Size-Hiding in Private Set Intersection: Existential Results and Constructions

Abstract. In this paper we focus our attention on Private Set Intersection. We show some impossibility and existential results, and we provide some explicit constructions. More precisely, we start by looking at the case in which *both* parties, client and server, in securely computing the intersection, would like to hide the sizes of their sets of secrets, and we show that:

- It is impossible to realize an unconditionally secure size-hiding set intersection protocol.
- In a model where a Trusted Third Party provides set up information to the two parties and disappears, unconditionally secure size-hiding private set intersection is possible.
- There exist computationally secure size-hiding private set intersection protocols.

Then, we provide some explicit constructions for *one-side* protocols, where the client learns the intersection but *hides* the size of her set of secrets. In the model with a Trusted Third Party, we design two protocols which are computationally secure under standard assumptions, and two very efficient protocols which are secure in the random oracle model. We close the paper with some remarks and by pointing out several interesting open problems.

Keywords: Set operations, private set intersection, size hiding, impossibility result, protocols.

1 Introduction

The Private Set Intersection Problem. (PSI) revolves around two parties, each holding a set of inputs drawn from a ground set, that wish to jointly compute the intersection of their sets, without leaking *any* additional information [14]. In particular, cryptographic solutions to PSI allow interaction between a server S and client C, with respective private input sets $\mathcal{C} = \{c_1, \ldots, c_v\}$, $\mathcal{S} = \{s_1, \ldots, s_w\}$, both drawn from a ground set $\mathcal{U}$. At the end of the interaction, C learns $\mathcal{S} \cap \mathcal{C}$ and $|\mathcal{S}|$, while S learns nothing beyond $|\mathcal{C}|$.

Real-life applications of PSI include, e.g., the Department of Homeland Security that wishes to check its list of terrorists against the passenger list of a flight operated by a foreign air carrier, federal tax authorities wishing to check if any suspect tax evader has a foreign bank account and other folklore case scenarios [8].

Related work. Freedman et al. [14] introduced the first PSI protocol based on Oblivious Polynomial Evaluation (OPE). The key intuition is that elements in the client's private set can be represented as roots of a polynomial, i.e., $P(x) = \prod_{i=1}^{v}(x - c_i) = \sum_{i=1}^{v} a_i x^i$. Hence, leveraging any additively homomorphic encryption scheme (e.g., [20]) the encrypted polynomial is obliviously evaluated by S on each element of her data set. In particular, S computes $\{u_j\}_{j=1,\ldots,w} = \{\mathsf{Enc}(r_j P(s_j) + s_j)\}_{j=1,\ldots,w}$ where $\mathsf{Enc}()$ is the encryption function of the additively homomorphic encryption scheme and r_j is chosen at random. Clearly, if $s_j \in \mathcal{S} \cap \mathcal{C}$, then C learns s_j upon decryption of the corresponding ciphertext (i.e., u_j); otherwise C learns a random value. OPE-based PSI protocols have been extended in [18,9,10] to support multiple parties and other set operations (e.g., union, element reduction, etc.).

Hazay et al. [16] proposed Oblivious Pseudo-Random Function (OPRF) [13] as an alternative primitive to achieve PSI. In [16], given a secret index k to a pseudo-random function family, S evaluates $\{u_j\}_{j=1,\ldots,w} = \{f_k(s_j)\}_{j=1,\ldots,w}$ and sends it to C. Later, C and S engage in v executions of the OPRF protocol where C is the receiver with private input $\mathcal{C}$ and S is the sender with private input k. As a result, C learns $\{f_k(c_i)\}_{i=1,\ldots,v}$ such that $c_i \in \mathcal{S} \cap \mathcal{C}$ if and only if $f_k(c_i) \in \{u_j\}_{j=1,\ldots,w}$.

Given $\mathcal{U}$ as the ground set where elements of $\mathcal{C}$ and $\mathcal{S}$ are drawn (i.e., $\mathcal{C}, \mathcal{S} \subseteq \mathcal{U}$), none of the above techniques prevents a client to run a PSI protocol on private input $\mathcal{C} \equiv \mathcal{U}$ in order to learn all elements in $\mathcal{S}$. To this end, Camenisch et al. extended PSI to *Certified Sets* [7], where a Trusted Third Party (TTP) ensures that private inputs are valid and binds them to each participant.

All of the above techniques reveal the size of the participants' sets. That is, C (resp. S) learns $|\mathcal{S}|$ (resp. $|\mathcal{C}|$), even if $\mathcal{S} \cap \mathcal{C} \equiv \emptyset$. To protect the size of private input sets, Ateniese et al. [1] proposed so-called Size-Hiding PSI (SHI-PSI)

protocols where C learns $\mathcal{S} \cap \mathcal{C}$ without leaking the size of $\mathcal{C}$. Their scheme is based on RSA accumulators and the property that the RSA function is an unpredictable function. The authors proved its security against honest but curious adversaries in the Random Oracle Model (ROM).

Contributions. This paper builds on top of [1] and explores PSI protocols where parties hide the size of their private sets, under different security models. We start looking at *unconditionally secure* SHI-PSI where *both* parties hide the size of their sets. In this context, we show that SHI-PSI protocols where both the client and the server hide the size of their sets are not achievable, while this is possible for the authorized flavor of PSI, namely APSI, where a TTP certifies the inputs to the protocol.

Then we move to computational security and show that there exist an APSI protocol where both parties hide the size of their sets.

Finally, we provide some explicit constructions for *one-side* protocols, where only the client hides the size of her set. More precisely, leveraging a TTP that authorizes private inputs, we design two protocols which are computationally secure under standard assumptions, and two very efficient protocols which are secure in the random oracle model. The following table summarizes our results.

Result	Model	Size-Hiding	Assumption	Efficiency	Rounds
Impossible	Client/Server	Two-side	None	$\times$	$\times$
Prot. Figure 1	C/S with TTP	Two-side	None	NO	2
Prot. Figure 2	Client/Server	Two-side	Standard Model	NO	2
Prot. Figure 3	Client/Server	Two-side*	Standard Model	YES	2
Prot. Figure 4	C/S with TTP	One-side	Standard Model	YES	1
Prot. Figure 5	C/S with TTP	One-side	Standard Model	YES	2
Prot. Figure 6	C/S with TTP	One-side	Random Oracle Model	YES	3
Prot. Figure 7	C/S with TTP	One-side	Random Oracle Model	YES	1

* an upper bound on the sizes of both sets (client's and server's) is needed

2 Preliminaries: Definitions and Tools

In this section we provide definitions and tools used in the rest of the paper.

2.1 Definitions

We refer to the formalization used in [1]. However, we slightly refine the definitions in order to deal with both computationally and unconditionally secure protocols, and to introduce the size-hiding constraint on both client and server side. Moreover, we will also consider the setting in which a trusted third party interacts with the parties before the actual PSI protocol is run.

Definition 1. *A party is referred to as honest-but-curious,* HBC *for short, if it correctly follows the steps of the protocol but eventually tries to get extra-knowledge from the transcript of the execution.*

A two-side size-hiding private set intersection protocol, enabling the client to compute the intersection of his set of inputs with the set of the server, while keeping secret the sizes of their respective sets of inputs, can be defined as follows:

Definition 2. *A* TS-SHI-PSI *is a scheme involving two parties, C and S, with two components, $Setup$ and $Interaction$, where*

- *$Setup$ is an algorithm that selects all global parameters*
- *$Interaction$ is a protocol between S and C on respective private input sets $\mathcal{S} = \{s_1, \ldots, s_w\}$ and $\mathcal{C} = \{c_1, \ldots, c_v\}$, which are subsets of a ground set $\mathcal{U} = \{u_1, \ldots, u_{|\mathcal{U}|}\}$,*

satisfying correctness, client privacy and server privacy.

Correctness is formalized by:

Definition 3. *A* TS-SHI-PSI *is correct if, when both parties are* HBC*, at the end of* $Interaction$*, run on inputs* $\mathcal{S}$ *and* $\mathcal{C}$*, with overwhelming probability* S *outputs* $\perp$ *and* C *outputs* $\mathcal{S} \cap \mathcal{C}$ *or* $\perp$ *if the intersection is empty.*

Note that, compared to the definition of correctness provided in [1], we do not set $|\mathcal{S}|$ as part of the client's output. In Section 5, in which we consider protocols hiding only the size of the client's set, we adhere to the definition of correctness from [1] and require C to output $\mathcal{S} \cap \mathcal{C}$ and $|\mathcal{S}|$ or just $|\mathcal{S}|$ if the intersection is empty.

Concerning client privacy, since the server does not get any output from the protocol, it is enough to require that the server, from the interaction, does not distinguish between cases in which the client has different input sets.

Definition 4. *Let* $View_S(\mathcal{C}, \mathcal{S})$ *be a random variable representing* S*'s view during the execution of Interaction with inputs* $\mathcal{C}$ *and* $\mathcal{S}$*. A* TS-SHI-PSI *guarantees client privacy if, for every* S^* *that plays the role of* S*, for every set* $\mathcal{S}$*, and for any two possible client input sets* $\mathcal{C}_0, \mathcal{C}_1$ *it holds that:*

$$View_{S^*}(\mathcal{C}_0, \mathcal{S}) \equiv View_{S^*}(\mathcal{C}_1, \mathcal{S}).$$

Note that in the above definition, when considering the unconditional setting, the parties S, C and S^* are unbounded and indistinguishable means that the two views are *perfectly indistinguishable*, i.e., they are identically distributed. On the other hand, in the computational setting, S, C and S^* are PPT machines and, hence, indistinguishable means that the two views are *computationally indistinguishable*.

Server privacy needs a bit more: the client gets the output of the protocol and, by analyzing the transcript of the execution, could get extra knowledge about the server's private set. Nevertheless, if the transcript can be simulated by using only input and output, then server privacy is achieved.

Definition 5. *Let* $View_C(\mathcal{C}, \mathcal{S})$ *be a random variable representing* C*'s view during the execution of Interaction with inputs* $\mathcal{C}$ *and* $\mathcal{S}$*. Then, the* TS-SHI-PSI *scheme guarantees server privacy if there exists an algorithm* C^* *such that*

$$\{C^*(\mathcal{C}, \mathcal{S} \cap \mathcal{C})\}_{(\mathcal{C},\mathcal{S})} \equiv \{View_C(\mathcal{C}, \mathcal{S})\}_{(\mathcal{C},\mathcal{S})}.$$

As before, in the unconditional setting the parties are unbounded and the transcript produced by C^* and the real view need to be identically distributed. On the other hand, in the computationally secure setting, the parties are PPT machines and the transcript produced by C^* and the real view are required to be computationally indistinguishable.

In our paper we will also consider a model where a trusted third party (TTP) interacts with client and server during a setup phase and disappears. The TTP might provide secret information to the parties, as well as it may act as a certification authority for the sets of secrets held by either the client or theserver. The model we consider is essentially the model considered by Rivest [22]. We introduce the presence of the TTP by modifying Definition 2 as follows:

Definition 6. *A* TS-SHI-PSI-TTP *is a scheme involving a* TTP *and two parties,* C *and* S*, with four components,* $Setup$*,* $SetupC$*,* $SetupS$*, and* $Interaction$*, where*

- $Setup$ *is an algorithm that selects all global parameters*
- $SetupC$ *is a protocol between* TTP *and* C *on input* TTP *secret data and* $\mathcal{C} = \{c_1, \ldots, c_v\}$
- $SetupS$ *is a protocol between* TTP *and* S *on input* TTP *secret data and* $\mathcal{S} = \{s_1, \ldots, s_w\}$
- $Interaction$ *is a protocol between* S *and* C *on respective input sets* $\mathcal{S} = \{s_1, \ldots, s_w\}$ *and* $\mathcal{C} = \{c_1, \ldots, c_v\}$.

satisfying correctness, client privacy and server privacy.

Roughly speaking, TS-SHI-PSI-TTP is a TS-SHI-PSI where, during the setup phase, client and server, one after the other, interact with the TTP and get some private information which are to privately compute the intersection of their sets.

We remark that in this paper we will always consider HBC parties, even though the above Definitions 4 and 5 are written without that restriction.

2.2 Tools

In this subsection we briefly mention some of the tools we will use in protocol design.

Homomorphic Encryption. An encryption scheme is *additively homomorphic* if, for any two encryptions $E(m_1)$ and $E(m_2)$ of any two messages m_1 and m_2, it holds that $E(m_1) \cdot E(m_2) = E(m_1 + m_2)$, where $\cdot$ is the group operation on ciphertexts. By repeated application of the property, for any integer c, it follows that $E(m_1)^c = E(cm_1)$. Paillier's encryption scheme [20] is a semantically secure public-key encryption scheme which exhibits such properties. It is easy to check that the following claim, already used in previous work, holds: given encryptions $E(a_0), \ldots, E(a_k)$ of the coefficients $a_0, \ldots, a_k$ of a polynomial P of degree k, and knowledge of a plaintext value y, it is possible to compute $E(P(y))$, i.e., an encryption of $P(y)$.

Oblivious Transfer. An oblivious transfer protocol is a two-party protocol between a sender and a receiver. The sender has two secrets, s_0 and s_1, while the receiver is interested in one of them. Her choice is represented by a bit σ. After running the protocol the receiver only learns s_σ, while the sender does not learn which secret the receiver has recovered. Introduced by Rabin [21], and later on redefined in different equivalent ways, it is a key-tool in secure two-party and multy-party computation. We will denote this primitive as $OT(s_0, s_1, \sigma)$.

3 Two-Side Size-Hiding: The unconditional case

In this section we deal with the unconditionally secure setting and we show two results. We start showing that it is impossible to provide an unconditionally secure set intersection protocol where both parties hide the sizes of their sets of secrets. Later we show that, by a proper set up phase, performed by a trusted third party, such a protocol does exist.

3.1 Impossibility in the Plain Model

The impossibility result for the unconditional secure setting follows by putting together some known results. First of all, let us observe that an unconditionally secure PSI where both parties hide the sizes of their sets of secrets exists *only if* an unconditionally secure PSI (without the privacy-preserving requirement on the sizes of the sets) exists. Then, by ruling out the possibility of the latter, we rule out the possibility of the former. To this aim, note that, in [14], the authors described a *reduction from* OT *to* PSI (therein referred to as PM). Then, due to the results of Impagliazzo and Rudich [17], they concluded that there is no black-box reduction of set intersection from one-way functions. On the other hand, it is well known (see [4], page 22, for a clear description) that *unconditionally secure oblivious transfer is impossible*. Hence, due to the former reduction, it follows that unconditionally secure TS-SHI-PSI is impossible, and we get our claim. Details can be found in Appendix A. In conclusion, we show that:

Theorem 1. *Unconditionally secure TS-SHI-PSI schemes do not exist.*

3.2 Feasibility in the Model with a setup by a TTP

The presence of a TTP, which sets up the system and disappears, makes unconditionally secure size-hiding set intersection possible. We prove the following result:

Theorem 2. *Unconditionally secure TS-SHI-PSI-TTP schemes do exist.*

Essentially, the idea of the protocol which proves our claim is the following: the TTP chooses two random bijections $f, g : \mathcal{P}(\mathcal{U}) \longrightarrow \{0,1\}^{|\mathcal{U}|}$. Without loss of generality, we assume the client is the first party to intercat with the TTP. C sends $\mathcal{C}$to TTP and receives an *identifier* computed by using the first random function, and a list of *sub-identifiers*, one for each possible subset of $\mathcal{C}$, computed through the second random function.

On the other hand, when S intecats with the TTP and provide her private set $\mathcal{S}$, the TTP sends back a two-column table. The first column has, for each possible subset E of the ground set, an identifier of E, computed with the first

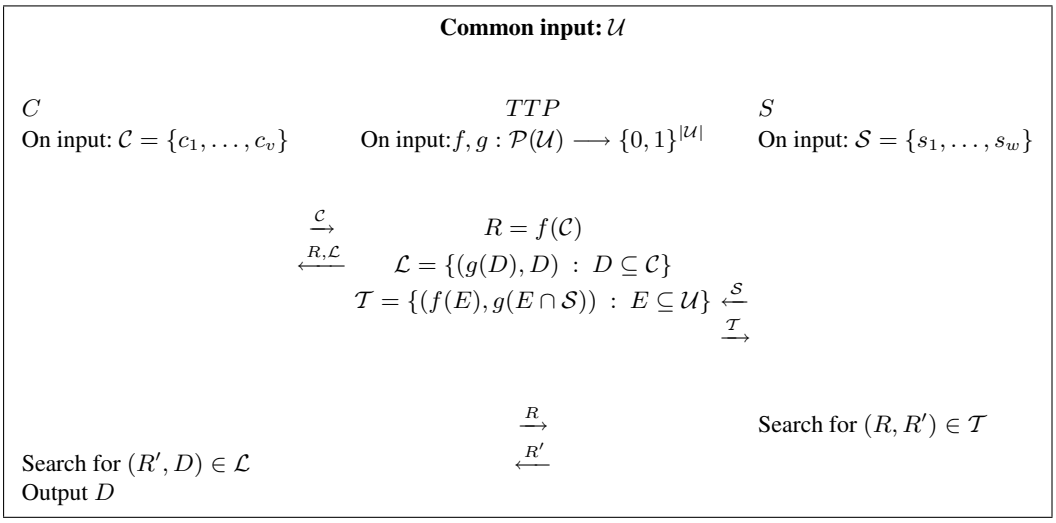

Fig. 1. Unconditionally secure protocol in the model with a TTP and unbounded parties.

random function; in the second there is an identifier, computed with the second random function, of the *intersection* $E \cap \mathcal{S}$.

The protocol between client and server is a simple two-round protocol: the client sends hes set identifier; the server looks up in the table the row with the received identifier, and sends back the identifier of the second column. Finally, the client looks up in the list of sub-identifiers and learns the intersection with the server. Details are given in Figure 1

It is easy to check that the protocol is correct. Similarly, the server privacy is unconditionally guaranteed: from the interaction the client only gets an identifier which determines a subset of his set of secrets. On the other hand, the client privacy needs a more accurate analysis. A condensed but rigorous argument is the following. First of all, notice that the server does not get any information about the correspondence value-subset, since the construction of the table is completely blind to her. Moreover, notice that, *independently of the client set of secrets*, the table the server gets has in the second column exactly $2^{|\mathcal{S}|}$ different random values, that is, the number of all possible subsets of $\mathcal{S}$. Each of these values appears exactly the same number of times, namely $2^{|\mathcal{U}|-|\mathcal{S}|}$. This follows from the fact that, for every $F \subseteq \mathcal{S}$,

$$\#\{E \subseteq \mathcal{U} \ : \ \mathcal{S} \cap E = F\} = \#\{F \cup E' \ : \ E' \subseteq \mathcal{U} \setminus \mathcal{S}\} = \#\{E' \ : \ E' \subseteq \mathcal{U} \setminus \mathcal{S}\} = 2^{|\mathcal{U}|-|\mathcal{S}|}$$

Hence, a request from a client only allows the server to learn the two values $(f(\mathcal{C}), g(\mathcal{C} \cap \mathcal{S}))$ which do not leak any information about the client private set of secrets or about its size.

4 Two-Side Size-Hiding: The computationally secure case

In this section we show that in the computational case, without a TTP, two-side size-hiding private set intersection is possible. The first construction is more or less an existential argument and has an interesting implication. The second can be useful in practice if the sizes of the sets of secrets are reasonable small and an upper bound is known a-priori.

4.1 An AND-based TS-SHI-PSI protocol

A private $AND(a, b)$ protocol is a two-party protocol, run by A and B, on private input a and b, respectively, at the end of which the players get the logical AND of their bits. It can be realized by using an $OT(b_0, b_1, s)$ protocol. Actually, it is enough to invoke the instance $OT(0, a, b)$, since the bit b_s that the receiver learns in an $OT(b_0, b_1, s)$

can be expressed as $b_s = (1 \oplus s)b_0 \oplus sb_1$. The key-idea underlying the protocol is that, if the set of secrets of C and S are represented by means of two characteristic vectors I_C and I_S of elements of $\mathcal{U}$ then, by running an $AND(I_{c_i}, I_{s_i})$ protocol for each bit of the vectors, C and S get the intersection and nothing else. Indeed, each $AND(I_{c_i}, I_{s_i}) = 1$ means that they share the i-th element of the ground set $\mathcal{U}$. Details are given in Figure 2.

Let n be a security parameter and let $\mathcal{U} = \{u_1, \ldots, u_{|\mathcal{U}|}\}$ be a ground set of size $poly(n)$. Assume that $\mathcal{C}$ (resp. $\mathcal{S}$) can be encoded in a characteristic vector I_C (resp. I_S), such that $I_C[j] = 1$ (resp. $I_S[j] = 1$) iff the $j - th$ element of $\mathcal{U}$ is in $\mathcal{C}$ (resp. $\mathcal{S}$).

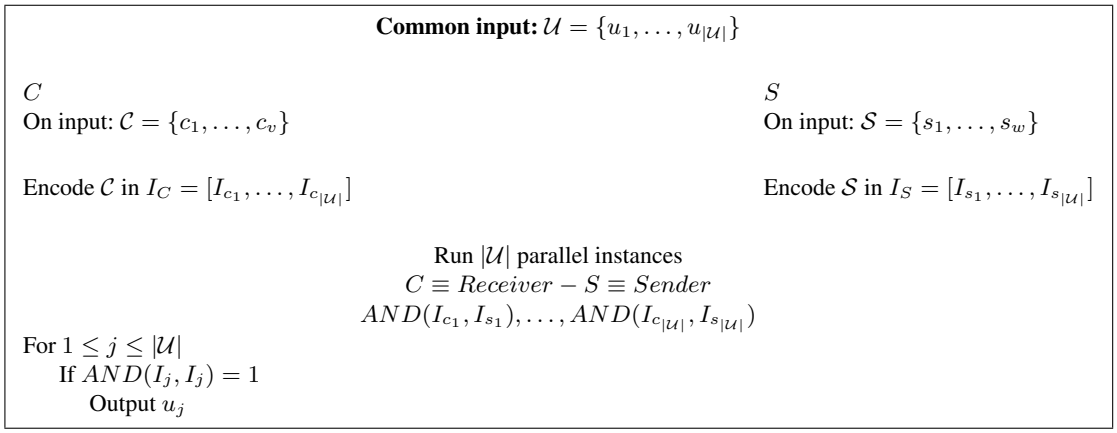

Fig. 2. A computationally secure size-hiding set intersection protocol.

It is easy to check that the protocol is correct. Moreover, it is secure as long as the AND protocol is secure. If we realize the AND protocol by using the OT construction proposed in [11] based on the existence of trapdoor permutations, since the executions are run by using independent randomness, we could use, for each execution, the simulator for the OT protocol. Thus, it is possible to show that the server cannot distinguish which set of secrets the client is using, and that there exists a simulator which, by using input and output of the client, provides transcripts which are indistinguishable from the real ones. Therefore, Definitions 4 and 5 are satisfied. More precisely, we prove the following result (details[1] and the proof can be found in Appendix B):

Theorem 3. *The protocol given in Figure 2, when instantiated with the OT protocol of Figure 8, realizes a computationally secure TS-SHI-PSI scheme.*

Remark. Notice that, since OT reduces to PSI but, as the protocol of Figure 2 shows, *also* PSI reduces to OT, it follows that OT and PSI are *equivalent*.

4.2 Threshold-based protocol

Assuming some a-priori information on the sizes of both sets $\mathcal{C}$ and $\mathcal{S}$ is known, more efficient protocols may be achieved. Here we make a proposal for the case when a known value M upper bounds the sizes of both client and server's sets. Actually, the smaller M is with respect to $|\mathcal{U}|$, the greater the interest of this construction (in particular, we need M of polynomial size but $|\mathcal{U}|$ may as well be exponential).

In a Setup phase, C generates public parameters $(params)$ and a key pair (sk, pk) for Paillier encryption. Let Enc and Dec be the encryption and decryption algorithms, respectively. C makes sure that the message space $\mathbb{Z}_n$, is

[1] Due to lack of space we provide a simplified (and somehow approximated) description. See [15] for a rigorous treatment of the protocol and the security analysis in the honest but curious model.

exponentially larger than $|\mathcal{U}|$. Further, she fixes an encoding of $\mathcal{U}$ into $\mathbb{Z}_n \setminus \{0\}$, denoted by $Encoding$. For the sake of readability, in Figure 3, elements of C and S are assumed to belong to $\mathbb{Z}_n \setminus \{0\}$. We will denote by π a random permutation of M elements.

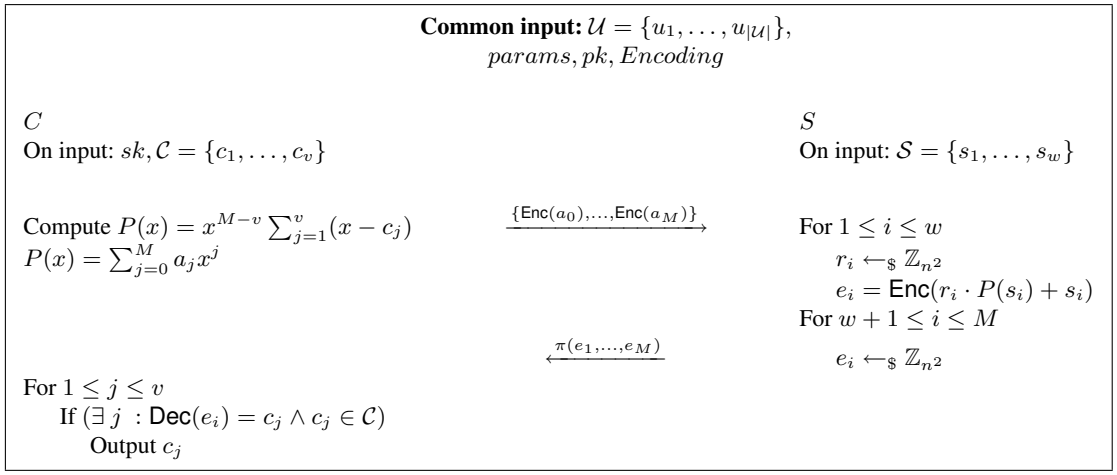

Fig. 3. Polynomial-based construction for $|\mathcal{C}|, |\mathcal{S}| \leq M$.

Our protocol is depicted in Figure 3, and it is actually a twist on the polynomial construction from [14], which main tool is a semantically secure (additively) homomorphic encryption scheme. As the authors of [14], we suggest to use Paillier encryption to this aim; further refinements of the protocol may of course be advisable if another encryption scheme is chosen. In the sequel, we set the notation $I := |\mathcal{C} \cap \mathcal{S}|$ and $L := w - I$.

Correctness. It is easy to see that the proposed protocol is correct, as the client's output is constructed by comparing his set $\mathcal{C}$ with the one consisting of $\mathcal{S} \cap \mathcal{C}$ plus the decryption of $M - I$ uniform random values from $\mathbb{Z}_{n^2}$. Namely, this sequence will consist of random values from $\mathbb{Z}_n$ which is exponentially larger than $\mathcal{U}$. As a result, the probability that they actually encode an element in $\mathcal{U}$ (disrupting thus the computation of the intersection) is negligible. [2]

Client Privacy. Due to the semantic security of Enc the distribution of $\{\mathsf{Enc}(a_0), \ldots, \mathsf{Enc}(a_M)\}$ is indistinguishable of that induced by selecting $M + 1$ elements independently and uniformly at random from $\mathbb{Z}_{n^2}$.

Server Privacy. In order to argue the existence of a pptm algorithm C^* which is able to simulate the clients view on input $\mathcal{C}$ and $\mathcal{C} \cap \mathcal{S}$, we modify C's view replacing the true input values from the server, constructed as encryptions involving values $s \in \mathcal{S} \setminus \mathcal{C}$ with encryptions of elements chosen uniformly and independently at random from $\mathbb{Z}_n \setminus \{0\}$. Consider thus the true distribution

$$\mathcal{D}_0 := \{\rho_0, \ldots, \rho_M, \mathsf{Enc}(r_{s_1} P(s_1) + s_1), \ldots, \mathsf{Enc}(r_{s_w} P(s_w) + s_w), \xi_1, \ldots, \xi_{M-w}\}$$

where for $i = 0 \ldots M$ each ρ_i denotes the random value involved in the Paillier encryption yielding $\mathsf{Enc}(a_i)$, namely, they are values chosen uniformly and independently at random from $\mathbb{Z}_n^*$, and

$$\mathsf{Enc}(r_{s_1} P(s_1) + s_1), \ldots, \mathsf{Enc}(r_{s_w} P(s_w) + s_w), \xi_1, \ldots, \xi_{M-w}\}$$

are constructed as in Figure 3 (w.l.o.g., we assume this sequence is not randomly permuted before output, and, moreover, that $\mathcal{S} \cap \mathcal{C} = \{s_1, \ldots, s_I\}$).

[2] At this, as we will do in the sequel, we are using the fact that Paillier encryption actually defines a trapdoor permutation from $\mathbb{Z}_n \times \mathbb{Z}_n^*$ into $\mathbb{Z}_{n^2}$. There is actually a negligible "loss" here, as we exclude 0 as a legitimate ciphertext.

Further, consider the distribution

$$\mathcal{D}_L = \{\rho_0, \ldots, \rho_M, \mathsf{Enc}(r_{s_1} P(s_1) + s_1), \ldots, \mathsf{Enc}(r_{s_I} P(s_I) + s_I), \nu_1, \ldots, \nu_L, \xi_1, \ldots, \xi_{M-w}\}$$

where $\nu_1, \ldots, \nu_L$ are elements chosen independently and uniformly at random from $\mathbb{Z}_{n^2}$.

Again from the semantic security of Enc it follows that this two distributions are computationally indistinguishable.

Efficiency. Having Paillier encryption in mind, we have designed the polynomial P in Step 2. of Round 1 (see 3), maximizing the number of its coefficients which are equal to zero (as encryptions of 0 with Paillier are cheap). That is the reason for excluding 0 from the domain when defining the encoding of $\mathcal{U}$ into $\mathbb{Z}_n$. Different refinements of this step may suit better if another encryption scheme is used, always ensuring that the resulting polynomial has no roots that may correspond to an encoding of an element outside $\mathcal{C}$ and yet in $\mathcal{U}$.

In [14], abstracting from the concrete homomorphic encryption scheme used, several modifications of their basic protocol are proposed in order to boost the efficiency, all of them geared towards reducing the number of products/exponentiations made by the server. Thus, their main goal is to avoid in the implementation the evaluation of polynomials with large exponents. This is first achieved by replacing the polynomial P computed by the client with several *low* degree polynomials with a nice trick consisting of hashing and using balanced allocations (see [14] for details). Further, the server evaluates the encrypted polynomials via Horner's rule, avoiding again *high* exponentiantions. Indeed, these ideas would also yield a significant speed up versus a naive implementation of our protocol.

5 One-Side Size-Hiding Set Intersection Protocols

In this section, we follow the spirit of [1] and provide *one-side* PSI protocols, i.e., protocols in which the client actually learns $|\mathcal{S}|$ from the interaction, while keeping $|\mathcal{C}|$ secret. We will thus, in the sequel, follow the definitions of correctness, client privacy and server privacy from [1].[3] In addition, our protocols also enjoy extra properties, some already considered in [1]. One of these properties is *unlinkability*, which holds for protocols in 5.2, 5.3 and 5.4. Another interesting one is a sort of *universal verifiability*, that is, any other party (another client who has received certified values from the TTP) can check the intersection of her set with $\mathcal{S}$ by having access to the values sent by the server in one protocol execution. This feature might be desirable in many client-server architectures where a multitude of clients interact with the same server. This last property holds for protocols in 5.1 and 5.4.

5.1 Pseudorandom function evaluation based protocols

First, we detail a protocol where, in the setup phase, a TTP obliviously evaluates a function (secretely chosen from a pseudorandom family) on the participants' inputs. This protocol follows the ideas in [16] but we delegate the function evaluation to the TTP. A description of the protocol is depicted in Figure 4. Here we make use of a pseudorandom function family $\{f_r\}_{r \in K}$ (with key set K) which can be evaluated in an oblivious way, that is, the TTP (holding the key r) learns nothing and a participant with input x learns $f_r(x)$.

Correctness. As f_r is a pseudorandom function, there is only negligible probability of that two different values from $\mathcal{U}$ are mapped to the same image, which is the only case in which C's computation of the intersection would not output $\mathcal{S} \cap \mathcal{C}$.

Client Privacy. This is straightforward to prove, as the client sends nothing to the server.

Server Privacy. This follows from the the pseudorandom property of the function family $\{f_r\}_{r \in K}$. Actually, a PPT machine C^* can simulate the client's view on input $\mathcal{C}, \mathcal{C} \cap \mathcal{S}$, and $|\mathcal{S}|$, by constructing a sequence $\{R_1^*, \ldots, R_w^*\}$ so that, for each $u_i \in \mathcal{C} \cap \mathcal{S}$, a corresponding R_i^* is defined as $f_r(u_i)$, while the rest are values chosen independently and uniformly at random in G.

[3] Correctness is defined including $|\mathcal{S}|$ as part of the client's output. The definition of client privacy matches the one we have given here for two-side protocols, while server's privacy ensures the client's view is polynomial-time simulatable on input $\mathcal{C}, \mathcal{S} \cap \mathcal{C}$ and $|\mathcal{S}|$ (the latter definition does not coincide with the one from [1], but $|\mathcal{S}|$ is needed in the simulation included in their proof).

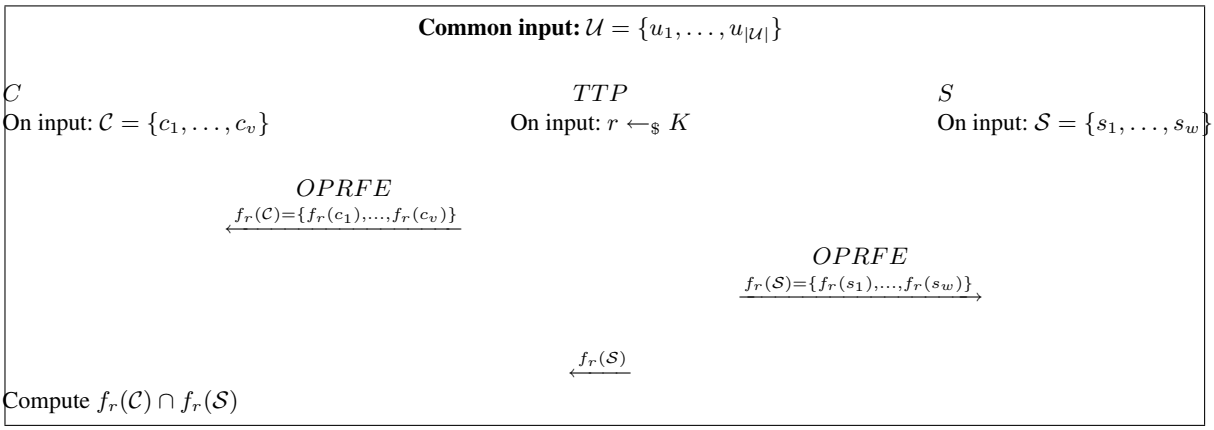

Fig. 4. One-Side protocol based on OPRFE.

The protocol of Figure 4 can be modified to achieve better efficiency, if the OPRFE is removed (see protocol from Appendix C, Figure 13); as a trade off, more trust on the TTP is required, as it learns the sets $\mathcal{C}$ and $\mathcal{S}$. The same is not possible in the protocols from [16] because client's privacy would be immediately lost. Finally, note that proofs of correctness and privacy for protocol in Figure 4 remain valid for protocol in Figure 13, as the only change is the way in which the pseudorandom function is evaluated.

In order to implement protocol of Figure 4 and Figure 13, the efficient proposal from [19] can be used. Let us briefly describe this pseudorandom function family: let G be a group of prime order q and g a generator of G for which the DDH assumption holds. We will denote by $K = (\mathbb{Z}_q)^n$, a set of keys for a function family. Elements in K have the form $r = (r_1, \ldots, r_n)$. The family of functions $\{f_r\}_{r \in K}$, defined as follows, is proven to be pseudorandom under the DDH assumption (see [19] for details)

$$f_r : \{0,1\}^n \longrightarrow G$$

$$f_r(x_1, \ldots, x_n) := g^{\prod_{i=1}^{n} x_i r_i} = g^{\prod_{x_i=1} r_i}.$$

Therefore, we will additionally need an encoding of the ground set $\mathcal{U}$ into the set $\{0,1\}^n$ for big enough n. Furthermore, the protocol proposed in [13] can be used to evaluate f_r in an oblivious way, suitable for protocol in Figure 4.

5.2 RSA-based protocol

Figure 5 shows a RSA based protocol along the lines of [1], but proven secure in the standard model. Once again, our construction makes use of a TTP which can go off-line after the Setup phase. Loosely speaking, the TTP uses a two-step process to map private input elements to values of a set V. During the first step, an element of the ground set is associated to an unpredictable value (RSA signatures); then, the TTP uses a strongly universal hash function to get *final* values, which are unrelated among each other and almost uniformly distributed over V. For details on strongly universal hash functions we refer to [3,23].

More precisely, in the Setup phase of the protocol, TTP executes an RSA key generation algorithm and keeps (N, e, d) private. We assume that elements of the ground set $\mathcal{U}$ are encoded as elements of $\mathbb{Z}_N^* \setminus \{1\}$. Further, TTP fixes a group $\mathbb{G}$ of prime order p (for p the smallest prime larger than N) and g a generator of $\mathbb{G}$. Finally, it selects and keeps private a strongly universal hash function $H : \mathbb{Z}_N^* \mapsto \mathbb{Z}_p$ from a given family, by selecting uniformly at random $a, b \in Z_p$ and setting $H_{a,b}(x) := ax + b \mod p$.

In the last round of the protocol, π denotes a permutation of w elements chosen u.a.r. by the server. An analogous choice is made in the schemes in subsections 5.3 and 5.4.

Before moving to the proof, let us recall the following:

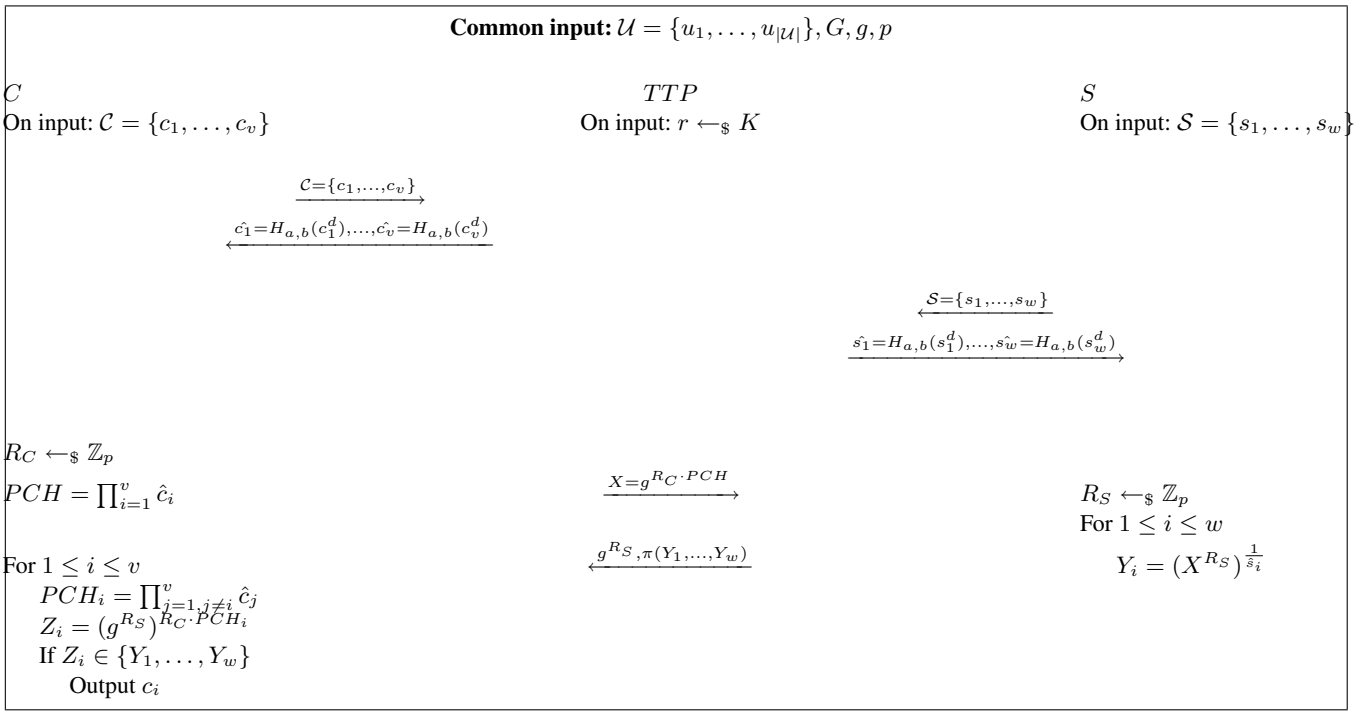

Fig. 5. RSA based protocol.

Definition 7. *A family $\mathcal{H} = \{h_s : \{0,1\}^\ell \rightarrow \{0,1\}^m\}$ of hash functions is called ϵ-almost strongly universal if and only if:*

1. $\forall a \in \{0,1\}^\ell, \forall b \in \{0,1\}^m$, *it holds that* $Pr_{s\in S}[h_s(a) = b] = 2^{-m}$
2. $\forall a_1 \neq a_2 \in \{0,1\}^\ell, \forall b_1, b_2 \in \{0,1\}^m$, *it holds that* $Pr_{s\in S}[h_s(a_2) = b_2 | h_s(a_1) = b_1] \leq \epsilon$.

Notice that, the first condition states that any input a is mapped to any hashed value b with probability $\frac{1}{2^m}$. A 2^{-m}-almost strongly universal hash function family $\mathcal{H}$ is called a strongly universal hash function family.

The function $H_{a,b}$ is used for two reasons: first, RSA signatures are malleable, e.g., from c_1^d and c_2^d it is immediate to compute the signature $(c_1c_2)^d$ for the product c_1c_2. Moreover, we cannot make any assumption on the distribution of the secrets $c_1, \ldots, c_v$ and $s_1, \ldots, s_w$. Since RSA is a permutation, then it preserves the input distribution. By post-processing the signatures we get randomized values through the hash function which are unrelated and almost uniformly distributed.

Correctness. Comes from the fact that, as H is a bijection, two different elements of the universe will never end up getting the same encoding from the TTP; as a result, the check up from the Client at the last step of the protocol will exactly result in the intersection.

Client Privacy. As X is the only public output of C, it suffices to argue that it is indistinguishable from a random group element from G. That is so, as G is cyclic of primer order and thus g^{PCH} generates G.

Server Privacy (Sketch). We again make use of the hybrid argument from [1], and gradually modify the client's view replacing values "outside" of $\mathcal{S} \cap \mathcal{C}$ with elements chosen uniformly and independently at random from G. Let $\mathcal{I} = \mathcal{S} \cap \mathcal{C}$ and $|\mathcal{I}| = t$ and consider distributions:

$$D_1 = \{R_C, g^{R_S}, Y_1, \ldots, Y_w\} \quad \text{and} \quad D_{w-t} = \{R_C, g^{R_S}, Y_1, \ldots, Y_t, R_1, \ldots, R_{w-t}\}$$

We assume w.l.o.g. that the elements from the intersection i.e., $Y_1, \ldots, Y_t$, come at the beginning in D_{w-t}, and they are constructed from the simulator exactly as in the real protocol; while $R_1, \ldots, R_{w-t}$ are generated as follows:

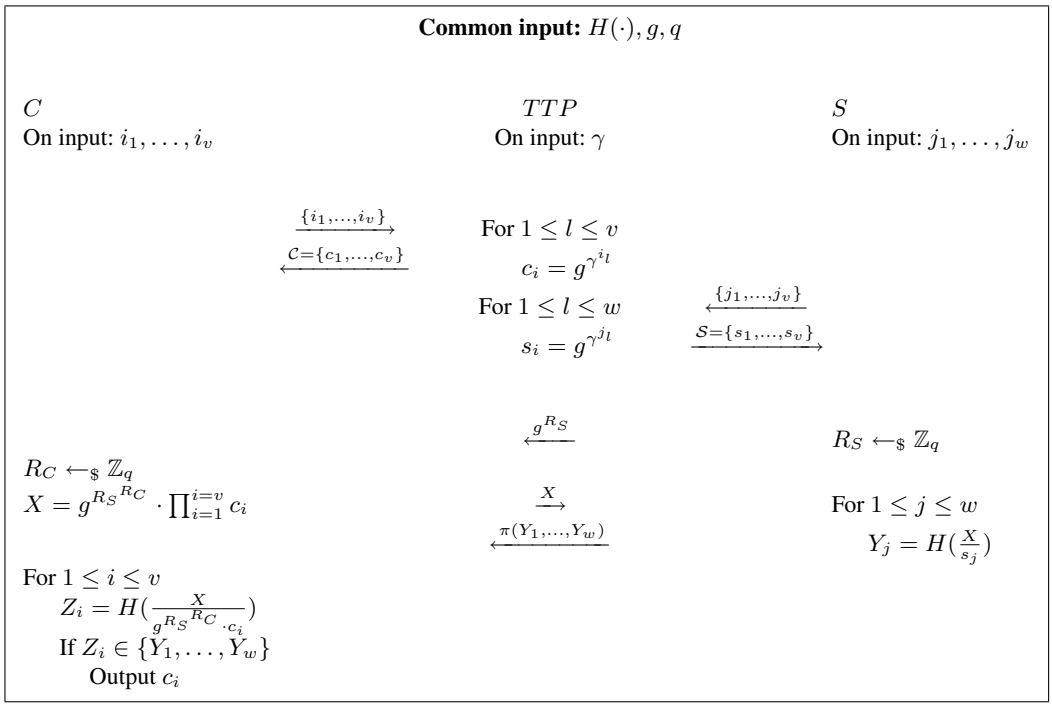

Fig. 6. Three-round Multiplicative protocol.

the simulator chooses $r_1, \ldots, r_{w-t}$ uniformly at random from $\mathbb{Z}_p$ and, for $i = 1, \ldots, w-t$, sets the value $R_i = ((g^{R_S})^{R_C PCH})^{\frac{1}{r_i}}$.

The difference between two distributions $\mathcal{D}_i$ and $\mathcal{D}_{i+1}$, for $i = 1, \ldots, w-t-1$, is the replacement of the i-th element; hence, a distinguisher between them should be able to distinguish an element of the form $(X^{R_S})^{\frac{1}{\hat{s}_j}} = ((g^{R_C PCH})^{R_S})^{\frac{1}{\hat{s}_j}}$ from the element $R_j = ((g^{R_S})^{R_C PCH})^{\frac{1}{r_j}}$, constructed by the simulator. However, due to the first property of a strongly universal hash function, the value $\hat{s}_j$ is associated to the input value s_j with probability $1/p$. Similarly, r_j is choosen uniformly at random, i.e., with probability $1/p$. Therefore, both $(X^{R_S})^{\frac{1}{\hat{s}_j}}$ and R_j are *uniformly distributed* over G. Hence, no PPT distinguisher can distinguish between them.

5.3 Three-round ROM based protocol

This protocol is inspired by [5]. The TTP chosen a full-domain hash function [2] $H(\cdot)$, sets up a group $\mathbb{G}$ of prime order q and randomly picks $\gamma \in \mathbb{Z}_q$ as her secret key. Elements of the ground set are encoded as integers in $\mathbb{Z}_q$. Element $x \in \mathbb{Z}_q$ is certified as g^{γ^x}. Let $i_1, \ldots, i_v$ and $j_1, \ldots, j_w$ be the set of (non-certified) elements held by C and S, respectively. The protocol is depicted in Figure 6. Both C and S get their elements certified by the TTP; S also uses a random permutation π. Later they interact so that C outputs $\mathcal{C} \cap \mathcal{S}$ and $|\mathcal{S}|$.

Client Privacy. During the protocol execution, C sends X to the server. Since R_C is chosen uniformly at random from $\mathbb{Z}_q$, and since G is a cyclic group of prime order q, then X is uniformly distributed element over G.

Server Privacy (Sketch). We use the same argument used before, that is, the simulator reproduces the client view from his input and the output of the protocol, by replacing values "outside" of $\mathcal{S} \cap \mathcal{C}$ with elements *chosen uniformly and independently at random*. Let $\mathcal{I} = \mathcal{S} \cap \mathcal{C}$ and $|\mathcal{I}| = t$ and consider the distributions:

$$D_I = \{(R_C, T) : R_C \leftarrow_{\$} \mathbb{Z}_q, T = (H(\frac{X}{s_1}), \ldots, H(\frac{X}{s_w}))\}$$

and

$$D_{w-t} = \{(R_C, T) : R_C \leftarrow_{\$} \mathbb{Z}_q, T = (H(\frac{X}{s_1}), \ldots, H(\frac{X}{s_t}), r_{t+1}, \ldots, r_w)\}$$

where $s_1, \ldots, s_w \in \mathcal{I}$ and $r_{t+1}, \ldots, r_w$ are values chosen uniformly at random. Since $H(\cdot)$ is a random oracle, there exists no distinguisher which is able to take apart D_I from D_{w-t}.

Remarks. As pointed out at the beginning of the section, the above protocol enjoys *unlinkability* by having the server contributing randomness at each run. If unlinkability *is not required* the protocol can be reduced to a two-round protocol, i.e., the first round is removed. In this case, the client might mask the product of her input with $R_C \leftarrow_{\$} \mathbb{G}$, i.e., computing $X = R_C \cdot \prod_{i=1}^{i=v} c_i$. Hence, the protocol would require no exponentiations and gain in efficiency.

5.4 One-round ROM based protocol

Figure 7 shows a very efficient protocol that allows for one-side SHI-APSI in the ROM. The TTP uses an RSA signature scheme $Sign(\cdot)$ to certify elements; the secret signing key is sk.

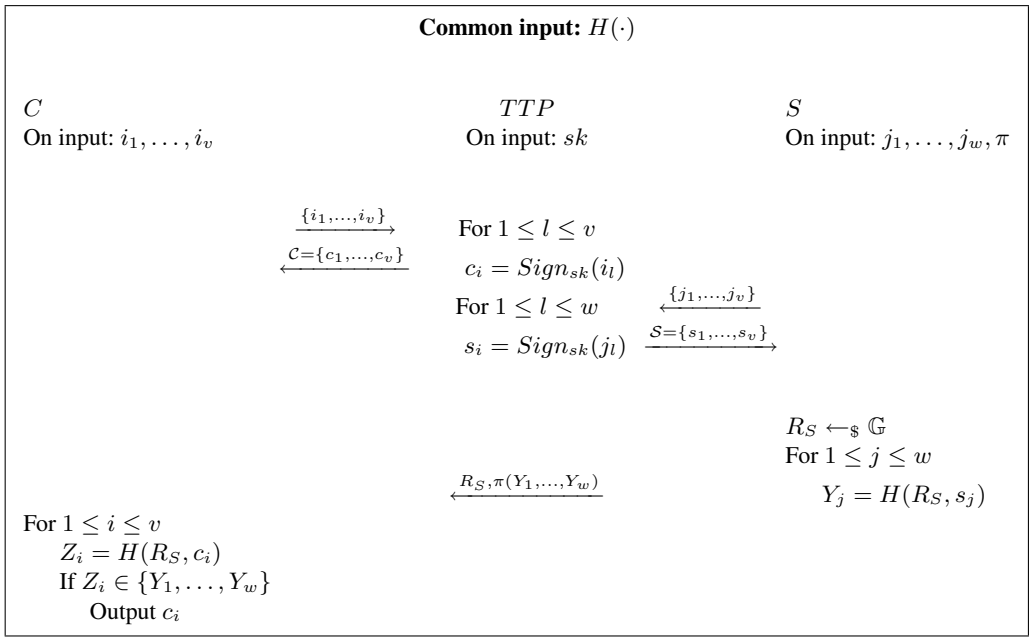

Fig. 7. One-round ROM based protocol.

Client Privacy. During the protocol C does not provide any input so its privacy is trivially preserved.

Server Privacy (Sketch). Since hash functions are modeled as a random oracle, the simulator reproduces the client view from his input and the output of the protocol, by replacing values "outside" of $\mathcal{S} \cap \mathcal{C}$ with elements *chosen uniformly and independently at random*. As before, we argue no distinguisher D can tell apart a real transcript from a simulated one.

Remarks. The protocol in Figure 7 enjoys *unlinkability* since the server picks, at each run, a random (R_S) that masks values s_j-s in the computation of Y_j-s. It also satisfies the kind of *universal verifiability* which was described at the beginning of the section.

6 Conclusions and Open Problems

This paper investigates Size-Hiding Private Set Intersection protocols and provides impossibility and existential results. Unconditionally secure two-side size-hiding private set intersection is impossible without assumptions, but it is possible with a setup phase performed by a TTP and unbounded parties. Computationally secure two-side size-hiding private set intersection protocols do exist. Moreover, we have provided some explicit constructions secure under standard assumptions or in the random oracle model. Several interesting open problems are left: concerning two-side size-hiding private set intersection, we miss an *efficient* unconditionally secure protocol in the model with a TTP and a computationally secure protocol which does not consider the whole ground set (or an impossibility result in that respect). Moreover, it is of interest to study the same problem in the malicious setting and to consider the extension of the problem to n parties.

References

1. G. Ateniese, E. De Cristofaro, and G. Tsudik, *(If) Size Matters: Size-Hiding Private Set Intersection*, PKC 2011, LNCS, Vol. 6571, pp. 156-173, 2011.
2. M. Bellare and P. Rogaway, *The Exact Security of Digital Signatures - How to Sign with RSA and Rabin*, EUROCRYPT 1996, LNCS, Vol. 1070, pp. 399416, 1996.
3. J.L. Carter and M.N. Wegman, *Universal classes of hash functions*, Journal of Computer and System Sciences, Vol. 18, 143154, 1079.
4. R. Cramer, *Introduction to Secure Computation*, Lectures on Data Security, LNCS, Vol. 1561, pp. 16-62, 1999.
5. J. Camenisch, M. Kohlweiss and C. Soriente, *An Accumulator Based on Bilinear Maps and Efficient Revocation for Anonymous Credentials*, PKC 2009, LNCS, Vol. 5443, pp. 481-500, 2009.
6. J. Camenisch and A. Lysyanskaya, *A Signature Scheme with Efficient Protocols*, SCN 2002, LNCS, Vol. 2576, pp., 268-289, 2003.
7. J. Camenish and G. M. Zaverucha, *Private Intersection of Certified Sets*, FC 2009, LNCS, Vol. 5628, pp., 108-127, 2009.
8. E. De Cristofaro and G. Tsudik, *Practical Private Set Intersection Protocols with Linear Complexity*, FC 2010, LNCS, Vol. 6052, pp. 143-159, 2010.
9. D. Dachman-Soled, T. Malkin, M. Raykova and M. Yung, *Efficient Robust Private Set Intersection*, 7th International Conference on Applied Cryptography and Network Security (ACNS), Vol. , pp. 125-142, 2009.
10. D. Dachman-Soled, T. Malkin, M. Raykova and M. Yung, *Secure Efficient Multiparty Computing of Multivariate Polynomials and Applications*, 9th International Conference on Applied Cryptography and Network Security (ACNS), Vol. , pp. 130-146, 2011.
11. S. Even, O. Goldreich, and A. Lempel, *A Randomized Protocol for Signing Contracts*, Communications of the ACM, Volume 28, Issue 6, pp. 637-647, 1985.
12. K. Frikken, *Privacy-Preserving Set Union*, ACNS 2007, Vol. , pp. 237-252, 2007.
13. M. J. Freedman, Y. Ishai, B. Pinkas and O. Reingold, *Keyword Search and Oblivious Psudorandom Functions*, TCC 2005, LLNC, Vol. 3378, pp. 303-324, 2005.
14. M. J. Freedman, K. Nissim, and B. Pinkas, *Efficient Private Matching and Set Intersection*, Eurocrypt 2004, LNCS, Vol. 3027, pp. 1-19, 2004.
15. O. Goldreich, *Foundations of Cryptography - Volume II Basic Applications*, Cambridge Press, 2004.
16. C. Hazay and Y. Lindell, *Efficient Protocols for Set Intersection and Pattern Matching with Security Against Covert Adversaries*, TCC 2008, LNCS, Vol. 4948, pp. 155 - 175, 2008.
17. R. Impagliazzo and S. Rudich, *Limits on the provable consequences of one-way permutations*, Proc. of the 21st Annual ACM Symposium on Theory of Computing, pp. 44-61, Seattle, Washington, May 1989.
18. L. Kissner and D. Song, *Privacy-Preserving Set Operations*, Crypto 2005, LNCS, Vol. 3621, pp. 241-257, 2005.
19. M. Naor and O. Reingold, *Number-theoretic constructions of efficient pseudo-random functions*, Journal of the ACM, Vol. 51, No. 2, pp. 231-262, 2004.
20. P. Pailler, *Public-key Cryptosystems based on composite degree residuosity classes*, Crypto 1999, LNCS, Vol. 1592, pp. 223-239, 1999.
21. M. Rabin, *How to exchange secrets by oblivious transfer*, Technical Report TR-81, Aiken Computation Laboratory, Harvard University, 1981.

22. R. Rivest, *Unconditionally Secure Commitment and Oblivious Transfer Schemes Using Private Channels and a Trusted Initializer*, unpublished manuscript, 11/8/1999 available at http://people.csail.mit.edu/rivest/publications.html
23. D. R. Stinson, *Universal hash families and the leftover hash lemma, and applications to cryptography and computing*, J. Combin. Math. Combin. Comput. Vol. 42, 3-31, 2002.

A Sketch of the Proof of Theorem 1

Let us start by describing the reduction from OT to PSI.

Reduction from OT to PSI. Let us assume that the sets of secrets of client and server have both two elements. It is possible to design a 1-out-of-2 bit OT protocol secure against honest but curious adversaries. In the OT protocol, the sender inputs are two bits, b_0 and b_1, while the receiver input is the bit σ. At the end of the protocol the receiver learns b_σ and nothing else, while the sender learns nothing. First the parties generate their respective inputs for the PSI protocol. The sender generates his set as $\{0|b_0, 1|b_1\}$, and the receiver generates the list $\{\sigma|0, \sigma|1\}$. Then, they run the PSI protocol, at the end of which the receiver learns $\sigma|b_\sigma$, from which she extracts b_σ. For example, if $b_0 = 0, b_1 = 1$ and $\sigma = 1$, then the set of secrets of the sender is $\{0|0, 1|1\}$ and the set of secrets of the receiver is $\{1|0, 1|1\}$. From the PSI protocol, the receiver gets $1|1$, and, hence, $b_1 = 1$.

Then, let us show why unconditionally secure OT is impossible.

Unconditionally Secure OT is impossible. Let us denote by $AND(a, b)$ a two-party protocol, run by A and B, at the end of which the players gets the logical AND of their bits and nothing else (i.e., a secure protocol for computing $a \cdot b$). For example if A's input is $a = 0$, then $a \cdot b = 0$ but, running the protocol, apart the result, A will never know whether B's input is $b = 0$ or $b = 1$. Moreover, let us denote by $OT(b_0, b_1, s)$ an OT protocol where the sender has two secret bits b_0 and b_1 and the receiver has the secret bit s. Notice that the output b_s the receiver gets from the $OT(b_0, b_1, s)$ protocol can be written as $b_s = (1 \oplus s)b_0 \oplus sb_1$, where the operations are the usual multiplication and addition of bits. Therefore, it is easy to check that $OT(0, a, b) = ab = AND(a, b)$. In other words, A and B can compute the logical AND of their bits by using the OT protocol: A uses as secret bits $0, a$ and B the bit b. Hence, if an unconditionally secure OT protocol exists, then an unconditionally secure AND protocol exists. By ruling out the possibility of the latter, we rule out also the possibility of the former.

It is possible to show that an unconditionally secure protocol for the logical AND does not exist, arguing as follows: the two programs held by A and B, given random strings and the input bits, operate deterministically. Moreover, they communicate over a perfect channel, the players have *infinite* computing power, and the protocol is *perfectly* correct and always halts. Let T be the sequence of messages exchanged in a complete protocol execution. Let us call T the *transcript*. Let $r_A \in \{0, 1\}^*$ and $r_B \in \{0, 1\}^*$ be the respective random strings used by A and B in the computation. We show that an honest but curious A having bit $a = 0$ can *always* figure out the bit b, held by B, thus contradicting the security conditions. Indeed, notice that:

- If $b = 0$, then *there exists* a random string r'_A such that the transcript T is consistent also with A having input $a = 1$ instead of $a = 0$. The existence of this string is guaranteed by the security property of the protocol, which states that B, having $b = 0$, running the protocol has no information about A's input. Clearly A, given the transcript T and fixing $a = 1$, since is computationally unbounded, can check for all possible random string $z \in \{0, 1\}^*$ and eventually find the string r'_A.
- If $b = 1$, then *it is impossible* that T is consistent also with $a = 1$ for some random string r''_A because, in this case, flipping A's input from 0 to 1, changes the logical AND $a \cdot b$ of the inputs. Since the protocol is perfectly correct, then T cannot be consistent with two pairs of inputs (a, b) whose respective logical AND is different.

Therefore, A decides that $b = 0$ if there exists a random string r'_A such that the transcript T is consistent with r'_A and $a = 1$, while decides that $b = 1$ if no such random string exists.

B Proof of Theorem 3

Let us start by describing the OT protocol used to compute the AND function.

Protocol steps (simplified description):

Let $b_0, b_1 \in \{0,1\}$ be the Sender secret bits, and let $i \in \{0,1\}$ be the Receiver choice. Moreover, let $\mathcal{F} = \{f_\alpha : D_\alpha \to D_\alpha\}_{\alpha \in \{0,1\}^{poly(n)}}$ be a family of trapdoor permutations.

- The Sender uniformly selects a trapdoor pair (α, t) and sends α to the receiver.
- The Receiver, uniformly and independently, selects $e_0, e_1 \in D_\alpha$, sets $y_i = f_\alpha(e_i)$ and $y_{1-i} = e_{1-i}$, and sends y_0, y_1 to the Sender.
- Using the trapdoor t and the inverting algorithm, for $j = 0, 1$, the sender computes

$$x_j = f_\alpha^{-1}(y_j) \text{ and } c_j = b_j \oplus b(x_j),$$

  where $b(\cdot)$ is a hardcore predicate for f_α. Then, he sends c_0, c_1 to the Receiver.
- The Receiver computes $b_i = c_i \oplus b(e_i)$.

Fig. 8. OT protocol based on trapdoor permutations.

OT Protocol based on trapdoor permutations.

It is easy to check that the protocol is correct. Indeed:

$$c_i \oplus b(e_i) = (b_i \oplus b(x_i)) \oplus b(e_i) = (b_i \oplus b(f_\alpha^{-1}(f_\alpha(e_i)))) \oplus b(e_i) = b_i.$$

Intuitively the protocol is secure because the Sender, indipendently of the bit i held by the Receiver, gets uniformly distributed values of D_α. Indeed, $y_{1-i} = e_{1-i}$ is chosen uniformly at random in D_α and $y_i = f_\alpha(e_i)$ is still uniformly distributed in D_α since e_i is chosen uniformly at random and f_α is a permutation on D_α. On the other hand, the Receiver does not get knowledge about the bit b_{1-i} because, since $b(\cdot)$ is a hardcore predicate for f_α, from the triplet $(\alpha, e_{1-i}, c_{1-i} = b_{1-i} \oplus b(x_{1-i}))$ it is infeasible to predict b_{1-i} better than at random. The above intuitions can be formalised by the simulators for the views of the sender and of the receiver given in Figures 9 and 10.

Sim_S: Simulator for the sender:

Input for the simulator: $b_0, b_1 \in \{0,1\}, \perp$

- Selects uniformly at random α and values $y_0, y_1 \in D_\alpha$.
- Outputs the values b_0, b_1, y_0 and y_1.

Fig. 9. Simulator for the sender

Notice that the output of the simulator in Figure 9 is *identically* distributed to the view of the Sender in a real execution of the protocol. Similarly, the output of the simulator in Figure 10, apart from c_{1-i}, is identically distributed to the Receiver view in a real execution of the protocol. The only difference is the computation of c_{1-i}. However, it is infeasible to distinguish between the two cases since the distinguisher does not get the trapdoor t and $b(\cdot)$ is a hardcore predicate.

By using the above simulators, we construct simulators for the TS-SHI-PSI protocol of Figure 2. See Figures 11 and 12.

Notice that, the view produced by Sim_{Server} is *identically* distributed to the view of the Sender in a real execution of the protocol and it is *independent of* the set of secrets held by the Client. Then, it easily follows that the Server does not distinguish between two executions in which the Client has two different sets of secrets.

Sim_R: Simulator for the receiver:

Input for the simulator: $i \in \{0,1\}, b_i \in \{0,1\}$

- Selects uniformly at random a trapdoor pair (α, t).
- Selects uniformly and independently $e_0, e_1 \in D_\alpha$ and computes $y_i = f_\alpha(e_i)$ and $y_{1-i} = e_{1-i}$.
- Sets $c_i = b_i \oplus b(e_i)$, and selects $c_{1-i} \in \{0,1\}$ uniformly at random.
- Outputs $(i, \alpha, e_0, e_1, c_0, c_1)$.

Fig. 10. Simulator for the receiver

Sim_{Server}: Simulator for the Server:

Input for the simulator: $I_S \in \{0,1\}^{|\mathcal{U}|}, \perp$

- For $i = 1, \ldots, |\mathcal{U}|$, run Sim_S on input $0, I_{s_i} \in \{0,1\}, \perp$
- Outputs the concatenation of the outputs of the $|\mathcal{U}|$ executions of Sim_S.

Fig. 11. Simulator for the Server

Sim_{Client}: Simulator for the Client:

Input for the simulator: $I_C \in \{0,1\}^{|\mathcal{U}|}, I_Z$

- For $i = 1, \ldots, |\mathcal{U}|$, run Sim_R on input $I_{c_i} \in \{0,1\}, I_{z_i}$
- Outputs the concatenation of the outputs of the $|\mathcal{U}|$ executions of Sim_R.

Fig. 12. Simulator for the Client

The view produces by Sim_{Client} is *indistinguishable* from the real view the Client gets in a run of the protocol. Indeed, it they were, it would be possible to show that there exists an efficient distinguisher which, with non-negligible probability, is able to distinguish the view of the Receiver in a run of the OT protocol from the transcript produced by Sim_R, and we get a contradiction.

C One-Side protocol based on (non oblivious) PRFE.

Let $\{f_r\}_{r \in K}$ be a pseudorandom function family (with key set K).

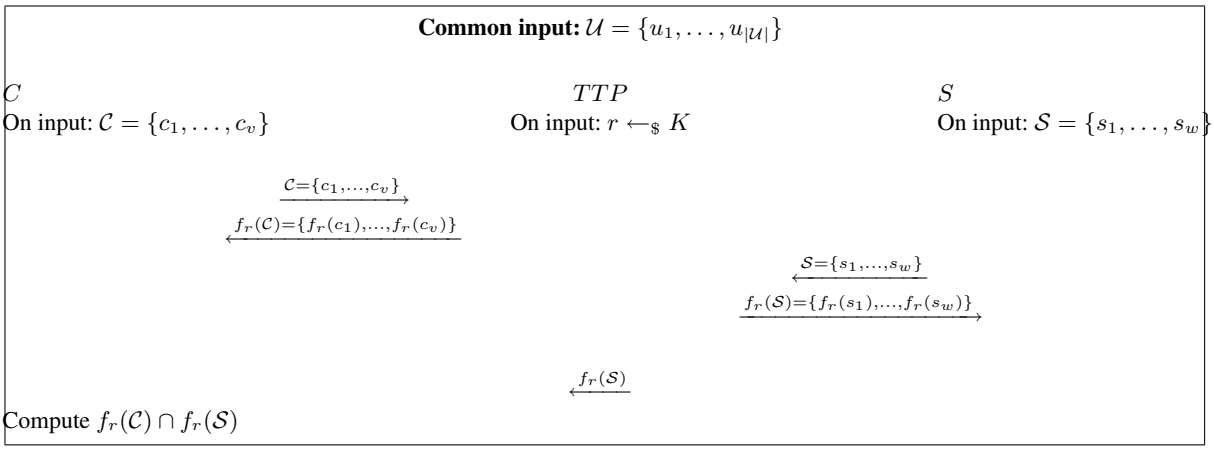

Fig. 13. One-Side protocol based on (non-oblivious) PRFE.