Size-Hiding in Private Set Intersection:

what can be done and how to do it without random oracles

Paolo D'Arco [*] María Isabel González Vasco [†] Ángel L. Pérez del Pozo [†]

Claudio Soriente [‡]

April 9, 2015

Abstract

In this paper we focus our attention on private set intersection protocols, through which two parties, each holding a set of inputs drawn from a ground set, jointly compute the intersection of their sets. Ideally, no further information than which elements are actually shared is compromised to the other party, yet the input set sizes are often considered as admissible leakage. Considering the (more restricted) size-hiding scenario, we are able to

- prove that it is impossible to realize an unconditionally secure set intersection protocol (size-hiding or not);
- prove that unconditionally secure size-hiding set intersection is possible in a model where a set up authority provides certain information to the two parties and disappears;
- provide several new computationally secure size-hiding set intersection protocols.

Regarding the latter, in particular we provide a new generic construction without random oracles for the *unbalanced* setting, where only the client *gets* the intersection and *hides* the size of its set of secrets. The main tool behind this design are smooth projective hash functions for languages derived from perfectly-binding commitments. We stand on the seminal ideas of Cramer–Shoup and Gennaro–Lindell, which have already found applications in several other contexts, such as password-based authenticated key exchange and oblivious transfer.

1 Introduction

The Private Set Intersection (PSI) problem, in a nutshell, concerns with 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 [26]. In particular, cryptographic solutions to the PSI problem allow interaction between a client C and a server S, with respective private input sets $\mathcal{C} = \{c_1, \ldots, c_v\}$ and $\mathcal{S} = \{s_1, \ldots, s_w\}$, both drawn from the 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 $|\mathcal{C}|$.

[*] Dipartimento di Informatica, University of Salerno, Italy. Email: pdarco@unisa.it

[†] Departamento de Matemática Aplicada, Ciencia e Ingeniería de los Materiales y Tecnología Electrónica, Universidad Rey Juan Carlos, Spain. Email: {mariaisabel.vasco, angel.perez}@urjc.es

[‡] Department of Computer Science, ETH Zurich, Switzerland. Email: claudio.soriente@inf.ethz.ch

Plenty of applications have been discussed since its introduction: 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, the federal tax authority wishing to check if any suspect tax evader has a foreign bank account and other folklore case scenarios have been the leading examples in many papers (see [20] for a detailed list). Remarkably, in many of these scenarios it is in the interest of the client (and maybe also of the server), that the size of its input set is not leaked through the interaction. Consider for instance the case in which two competing companies wish to find out the set of common clients in a certain geographical area, while they might rather not disclose to what extent they are already established there. If size-hiding is a requirement, many of the techniques proposed for private set intersection fail to do the job.

History of PSI: Results and Techniques. Freedman et al. [26] 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., [39]), the encrypted polynomial is obliviously evaluated by S on each element of its data set. In particular, S computes $\{u_j\}_{j=1,\ldots,w} = \{E(r_j P(s_j) + s_j)\}_{j=1,\ldots,w}$ where $E()$ 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 [35, 24, 21, 22] to support multiple parties and other set operations (e.g., union, element reduction, etc.).

Hazay et al. [31] proposed Oblivious Pseudo-Random Function (OPRF) [25] as an alternative primitive to achieve PSI. In [31], 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 $f_k(c_i) \in \{u_j\}_{j=1,\ldots,w}$. Improvements by using the same approach were provided in [29, 30].

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 the elements in $\mathcal{S}$. To this end, Camenisch et al. extended PSI to *Certified Sets* [15], where a Trusted Third Party (TTP) ensures that private inputs are valid and binds them to each participant. The certification issue was also addressed in [18], where a related problem to PSI was considered. Moreover, the same extension, under a different name, *Authorised PSI*, was considered in [20], where protocols that use modular exponentiation, multiplication and hash evaluation were described.

Efficiency of the protocols on large data sets is an important practical issue, and it has been addressed in several papers in the last years. One of the most relevant applications in this respect come from the bursting field of genomic computing: the user wants to protect the privacy of sensitive information coded in her genomic sequence but at the same time wishes to engage in private computations with other parties, in order to get some advantage, e.g., understand whether she has a predisposition to certain diseases or whether some medicines could be useful to improve her state of health. The interested reader is referred to [3] and [6] (and the references therein quoted) for an in-depth introduction to the area. In [19] linear-complexity private set intersection protocols were proposed for malicious adversaries. Along the line of [37], to gain efficiency, Bloom filters have been applied in [34, 17]. The protocols proposed in [34] are elegant and one of them is designed for an outsourced scenario. On the other hand, the protocols described in [17] and their optimizations suggested in [40] are currently, with respect to semi-honest adversaries, the most efficient available solutions on the market.

Related work. 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. [2] proposed a size-hiding PSI protocol where C can privately learn $\mathcal{S} \cap \mathcal{C}$ and $|\mathcal{S}|$ 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.

Very recently, general results on hiding the input-size in two party computation were given in [36]; at this, the authors prove that hiding the input size of both participants involved in a set intersection protocol is impossible in the cryptographic model, considering honest but curious adversaries, if after the interaction:

- both C and S get $\mathcal{S} \cap \mathcal{C}$ and nothing else,
- client only gets $\mathcal{S} \cap \mathcal{C}$, server gets nothing
- client gets $\mathcal{S} \cap \mathcal{C}$, server gets its size, $|\mathcal{S} \cap \mathcal{C}|$.

Further, they are able to evidence that hiding one party's input size is possible using fully homomorphic encryption; this construction is essentially theoretical but is the first to encourage searching for new designs without random oracles.

Another (loosely related) work in which the input-size issue has been addressed is [7], where the authors present a simple database commitment functionality where besides the standard security properties the input size of the committed database set is not revealed. They consider malicious users and use as a main tool *universal arguments of quasi-knowledge*.

Contributions. This paper focuses on PSI protocols where honest-but-curious parties hide the size of their private sets, under different security models. It is an improved and revised full version of [16]; note however we have refined and extended some results, and Section 5 is new.

We start looking at *unconditionally secure* SHI-PSI where *both* parties hide the size of their sets (Section 3). In this context, we show that PSI protocols are not achievable, while size-hiding PSI is possible involving a trusted third party which disappears after the set up phase (see Figure 1). Then, in Section 4 we move to computational security and show that there exists a private set intersection protocol where both parties hide the size of their sets (Figure 2); this does not contradict [36], for our protocol is not general, as it cannot be used unless the universe size $\mathcal{U}$ is polynomial (as computations have to actually run over the full universe). Further, we prove that the polynomial construction of [26] can be easily twisted to hide the input size of both participants if an upper bound on $|\mathcal{C}|$ and $|\mathcal{S}|$ is known (see Figure 4).

Finally, in Section 5 we provide an explicit construction of an *unbalanced* protocol where only the client hides the size of its set. Our construction builds on top of the protocol from [2], but is actually designed without random oracles. As far as we know, this is the first practical construction along these lines without random oracles, as the (somewhat conceptual) design of [36] is geared towards an existential result that can be applied generically to a large class of functions.

2 Preliminaries

In this section we provide definitions and tools used in the rest of the paper. We essentially follow the model from [2]; however, we refine their definitions slightly in order to deal with both computationally and unconditionally secure protocols, and to introduce the size-hiding constraint on both the client and the

server side. Moreover, we generalize the underlying scenario by considering a finite set of clients $\mathbf{C} = \{C_1, \ldots, C_n\}$ and a finite set of servers $\mathbf{S} = \{S_1, \ldots, S_k\}$ (possibly not disjoint) holding corresponding data sets $\mathcal{C}_i, i = 1, \ldots n$ and $\mathcal{S}_j, j = 1, \ldots, k$ which elements are drawn from a common universe $\mathcal{U}$. Each client C_i may interact with each server S_j in order to compute the intersection $\mathcal{C}_i \cap \mathcal{S}_j$. Note that as the set of clients and the set of servers may not be disjoint, we capture the situation in which the same participants may be enrolled in various executions playing a different role each time.

Furthermore, in some cases we also introduce a set-up authority SA which is fully trusted and interacts with the parties in a setup phase before the actual protocol execution takes place (so, it is in a sense a *Trusted Initializer* as introduced by Rivest in [42]). This authority might provide secret information to the parties, and its presence is actually unavoidable if participants cannot be assumed honest but curious, to certify the sets of secrets held by the parties in order to prevent any party to execute the protocol with a different set than the one it owns (for instance, the full universe).

Definition 2.1 *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 size-hiding private set intersection protocol, enabling two participants $C_i \in \mathbf{C} = \{C_1, \ldots, C_n\}$ and $S_j \in \mathbf{S} = \{S_1, \ldots, S_k\}$ to compute the intersection of their set of inputs, while revealing no further information (in particular, keeping secret all sizes of the involved sets) can be defined as follows:

Definition 2.2 *A* SH-PSI *is a scheme involving $C_i \in \boldsymbol{C} = \{C_1, \ldots, C_n\}$, $S_j \in \boldsymbol{S} = \{S_1, \ldots, S_k\}$ and (possibly) a fully trusted set-up authority SA, with two components, $Setup$ and $Interaction$, where*

- *$Setup$ is an algorithm that selects all global parameters, run by the server S_j and the client C_i an possibly involving the set-up authority SA.*
- *$Interaction$ is a protocol involving only S_j and C_i on respective input sets $\mathcal{S}_j = \{s_1^j, \ldots, s_{w_j}^j\}$ and $\mathcal{C}_i = \{c_1^i, \ldots, c_{v_i}^i\}$, which are subsets of a ground set $\mathcal{U} = \{u_1, \ldots, u_{|\mathcal{U}|}\}$,*

satisfying correctness, client privacy and server privacy. [1].

Correctness is formalized by:

Definition 2.3 *A scheme specified in Definition 2.2 is correct if, when run by* HBC *parties, $C \in \boldsymbol{C}$ and $S \in \boldsymbol{S}$, at the end of $Interaction$, run on corresponding 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.*

Notice that, compared to the definition of correctness provided in [2], we do not require that $|\mathcal{S}|$ is part of the client's output. In Section 5 in which we will apply the size-hiding restriction only on the client's side, we will stick to the definition of correctness from [2] and require the client C to output $\mathcal{S} \cap \mathcal{C}$ and $|\mathcal{S}|$ or just $|\mathcal{S}|$ if the intersection is empty. In the sequel, we will refer to such protocols as *unbalanced size-hiding*, thus coining the term USH-PSI to address schemes fulfilling the definition of correctness from [2]. Unless otherwise specified, the definition of correctness used in the sequel is the one above.

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.

[1] when specifying the set or concrete elements from a client's or servers's set the index i may be dropped – if that yields no ambiguity

Definition 2.4 *Let $View_S(\mathcal{C}, \mathcal{S})$ be a random variable representing S's view during the execution of Interaction with inputs $\mathcal{C}$ (from a client in $\mathbf{C}$) and $\mathcal{S}$. A scheme specified as in Definition 2.2 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 $\mathcal{C}, \widehat{\mathcal{C}}$ it holds that:*

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

Notice 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 probabilistic polynomial time turing 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 using its input and the output, by analyzing the transcript of the execution, could get extra-knowledge about the server's secrets. Nevertheless, if the transcript can be simulated by using only input and output, then server privacy is achieved.

Definition 2.5 *Let $C \in \mathbf{C}$ and $View_C(\mathcal{C}, \mathcal{S})$ be a random variable representing C's view during the execution of Interaction with inputs $\mathcal{C}$ and $\mathcal{S}$. Then, a scheme defined as in Definition 2.2 guarantees server privacy if, for any $C \in \mathbf{C}$ 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 probabilistic polynomial time turing machines and the transcript produced by C^* and the real view are required to be computationally indistinguishable.

We remark that in this paper we will always consider HBC parties, even though the above Definitions 2.4 and 2.5 are written without that restriction.

3 SH-PSI: the unconditionally secure case

In this section we deal with the unconditionally secure setting and show two results: first, we evidence that unconditionally secure 2-party PSI protocols do not exist. Further, we prove that unconditionally secure size-hiding PSI can indeed be achieved if we allow the involvement of SA in the $Setup$ algorithm.

3.1 Impossibility of unconditionally secure 2-party PSI

We here state that there is no 2-party PSI protocol which is unconditionally secure. This result is true for any flavor of PSI, namely, it does not matter whether one or both players receive the intersection as an output neither if the size of the input sets is revealed or not, our claim remains true. The proof is done in two steps. First we show that, from a PSI secure protocol Π which outputs the intersection and the set sizes to both participants, one can construct a protocol which securely computes the AND function. The existence of the latter is known to be impossible in the unconditionally secure setting (see [10], page 22), therefore Π cannot exist. We conclude by pointing out that the rest of output-variants of PSI protocols can actually be obtained from a protocol like Π, thus completing our argument.

Theorem 3.1 *Let $F = (F_1, F_2)$ be the functionality that, on input a pair of sets $(\mathcal{C}, \mathcal{S})$, outputs $F_1(\mathcal{C}, \mathcal{S}) = (\mathcal{C} \cap \mathcal{S}, |\mathcal{S}|)$ to the first participant and $F_2(\mathcal{C}, \mathcal{S}) = (\mathcal{C} \cap \mathcal{S}, |\mathcal{C}|)$ to the second one. Then an unconditionally secure 2-party protocol which computes F does not exist.*

PROOF. Let us reason by contradiction and assume that a secure 2-party protocol Π which computes F does exist. From Π we will build a secure private AND protocol Γ. A private AND protocol is a two-party protocol, run by users A and B, at the end of which the players get the logical AND of their input bits and nothing else (i.e., a secure protocol for computing $a \cdot b$ from private inputs $a, b \in \{0, 1\}$).

Indeed, assume that, for $i \in \{1, 2\}$, P_i is holding as input for Γ a bit $b_i \in \{0, 1\}$. For the PSI protocol Π, choose as the universe the set of integers $\mathcal{U} = \{0, 1, 2, 3\}$. Each player P_i constructs its input set X_i as follows:

- If $b_i = 0$, then X_i is chosen as $\{0, i\}$.
- If $b_i = 1$, then X_i is chosen as $\{0, 3\}$.

Then P_1 and P_2 run the protocol Π for PSI with inputs X_1 and X_2 respectively. Each player, depending on the output of Π, sets the output of Γ as follows:

- If the intersection $X_1 \cap X_2 = \{0\}$, then the output of Γ is set to 0.
- If the intersection $X_1 \cap X_2 = \{0, 3\}$, then the output of Γ is set to 1.

It is easy to check that Γ is a secure protocol for the computation of $AND(b_1, b_2)$:

Correctness. Note that $X_1 \cap X_2 = \{0, 3\}$ if and only if $b_1 = b_2 = 1$ and $X_1 \cap X_2 = \{0\}$ otherwise, thus Γ correctly computes $AND(b_1, b_2)$.

Security. First note that all the possibly involved sets have two elements, therefore the size of the other player's set does not provide any information. Let us analyze the information leaked from $X_1 \cap X_2$ to player P_i. If the input of player P_i is $b_i = 0$ then $X_1 \cap X_2 = \{0\}$, regardless of the input of the other player. Thus P_i does not gain any information. On the other hand, if $b_i = 1$, the intersection $X_1 \cap X_2$ allows P_i to learn the input of the other player, but this is also what happens in an ideal execution of an AND protocol.

As Π is supposed to be an unconditionally secure protocol, players do not learn anything beyond the output, so Γ is also an unconditionally secure protocol, which concludes the proof.

□

Corollary 3.2 *Let $G = (G_1, G_2)$ be any functionality that, on input a pair of sets $(\mathcal{C}, \mathcal{S})$, outputs $G(\mathcal{C}, \mathcal{S})$ such that*

- *G_i consists in both, one or none of the components of F_i for $i \in \{1, 2\}$, where F is the functionality described in Theorem 3.1.*
- *$\mathcal{C} \cap \mathcal{S}$ is present in at least one of the two outputs $G_1(\mathcal{C}, \mathcal{S})$ or $G_2(\mathcal{C}, \mathcal{S})$.*

Then an unconditionally secure 2-party protocol which computes G does not exist.

PROOF. From a protocol Λ which securely computes G is easy to build a protocol Π which securely computes F. For that purpose, each player simply sends the missing outputs to the other player. Namely, the player holding the intersection $\mathcal{C} \cap \mathcal{S}$ sends it to the other player, if needed. Further, each player sends the size of its input to the other player. Theorem 3.1 states no such protocol Π can exist; as a result, neither can Λ as stated above. □

3.2 Unconditionally secure SH-PSI with a trusted initializer

Allowing the involvement of SA we can actually achieve size-hiding in the unconditional setting; such a construction can be seen in Figure 1.

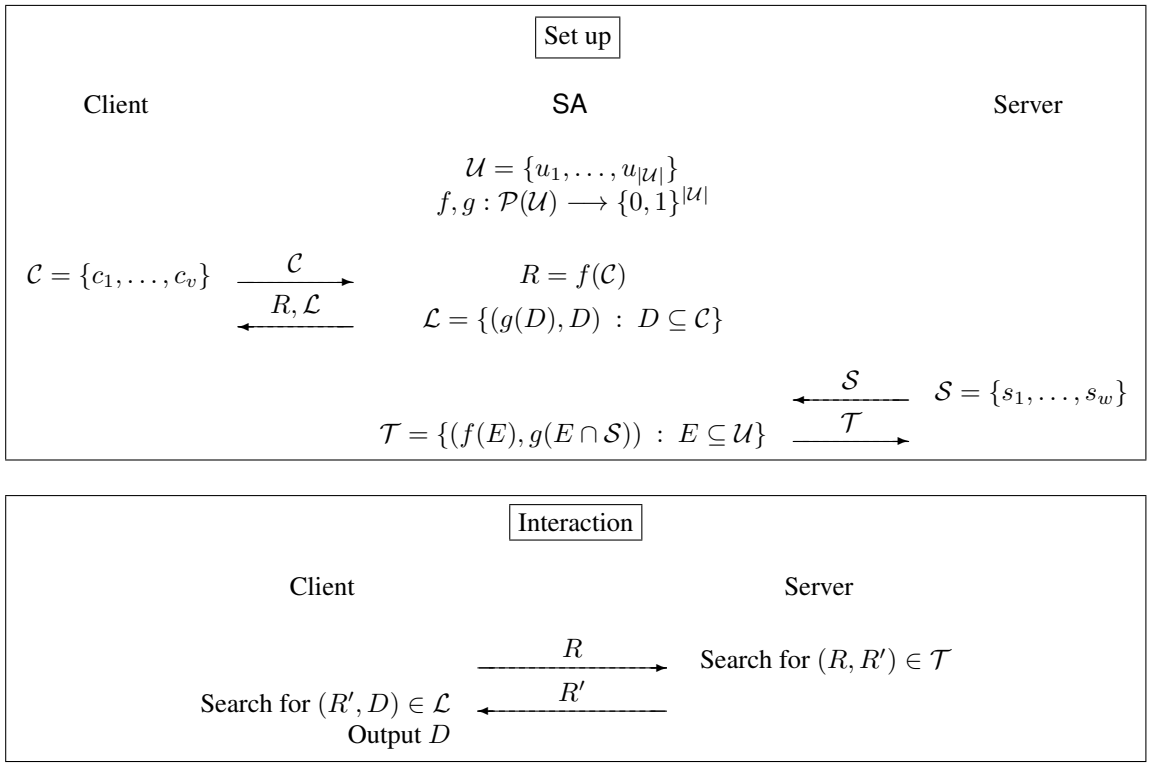

Figure 1: An unconditionally secure size-hiding set intersection protocol.

The idea of the scheme is the following: in the $Setup$, the SA chooses two random bijections $f, g : \mathcal{P}(\mathcal{U}) \longrightarrow \{0,1\}^{|\mathcal{U}|}$. When the SA interacts with a client, once received the client's set of secrets, the SA sends the client an *identifier*, computed by using the first random function, and a list of *sub-identifiers*, one for each possible subset of the client's set of secrets, computed through the second random function.

On the other hand, when the SA interacts with a server and receives its set of secrets, it constructs a two-column table: in the first column there is, for each possible subset E of the ground set, an identifier of E, computed with the first random function; the second column has an identifier, computed with the second random function, of the *intersection* $E \cap \mathcal{S}$ of the subset E and the server's set of secrets $\mathcal{S}$. The table is given to the server. The $Interaction$ between a client and a server is a simple two-round protocol: the client

sends its set identifier; the server looks up in the table the row with the received identifier, and sends it back the identifier of the second column. Finally, the client looks up in the list of sub-identifiers and determines the intersection with the server. We emphasize that the goal of this construction is only to show that the construction of SH-PSI schemes is possible in the unconditionally setting, but the scheme we provide is, by no means, of practical interest for a universe of large size.

Theorem 3.3 *Let $f, g : \mathcal{P}(\mathcal{U}) \longrightarrow \{0,1\}^{|\mathcal{U}|}$ be two random bijections. The protocol described in Figure 1 is a SHI-PSI protocol achieving correctness, client privacy and server privacy in the unconditional model.*

PROOF.

Correctness. It is easy to check that the protocol is correct; indeed, the table $\mathcal{T}$ contains the pair $(R, g(\mathcal{C} \cap \mathcal{S}))$ and the client will indeed retrieve $\mathcal{C} \cap \mathcal{S}$ by looking up in $\mathcal{L}$.

Server Privacy. Note that the client only gets a (random) identifier from the interaction, which can indeed be simulated without $\mathcal{S}$.

Client Privacy. 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 it. Moreover, *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's set of secrets nor its size. □

4 SH-PSI: the computationally secure case

In this section we show that in the computational case size-hiding private set intersection is possible. To this aim we give two constructions. The first one is valid when the size of the ground set is polynomial in the security parameter; interestingly, it allows to state the equivalence between PSI, oblivious transfer (OT) and the secure computation of the AND funcion. The second one may be useful in practice if the sizes of the sets of secrets are *reasonably* small and an upper bound on them is a-priori known.

4.1 AND-based SH-PSI protocol

An oblivious transfer protocol is a two party protocol involving a sender and a receiver; the sender has two secrets, s_0 and s_1, while the receiver is interested in one of them. Its choice is represented by a bit σ. After running the protocol the receiver gets s_σ and nothing else, while the sender does not learn which secret the receiver has recovered. Introduced by Rabin [41], 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 $OT(s_0, s_1, \sigma)$.

Note that a private AND protocol can be realized by using an $OT(b_1, b_2, s)$ protocol. Indeed, the bit b_s that the receiver gets can be expressed as $b_s = (1 \oplus s)b_0 \oplus sb_1$. As a result, invoking the instance $OT(0, a, b)$, the receiver will get $a \cdot b$. Now, the key-idea underlying our construction 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 $poly(n)$ (both known and fixed at the Set-up phase). 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. $\mathcal{S}$).

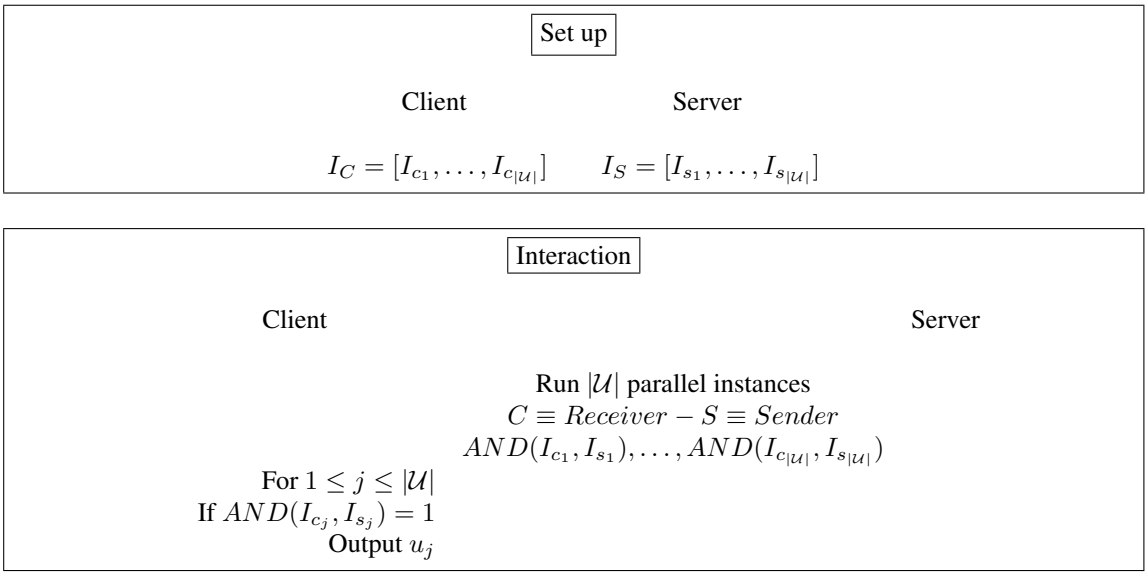

Figure 2: A computationally secure size-hiding set intersection protocol.

It is easy to check that the protocol is correct. Moreover, it is secure as long as the AND protocol is. Thus, if we use the OT construction proposed in [23], which relies on the existence of enhanced trapdoor permutations[2], we yield the following result:

Theorem 4.1 *The protocol given in Figure 2, when instantiated with the OT protocol of Figure 3, realizes SH-PSI scheme achieving correctness, client privacy and server privacy.*

PROOF.

Correctness. It is easy to check that the SH-PSI protocol from Figure 2 is correct as long as the OT protocol is. To see this, just note that indeed:

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

[2]see Definition C.1.1. from [27]

Protocol steps (simplified description):

Let $b_0, b_1 \in \{0,1\}$ be the sender's secret bits, and let $i \in \{0,1\}$ be the receiver's choice.
Moreover, let $\mathcal{F} = \{f_\alpha : D_\alpha \to D_\alpha\}_{\alpha \in \{0,1\}^{poly(n)}}$ be a family of enhanced 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 $y_{1-i} = e_{1-i}$;
  - 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, it sends c_0, c_1 to the Receiver.
- The Receiver computes $b_i = c_i \oplus b(e_i)$.

Figure 3: OT protocol based on trapdoor permutations.

Client Privacy. Intuitively client privacy is guaranteed because in the OT protocol the sender, independently 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.

More precisely, a simulator Sim_S for the sender in the OT protocol will choose $y_0, y_1 \in D_\alpha$ uniformly at random. From this one can construct a simulator for the server just running independently Sim_S for each instance of the AND computation. Note that the view produced by this simulator 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.

Server Privacy. On the other hand, in the OT protocol the Receiver gains no knowledge on the bit b_{1-i} because $b(\cdot)$ is a hardcore predicate for f_α; as a result, 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.

Based on this, we may construct a simulator Sim_R which acts as the receiver in the real protocol OT except that it chooses $c_{i-1} \in \{0,1\}$ uniformly at random. This in turn yields a simulator for the client of the SH-PSI protocol, by running Sim_R for each execution of the AND protocol. □

Remark 4.2 *Notice that, in this section we have described reductions from PSI to AND and from AND to OT. On the other hand, a reduction from OT to PSI is constructed in [26]. Therefore it follows that OT, PSI and AND are* equivalent.

4.2 Threshold-based protocol

Assuming some a-priori information on the sizes of the sets involved is known, more efficient protocols may be achieved. Here we assume that a known value M upper bounds the sizes of all client and server's sets. Indeed, the smaller M is with respect to $|\mathcal{U}|$, the greater the interest of this construction (actually, we need M of polynomial size but $|\mathcal{U}|$ may as well be exponential).

As our scheme is a simple twist of [26], the main tool used is also additively homomorphic encryption. Informally, a public key encryption scheme is (additively) homomorphic if, for any two encryptions $Enc(m_1)$ and $Enc(m_2)$ of any two messages m_1 and m_2, it holds that $Enc(m_1) \cdot Enc(m_2) = Enc(m_1 + m_2)$, where $\cdot$ is the group operation on ciphertexts. By repeated application of the property, for any integer c, it follows that $Enc(m_1)^c = Enc(cm_1)$; as a result, given encryptions $Enc(a_0), \ldots, Enc(a_k)$ of the coefficients $a_0, \ldots, a_k$ of a polynomial P of degree k, and a plaintext value y, it is possible to compute $Enc(P(y))$, i.e., an encryption of $P(y)$. Paillier's cryptosystem [39] exhibits such properties and achieves semantic security under chosen plaintext attacks under the decisional composite residuosity assumption (see, for instance, Section 11.3 of [33]).

The proposed scheme is depicted in Figure 4. At this, Enc and Dec denote the Paillier encryption and decryption algorithms, respectively. As the client is assumed to be honest, it will execute in the set up phase the key generation algorithm for Paillier encryption, at which, in particular two l- bit primes p, q are fixed. At this point, it makes explicit an encoding of $\mathcal{U}$ into $\mathbb{Z}_n \setminus \{0\}$. For the sake of readability in Figure 4 elements of $\mathcal{U}$ are assumed to belong to $\mathbb{Z}_n \setminus \{0\}$.

Theorem 4.3 *The protocol given in Figure 4, is a SH-PSI protocol achieving correctness, client privacy and server privacy, under the assumption that the Paillier encryption scheme is IND-CPA.*

PROOF.

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 its 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. [3]

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 $\mathbb{Z}_n \setminus \{0\}$.

[3] We leverage 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.

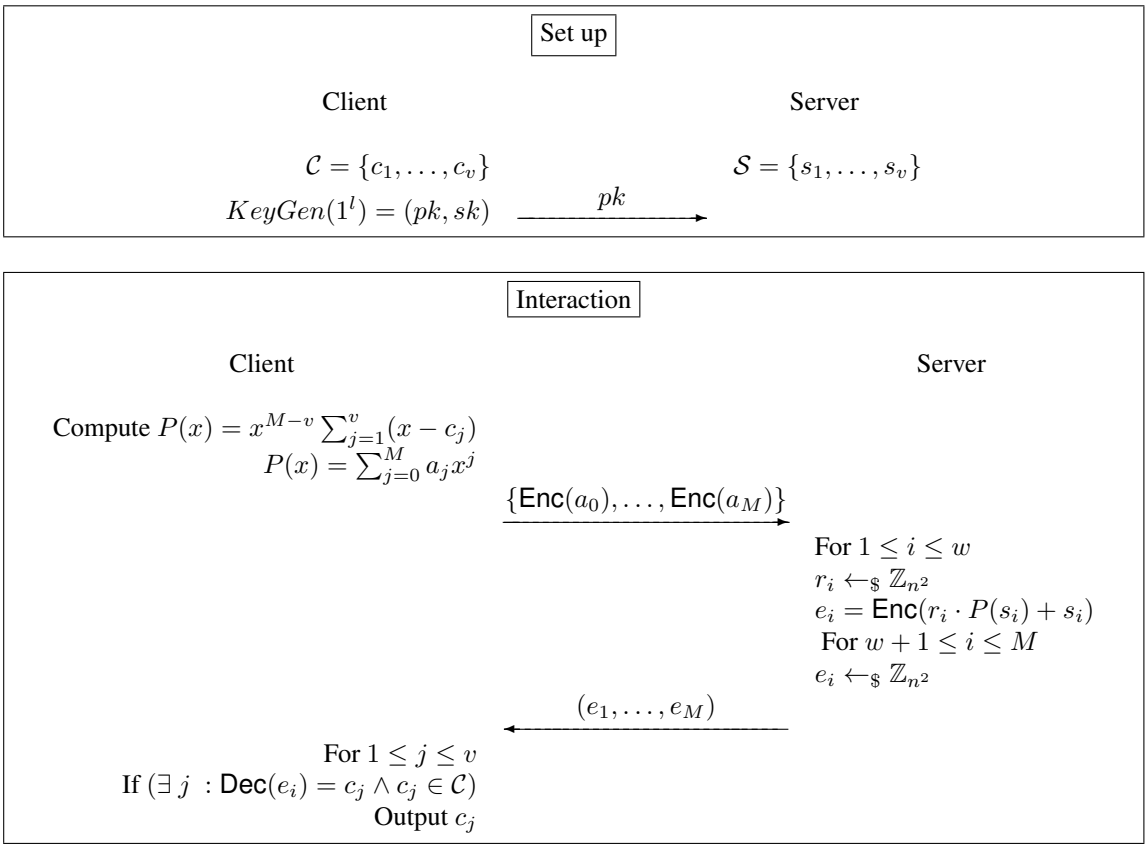

Figure 4: Polynomial-based construction for $|\mathcal{C}|, |\mathcal{S}| \leq M$.

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 4 (w.l.o.g., we assume that $\mathcal{S} \cap \mathcal{C} = \{s_1, \ldots, s_I\}$).
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.

□

Remark 4.4 *Having Paillier encryption in mind, we have designed the polynomial P in Step 2. of Round 1 (see 4), 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 [26], abstracting from the concrete homomorphic encryption scheme used, several modifications of their basic protocol are proposed in order to boost efficiency all of them geared towards reducing the number of products and exponentiations made by the server. These ideas would also yield a significant speed up versus a naïve implementation of the above protocol.*

5 Unbalanced Size-Hiding: a construction without random oracles

In this section, we follow the spirit of [2] and try to provide *unbalanced* private set intersection 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 [2].[4]

The main tool behind our construction are smooth projective hash functions from non-malleable commitments, as proposed by Gennaro and Lindell in [28]. We summarize here informally the main definitions and results, and refer the reader to [28, 1] for precise definitions and concrete constructions.

Let Com be a non-interactive, non-malleable commitment scheme which is computationally hiding and perfectly binding in the common reference string model[5]. By $\mathsf{Com}_\rho(u, r)$ we denote a commitment to an element u using the common reference string ρ and the random coins r; further, denote by Com_ρ the space of all strings that may be output by Com when the common reference string is ρ. Now, define the sets:

- $\mathcal{U}_\rho = \mathcal{U} \times \mathsf{Com}_\rho$
- $L_\rho = \{(u, \mathsf{com}) \| \ \exists r \ s.t. \ \mathsf{com} = \mathsf{Com}_\rho(u, r)\}$

In such a situation, consider at hand a smooth projective hash family F for the induced hard partitioned subset membership problem. This essentially means we have a hash family F indexed by a key space K, that is $F = \{f_k\}_{k \in K}$, so that for every $k \in K$, we have

$$f_k : \mathcal{U}_\rho \longmapsto G$$

for a set G which is of superpolynomial size in the security parameter. For our purposes later, G can be seen as $\{1, \ldots, N^2\}$. Let P be a fixed set and α an (efficiently computable) projection function defined over $K \times \mathsf{Com}_\rho$ with range in P. Now given all these ingredients we get the following:

1. *Efficient hashing from k:* there exists an efficient algorithm $\mathsf{KeyHash}$ which on input a pair (u, com) from $\mathcal{U}_\rho$, and a key $k \in K$ outputs $f_k(u, \mathsf{com})$.
2. *Efficient hashing from projection and witness:* there exists an efficient algorithm $\mathsf{ProjHash}$ which on input a pair $(u, \mathsf{com}) \in \mathcal{U}_\rho$, a random value r so that $\mathsf{com} = \mathsf{Com}_\rho(u, r)$ and a projection $\alpha(k, \mathsf{com})$

[4] Correctness is defined including $|\mathcal{S}|$ as part of the client's output, the definition of client privacy coincides with the one we have given here for SH-PSI 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 [2], but $|\mathcal{S}|$ is needed in the simulation included in their proof).

[5] (it may actually be derived, as argued in [28], from any IND-CCA encryption scheme)

outputs $f_k(u, \mathsf{com})$. We may actually assume the input of ProjHash to be the projection $\alpha(k, \mathsf{com})$ and u, r, provided that Com and ρ are well defined and publicly available.

3. *Smoothness:* for every $u \in \mathcal{U}$, for $k \in K$ chosen u.a.r. and uniformly chosen coins r, it holds that

$$\{\mathsf{com} = \mathsf{Com}_\rho(u, r), u, \alpha(k, \mathsf{com}), f_k(u, \mathsf{com})\}$$

and

$$\{\mathsf{com} = \mathsf{Com}_\rho(u, r), u, \alpha(k, \mathsf{com}), g\}$$

are computationally indistinguishable, where g is drawn u.a.r. from G.

Let us now introduce our protocol, building on a non-interactive non-malleable commitment scheme Com and a smooth projective hash family $F = \{f_k\}_{k \in K}$ as described above. These can actually be derived from IND-CCA encryption schemes, like Cramer-Shoup (see, again [28]).We may, for instance take the Cramer-Shoup scheme based on quadratic residuosity as a base (see [11]), which fits our construction nicely. Further, we need to make another computational assumption, namely the Strong RSA assumption as introduced in [4], which states that given a randomly chosen RSA modulus N, and a random element $y \in \mathbb{Z}_N^*$, it is computationally hard to find a pair (e, r), with $e > 1$, so that $r^e = y \pmod N$.

Further, during the set up phase, a pseudorandom function $h : \mathcal{U} \mapsto \{1, \ldots, N^2\}$ is selected, which can be efficiently evaluated by both users. Moreover, assuming the sizes of the underlying sets are large enough, we can assume h to be *division intractable* (see Theorem 2 of [9]). Thus in the sequel we assume that it is computationally hard to find distinct inputs $x_1, \ldots, x_n, y \in \mathcal{U}$ such that $h(y)$ divides the product of the $h(x_i)$'s (see Definition 2 in [44]). The key ideas in our design are similar to those in [2], in a nutshell:

- During set up, server generates two safe primes p and q and publishes the product N, further, the two agree upon a randomly chosen generator from the group of quadratic residues modulo N and a suitable pseudorandom function h. Also the projective hash family and the corresponding commitment scheme are agreed upon and made explicit.
- In the first round, the client sends to the server a witness for its set $X = g^{R_c \prod_{i=1}^{v} h(c_i)} \pmod N$, by means of an RSA accumulator. where R_c is selected uniformly at random in $\{1, \ldots, N^2\}$.
- The server, being able to compute $h-$roots modulo N, constructs for each $s \in \mathcal{S}$, a value $y = X^{\frac{1}{h(s)}}$ which, if s is in the intersection, constitutes a witness for the set $\mathcal{C} \setminus \{s\}$ —and provides no information otherwise. Further, it commits to these values and sends in the second round, for each of them, a triplet consisting of: a hash value $f_k(h(s, y), \mathsf{com})$ (for a randomly selected f_k from F), the random nonce involved in the commitment and the projection $\alpha(k, \mathsf{com})$. Note that from the random nonce and the given projection the client just has enough to evaluate f_k on exactly the right input derived from s – for other inputs, we may assume the algorithm ProjHash gives a random output.
- Finally, the client can identify matching sets $\mathcal{C} \setminus \{s\}$ for each $s \in \mathcal{C} \cap \mathcal{S}$, and, as a result, retrieve the intersection. Note that this is all it learns, as it has no information on the evaluations of f_k in any element outside the intersection.

A detailed description can be seen in Figure 5; note that evaluations of f_k computed by the server are actually executions of KeyHash, while evaluations by the client are executions of ProjHash.

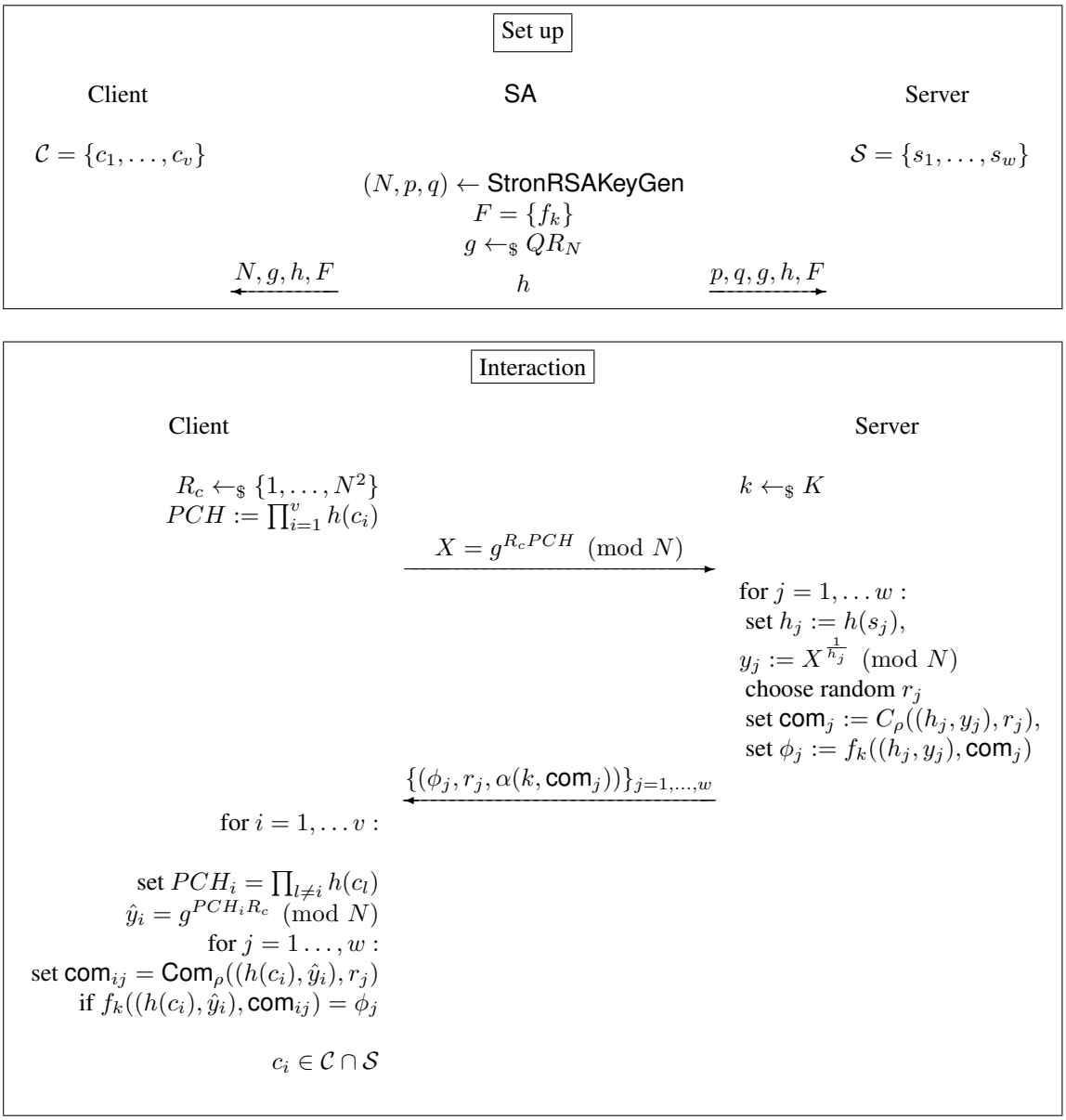

Figure 5: An USH-PSI protocol in the CRS model.

Theorem 5.1 *The protocol given in Figure 5, is a USH-PSI protocol in the common reference string model achieving correctness, client privacy and server privacy, under the strong RSA assumption, provided that* Com *is a non-interactive and non-malleable perfectly binding and computationally hiding commitment scheme and F is a family of smooth projective hash functions.*

Correctness. To see that the protocol is correct, let us first argue that whenever there are $i \in \{1, \ldots, v\}$ and $j \in \{1, \ldots, w\}$ such that $c_i = s_j$ the client will set $c_i \in \mathcal{C} \cap S$. Indeed, if that is the case $PCH_i = \frac{PCH}{h_j}$,

and as a result $\hat{y}_i = y_j$, so client gets $f_j = f_k((h(c_i), \hat{y}_i), \mathsf{com}_j)$ from $\mathsf{ProjHash}(\alpha(k, \mathsf{com}_j), (h(c_i), \hat{y}_i), r_j)$.

Furthermore, if c_i is not in $\mathcal{S}$, for each $j = 1, \ldots, w$, due to the division intractability of H we have that with overwhelming probability $PCH_i \neq \frac{PCH}{h_j}$; and as a result there is only negligible probability that the client includes c_i in the intersection.
Indeed, with overwhelming probability, $\mathsf{ProjHash}(\alpha(k, \mathsf{com}_j), (h(c_i), \hat{y}_i), r_j)$ will give a random output that will not coincide with ϕ_j.

Client Privacy. As PCH is, with overwhelming probability, coprime with $p'q'$ and R_c is selected uniformly at random, X is (except with negligible probability) uniformly distributed over QR_N (our argument is exactly the same as in [2]).

Server Privacy. Let us argue how C^* may simulate the client's view. Without loss of generality, let us "reorder" the client's set and assume the input of C^* is $\mathcal{C} = \{c_1, \ldots, c_v\}$ and $\mathcal{C} \cap \mathcal{S} = \{c_1, \ldots, c_t\}$. C^* selects u.a.r. a key k for the projective hash family F, and R_C in $\{1, \ldots, N^2\}$.

Then it:

- selects random values r_j $j = 1, \ldots, w$,
- for $i = 1, \ldots v$, compute $h(c_i)$,
- set $PCH := \prod_{l=1}^{v} h(c_l)$, $X := g^{R_c PCH} \pmod N$
- for $i = 1, \ldots, t$ set $PCH_i := \prod_{l \neq i} h(c_l)$, $y_i := g^{R_c PCH_i}$, $\mathsf{com}_i = C_\rho((h(c_i), y_i), r_i)$ and $f_i = f_k((h(c_i), y_i), \mathsf{com}_i)$
- for $j = t+1, \ldots, w$ choose u.a.r values $y_j^* \in QR_N$, h_j^* in the range of h and set $\mathsf{com}_j^* := C_\rho((h_j^*, y_j^*), r_j)$. Further, select f_j^* u.a.r. from the range of f_k.

Now, let us see that the distribution:

$$\mathsf{simview} := \{R_c, \{f_i, r_i, \alpha(k, \mathsf{com}_i)\}_{i=1,\ldots,t}, \{(f_j^*, r_j, \alpha(k, \mathsf{com}_j^*))\}_{j=t+1,\ldots,w}\}$$

is computationally indistinguishable from the real clients view:

$$\mathsf{D}_0 := \{R_c, \{f_k((h_j, y_j), \mathsf{com}_j), r_j, \alpha(k, \mathsf{com}_j)\}_{j=1,\ldots,w}\}$$

where $y_j = X^{\frac{1}{h_j}}$ and $\mathsf{com}_j = C_\rho((h_j, y_j), r_j)$.

Our argument mimics the reasoning from Theorem 1 in [2]: if we consider a series of intermediate distributions D_ν for $\nu = 1, \ldots, w - (t+1)$ where for each $1 \leq \nu \leq w - (t+1)$ we have that the first ν values corresponding to elements in $\mathcal{S}$ but not in the intersection are simulated by selecting uniformly at random a nonce r, an element $y^* \in QR_N$, h^* in the range of H and f^* in the range of f_k and further computing $\mathsf{com} := C_\rho((h^*, y^*), r)$. Namely, $\mathsf{simview} = D_{w-t}$. Indeed, using a hybrid argument, we claim that the probability of distinguishing D_ν from $D_{\nu+1}$ is negligible in the security parameter ℓ.

Indeed, due to the smoothness of F we have that if a distinguisher D is able to tell apart these two distributions, it must have, on input D_ν, constructed the elements $h(s_\nu)$ and $y_\nu = X^{\frac{1}{h(s_\nu)}}$. Nothing else

being sent helps in this distinction, as the nonces are truly random and nothing is leaked from the projections (due to the hiding property of the commitment scheme).

This is because the division intractability of H guarantees that $\frac{R_C PCH}{h(s_\nu)}$ is not an integer with overwhelming probability, and thus it must have extracted an $h(s_\nu)$-root of $X \pmod N$ without knowing the factorization of N. Now, as a result, such a distinguisher can be used (as in [44]) to violate the RSA assumption. Indeed, we can build an algorithm $\mathcal{A}$ which on input a Strong RSA challenge of the form (N, y) will (making use of D) output values (r, e) so that $r^e = y \pmod N$.

For that purpose, $\mathcal{A}$ may simply set $g = y$ and select $x_1, \ldots, x_v \in \mathcal{U}$, and select a random $u \in \mathcal{U} \setminus \mathcal{C}$ which will play the role of the distinguishing element. Further, select R_c uniformly at random in $\{1, \ldots, N^2\}$ and set $PCH = \prod_{i=1,\ldots,v} h(x_i)$. Now present D with the derived distributions D_ν and $D_{\nu+1}$. Note that they only differ in the last triplet, which in D_ν is $(f_k((h, y), \mathsf{com}), r, \alpha(k, \mathsf{com}))$ for values selected as the server would do following the protocol specification, while in $D_{\nu+1}$ it is $(f^*, r, \alpha(k, \mathsf{com}^*))$, where the value r is selected properly but f^* and com^* are incorrelated, as explained above.

Now D outputs $h^* = h(u)$ and $X^{\frac{1}{h^*}} \pmod N$. Note that, due to the division intractability of $\mathcal{H}$ we have that h^* does not divide $R_c PCH$ with overwhelming probability.

As a result, let $d = g.c.d.(R_c PCH, h^*)$. Then there must be b, e with $gcd(b, e) = 1$ and so that $h^* = ed$ and $R_c PCH = bd$. Now, using the extended Euclidean algorithm $\mathcal{A}$ may compute α, β so that $\alpha e + \beta b = 1$. Note that the element $X^{\frac{1}{h^*}} \pmod N$ output by the distinguisher is actually $g^{\frac{b}{e}}$, from which (using Shamir's trick) $\mathcal{A}$ derives $g^{\frac{1}{e}}$ as $(g^{\frac{b}{e}})^\alpha g^\beta$.

Remark 5.2 *Note also that a very simple solution in the unbalanced scenario comes at hand when we are willing to involve a set up authority SA who will simply select function f from a pseudorandom function family and provide each participant with f-evaluations of the elements in its private input set. Further, users may exchange these evaluations in order to identify common elements . Indeed, such a trivial solution may come handy in many scenarios, however if the SA is not fully trusted the evaluation of the pseudorandom function must be done in an oblivious way. Following the ideas of [31], two protocols along these lines can be found in our prior work [16].*

6 Conclusion

We have explored the private set intersection problem when hiding the sizes of the input sets is relevant; for SH-PSI, we provide a (conceptual) protocol in the unconditional setting (with the help of a trusted party in a set up phase). Furthermore, we proved that PSI is impossible in the unconditional setting, making explicit its relation to AND. In addition, in the computational scenario, we have both given a theoretical construction (only applicable for polynomial universes) and a practical one which is a simple twist of the well known polynomial scheme of Freedman et al. [26].

For USH-PSI we have given a generic construction that may be implemented using different suitable projective hash functions from convenient commitment schemes. Depending on the underlying primitives actually selected, the resulting scheme will turn out to be more or less efficient and its security guarantees stronger or weaker depending on the underlying computational assumptions required (as explicited above, we need at least to assume that the Strong RSA assumption holds).

Acknowledgements

The first three authors were partially supported by the Spanish "Ministerio de Economía y Competitividad" through the project grant MTM-2010-15167. This research is also partially supported by the Italian PRIN project GenData 2020.

References

[1] M. Abdalla, F. Benhamouda, O. Blazy, C. Chevalier, and D. Pointcheval, *SPHF-Friendly Non-interactive Commitments*, Asiacrypt 2013, LNCS, Vol. 8269, pp. 214–234, 2013.

[2] 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.

[3] 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.

[4] N. Baric and B. Pfitzmann. *Collision-free accumulators and Fail-stop signature schemes without trees* In Eurocrypt 1997, LNCS, Vol. 1233, pp. 480-494, 1997.

[5] 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.

[6] E. De Cristofaro, S. Faber, G. Tsudik, *Secure Genomic Testing with Size- and Position-Hiding Private Substring Matching*, WPES 13, 2013.

[7] M. Chase and I. Visconti, *Secure database commitments and universal arguments of quasi knowledge*, In CRYPTO, pp. 236254, 2012.

[8] J.L. Carter and M.N. Wegman, *Universal classes of hash functions*, Journal of Computer and System Sciences, Vol. 18, 143154, 1079.

[9] J.S. Coron and D. Naccache, *Security analysis of the Gennaro-Halevi-Rabin signature scheme*, Eurocrypt 2000, LNCS, Vol. 1807, 91101, 2000.

[10] R. Cramer, *Introduction to Secure Computation*, Lectures on Data Security, LNCS, Vol. 1561, pp. 16-62, 1999.

[11] R. Cramer and V. Shoup. *Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption*, Eurocrypt 2002, LNCS, Vol. 2332, pp. 45-64, 2002.

[12] 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.

[13] J. Camenisch and A. Lysyanskaya, *A Signature Scheme with Efficient Protocols*, SCN 2002, LNCS, Vol. 2576, pp., 268-289, 2003.

[14] J. Coron and D. Naccache, *Security analysis of the Gennaro-Halevi-Rabin signature scheme*, Eurocrypt 1996, LNCS, Vol. 1070, pp. 399-416, 1996.

[15] J. Camenish and G. M. Zaverucha, *Private Intersection of Certified Sets*, FC 2009, LNCS, Vol. 5628, pp., 108-127, 2009.

[16] P. DArco, M.I. Gonzlez-Vasco, A. L. Prez del Pozo, and C. Soriente. *Size Hiding in Private Set Intersection: Existential Results and Constructions*, Africacrypt 2012, LNCS, Vol. 7374, pp. 378394, 2012.

[17] C. Dong, L. Chen, and Z. Wen, *When Private Set Intersection Meets Big Data: An Efficient and Scalable Protocol*, CCS13, pp. 789-800, 2013.

[18] E. De Cristofaro, S. Jarecki, J. Kim, G.Tsudik. *Privacy-Preserving Policy-Based Information Transfer*. Privacy Enhancing Technologies (PES09), LNCS, Vol. 5672, pp. 164184, 2009.

[19] E. De Cristofaro, J. Kim, and G. Tsudik, *Linear-Complexity Private Set Intersection Protocols Secure in Malicious Model*, ASIACRYPT 2010, LNCS, Vol. 6477, pp. 213231, 2010.

[20] E. De Cristofaro and G. Tsudik, *Practical Private Set Intersection Protocols with Linear Complexity*, FC 2010, LNCS, Vol. 6052, pp. 143-159, 2010.

[21] 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.

[22] 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.

[23] 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.

[24] K. Frikken, *Privacy-Preserving Set Union*, ACNS 2007, Vol. , pp. 237-252, 2007.

[25] 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.

[26] M. J. Freedman, K. Nissim, and B. Pinkas, *Efficient Private Matching and Set Intersection*, Eurocrypt 2004, LNCS, Vol. 3027, pp. 1-19, 2004.

[27] O. Goldreich, *Foundations of Cryptography - Volume II Basic Applications*, Cambridge Press, 2004.

[28] R. Gennaro and Y. Lindell *A Framework for Password-Based Authenticated Key Exchange (Extended Abstract)*, Eurocrypt 2003, LNCSI, Vol. 2656, pp. 524–543, 2003.

[29] S. Jarecki, X. Liu. *Efficient oblivious pseudorandom function with ap- plications to adaptive OT and secure computation of set intersection*, TCC 2009, LNCS, Vol. 5444, pp. 577594, 2009.

[30] S. Jarecki, X. Liu. *Fast and Secure Computation of Set Intersection*, SCN 2010, LNCS 6280, pp. 418435, 2010.

[31] 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.

[32] 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.

[33] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*, Chapman & Hall/Crc Cryptography and Network Security Series, 2007.

[34] F. Kerschbaum, *Outsourced Private Set Intersection Using Homomorphic Encryption*, ASIACCS 12, 2012.

[35] L. Kissner and D. Song, *Privacy-Preserving Set Operations*, Crypto 2005, LNCS, Vol. 3621, pp. 241-257, 2005.

[36] Y. Lindell, K. Nissim, and C. Orlandi, *Hiding the Input-Size in Secure Two-Party Computation*, ASIACRYPT 2013, LNCS, Vol. 8270, pp. 421440, 2013.

[37] R. Nojima, Y. Kadobayashi, *Cryptographically Secure Bloom-Filter*, Transactions on Data Privacy, Vol. 2, pp. 131139, 2009.

[38] 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.

[39] P. Pailler, *Public-key Cryptosystems based on composite degree residuosity classes*, Crypto 1999, LNCS, Vol. 1592, pp. 223-239, 1999.

[40] B. Pinkas, T. Schneider, M. Zohner, 23rd USENIX Security Symposium (USENIX Security'14), *On the Performance of Private Set Intersection*, 2014.

[41] M. Rabin, *How to exchange secrets by oblivious transfer*, Technical Report TR-81, Aiken Computation Laboratory, Harvard University, 1981.

[42] 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

[43] 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.

[44] R. Gennaro, S. Halevi, and T. Rabin, *Secure hash-and-sign signatures without the random oracle*, Eurocrypt 1999, LNCS, Vol. 1592, pp. 332–350, 1999.