Fast and Private Computation of Set Intersection Cardinality

Emiliano De Cristofaro, Paolo Gasti, and Gene Tsudik

University of California, Irvine

Abstract. The use of sensitive electronic information has increased tremendously in recent years. In many realistic application scenarios, legitimate needs for sensitive information must be reconciled with privacy concerns. This has motivated various privacy-protecting cryptographic techniques, such as Private Set Intersection (PSI) and Private Set Union (PSU). Such techniques involve two parties – a client and a server – each holding a private data set: the client learns only the intersection (or union) of the two respective sets, while the server learns nothing. However, it is often imprudent to use PSI and PSU protocols alone, since their use may offer little or no privacy for the server. Instead, prospective participants in PSI/PSU protocols should use their respective policies to decide whether or not to participate. Policies can be based on variables, such as the size of set intersection and its relationship to the entire set size. To enable policy considerations in advance of private set operations, we need a cryptographic primitive called *Private Set Intersection Cardinality* (PSI-CA), which yields only the size of set intersection. PSI-CA protocols are also particularly appealing in numerous realistic scenarios where it is crucial to obtain the magnitude – rather than the content – of the set intersection.
This paper motivates the need for PSI-CA and constructs a very efficient (i.e., linear-complexity) protocol. Efficiency claims are supported by experiments with prototype implementations. Finally, an extension to support authorization of client input is also sketched.

1 Introduction & Motivation

Proliferation of, and growing reliance on, electronic information fuel the increase of sensitive data in cyberspace. Privacy concerns have motivated the need for efficient privacy-preserving cryptographic mechanisms, such as Private Set Intersection (PSI) [11,22,14,19,15,20,9,8], and Private Set Union (PSU) [22,15,17,12].

PSI techniques allow one party (client) to compute the intersection of its input set with that of another party (server), such that: (i) the server learns nothing about client input, and (ii) the client learns no information about server input, beyond the intersection. Recently, efficient PSI protocols have been used as the main building block for many privacy-focused systems and applications, including location sharing [25], collaborative botnet detection [24], smartphone applications [6], on-line gaming [4], and intelligence-community systems [18].

Nonetheless, in certain scenarios, PSI and PSU techniques offer limited privacy for the server. Consider, for instance, an extreme case where the set intersection learned by the client corresponds to the entire server input. In this case, server privacy is non-existent, whereas, client privacy is fully preserved. Similarly, suppose that the size of the union equals the sum of the two respective set sizes: the server has no privacy since its entire input is disclosed to the client. Both examples illustrate the need for the server to establish and enforce its own policy. Such a policy might require the server to determine (in privacy-preserving manner) cardinality of set intersection before deciding whether or not to engage in PSI or PSU interactions with the client.

Moreover, private computation of cardinality of set intersection is crucial in many scenarios where the client is allowed to only obtain the magnitude – rather than the content – of the set intersection. Recently, the work in [2] advocates the availability of efficient PSI-CA constructs to perform privacy-preserving paternity testing. PSI-CA also appeals to the case where two parties want to compare their respective sets of friends and their policy is to become friends if they have at least a given number of friends in common. Other interesting applications of PSI-CA include role-based affiliation-hiding authentication [1] as well as association rule mining [21].

Roadmap. In this paper, we investigate ***Private Set Intersection Cardinality (PSI-CA)*** – a cryptographic primitive involving a server S (on input of a private set $\mathcal{S}$) and a client C (on input of a private set $\mathcal{C}$). It results in the client outputting $|\mathcal{S} \cap \mathcal{C}|$. Prior work yielded some PSI-CA constructs, however, as demonstrated by this paper, none is efficient enough for real-world applications. In fact, available PSI-CA protocols involve a number of cryptographic operations (such as modular exponentiations) quadratic in the size of participants' sets. Also, some constructs additionally incur quadratic communication overhead as well. On the contrary, this paper presents the first PSI-CA technique with *linear* computation and communication complexity. Proposed technique uses efficient cryptographic operations involving only a linear number of short-exponent modular exponentiations and provides provably secure privacy guarantees. We also show that PSI-CA can be used as a building block for policy-based privacy-preserving set operations and present an efficient three-round ***Policy-based Private Set Intersection*** protocol. In it, the server fixes a policy the disclosure of set intersection – bounded to the size of the intersection. Finally, we sketch a protocol variant, called ***Authorized Private Set Intersection Cardinality (APSI-CA)***, where client input must be authorized (signed) by some recognized and mutually trusted authority (CA).

Paper Organization. Next section reviews related work on privacy-preserving set operations. Then, Section 3 presents our novel protocol for Private Set Intersection Cardinality (PSI-CA), along with security proofs and performance evaluation (and comparison to prior work). In Section 4, we propose a three-round policy-based Private Set Intersection based on our PSI-CA construct, and

conclude in Section 5. Finally, Appendix A sketches a protocol for Authorized Private Set Intersection Cardinality (APSI-CA).

2 Related Work

Prior work yielded several approaches to privacy-preserving set operations, which we review below.

Private Set Intersection and Union

The work in [11] introduces the Private Set Intersection (PSI) problem and present techniques using Oblivious Polynomial Evaluations (OPE-s) and additively homomorphic encryption (e.g., Paillier [26]). The intuition is to represent a set as a polynomial, and its elements – as the polynomial's roots. The client uses an additively homomorphic cryptosystem to encrypt the coefficients, that are then evaluated homomorphically by the server, such that the client learns the intersection (and nothing else) upon decrypting. Assuming that server and client sets contain w and v items, respectively, client's computation complexity amounts to $O(w + v)$ exponentiations, and server's – $O(wv)$ exponentiations. [11] also proposes techniques to asymptotically reduce server's workload to $O(w \log\log v)$ using Horner's rule and balanced bucket allocation. Next, [15] obtains similar complexities but also extend their techniques to the PSU problem, while [22] extends OPE-s to more than two players, all learning the intersection/union, with quadratic computational and linear communication complexities.

Other PSI constructs, such as [14,19], 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) obtain very efficient PSI protocols, also with linear complexities but using much more efficient cryptographic tools: they replace OPRFs with unpredictable functions [20] and blind signatures [9], with security under *One-More-DH* and *One-More-RSA* assumptions [3], respectively. Finally, the work in [8] achieves linear communication and computational complexities, using short exponents, with security under the DDH assumption.

Authorized Private Set Intersection

Authorization of client input was first investigated, independently, in [5] and [7] (both with quadratic complexities). Later, this concept was formalized as Authorized Private Set Intersection (APSI) in [9] and [8], where efficient techniques with linear computation and communication overhead are proposed, with security under the RSA assumption.

Private Set Intersection Cardinality

Prior work has proposed a number of Private Set Intersection Cardinality (PSI-CA) protocols, which we review below. Note that none of these achieve practical—specifically, linear-complexity—overhead.

- Note that the PSI protocol by Freedman, Nissim, and Pinkas [11] can actually be extended to PSI-CA, with similar complexities – i.e., $O(w \log\log v)$ computational and $O(w+v)$ communication overhead (again, assuming that server and client sets contain w and v items).
- Hohenberger and Weis [16] present a PSI-CA protocol based on [11], thus, achieving similar (sub-quadratic) complexities. [16] also shows that one can use the additively homomorphic Elgamal variant (rather than Paillier), hence, using shorter exponents and speeding up computation.
- Kissner and Song [22] propose a PSI-CA construct for multiple (n) players, incurring $O(n^2 \cdot v)$ communication and $O(v^2)$ computational overhead (assuming all parties hold set of size v).
- Vaidya and Clifton [29] also propose a multi-party PSI-CA, based on commutative one-way hash functions [23], using Pohlig-Hellman [27], and incurring n rounds, $O(n^2 \cdot v)$ communication and $O(vn)$ computational overhead.
- Narayanan, et al. [28] present a PSI-CA protocol with information-theoretic security, however, with high computation and communication overhead. [28] also requires the presence of a semi-trusted third party.
- Finally, Camenisch and Zaverucha [5] present an APSI variant (specifically, so-called intersection of certified sets) that computes the cardinality of (certified) set intersection and incurs quadratic communication and computation complexity.

3 Efficient Private Set Intersection Cardinality

In this section, we define the Private Set Intersection Cardinality (PSI-CA) functionality, along with its privacy requirements. Next, we present our efficient construct.

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

PSI-CA entails the following (informal) privacy requirements:

- *Server Privacy.* The client learns no information beyond what can be inferred from the actual protocol output, i.e., (1) cardinality of set intersection 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).

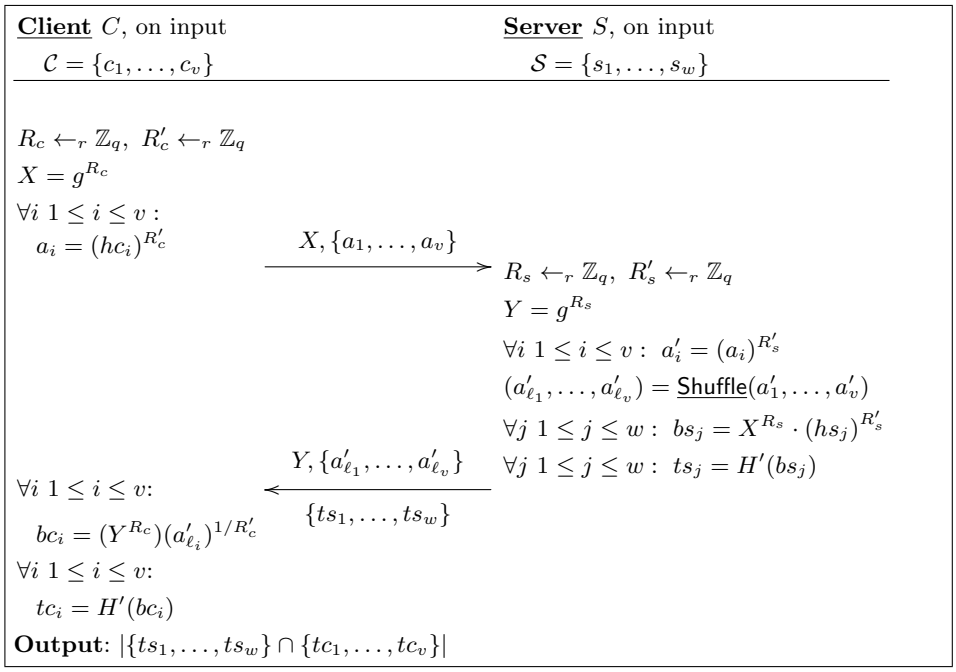

Figure 1: Proposed PSI-CA Protocol. All computation is mod p.

- *Unlinkability.* Neither the server nor the client can determine if any two instances of the protocol are related, i.e., executed on the same input by the client or the server.

The Protocol

We present our PSI-CA construct in Figure 1. Note that the protocol is executed on common input of two primes p, q, s.t., $q|p-1$, a generator g of a subgroup of size q, and two hash functions (modeled as random oracles), $H : \{0,1\}^* \to \mathbb{Z}_p^*$ and $H' : \{0,1\}^* \to \{0,1\}^\kappa$, given the security parameter κ. We also assume that server's input ($\mathcal{S}$) is randomly permuted before protocol execution to mask any *ordering* of the items contained in it. Finally, observe that hc_i and hs_j denote, respectively, $H(c_i)$ and $H(s_j)$.

This section presents security proofs for the PSI-CA protocol, introduced in Section 3. Whereas, security proofs for APSI-CA are left as part of future work. Below, we discuss our adversarial model and computational assumptions.

Semi-Honest Participants. We start with semi-honest security: we introduce our security definitions and present formal security proofs. 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.

One-More-DH Assumption. Informally, the One-More-DH assumption [3] indicates that the DH problem is hard even if the adversary is given access to a DH oracle. Formally, let $(\mathbb{G}, q, g) \leftarrow KeyGen(\kappa)$ the Key-Generation algorithm outputting a multiplicative group of order q and assume $z \leftarrow \mathbb{Z}_q$. We say that the One-More-DH problem is (τ, t)-hard if for every algorithm $\mathcal{A}$ that runs in time t we have:

$$Pr[\{(g_i, (g_i)^x)\}_{i=1,\cdots v+1} \leftarrow \mathcal{A}^{DH_z(\cdot)}(g_1, \cdots, g_{ch})] \leq \tau$$

where $\mathcal{A}$ made at most v queries to the $DH_x(\cdot)$ oracle.

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

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

is a negligible function of κ.

Definitions

We assume semi-honest parties and use general definitions of secure computation given in [13]. We prove that protocol in Figure 1 satisfies the following requirements.

Definition 2 (Correctness). *If both parties are honest, at the end of the protocol, run on inputs* $((\mathcal{S}, v), (\mathcal{C}, w))$*,* S *outputs* $\perp$*, and* C *outputs* $(|\mathcal{S} \cap \mathcal{C}|)$*.*

The following client and server privacy definitions follow from those in related work [11,10,14]. In particular, Goldreich ([13], Sec. 7.2.2) states that, in case of semi-honest parties, the general "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 3 (Client Privacy). *For every* PPT S^* *that plays the role of* S*, for every* $\mathcal{S}$*, and for any client input set* $(\mathcal{C}^{(0)}, \mathcal{C}^{(1)})$*, s.t.* $|\mathcal{C}^{(0)}| = |\mathcal{C}^{(1)}|$*, two views of* S^* *corresponding to* C*'s inputs:* $\mathcal{C}^{(0)}$ *and* $\mathcal{C}^{(1)}$*, are computationally indistinguishable.*

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

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

In other words, on each possible pair of inputs $(\mathcal{C}, \mathcal{S})$, C'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 [13], we claim that the two distributions implicitly defined above are computationally indistinguishable.

Proofs

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

$$\begin{aligned} tc_{\ell_i} &= H'(bc_i) = H'(Y^{R_c} \cdot {a_{\ell_i}}^{(1/R'_c)}) = H'(g^{R_c R_s} \cdot {hs_j}^{R'_s}) \\ ts_j &= H'(bs_j) = H'(X^{R_s} \cdot hs_j^{R'_s}) = H'(g^{R_c R_s} \cdot {hs_j}^{R'_s}) \end{aligned}$$

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

Client Privacy. Client's messages to the server are $X = g^{R_c}$ – which is independent from its input – and $a_i = H(c_i)^{R'_c}$ for $i = 1, \ldots, v$, where H is modeled as a random oracle. It is easy to see that, since R'_c is chosen uniformly at random from $\mathbb{Z}_q$ (with $q|p-1$), the distribution of a_i-s are essentially equivalent to that of random elements in $\mathbb{Z}_p$. □

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

1. SIM_c builds two tables $T_1 = (q, h)$ and $T_2 = (q', h')$ to answer the H and H' queries respectively. SIM_c responds to a query q (resp. q') with a value in $h \leftarrow_r \mathbb{Z}_p$ for H ($h' \leftarrow_r \mathbb{Z}_p$ for H'), and stores (q, h) in T_1 ((q', h') in T_2 resp.). SIM_c uses T_1, T_2 to respond consistently to the queries from the client.
2. SIM_c constructs a set $TS = \{ts_1, \ldots, ts_w\}$, where $ts_i \leftarrow_r \mathbb{Z}_p$, and a random subset $TS' = \{ts'_1, \ldots, ts'_{|\mathcal{I}|}\} \subseteq TS$ such that $|TS'| = |\mathcal{I}|$.
3. Upon receiving $X, \{a_1, \ldots, a_v\}$ from the client, SIM_c picks $R_s \leftarrow_r \mathbb{Z}_q$ and $R'_s \leftarrow_r \mathbb{Z}_q$, computes $a'_i = a_i^{R'_s}$ and adds $(a', {a'}^{R'_s})$ to T_3. Finally SIM_c sends $Y = X^{Rs}$, $\mathsf{Shuffle}(a'_i, \ldots, a'_v)$ and $\{ts_1, \ldots, ts_w\}$ to the client.
4. SIM_c adds $|\mathcal{I}|$ pairs $((X^{R_s} \cdot H(c_i)^{R'_s}, ts'_i \in TS')$ in T_2 and continues to answers queries to H and H' consistently using T_1 and T_2 as defined in Step 1.

5. SIM_c answers to the queries to H and H' as follows:
   (a) For query q to H never asked before, SIM_c responds with a value in $h \leftarrow_r \mathbb{Z}_p$ and adds (q, h) to T_1.
   (b) For query q' to H' never asked before, if there is a pair (q, h) in T_1 such that $h^{R'_s} = (q'/X^{R_s})$, add $(h, (q'/X^{R_s}))$ to table T_3 and aborts; otherwise SIM_c responds with $h' \leftarrow_r \mathbb{Z}_p$ and adds (q', t') to T_2.

Any efficient honest-but-curious client cannot distinguish between the interaction with an honest server and SIM_c. By construction, client's view differs from the interaction with an honest server only when SIM_c aborts. In particular, the existence of an efficient honest-but-curious client C^* that can cause SIM_c to abort allows us to construct an efficient adversary for the Gap-One-More-DH problem. The reduction $\overline{\mathsf{SIM}}_c$ is constructed modifying SIM_c as follows:

- Given a Gap-One-More-DH challenge $\mathsf{Ch} = (g_1, \ldots, g_n)$ and access to oracles $(\cdot)^x$ and $DL_x(\cdot, \cdot)$, $\overline{\mathsf{SIM}}_c$ responds to a query q for H, where q was never asked before, with a fresh g_i rather than with a random element in $\mathbb{Z}_p$ as defined in Step 1.
- In Step 3, upon receiving $X, \{a_1, \ldots, a_v\}$ from the client, $\overline{\mathsf{SIM}}_c$ picks $R_s \leftarrow_r \mathbb{Z}_q$ and uses the oracle $(\cdot)^x$ to compute $a'_i = a_i^x$ and adds (a', a'^x) to T_3 (i.e. $\overline{\mathsf{SIM}}_c$ invokes the oracle $(\cdot)^x$ exactly v times). Then SIM_c sends $Y = X^{Rs}$, $\mathsf{Shuffle}(a'_i, \ldots, a'_v)$ and $\{ts_1, \ldots, ts_w\}$ to the client.
- In Step 4, $\overline{\mathsf{SIM}}_c$ does not add any new pair to T_2 and simply answers queries to H and H' consistently.
- In Step 5.(a), $\overline{\mathsf{SIM}}_c$ responds to a new query q for H with a fresh element $g_i \in \mathsf{Ch}$
- In Step 5.(b), upon receiving a query q' for H', $\overline{\mathsf{SIM}}_c$ computes $t = q'/X^{R_s}$. If there exists an element $g_i \in \mathsf{Ch}$ such that $DL_x(g_i, t) = 1$ and $(g_i, t) \notin T_3$, then $\overline{\mathsf{SIM}}_c$ adds (g_i, t) to T_3

At the end of the execution, we have that $|T_3| > v$. $\overline{\mathsf{SIM}}_c$ can now use $|T_3|$ to answer the Gap-One-More-DH challenge. Since this contradicts the hardness of the Gap-One-More-DH problem, SIM_c only aborts – and therefore C^* detects the simulation – with only negligible probability. □

Performance Evaluation

The PSI-CA protocol proposed in Figure 1 has been implemented in `C++`, using the OpenSSL library, and tested on a PC with 2.33GHz CPU and 8GB RAM. Source code is available upon request.

We now compare its performance to the most efficient available PSI-CA construct – i.e., the one in [11] (denoted FNP below). Recall that, while PSI-CA protocol proposed in this paper incurs linear communication and computation overhead, FNP incurs sub-quadratic computational overhead.

In Figure 2, we compare running times of both tested protocols, instantiated in ROM, using 1024-bit moduli and exponents from subgroups of 160-bit order. We use increasing set sizes (from 1 to 1000) and plot total running times, in seconds, using a *logarithmic scale*. Communication overhead is omitted since it is the same for both protocols (linear in the size of sets).

Results confirm that our PSI-CA construct remarkably improves efficiency of prior work (of several order of magnitudes). For instance, if client and server sets contain 1000 items, our protocol executes in about $1s$, compared to more than $1000s$ using FNP [11].

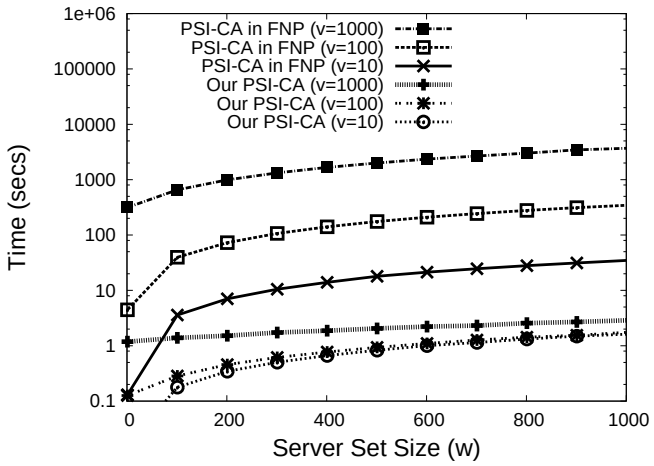

Figure 2: Total computation overhead (in seconds) of PSI-CA in FNP [11] and our PSI-CA in Figure 1.

4 Policy-based Private Set Intersection

We observe that the Private Set Intersection Cardinality (PSI-CA) primitive can be used not only for privately computing intersection size of two sets. We argue, in fact, that our PSI-CA protocol (proposed in Section 3) provides a powerful tool to construct *policy-based* privacy-preserving protocols. Consider the scenario discussed in Section 1 where the execution of Private Set Intersection (PSI) results in the client learning an intersection with size close to the entire server set (i.e., $|\mathcal{S} \cap \mathcal{C}| \approx |\mathcal{S}|$).

We now show how our PSI-CA protocol can be used, together with PSI protocols, to design a policy-based PSI technique. In it, the client obtains the intersection only if a server's *policy* is satisfied. Specifically, before engaging in PSI, parties first run the PSI-CA protocol with their (client/server) roles reversed. This way, the server learns the cardinality of the intersection and decides whether or not to let the client obtain the set intersection.

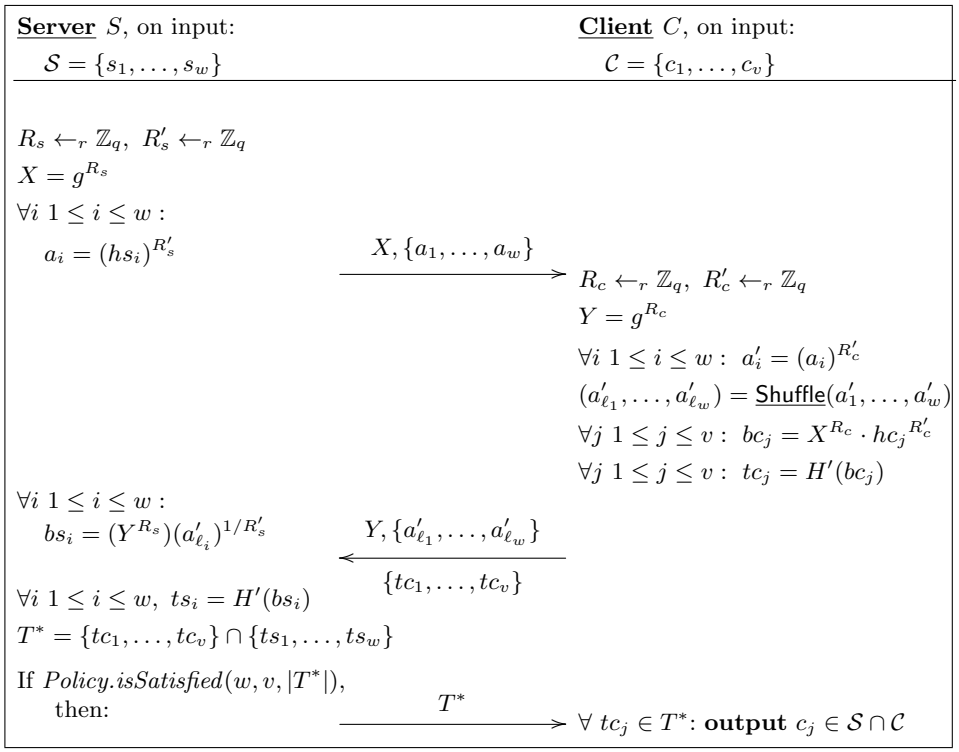

Figure 3: Three-round Policy-based Private Set Intersection protocol. All computation is mod p.

Proposed PSI-CA protocol, assuming semi-honest adversaries, can be easily composed into such a policy-based private set intersection primitive. After learning the magnitude of the intersection, the server decides whether or not its policy is satisfied; in this case, it needs to send the client only one last message, allowing the client to recover the set intersection.

The resulting protocol, which we denote as ***Policy-based Private Set Intersection***, is presented in Figure 3. In the first two rounds, the server and the client run a PSI-CA with their roles reversed, and the last round allows the client to learn the set intersection. The same approach can be used for other operations on private sets, such as Private Set Union [15]. 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 exchanged roles.

5 Conclusion

This paper presented a novel protocol for Private Set Intersection Cardinality (PSI-CA) that is appreciably more efficient than state of the art. Performance

evaluation of prototype implementations confirmed our efficiency claims. Further, we showed how to use proposed PSI-CA construct to realize a three-round policy-based Private Set Intersection protocol, allowing the server to determine (in privacy-preserving manner) cardinality of set intersection before deciding whether or not to engage in a PSI interaction with the client.

We proved security of proposed constructs in the presence of semi-honest models, while we leave as part of future work to prove their security against malicious adversaries. In particular, note that the composition of protocols for operations on private set is not a trivial exercise in presence of malicious adversaries. In fact, there is no straightforward guarantee that malicious parties maintain the same input over multiple interactions, thus, proving Policy-based Private Set Intersection with malicious-security is an interesting challenge that calls for further research in the area.

References

1. G. Ateniese, M. Blanton, and J. Kirsch. Secret handshakes with dynamic and fuzzy matching. In *NDSS*, 2007.
2. P. Baldi, R. Baronio, E. De Cristofaro, P. Gasti, and G. Tsudik. Countering GATTACA: Efficient and Secure Testing of Fully-Sequenced Human Genomes. In *To Appear in ACM Conference on Computer and Communications Security (CCS)*, 2011.
3. 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.
4. E. Bursztein, J. Lagarenne, M. Hamburg, and D. Boneh. OpenConflict: Preventing Real Time Map Hacks in Online Games. In *IEEE Security and Privacy Symposium*, 2011.
5. J. Camenisch and G. M. Zaverucha. Private intersection of certified sets. In *Financial Cryptography*, 2009.
6. E. De Cristofaro, A. Durussel, and I. Aad. Reclaiming Privacy for Smartphone Applications. In *PerCom*, 2011.
7. E. De Cristofaro, S. Jarecki, J. Kim, and G. Tsudik. Privacy-preserving policy-based information transfer. In *PETS*, 2009.
8. E. De Cristofaro, J. Kim, and G. Tsudik. Linear-Complexity Private Set Intersection Protocols Secure in Malicious Model. In *Asiacrypt*, 2010.
9. E. De Cristofaro and G. Tsudik. Practical private set intersection protocols with linear complexity. In *Financial Cryptography*, 2010.
10. M. Freedman, Y. Ishai, B. Pinkas, and O. Reingold. Keyword search and oblivious pseudorandom functions. In *TCC*, 2005.
11. M. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Eurocrypt*, 2004.
12. K. Frikken. Privacy-Preserving Set Union. In *ACNS*, 2007.
13. O. Goldreich. *Foundations of Cryptography*. Cambridge U. Press, 2004.
14. C. Hazay and Y. Lindell. Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. In *TCC*, 2008.
15. C. Hazay and K. Nissim. Efficient Set Operations in the Presence of Malicious Adversaries. In *PKC*, 2010.

16. S. Hohenberger and S. Weis. Honest-verifier private disjointness testing without random oracles. In *PET*, 2006.
17. 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/`.
18. Intelligence Advanced Research Projects Activity. Automatic Privacy Protection Program. `http://sprout.ics.uci.edu/projects/iarpa-app/`.
19. S. Jarecki and X. Liu. Efficient Oblivious Pseudorandom Function with Applications to Adaptive OT and Secure Computation of Set Intersection. In *TCC*, 2009.
20. S. Jarecki and X. Liu. Fast secure computation of set intersection. In *SCN'10*, 2010.
21. M. Kantarcioglu, R. Nix, and J. Vaidya. An efficient approximate protocol for privacy-preserving association rule mining. *KDD*, 2009.
22. L. Kissner and D. Song. Privacy-preserving set operations. In *Crypto*, 2005.
23. A. Menezes, P. V. Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC, 1997.
24. S. Nagaraja, P. Mittal, C. Hong, M. Caesar, and N. Borisov. BotGrep: Finding Bots with Structured Graph Analysis. In *Usenix Security*, 2010.
25. A. Narayanan, N. Thiagarajan, M. Lakhani, M. Hamburg, and D. Boneh. Location Privacy via Private Proximity Testing. In *NDSS*, 2011.
26. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Eurocrypt*, 1999.
27. 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.
28. G. Sathya Narayanan, T. Aishwarya, A. Agrawal, A. Patra, A. Choudhary, and C. Pandu Rangan. Multi Party Distributed Private Matching, Set Disjointness and Cardinality of Set Intersection with Information Theoretic Security. In *CANS*, 2009.
29. J. Vaidya and C. Clifton. Secure set intersection cardinality with application to association rule mining. *Journal of Computer Security*, 13(4), 2005.

A Efficient Authorized Private Set Intersection Cardinality

In this appendix, we introduce the concept of Authorized Private Set Intersection Cardinality (APSI-CA). It extends PSI-CA to enforce authorization of client input. Similar to Authorized Private Set Intersection, APSI-CA involves an (offline) trusted third party, e.g., a Certification Authority (CA).

Definition 5 (Authorized Private Set Intersection Cardinality (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 (signatures), $\mathcal{C} = \{(c_1, \sigma_i) \cdots, (c_v, \sigma_v)\}$. It results in the client outputting outputting $|\boldsymbol{\mathcal{I}^*}|$, where:*

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

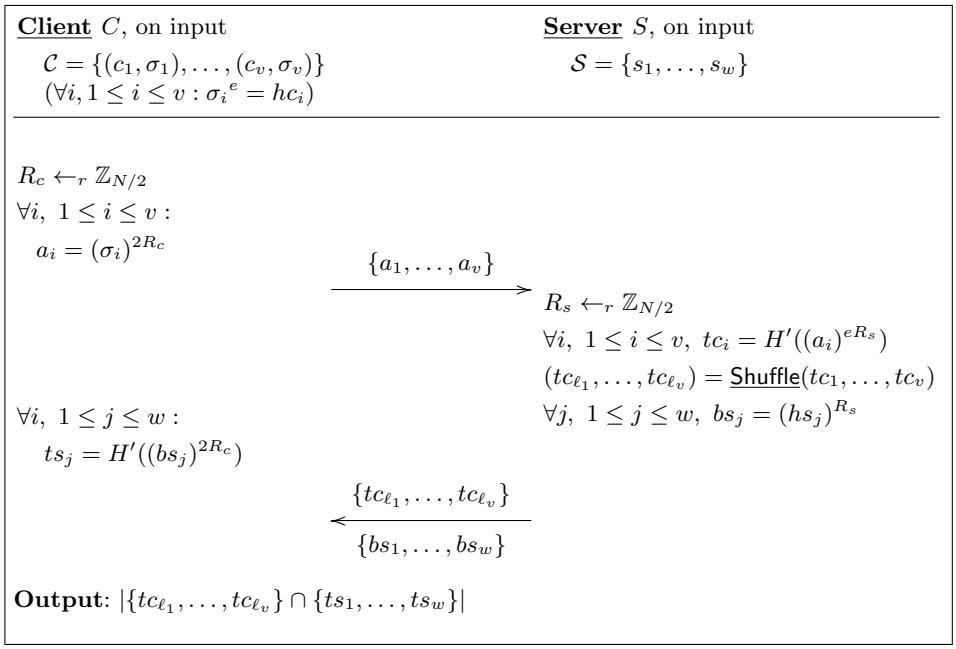

Figure 4: Authorized Private Set Intersection Cardinality. All Computation is mod N.

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 $\mathcal{S}$.
- *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 the server nor the client can determine if any two instances of the protocol are related, i.e., executed on the same input by the client or the server.

Proposed APSI-CA Construct. We illustrate our APSI-CA protocol in Figure 4. Note that the CA is responsible for generating public parameters: on input of security parameter κ, it executes $(N, e, d, g) \leftarrow RSA.KGen(\kappa)$, where g is a generator of QR_N, and selects $H : \{0,1\}^* \rightarrow {\mathbb{Z}_N}^*$ (Full-Domain Hash) and $H' : \{0,1\}^* \rightarrow \{0,1\}^\kappa$ (modeled as a 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 the client and the 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, observe that 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 the client and s_j held by the 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 Fig.1 of [8], 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, the client performs $O(w)$ exponentiations and the server – $O(w+v)$. 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 such a complexity is remarkably lower compared to related work, i.e., [5], which incurs quadratic—$O(wv)$—communication and computation overhead.