when A and Ar are corrupted or steps (i)-(iv) of the case when B and Ar are corrupted. The only change is that $\bar{Ar}$ is honest and always sends $\circ$ to T in these cases.

Table 2 : Complexity of our mPSI protocol

	Party A	Party B	Arbiter Ar	Total
Exp	$21v + 30w + 12$	$30v + 46w + 13$	$4v + 6w + 2$	$55v + 82w + 27$
GE	$13v + 22w + 8$	$9v + 9w + 11$	1	$22v + 31w + 20$
Inv	$v + w$	$2v + 2w$		$3v + 3w$
H	w		w	$2w$

GE=number of group elements, Exp=number of exponentiations,
Inv=number of inversions, H=number of hash query.

5 Efficiency

The computation overhead of our mPSI and mPSI-CA is measured by modular exponentiation, modular inversion and hash function evaluation. On the other hand, the number of group elements transmitted publicly by the users in our mPSI and mPSI-CA incurs the communication cost. The complexities of our mPSI and mPSI-CA are exhibited in Table 2 and Table 3 respectively, where $\pi_1, \pi_2, \pi_3, \pi_4, \pi_5, \hat{\pi}_2, \hat{\pi}_3$ are associated interactive zero-knowledge proofs. As far

Table 3 : Complexity of our mPSI-CA protocol

	Party A	Party B	Arbiter Ar	Total
Exp	$29v + 39w + 4$	$30v + 42w + 6$	$4v + 6w + 2$	$63v + 87w + 12$
GE	$14v + 26w + 5$	$24v + 24w + 8$	$v + 2w + 1$	$39v + 52w + 14$
Inv	$v + w$	$2v + 2w$	w	$3v + 4w$
H	w		w	$2w$

GE=number of group elements, Exp=number of exponentiations,
Inv=number of inversions, H=number of hash query.

as we are aware of, till now the most efficient fair mPSI protocols are [14, 15]. We compare our mPSI protocol with the construction of [14, 15] in Table 4.

Table 4 : Comparison summary in terms of GE, Exp, fairness, optimistic and order of underlying group

Protocol	GE	Exp	Fairness	Optimistic	Group order
[15]	$7w + 53v + 2wv + 5$	$11w + 96v + 12wv$	yes	yes	prime
[14]	$77(w + v)$	$156(w + v)$	yes	yes	composite
our mPSI	$22v + 31w + 20$	$55v + 82w + 27$	yes	yes	prime

6 Conclusion

We have designed a fair mPSI protocol with *linear* complexity over *prime* order group in standard model. The security of this protocol is achieved in presence of malicious parties under DDH assumption. Our mPSI achieves fairness in the optimistic way i.e., by using an off-line semi trusted third party (arbiter). Particularly, our mPSI is more efficient than existing mPSI protocols. Further, we have proposed that utilizing two random permutations our mPSI can be extended to mPSI-CA, where the security properties remain invariant. To the best of our knowledge, our mPSI-CA is the *first* mPSI-CA achieving *linear complexity*. Note that the communication cost of each of our constructions can be further reduced by transforming each of the interactive zero-knowledge proofs into non-interactive using Fiat-Shamir technique [18]. But in that case the security model of our protocol will be changed to the random oracle model (ROM).

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] M. Bellare and O. Goldreich. On defining proofs of knowledge. In *Advances in CryptologyCRYPTO92*, pages 390–420. Springer, 1993.

[3] D. Boneh. The decision diffie-hellman problem. In *Algorithmic number theory*, pages 48–63. Springer, 1998.

[4] F. Brandt. Efficient cryptographic protocol design based on distributed el gamal encryption. In *Information Security and Cryptology-ICISC 2005*, pages 32–47. Springer, 2006.

[5] J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *Advances in Cryptology-CRYPTO 2003*, pages 126–144. Springer, 2003.

[6] J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *Advances in CryptologyCRYPTO'97*, pages 410–424. Springer, 1997.

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

[8] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in CryptologyCRYPTO'98*, pages 13–25. Springer, 1998.

[9] I. Damgård and M. Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *Public Key Cryptography*, pages 119–136. Springer, 2001.

[10] 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.

[11] 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.

[12] E. De Cristofaro and G. Tsudik. Practical private set intersection protocols with linear complexity. In *Financial Cryptography and Data Security*, pages 143–159. 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] S. K. Debnath and R. Dutta. A fair and efficient mutual private set intersection protocol from a two-way oblivious pseudorandom function. In *Information Security and Cryptology-ICISC 2014*, pages 343–359. Springer, 2014.

[15] C. Dong, L. Chen, J. Camenisch, and G. Russello. Fair private set intersection with a semi-trusted arbiter. In *Data and Applications Security and Privacy XXVII*, pages 128–144. Springer, 2013.

[16] 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.

[17] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology*, pages 10–18. Springer, 1985.

[18] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in CryptologyCRYPTO86*, pages 186–194. Springer, 1987.

[19] 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.

[20] J. Furukawa. Efficient and verifiable shuffling and shuffle-decryption. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 88(1):172–188, 2005.

[21] C. Hazay. Oblivious polynomial evaluation and secure set-intersection from algebraic prfs. *IACR Cryptology ePrint Archive*, 2015:4, 2015.

[22] C. Hazay and Y. Lindell. Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. In *Theory of Cryptography*, pages 155–175. Springer, 2008.

[23] 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.

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

[25] 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.

[26] S. Jarecki and X. Liu. Efficient oblivious pseudorandom function with applications to adaptive ot and secure computation of set intersection. In *Theory of Cryptography*, pages 577–594. Springer, 2009.

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

[28] M. Kim, H. T. Lee, and J. H. Cheon. Mutual private set intersection with linear complexity. In *Information Security Applications*, pages 219–231. Springer, 2012.

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

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