Fast and Private Computation of Cardinality of Set Intersection and Union

Emiliano De Cristofaro[1], Paolo Gasti[2], and Gene Tsudik[3]

[1] Palo Alto Research Center – `Emiliano.DeCristofaro@parc.com`
[2] New York Institute of Technology – `pgasti@nyit.edu`
[3] University of California Irvine – `gts@ics.uci.edu`

Abstract. In many everyday scenarios, sensitive information must be shared between parties without complete mutual trust. Private set operations are particularly useful to enable sharing information with privacy, as they allow two or more parties to jointly compute operations on their sets (e.g., intersection, union, etc.), such that only the minimum required amount of information is disclosed. In the last few years, the research community has proposed a number of secure and efficient techniques for Private Set Intersection (PSI), however, somewhat less explored is the problem of computing the *magnitude*, rather than the contents, of the intersection – we denote this problem as Private Set Intersection Cardinality (PSI-CA). This paper explores a few PSI-CA variations and constructs several protocols that are more efficient than the state-of-the-art.

1 Introduction

Proliferation of, and growing reliance on, electronic information generate an increasing amount of sensitive data stored and processed in the cyberspace. Consequently, there is a compelling need for efficient cryptographic techniques that allow sharing information while protecting privacy. Among these, Private Set Intersection (PSI) [14,28,17,25,18,26,11,10,21], and Private Set Union (PSU) [28,18,20,15,35] have recently attracted a lot of attention from the research community. In particular, PSI allows one party (client) to compute the intersection of its set with that of another party (server), such that: (i) server does not learn client input, and (ii) client learns no information about server input, beyond the intersection. Efficient PSI protocols have been used as building blocks for many privacy-oriented applications, e.g., collaborative botnet detection [31], denial-of-service identification [2], on-line gaming [7], intelligence-community systems [23], location sharing [32], just to cite a few.

Nonetheless, in certain information-sharing settings, PSI and PSU functionalities offer very limited privacy to server. Consider the following scenario where, after running PSI, the set intersection learned by client corresponds to entire server input: server privacy is actually non-existent, while client's is fully preserved. This illustrates the need for server to enforce a policy, based on the cardinality of set intersection/union, that governs whether it is willing to take part in PSI or PSU protocols. (We explore this intuition in Section 6.)

This paper investigates Private Set Intersection Cardinality (PSI-CA) and Private Set Union Cardinality (PSU-CA). These functionalities are appealing in scenarios where client is only allowed to learn the *magnitude* – rather than the *content* – of set intersection/union. For instance, PSI-CA is useful in social networking, e.g., when two parties want to privately determine the number of common connections (or interests) in order to decide whether or not to become friends. Moreover, PSI-CA is useful to privately compare equal-size low-entropy vectors, e.g., to realize private computation of Hamming Distance between two strings on an arbitrarily large alphabet: two parties may use PSI-CA, by treating each symbol, together with its position in the string, as a unique set element, such that client privately learns the number of elements (symbols) in common (thereby also obtaining the Hamming Distance). Other relevant applications of PSI-CA include role-based association rule mining [27], affiliation-hiding authentication [3], as well as to estimating the similarity of sample sets [6]. Finally, efficient PSI-CA protocols are becoming instrumental to privacy-preserving genomic tests, as recently showed in [4].

1.1 Contributions

This paper focuses on PSI-CA – a cryptographic primitive, involving server (on input of a private set $\mathcal{S}$) and client (on input of a private set $\mathcal{C}$), that results in client outputting $|\mathcal{S} \cap \mathcal{C}|$. Computation of PSI-CA naturally implies that of PSU-CA, since $|\mathcal{S}|, |\mathcal{C}|$ are always mutually disclosed and $|\mathcal{S} \cup \mathcal{C}| = |\mathcal{S}| + |\mathcal{C}| - |\mathcal{S} \cap \mathcal{C}|$.

Although prior work has yielded some PSI-CA techniques (see Section 2), a number of open problems still remain to be addressed. This paper presents the following contributions:

1. We present a very efficient PSI-CA protocol that incurs computational and communication complexities linear in the set sizes. Our protocol is secure under the DDH assumption in the Random Oracle Model (ROM) against semi-honest adversaries. This protocol is a very close variant of the protocol of Agrawal, Evfimievski, and Srikant [1], and our merit is really a security analysis of this modification rather than the protocol itself.
2. We introduce the concept of *Authorized* PSI-CA (APSI-CA), whereby client input must be pre-authorized by an off-line mutually-trusted authority, and present an appropriate protocol extension with linear complexities (as opposed to quadratic in related prior techniques).
3. We show how to combine PSI-CA with PSI such that server can decide whether to allow client to obtain the set intersection according to its policy, based on the size of the intersection itself (privately obtained using PSI-CA). This first-of-a-kind approach is very efficient, as it requires only *one* additional message on top of PSI-CA protocol.

Paper organization. Next section reviews related work. After preliminaries in Section 3, Section 4 presents our PSI-CA protocol. Then, Section 5 constructs a variant for APSI-CA, and finally, Section 6 sketches a three-round policy-based PSI variant. The paper concludes in Section 7.

2 Related Work

2.1 (Authorized) Private Set Intersection and Union

Agrawal, Evfiviemski, and Srikant [1] introduce a Private Set Intersection (PSI) construction based on commutative encryption.[4] The protocol has linear complexity – that is, assuming that server and client sets contain w and v items, respectively, computation and communication complexity amounts to $O(w+v)$. [1] also presents a variant that only discloses the size of the intersection – we review it in Section 2.2 below.

The work in [14] propose a few protocols for Private Set Intersection (PSI) based on Oblivious Polynomial Evaluations (OPE-s) and additively homomorphic encryption (e.g., Paillier [33]). The intuition is to represent a set as a polynomial and its elements – as the polynomial's roots. Client encrypts the coefficients, that are then evaluated homomorphically by server. As a result, client learns the intersection and nothing else. Client's computation complexity amounts to $O(w + v)$, and server's to $O(wv)$ exponentiations. [14] proposes techniques to asymptotically reduce server workload to $O(w \log\log v)$, by using Horner's rule and balanced bucket allocation. [18] obtains similar complexities while also offering PSU techniques. Whereas, [28] extends OPE-s to more than two players, all learning the intersection/union, with quadratic computational and linear communication complexities. Additional PSU constructs appear in [20,15,35].

Other PSI constructs, such as [17,25], rely on Oblivious Pseudo-Random Functions (OPRF-s) and reduce computation overhead to a *linear* number of exponentiations. Recent results in the Random Oracle Model (ROM) have led to very efficient PSI protocols, also with linear complexities, while using much more efficient cryptographic tools. They replace OPRFs with unpredictable functions [26] and blind signatures [11], with security under *One-More-DH* and *One-More-RSA* assumptions [5], respectively. Finally, [10] achieves linear communication and computational complexities, using short exponents, with security in the malicious model, while [21] shows a construction in the semi-honest model based on garbled circuits [38] which, leveraging so-called Oblivious Transfer Extension [24], scales relatively gracefully for very large security parameters.

Authorization of client input in PSI has been first investigated in [8] and [9]. Authorized Private Set Intersection (APSI) is later formalized in [11] and [10] that construct efficient techniques with linear complexity in the presence of, respectively, semi-honest and malicious adversaries. Finally, [36] proposes Policy-

[4] It is quite interesting to observe that several PSI papers (e.g., [14,26]) erroneously cite the work by Evfiviemski, Gerke, and Srikant [12] as the work introducing commutative-encryption based PSI, which is, in fact, [1]. Also, observe that protocols in [1] are essentially the same as those sketched earlier, in [22], although the latter provided no security analysis.

Enhanced PSI, allowing two parties to privately share information while enforcing complex policies. In this model, both parties' sets must be authorized, and both parties obtain the intersection.

2.2 Private Set Intersection Cardinality

Prior work yielded several PSI-CA protocols:

- Agrawal, Evfimievski, and Srikant [1] present an adaptation of their PSI protocol to PSI-CA, also secure under the DDH assumption in the presence of semi-honest adversaries. Their construction is actually similar to ours (presented in Figure 1), although we also present two extensions.
- The PSI protocol by Freedman, Nissim, and Pinkas [14] can be extended to PSI-CA with the same complexity, i.e., $O(w \log\log v)$ computation and $O(w+v)$ communication.
- Hohenberger and Weis [19] present a PSI-CA construction, also based on [14], and with similar (sub-quadratic) complexities.
- Kissner and Song [28] proposes a PSI-CA protocol for multiple ($n \geq 2$) parties, incurring $O(n^2 \cdot v)$ communication and $O(v^2)$ computational overhead.
- Vaidya and Clifton [37] construct a multi-party PSI-CA protocol, based on commutative one-way hash functions [30] and Pohlig-Hellman encryption [34]. It incurs n rounds, and involves $O(n^2 \cdot v)$ communication and $O(vn)$ computational overhead.
- Camenisch and Zaverucha [8] present an APSI variant (private intersection of certified sets) that computes the cardinality of (certified) set intersection and incurs *quadratic* communication and computation complexity.

3 Preliminaries

This section defines PSI-CA/PSU-CA functionalities, along with their privacy requirements, and introduces computational assumptions.

DDH Assumption. Let $\mathbb{G}$ be a cyclic group and g be its generator. We assume that bit-length of group size is l. The DDH problem is hard in $\mathbb{G}$ if, for every efficient algorithm $\mathcal{A}$, the following probability is a negligible function of κ:

$$\left|\Pr[x, y \leftarrow \{0,1\}^l : \mathcal{A}(g, g^x, g^y, g^{xy}) = 1] - \Pr[x, y, z \leftarrow \{0,1\}^l : \mathcal{A}(g, g^x, g^y, g^z) = 1]\right|$$

Definition 1 (Private Set Union Cardinality (PSU-CA)). *A protocol involving server, on input a set of w items $\mathcal{S} = \{s_1, \ldots, s_w\}$, and client, on input a set of v items $\mathcal{C} = \{c_1, \cdots, c_v\}$. It results in the latter outputting $|\mathcal{U}|$, where: $\mathcal{U} = \mathcal{S} \cup \mathcal{C}$.*

Definition 2 (Private Set Intersection Cardinality (PSI-CA)). *A protocol involving server, on input a set of w items $\mathcal{S} = \{s_1, \ldots, s_w\}$, and client, on input a set of v items $\mathcal{C} = \{c_1, \cdots, c_v\}$. It results in the latter outputting $|\mathcal{I}|$, where: $\mathcal{I} = \mathcal{S} \cap \mathcal{C}$.*

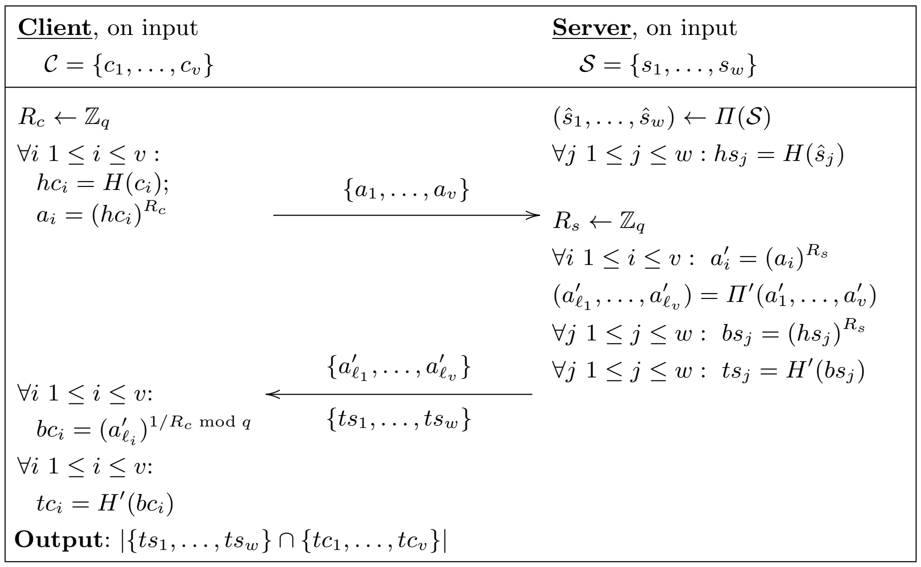
Client, on input
$\mathcal{C} = \{c_1, \ldots, c_v\}$
Server, on input
$\mathcal{S} = \{s_1, \ldots, s_w\}$
$R_c \leftarrow \mathbb{Z}_q$
$\forall i \; 1 \leq i \leq v :$
$hc_i = H(c_i);$
$a_i = (hc_i)^{R_c}$
$(\hat{s}_1, \ldots, \hat{s}_w) \leftarrow \Pi(\mathcal{S})$
$\forall j \; 1 \leq j \leq w : hs_j = H(\hat{s}_j)$
$\{a_1, \ldots, a_v\}$
$R_s \leftarrow \mathbb{Z}_q$
$\forall i \; 1 \leq i \leq v : \; a'_i = (a_i)^{R_s}$
$(a'_{\ell_1}, \ldots, a'_{\ell_v}) = \Pi'(a'_1, \ldots, a'_v)$
$\forall j \; 1 \leq j \leq w : \; bs_j = (hs_j)^{R_s}$
$\forall j \; 1 \leq j \leq w : \; ts_j = H'(bs_j)$
$\{a'_{\ell_1}, \ldots, a'_{\ell_v}\}$
$\{ts_1, \ldots, ts_w\}$
$\forall i \; 1 \leq i \leq v$:
$bc_i = (a'_{\ell_i})^{1/R_c \bmod q}$
$\forall i \; 1 \leq i \leq v$:
$tc_i = H'(bc_i)$
Output: $|\{ts_1, \ldots, ts_w\} \cap \{tc_1, \ldots, tc_v\}|$

Figure 1: Proposed PSI-CA Protocol. All computation is mod p. Π and Π' are random permutations.

Informally, both PSI-CA and PSU-CA entail the following privacy requirements:

- *Server Privacy.* Client learns no information beyond: (1) cardinality of set intersection/union and (2) upper bound on the size of $\mathcal{S}$.
- *Client Privacy.* No information is leaked about client set $\mathcal{C}$, except an upper bound on its size.
- *Unlinkability.* Neither party can determine if any two instances of the protocol are related, i.e., executed on the same input by client or server, unless this can be inferred from the actual protocol output.

Remark: As mentioned earlier in the paper, for any $\mathcal{C}$ and $\mathcal{S}$, the size of $\mathcal{C} \cup \mathcal{S}$ can be computed as $|\mathcal{C}| + |\mathcal{S}| - |\mathcal{C} \cap \mathcal{S}|$. Thus, privately computing cardinality of the *intersection* of $\mathcal{C}$ and $\mathcal{S}$ allows one to privately compute the cardinality of their *union* as well. Consequently, the rest of the paper only focuses on PSI-CA.

4 New PSI-CA and PSU-CA

This section presents our PSI-CA construction, secure in the presence of semi-honest adversaries in the Random Oracle Model (ROM). We outline it in Figure 1. Protocol executes on common input of two primes p, q (where $q|p-1$), a generator g of a subgroup of size q, and two hash functions (modeled as random oracles), $H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ and $H' : \{0,1\}^* \rightarrow \{0,1\}^\kappa$, given the security parameter κ. (Notation $a \leftarrow A$ denotes that a is chosen uniformly at random from A).

Intuition. First, client masks its set items (c_i-s) with a random exponent (R_c) and sends resulting values (a_i-s) to server, which "blindly" exponentiates them with its own random value R_s. Server shuffles the resulting values (a'_i-s) and sends them to client. Then, server sends client the output of a one-way function, $H'(\cdot)$, computed over the exponentiations of server's items (s_j-s) to randomness R_s. Finally, client tries to match one-way function outputs received from server with one-way function outputs computed over the shuffled (a'_i-s) values, stripped of the initial randomness R_c. *Client* learns the set intersection cardinality (and nothing else) by counting the number of such matches. As showed below, unless they correspond to items in the intersection, one-way function outputs received from server cannot be used by client to learn related items in server's set (under the DDH assumption). Also, client does not learn *which* items are in the intersection as the matching occurs using *shuffled* a'_i values.

Complexity. Protocol complexity is *linear* in the sizes of the two sets. Let $|\mathcal{S}| = w$ and $|\mathcal{C}| = v$. Client performs $2(v+1)$ exponentiations with short, i.e., $|q|$-bit, exponents modulo $|p|$-bit and v modular multiplications. Server computes $(v+w)$ modular exponentiations with short exponents and w modular multiplications. In practice, one can select $|p| = 1024$ or $|p| = 2048$, and $|q| = 160$ or $|q| = 224$. Communication overhead amounts to $2(v+1)$ $|p|$-bit and w κ-bit values.

Semi-Honest Participants. We start with security in the semi-honest model. Note that the term *adversary* refers to insiders, i.e., protocol participants. Outside adversaries are not considered, since their actions can be mitigated via standard network security techniques.

Definition 3 (Correctness). *If both parties are honest, at the end of the protocol, executed on inputs $((\mathcal{S}, v), (\mathcal{C}, w))$, server outputs $\perp$, and client outputs $(|\mathcal{S} \cap \mathcal{C}|)$.*

The following client and server privacy definitions follow from those in related work [14,13,17]. In particular, as formalized in [16] (Sec. 7.2.2), in case of semi-honest parties, the traditional "real-versus-ideal" definition framework is *equivalent* to a much simpler framework that extends the formulation of honest-verifier zero-knowledge. Informally, a protocol privately computes certain functionality if whatever can be obtained from one party's view of a protocol execution can be obtained from input and output of that party. In other words, the view of a semi-honest party (including $\mathcal{C}$ or $\mathcal{S}$, all messages received during execution, and the outcome of that party's internal coin tosses), on each possible input $(\mathcal{C}, \mathcal{S})$, can be efficiently simulated considering only that party's own input and output.

Definition 4 (Client Privacy). *Let $\mathsf{View}_S(\mathcal{C}, \mathcal{S})$ be a random variable representing server's view during execution of PSI-CA with inputs $\mathcal{C}, \mathcal{S}$. There exists a* PPT *algorithm S^* such that:*

$$\{S^*(\mathcal{S}, |\mathcal{S} \cap \mathcal{C}|)\}_{(\mathcal{C},\mathcal{S})} \overset{c}{\equiv} \{\mathsf{View}_S(\mathcal{C}, \mathcal{S})\}_{(\mathcal{C},\mathcal{S})}$$

Definition 5 (Server Privacy). *Let* $\mathsf{View}_C(\mathcal{C},\mathcal{S})$ *be a random variable representing client's view during execution of PSI-CA with inputs* $\mathcal{C},\mathcal{S}$*. There exists a* PPT *algorithm* C^* *such that:*

$$\{C^*(\mathcal{C},|\mathcal{S}\cap\mathcal{C}|)\}_{(\mathcal{C},\mathcal{S})} \overset{c}{\equiv} \{\mathsf{View}_C(\mathcal{C},\mathcal{S})\}_{(\mathcal{C},\mathcal{S})}$$

In other words, on each possible pair of inputs $(\mathcal{C},\mathcal{S})$, client's view can be efficiently simulated by C^* on input: $\mathcal{C}$ and $|\mathcal{S}\cap\mathcal{C}|$ (as well as v,w). Thus, as in [16], we claim that the two distributions implicitly defined above are computationally indistinguishable. (Notation "$\overset{c}{\equiv}$" indicates computational indistinguishability.)

We claim that the protocol in Figure 1 is correct under Definition 3 and secure under Definitions 4 and 5 above. Proof of such claims is provided next.

4.1 Proofs

Correctness. For any c_i held by client and s_j held by server, if $c_i = s_j$, hence, $hc_i = hs_j$, we obtain:

$$tc_{\ell_i} = H'(bc_{\ell_i}) = H'({a_{\ell_i}}^{(1/R_c)}) = H'({hc_i}^{R_s}) = H'({hs_j}^{R_s}) = H'(bs_j) = ts_j$$

Hence, client learns set intersection cardinality by counting the number of matching pairs (tc_{ℓ_i}, ts_j). □

Client Privacy. We claim that the views of server – i.e., $\mathcal{S}$ and $a_i = H(c_i)^{R_c}$ for $i = 1,\ldots,v$ where H is modeled as a random oracle – is indistinguishable from $r_1,\ldots,r_v$ with $r_i \leftarrow \mathbb{Z}_p$. Therefore it is possible to construct a PPT algorithm S^* such that $\{S^*(\mathcal{S},|\mathcal{S}\cap\mathcal{C}|)\}_{(\mathcal{C},\mathcal{S})} \overset{c}{\equiv} \{\mathsf{View}_S(\mathcal{C},\mathcal{S})\}_{(\mathcal{C},\mathcal{S})}$.

When $v = 1$, for any $hc_1 = H(c_1)$ there exists R_{c_1} such that $a_1 = hc_1^{R_{c_1}}$. Therefore, a_1 is uniformly distributed – i.e., distributed identically to r_1.

For $v \geq 2$, elements $a_1,\ldots,a_v$ are indistinguishable from $r_1,\ldots,r_v$ assuming the hardness of DDH. In particular, the existence of an efficient distinguisher $\mathcal{D}$ that outputs 0 when presented with $r_1,\ldots,r_v$ and outputs 1 when it observes $a_1,\ldots,a_v$ allows us to construct a simulator $\overline{\mathsf{SIM}}_s$ that violates the DDH assumption, as follows.

Upon receiving a DDH challenge $(\bar{g},\bar{g}^x,\bar{g}^y,\bar{g}^z)$, $\overline{\mathsf{SIM}}_s$:

- Selects random set $\overline{\mathcal{C}}$ composed of v elements $\overline{\mathcal{C}} = \{c_1,\ldots,c_v\}$, $v-2$ random values $d_1,\ldots,d_{v-2}$ from $\mathbb{Z}_q$ and R_c at random from $\mathbb{Z}_q$.
- Sends $\{\overline{a}_1,\ldots,\overline{a}_v\} = \{\bar{g}^y,\bar{g}^z,(\bar{g}^y)^{d_1},\ldots,(\bar{g}^y)^{d_{v-2}}\}$ to $\mathcal{D}$.
- Answers queries for H as follows: $H(c_1) = \bar{g}$; $H(c_2) = \bar{g}^x$; $H(c_i) = \bar{g}^{d_{i-2}}$ for $3 \leq i \leq v$ and with a random value otherwise. Queries and responses to H are stored by $\overline{\mathsf{SIM}}_S$ for consistency.

Note that if $(\bar{g},\bar{g}^x,\bar{g}^y,\bar{g}^z)$ is a Diffie-Hellman tuple, i.e. $z = xy$, then $\overline{a}_1,\ldots,\overline{a}_v$ is distributed like $a_1,\ldots,a_v$; thus, $\mathcal{D}$ must output 1. If $(\bar{g},\bar{g}^x,\bar{g}^y,\bar{g}^z)$ is not a Diffie-Hellman tuple, then $\overline{a}_1,\ldots,\overline{a}_v$ is not properly distributed (since $a_2 \neq (H(c_2))^y$)

and therefore $\mathcal{D}$ must output 0. As a result, $\overline{\mathsf{SIM}}_s$ can use $\mathcal{D}$'s output to respond to the DDH challenge correctly iff $\mathcal{D}$'s output is correct. Therefore, $\mathcal{D}$ can only answer correctly with negligible advantage over random guessing. $\square$

Server Privacy. We show that client's view can be efficiently simulated by a PPT algorithm SIM_C, i.e., $\{\mathsf{SIM}_C(\mathcal{C}, |\mathcal{S} \cap \mathcal{C}|)\}_{(\mathcal{C},\mathcal{S})} \overset{c}{\equiv} \{\mathsf{View}_C(\mathcal{C}, \mathcal{S})\}_{(\mathcal{C},\mathcal{S})}$. The simulator is constructed as follows:

1. SIM_C builds two tables $T_1 = (u, h)$ and $T_2 = (u', h')$ to answer the H and H' queries respectively. SIM_C responds to a query u (resp. u') with a value in $h \leftarrow \mathbb{Z}_p$ for H (resp. $h' \leftarrow \mathbb{Z}_p$ for H'), and stores (u, h) in T_1 ((u', h') in T_2 resp.). SIM_C uses T_1, T_2 to respond consistently to queries from client.
2. SIM_C constructs a set $TS = \{ts_1, \ldots, ts_w\}$, where $ts_i \leftarrow \{0,1\}^\kappa$, and a random subset $TS' = \{ts'_1, \ldots, ts'_{|\mathcal{S}\cap\mathcal{C}|}\} \subseteq TS$, such that $|TS'| = |\mathcal{S} \cap \mathcal{C}|$.
3. Then, SIM_C adds $|\mathcal{S} \cap \mathcal{C}|$ distinct pairs $(H(c_i)^{R_s}, ts'_i \in TS')$ to T_2 and continues to answer queries to H and H' consistently using T_1 and T_2 as defined in Step 1.
4. Upon receiving $\{a_1, \ldots, a_v\}$ from client, SIM_C picks $R_s \leftarrow \mathbb{Z}_q$ and computes $a'_i = a_i^{R_s}$. Finally SIM_C sends $\Pi'(a'_i, \ldots, a'_v)$ and $\{ts_1, \ldots, ts_w\}$ to client.

Any efficient semi-honest client C^* cannot distinguish between an interaction with an honest server with input $\mathcal{S} = \{s_1, \ldots, s_w\}$ and SIM_C.

By construction, C^*'s view differs from the interaction with an honest server only in the way elements $\{ts_1, \ldots, ts_w\}$ are constructed. Let distinguisher $\mathcal{D}$ be an algorithm that outputs 0 on input an element from distribution:

$$\begin{aligned}D_0 =\{&(H(s_1), \ldots, H(s_w)), \Pi(ts_1 = H'(H(s_1)^{R_s}), \ldots, ts_w = H'(H(s_w)^{R_s})),\\ &(a_1 = H(c_1)^{R_c}, \ldots, a_v = H(c_v)^{R_c}), \Pi'(a'_1 = H(c_1)^{R_cR_s}, \ldots,\\ &a'_v = H(c_v)^{R_cR_s})\}_{hs_i}\end{aligned}$$

and 1 on input an element from:

$$\begin{aligned}D_1 =\big\{&(hs_1, \ldots, hs_w), \Pi\big(ts_1 = H'(H(c_1)^{R_s}), \ldots, ts_{|\mathcal{S}\cap\mathcal{C}|} = H'(H(c_{|\mathcal{S}\cap\mathcal{C}|})^{R_s}),\\ &ts_{|\mathcal{S}\cap\mathcal{C}|+1} = H'(r_{|\mathcal{S}\cap\mathcal{C}|+1}), \ldots, ts_w = H'(r_w)\big),\\ &(a_1 = H(c_1)^{R_c}, \ldots, a_v = H(c_v)^{R_c}),\\ &\Pi'(a'_1 = H(c_1)^{R_cR_s}, \ldots, a'_v = H(c_v)^{R_cR_s})\}_{hs_i}\end{aligned}$$

with $r_{|\mathcal{S}\cap\mathcal{C}|+1}, \ldots, r_w$ random elements from $\mathbb{Z}_p$ and where $\mathcal{D}$ is allowed to select the elements in sets $\mathcal{C} = \{c_1, \ldots, c_v\}$ and $\mathcal{S} = \{s_1, \ldots, s_v\}$. The existence of $\mathcal{D}$ violates the hardness assumption of DDH: Let (g, g^x, g^y, g^z) be a DDH challenge for simulator $\overline{\mathsf{SIM}}$, which interacts with $\mathcal{D}$ as follows: $\overline{\mathsf{SIM}}$ responds to $H(x)$ queries from $\mathcal{D}$ with g^{rh_i} for a random $rh_i \in \mathbb{Z}_q$, and stores (x, g^{rh_i}) in table T_H for consistency and to queries $H'(x)$ with a random string, using $T_{H'}$ to store queries-response for consistency.

Client, on input $\mathcal{C} = \{(c_1, \sigma_1), \ldots, (c_v, \sigma_v)\}$ $(\forall i, 1 \le i \le v : {\sigma_i}^e = hc_i)$		**Server**, on input $\mathcal{S} = \{s_1, \ldots, s_w\}$
$R_c \leftarrow \mathbb{Z}_{N/2}$		
$\forall i,\ 1 \le i \le v :$		
$a_i = (\sigma_i)^{2R_c}$	$\xrightarrow{\{a_1, \ldots, a_v\}}$	
		$R_s \leftarrow \mathbb{Z}_{N/2}$
		$\forall i,\ 1 \le i \le v,\ tc_i = H'((a_i)^{eR_s})$
		$(tc_{\ell_1}, \ldots, tc_{\ell_v}) = \Pi(tc_1, \ldots, tc_v)$
$\forall i,\ 1 \le j \le w :$		$\forall j,\ 1 \le j \le w,\ bs_j = (hs_j)^{R_s}$
$ts_j = H'((bs_j)^{2R_c})$		
	$\xleftarrow[\{bs_1, \ldots, bs_w\}]{\{tc_{\ell_1}, \ldots, tc_{\ell_v}\}}$	
Output: $\lvert\{tc_{\ell_1}, \ldots, tc_{\ell_v}\} \cap \{ts_1, \ldots, ts_w\}\rvert$		

Figure 2: Authorized PSI-CA. All computation is mod N.

W.l.o.g., let $H(c_i) = g^{rh_i}$; $\overline{\mathsf{SIM}}$ computes $a'_i = (g^y)^{rh_i \cdot R_c}$ and constructs $ch = ((g, g^x, g^{r_3}, \ldots, g^{r_w}), \Pi(ts_1 = H'(g^y), ts_2 = H'(g^z), ts_3 = H'((g^y)^{r_3}), \ldots, ts_w = H'((g^y)^{r_w})), (a_1, \ldots, a_v), (a'_1, \ldots, a'_v))$ with $r_3, \ldots, r_w$ random elements in $\mathbb{Z}_p$. Note that ch belongs to distribution D_0 iff (g, g^x, g^y, g^z) is a proper Diffie-Hellman tuple, i.e., $z = xy$ and to D_1 otherwise. Moreover, while $\mathcal{D}$ can test for which elements $H'(a'_i) = ts_j$, pairs i, j are distributed as expected because of the permutation Π'. Therefore, $\mathcal{D}$ has only negligible advantage in distinguishing the two distributions. $\square$

5 Fast Authorized PSI-CA

We now introduce the concept of Authorized PSI-CA (APSI-CA). It extends "plain" PSI-CA to enforce (pre-)*authorization* of client input. Similar to APSI [11] (reviewed in Section 2), APSI-CA involves an offline trusted third party – Certification Authority (CA) – that provides client with authorizations (in practice, signatures) to input into the set intersection cardinality computation.

Definition 6 (Authorized PSI-CA (APSI-CA)). *A protocol involving a server, on input of a set of w items: $\mathcal{S} = \{s_1, \cdots, s_w\}$, and a client, on input of a set of v items with associated authorizations (i.e., signatures), $\mathcal{C} = \{(c_1, \sigma_i) \cdots, (c_v, \sigma_v)\}$. It results in client outputting $|\mathcal{I}^*|$, where:*

$$\mathcal{I}^* = \{s_j \in \mathcal{S} \mid \exists (c_i, \sigma_i) \in \mathcal{C} \ s.t.\ c_i = s_j \wedge \mathsf{Verify}(\sigma_i, c_i) = 1\}.$$

APSI-CA entails the following informal privacy requirements:

- *Server Privacy (APSI-CA).* The client learns no information beyond what can be inferred from the protocol output, i.e., (1) cardinality of set intersection on authorized items and (2) upper bound on the size of $\mathcal{S}$ (the server could conceivably add "dummies" to its input; such dummies do not alter the output of the protocol, but conceal the exact number of elements in the server's set).
- *Client Privacy (APSI-CA).* No information is leaked about items or authorizations in client set (except an upper bound on their number).
- *Unlinkability.* Similar to PSI-CA, we require that neither server nor client can determine if any two instances of the protocol are related, i.e., executed on the same input by client or server.

We illustrate our APSI-CA protocol in Figure 2. Observe that the CA is responsible for generating all public parameters: on input the security parameter κ, it executes $(N, e, d, g) \leftarrow RSA.KGen(\kappa)$, where g is a generator of QR_N, and selects $H : \{0,1\}^* \to \mathbb{Z}_N{}^*$ (Full-Domain Hash) and $H' : \{0,1\}^* \to \{0,1\}^\kappa$ (random oracles). The CA authorizes client input c_i by issuing $\sigma_i = H(c_i)^d \bmod N$ (i.e., an RSA signature). The protocol is executed between client and server, on common input (N, e, H, H'). We assume that server's input ($\mathcal{S}$) is randomly permuted before protocol execution to mask any *ordering* of the items contained in it. Finally, hc_i and hs_j denote, respectively, $H(c_i)$ and $H(s_j)$.

Similar to its PSI-CA counterpart, this APSI-CA has the following properties:

- ***Correctness.*** For any (σ_i, c_i) held by client and s_j held by server, if: (1) σ_i is a genuine CA signature on c_i, and (2) $c_i = s_j$, hence, $hc_i = hs_j$, we obtain: $tc_{\ell_i} = H'((\sigma_i)^{2eR_cR_s}) = H'((hc_i)^{2R_cR_s}) = ts_j$.
- ***Privacy.*** In this version of the paper, we only provide some intuition for our security arguments, and defer to future work formal proofs. Client privacy is based on its input being statistically indistinguishable from a random distribution in QR_N. Arguments regarding server privacy are similar to those for PSI-CA, thus, we do not repeat them here. We argue that if one could violate APSI-CA server privacy, then the one would also violate server privacy of the APSI construct in Figure 1 of [10], proven secure under the RSA and DDH assumptions. Finally, note that the protocol is unlinkable, given that random values, R_c, R_s, are selected fresh for each protocol execution.
- ***Efficiency.*** This APSI-CA protocol incurs linear computation (for both parties) and communication complexity. Specifically, client and server perform respectively $O(w)$ and $O(w + v)$ modular exponentiations. However, exponents are now taken in the RSA settings, while in PSI-CA can be taken from a smaller group, thus, be much shorter (e.g., 160-bit vs 1024-bit long). Communication complexity amounts to $O(w + v)$. Note that this is significantly lower than related work, i.e., [8], which incurs quadratic overhead (see Section 2.2).

6 Combining PSI-CA and PSI

As mentioned in Section 1, it is often desirable to privately assess the magnitude of the set intersection before engaging in an actual (private) set intersection computation. We are motivated by potential concerns with respect to server privacy, arising in PSI executions where the intersection obtained by client is close to the entire server set (i.e., $|\mathcal{S} \cap \mathcal{C}| \approx |\mathcal{S}|$).

We now show how to combine our proposed PSI-CA construct with with PSI functionality, in order to address such concerns. Specifically, rather than engaging in PSI, parties first run the PSI-CA protocol with their client/server roles reversed. This way, server learns (only) the intersection cardinality and the size of the parties' inputs, and uses this information to decide whether to proceed with PSI. In case it decides to proceed, client only needs to receive one more message from server to compute the intersection. In other words, server defines a *policy* – based on the size of (i) the two sets and (ii) the intersection – and only if the policy is satisfied, server engages in PSI protocol (thus, client privately obtains the set intersection).

The resulting protocol is presented in Figure 3. In the first two rounds, server and client run PSI-CA with their *roles reversed* (i.e., server learns the cardinality of the intersection), and, assuming server's policy is satisfied, the last round allows client to learn the set intersection. The same approach can be used for other private set operations, such as PSU [18]. Indeed, similar concerns about server privacy occur in a scenario where $|\mathcal{C} \cup \mathcal{S}| \approx |\mathcal{C}| + |\mathcal{S}|$, and can again be addressed by running PSI-CA with roles reversed. Observe that protocol in Figure 3 incurs complexities comparable to the underlying PSI-CA (illustrated in Figure 1): only one additional message must be sent to realize policy-based PSI.

The security of this protocol, in presence of semi-honest adversaries, trivially stems from that of the underlying PSI-CA. Nonetheless, we defer to future work extending our constructions to malicious security. In fact, there is no guarantee that malicious parties maintain the same input over multiple interactions or do not abort execution prematurely. This constitutes an interesting open challenge that we defer to future work.

Remark: Our technique in Figure 3 is not to be confused with the concept of Policy-Enhanced PSI, recently proposed by [36]. Using the latter, two parties privately obtain the intersection of their sets, while enforcing policies pertaining what/how to share, based on policies and authorizations related to single items. Whereas, policy enforced by server in our protocol is much simpler – it is based on the cardinality of set intersection: depending on this (and on its relationship to set size), server decides whether or not to disclose set intersection's content to client. A vaguely comparable approach is so-called Knowledge-oriented Multiparty Secure Computation [29], where each participating party is able to reason about the increase in knowledge that other parties could gain as a result of the secure computation, and may choose not to participate to restrict that gain.

Server, on input: $\mathcal{S} = \{s_1, \dots, s_w\}$		**Client**, on input: $\mathcal{C} = \{c_1, \dots, c_v\}$
$R_s \leftarrow \mathbb{Z}_q$		$(\hat{c}_1, \dots, \hat{c}_v) \leftarrow \Pi(\mathcal{C})$, with Π random permutation
$\forall i\ 1 \le i \le v$: $a_i = (hs_i)^{R_s}$	$\xrightarrow{\{a_1, \dots, a_w\}}$	$\forall i\ 1 \le i \le v : hc_i = H(\hat{c}_i)$ $R_c \leftarrow \mathbb{Z}_q$ $\forall i\ 1 \le i \le w :\ a'_i = (a_i)^{R_c}$
		$(a'_{\ell_1}, \dots, a'_{\ell_w}) = \Pi'(a'_1, \dots, a'_w)$ $\forall j\ 1 \le j \le v :\ bc_j = {hc_j}^{R_c}$ $\forall j\ 1 \le j \le v :\ tc_j = H'(bc_j)$
$\forall i\ 1 \le i \le w$: $bs_i = (a'_{\ell_i})^{1/R_s}$ $\forall i\ 1 \le i \le w,\ ts_i = H'(bs_i)$	$\xleftarrow[\{tc_1, \dots, tc_v\}]{\{a'_{\ell_1}, \dots, a'_{\ell_w}\}}$	
$T^* = \{tc_1, \dots, tc_v\} \cap \{ts_1, \dots, ts_w\}$		
If *Policy.isSatisfied*$(w, v, \lvert T^* \rvert)$:	$\xrightarrow{T^*}$	$\forall\ tc_j \in T^*$: **Output** $c_j \in \mathcal{S} \cap \mathcal{C}$

Figure 3: Combining PSI-CA and PSI for a three-round policy-based Private Set Intersection protocol. (All computation is mod p).

7 Conclusion

This paper presented a protocol for PSI-CA, with *linear* computational and communication complexities. It can be used to compute PSU-CA, without introducing any additional overhead. Then, we presented two novel extensions. First, we introduced Authorized PSI-CA, or APSI-CA, that is useful in settings where client input must be authorized by a certification authority. Then, we used PSI-CA to realize a PSI protocol where server determines (in privacy-preserving manner) cardinality of set intersection before deciding whether or not to engage in a PSI interaction with client. Such an approach is very efficient, as it requires only *one* additional message on top of our PSI-CA protocol.

We will release an optimized implementation of all protocols presented in this paper along with the final version of the paper. As part of future work, we plan to investigate extensions to guarantee security in the presence of malicious adversaries and in the UC framework. **Acknowledgments.** We wish to thank Stanislaw Jarecki for his valuable feedback.

References

1. R. Agrawal, A. Evfimievski, and R. Srikant. Information sharing across private databases. In *SIGMOD*, 2003.
2. B. Applebaum, H. Ringberg, M. Freedman, M. Caesar, and J. Rexford. Collaborative, Privacy-preserving Data Aggregation at Scale. In *PETS*, 2010.

3. G. Ateniese, M. Blanton, and J. Kirsch. Secret handshakes with dynamic and fuzzy matching. In *NDSS*, 2007.
4. P. Baldi, R. Baronio, E. De Cristofaro, P. Gasti, and G. Tsudik. Countering GATTACA: Efficient and Secure Testing of Fully-Sequenced Human Genomes. In *CCS*, 2011.
5. M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko. The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. *Journal of Cryptology*, 16(3), 2003.
6. C. Blundo, E. De Cristofaro, and P. Gasti. EsPRESSo: Efficient Privacy-Preserving Evaluation of Sample Set Similarity. In *DPM*, 2012.
7. E. Bursztein, J. Lagarenne, M. Hamburg, and D. Boneh. OpenConflict: Preventing Real Time Map Hacks in Online Games. In *S&P*, 2011.
8. J. Camenisch and G. M. Zaverucha. Private intersection of certified sets. In *Financial Cryptography*, 2009.
9. E. De Cristofaro, S. Jarecki, J. Kim, and G. Tsudik. Privacy-preserving policy-based information transfer. In *PETS*, 2009.
10. E. De Cristofaro, J. Kim, and G. Tsudik. Linear-Complexity Private Set Intersection Protocols Secure in Malicious Model. In *Asiacrypt*, 2010.
11. E. De Cristofaro and G. Tsudik. Practical private set intersection protocols with linear complexity. In *Financial Cryptography*, 2010.
12. A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *PODS*, 2003.
13. M. Freedman, Y. Ishai, B. Pinkas, and O. Reingold. Keyword search and oblivious pseudorandom functions. In *TCC*, 2005.
14. M. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Eurocrypt*, 2004.
15. K. Frikken. Privacy-Preserving Set Union. In *ACNS*, 2007.
16. O. Goldreich. *Foundations of Cryptography*. Cambridge U. Press, 2004.
17. C. Hazay and Y. Lindell. Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. In *TCC*, 2008.
18. C. Hazay and K. Nissim. Efficient Set Operations in the Presence of Malicious Adversaries. In *PKC*, 2010.
19. S. Hohenberger and S. Weis. Honest-verifier private disjointness testing without random oracles. In *PET*, 2006.
20. J. Hong, J. W. Kim, J. Kim, K. Park, and J. H. Cheon. Constant-Round Privacy Preserving Multiset Union. Cryptology ePrint Archive, Report 2011/138, 2011. `http://eprint.iacr.org/2011/138`.
21. Y. Huang, D. Evans, and J. Katz. Private Set Intersection: Are Garbled Circuits Better than Custom Protocols? In *NDSS*, 2012.
22. B. Huberman, M. Franklin, and T. Hogg. Enhancing privacy and trust in electronic communities. In *ACM Conference on Electronic commerce*, 1999.
23. Intelligence Advanced Research Projects Activity (IARPA). Automatic Privacy Protection and Security And Privacy Assurance Research Programs. `https://www.fbo.gov/utils/view?id=920029a5107a9974c2e379324a1dcc4e`.
24. Y. Ishai, J. Kilian, K. Nissim, and E. Petrank. Extending oblivious transfers efficiently. *CRYPTO*, 2003.
25. S. Jarecki and X. Liu. Efficient Oblivious Pseudorandom Function with Applications to Adaptive OT and Secure Computation of Set Intersection. In *TCC*, 2009.
26. S. Jarecki and X. Liu. Fast secure computation of set intersection. In *SCN*, 2010.

27. M. Kantarcioglu, R. Nix, and J. Vaidya. An efficient approximate protocol for privacy-preserving association rule mining. *KDD*, 2009.
28. L. Kissner and D. Song. Privacy-preserving set operations. In *Crypto*, 2005.
29. P. Mardziel, M. Hicks, J. Katz, and M. Srivatsa. Knowledge-oriented secure multiparty computation. In *PLAS*, 2012.
30. A. Menezes, P. V. Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC, 1997.
31. S. Nagaraja, P. Mittal, C. Hong, M. Caesar, and N. Borisov. BotGrep: Finding Bots with Structured Graph Analysis. In *Usenix Security*, 2010.
32. A. Narayanan, N. Thiagarajan, M. Lakhani, M. Hamburg, and D. Boneh. Location Privacy via Private Proximity Testing. In *NDSS*, 2011.
33. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Eurocrypt*, 1999.
34. S. Pohlig and M. Hellman. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. *IEEE Transactions on information Theory*, 24(1), 1978.
35. J. H. Seo, J. H. Cheon, and J. Katz. Constant-Round Multi-Party Private Set Union using Reversed Laurent Series. In *PKC*, 2012.
36. E. Stefanov, E. Shi, and D. Song. Policy-enhanced private set intersection: Sharing information while enforcing privacy policies. In *PKC*, 2012.
37. J. Vaidya and C. Clifton. Secure set intersection cardinality with application to association rule mining. *Journal of Computer Security*, 13(4), 2005.
38. A. Yao. Protocols for secure computations. In *FOCS*, 1982.