(If) Size Matters: Size-Hiding Private Set Intersection

Giuseppe Ateniese[1], Emiliano De Cristofaro[2], and Gene Tsudik[2]

[1] Computer Science Department, Johns Hopkins University
[2] Computer Science Department, University of California, Irvine

Abstract. Modern society is increasingly dependent on, and fearful of, the availability of electronic information. There are numerous examples of situations where sensitive data must be – sometimes reluctantly – shared between two or more entities without mutual trust. As often happens, the research community has foreseen the need for mechanisms to enable limited (privacy-preserving) sharing of sensitive information and a number of effective solutions have been proposed. Among them, Private Set Intersection (PSI) techniques are particularly appealing for scenarios where two parties wish to compute an intersection of their respective sets of items without revealing to each other *any other information*. Thus far, "any other information" has been interpreted to mean any information about items not in the intersection.
In this paper, we motivate the need for Private Set Intersection with a stronger privacy property of *hiding the size* of the set held by one of the two entities ("client"). We introduce the notion of Size-Hiding Private Set Intersection (SHI-PSI) and propose an efficient construction secure under the RSA assumption in the Random Oracle Model. We also show that input size-hiding is attainable at very low additional cost.

1 Introduction

Operations that involve sharing sensitive or private information are increasingly encountered in everyday life. A typical scenario involves two entities: one that seeks certain information, and the other – that possibly has this information and is either willing or is compelled to share it. At the same time, each entity wants to maximize privacy of its information, beyond the minimum disclosure necessary to complete the required operation. To motivate the problem, we present three concrete (and only slightly contrived) examples illustrating nuanced requirements of such privacy-preserving operations.

Example 1: U.S. Department of Homeland Security (DHS) maintains a dynamic database of suspected terrorists (TWL: Terror Watch List). For every flight, DHS needs to know whether the flight passenger manifest and TWL have any names in common. Airlines are reluctant to unconditionally share passenger information. Some airlines are foreign and some flights might be transit, i.e., they merely fly over, but not land in, United States. At the same time, compliance with DHS is mandatory, meaning that names of any flight passengers that also appear

in TWL must be supplied to DHS. For its part, DHS treats TWL as secret information and is absolutely unwilling to reveal it to any airline.

Example 2: U.S. Central Intelligence Agency (CIA) has a requirement to periodically (e.g., every year) check whether any of its agents have been arrested or convicted any crimes. It thus needs to approach every state and, ideally, compare its list of employees against the state's list of arrestees and/or convicted offenders. A state (e.g., South Dakota) is unwilling to reveal its information for fear of misuse and legal liability. Whereas, CIA is mandated by law not to reveal the names of *any* of its agents.

Example 3: U.S. Center of Disease Control and Prevention (CDC) maintains a list of people, per city, afflicted by certain contagious diseases, e.g., the H1N1 virus. CDC needs to monitor high or unusual concentrations of infected people in schools, since that might indicate the start of an epidemic. To this end, it periodically needs to cross-check its list with student rosters in each school district. Privacy regulations prevent schools from granting wholesale access to student data. However, information regarding students with highly infectious diseases needs to be disclosed.

1.1 Why Size Matters?

All examples above have some features in common: neither party can reveal its information in its entirety. What they are willing to reveal is limited to common information, i.e., items appearing on both parties' lists. Specifically, our examples involve only *one-way* information sharing, i.e., airlines allow DHS to learn names that appear on both lists, whereas, DHS does not allow airlines to learn the same.

Another important, but more subtle, feature common to our examples is the need to keep client input size secret. Specifically, DHS does not reveal the number of names on the TWL. This list is dynamic (names can be added and removed frequently) and revealing its size would leak sensitive information. Likewise, by law, CIA cannot divulge the number of its agents, for obvious reasons. Finally, the number of infected school-kids in a city (school district) is extremely sensitive: its disclosure can cause wide-spread panic and/or prompt a health insurance rate hikes for that location.

We conclude that there are solid reasons for parties in certain privacy-preserving operations to keep sizes of their inputs secret. The most common reason is that input size represents sensitive information. A related reason is that, given multiple interactions between the same two parties, fluctuations in input size are equally (or even more) sensitive. Another factor motivating input size secrecy is related to the amount of computation imposed on the other party; we discuss it later, in Section 6.

We note, from the outset, that there are limits to input size hiding. For instance, both parties cannot hide their respective input sizes. One obvious reason is that, in all examples above, (at least) one party learns the intersection of its

input with that of the other party. The intersection itself is a list (or a set) and its size leaks information about overall input size.

In this paper, we focus on privacy-preserving interactions characterized by the examples above, where one party (client, the one that learns the intersection) aims to keep the size of its input secret. Next, we argue that no current cryptographic primitive, (including generic secure two-party computation [18]) supports input size-hiding for private set intersection. We also discuss the inadequacy of some trivial approaches.

1.2 Size Hiding with Current Tools?

Private Set Intersection (PSI) is a cryptographic primitive, introduced in [13], that allows two parties: server S and client C, to interact on their respective private input sets: $\mathcal{S}$ and $\mathcal{C}$, such that, C learns $(\mathcal{S} \cap \mathcal{C}, |\mathcal{S}|)$ and S learns nothing beyond $|\mathcal{C}|$.[1] Over the last few years, the research community has devised a number of PSI techniques that vary in costs, security assumptions and adversarial models. We discuss prior work on PSI in Section 2. One common feature of all current PSI protocols is that client input size (# of elements in client set) is revealed to server. It is unclear whether they can be extended or amended to support client input size-hiding. Also, generic secure multi-party computation tools [35] are not applicable as they provide all players with the sizes of other players' inputs.

One trivial approach is for client to employ fixed-size input, i.e., pad its input with chaff up to a certain fixed size. However, this has several drawbacks. First and foremost, this always leaks the *upper bound* of input size. Second, if client input is a dynamic set, the fixed size must reflect the maximum possible set size (otherwise, fluctuations would leak information), which entails wasted computation and communication.

1.3 Roadmap and Contributions

In this paper, we introduce the concept of *Size-Hiding Private Set Intersection* (SHI-PSI).We then present the first concrete SHI-PSI construction, offering provable security and efficient operation. Next, we discuss possible extensions to reduce protocol overhead and adapt SHI-PSI to scenarios where client needs to learn data records associated withe each item in the intersection. Finally, we compare costs of our SHI-PSI approach to prior work (on non size-hiding PSI) and show that size-hiding is attainable at very low additional costs.

Notable SHI-PSI features include:

1. It offers a superset of privacy properties of prior PSI protocols: SHI-PSI additionally hides client input size from server. Client input is hidden unconditionally, without relying on any computational assumption – a contribution in its own right.

[1] This is sometimes referred to as one-way PSI; a mutual version involves each party learning the intersection.

2. It is secure under the standard RSA assumption in the Random Oracle Model (ROM).
3. It is very efficient: (1) communication overhead is linear in server input size *only*, (2) server computation complexity is linear in the size of its input *only*, (3) client computation complexity is almost linear – $O(v \cdot \log v)$ – in the size of its input. (We note that this is remarkably low, since the most efficient PSI offers linear complexity; hence, the only "penalty" for size-hiding is a small increase in client computation: $O(v \cdot \log v)$ vs $O(v)$.)
4. It is particularly attractive for scenarios where server is mandated (e.g., by law) to take part in the interaction with client(s). In such scenarios, it makes sense to minimize burden on server, especially, when client input is large. Current PSI schemes involve server computation proportional to client input size. Whereas, in SHI-PSI, server computation depends only on its own input size.

Organization: After reviewing related work in Section 2, Section 3 defines SH-PSI as a concept and specifies its desired properties. Then, Section 4 presents a concrete SHI-PSI protocol and argues its security. Section 5 discusses possible extensions and Section 6 considers performance issues. The paper concludes with a laundry-list of future work items in Section 7.

2 Related Work

We now overview related cryptographic primitives and prior work.

Two-Party Computation (2PC). Private set operations can be performed using secure two-party computation [18,35]. 2PC allows two parties, each with its own private input, to privately evaluate a generic public function, such that nothing else is revealed. However, standard 2PC definitions (see [17]) provide both parties with the length of the other party's input. This contradicts our input size-hiding goal. Furthermore, 2PC incurs several rounds and relatively high computational overhead. Recent techniques, such as [33] and [22], proposed efficient tools for 2PC. However, special-purpose protocols for private set operations are still much faster. For instance, [22] reports the overhead of 12.8 seconds for computing the intersection of two sets with only 100 items using their 2PC-based techniques. In contrast, [11] shows the overhead of only 6 seconds for computing private set intersection for two sets with $5,000$ items (on comparable hardware), using specialized PSI tools.

Private Set Intersection (PSI). Freedman, et al. [13] devised a suite of private set operation protocols based on Oblivious Polynomial Evaluations (OPE-s) [31]. The main idea is to represent a set as a polynomial, and set elements – as its roots. Client uses an additively homomorphic cryptosystem (e.g., Paillier [32]) to encrypt coefficients, that are then evaluated by server, such that client learns the intersection (and nothing else) upon decrypting.

Assuming that client and server sets contain v and w items, respectively, their respective computation overheads amount to $O(v + w)$ and $O(w \log \log v)$

exponentiations. The protocol is secure against semi-honest adversaries in the standard model, and against malicious adversaries in ROM (with increased cost). There are several improvements to this construction against malicious adversaries in the standard model, with linear communication and quadratic computation [7], and linear communication and $O(w \log\log v)$ computation in [20] (all under DDH). Also, [27] extends OPE-s to more than two players, all learning the intersection, with quadratic computational and linear communication complexities.
Other PSI constructs rely on Oblivious Pseudo-Random Functions (OPRF-s) [12]. An OPRF is a two-party protocol that securely evaluates a pseudorandom function $f_k(\cdot)$ on key k contributed by sender and input x contributed by receiver, such that the former learns nothing from the interaction, and the latter only learns $f_k(x)$. OPRF-s can be used to obtain linear complexity PSI, e.g., [19] and [24], secure in the standard model against, respectively, covert [1] and malicious adversaries. Recent PSI results in the Random Oracle Model (ROM) yielded very efficient constructs with linear complexity and short exponentiations. They replace OPRF-s with unpredictable functions [25] and blind signatures [11]. These techniques are secure under *One-More-DH* and *One-More-RSA* assumptions [2], respectively. Very recent work in [10] achieves linear computational and communication complexity (also using short exponents) against malicious adversaries, under the DDH assumption in ROM.

Branching Programs (BP). Ishai and Paskin [23] consider the following problem: given a branching program P (held by server) and encryption c of input x (held by client), is it possible to compute ciphertext c' from which $P(x)$ can be decoded using the secret key? Note that size of c' depends, polynomially, on sizes of x and P. Thus, neither client computation nor protocol communication overhead depends on server input size P, that remains secret to client. Although one can implement PSI with a branching program and thus hide *server* input size (whereas, we focus on hiding client input size), we argue that this generic construction would involve much higher computational overhead – polynomial in the size of inputs. Also, it would require v parallel executions, where v is client input size.

Secure Pattern Matching Some recent work addressed a somewhat related problem: secure computation of *pattern matching* [19,16,26,21]. One party (P_1) holds a pattern and the other party (P_2) holds a text string. The goal of P_1 is to learn where the pattern appears in the text, without revealing it to P_2 or learning anything else about P_2's input. However, the size of P_1's pattern is always revealed to P_2. [21] sketches a possible way to hide pattern size, however, only by means of random padding. (As discussed earlier, this is an unsatisfying approach that exposes the upper bound.) It also imposes a substantial performance penalty: from linear to quadratic complexity.

Zero-Knowledge Sets. The only cryptographic primitive in the context of which the need for hiding input sizes was discussed is *Zero-Knowledge Sets* [30]. In it, server publishes a short snapshot of its private database, i.e., a commit-

ment. Later, client can request server to prove whether a given item belongs to the committed set. Note that neither the commitment nor the proof reveals server database. However, the problem addressed by ZK-Sets is quite different from (size-hiding) PSI. In fact, in ZK-Sets, client input is not private.

3 Definitions

We now formalize the concept of *Size-Hiding Private Set Intersection* (SHI-PSI). Informally, SHI-PSI extends PSI with an additional privacy feature that client input size must not be revealed to server. Clearly, SHI-PSI implies (one-way) PSI. For ease of presentation, instead of first defining PSI and then adding size-hiding, we begin by defining SHI-PSI directly.

Definition 1. (**SHI-PSI.**) *A scheme satisfying correctness, server privacy and client privacy (per Definitions 2, 3 and 4 below), involving two parties: C and S, and two components:* {Setup,Interaction}*, where:*

- Setup*: an algorithm that selects all global parameters.*
- Interaction*: a protocol between S and C on respective inputs: $\mathcal{S} = \{s_1, \cdots, s_w\}$ and $\mathcal{C} = \{c_1, \cdots, c_v\}$.*

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

We assume semi-honest parties and use general definitions of secure computation [17]. Specifically, we define SHI-PSI as a secure two-party protocol realizing functionality described above. Our client and server privacy definitions follow from those in related work [28,13,12,19]. In particular, Goldreich ([17], Sec. 7.2.2) states that, in case of semi-honest parties, the general "real-versus-ideal" definition framework is **equivalent** to a much simpler framework that extends the formulation of honest-verifier zero-knowledge. Informally, a protocol privately computes certain functionality if whatever can be obtained from one party's view of a protocol execution can be obtained from input and output of that party. In other words, the view of a semi-honest party (including $\mathcal{C}$ or $\mathcal{S}$, all messages received during execution, and the outcome of that party's internal coin tosses), on each possible input $(\mathcal{C}, \mathcal{S})$, can be efficiently simulated considering only that party's own input and output. This is equivalent to the following formulation:

Definition 3. (**Client Privacy.**) *For every* PPT S^* *that plays the role of S, for every $\mathcal{S}$, and for any client input set $(\mathcal{C}^{(0)}, \mathcal{C}^{(1)})$, two views of S^* corresponding to C's inputs: (1) $\mathcal{C}^{(0)}$ and (2) $\mathcal{C}^{(1)}$, are computationally indistinguishable.*

Client privacy is guaranteed if no information is leaked about client input. That is, S^* cannot distinguish between $\mathcal{C}^{(0)}$ and $\mathcal{C}^{(1)}$. S^* cannot even determine whether $|\mathcal{C}^{(0)}| \neq |\mathcal{C}^{(1)}|$. In fact, Definition 3 is strictly stronger than client

privacy definition for standard PSI protocols that reveal client input size. In this case, indistinguishability would be relaxed by the constraint $|\mathcal{C}^{(0)}| = |\mathcal{C}^{(1)}|$.

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

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

In other words, on each possible pair of inputs $(\mathcal{C}, \mathcal{S})$, C's view can be efficiently simulated by C^* on input: $\mathcal{C}$ and $\mathcal{S} \cap \mathcal{C}$. Thus, as in [17], we claim that the two distributions implicitly defined above are computationally indistinguishable.

Remark. As mentioned earlier, we consider security in the presence of semi-honest parties, i.e., parties that faithfully follow protocol specifications. However, during or after protocol execution, they might (passively) attempt to infer additional information about the other party's input. Note that this models precisely the class of adversaries considered in our applications. For instance, in our Example 1 in Section 1, DHS and airlines have no incentive to deviate from protocol specifications, because they might be subject to auditing and could face severe penalties for non-compliance. Nonetheless, airline personnel, system administrators, or other malicious insiders might seek to surreptitiously obtain information about contents or size of the DHS Terror Watch List (TWL).

The RSA Assumption on safe moduli. Let τ be a security parameter and let $RSA.Setup(\tau)$ be an algorithm that outputs so-called safe RSA instances, i.e., pairs (N, e), where: (1) $N = pq$ where p and q are random distinct τ-bit primes, such that $p = 2p' + 1$ and $q = 2q' + 1$ for distinct primes p', q', and (2) $e < \phi(N)$ is a random positive integer, such that $GCD(e, \phi(N)) = 1$. The RSA problem is (τ, t)-hard on 2τ-bit safe RSA moduli, if, for each algorithm $\mathcal{A}$ that runs in time t, it holds that:

$$\Pr[(N, e) \leftarrow_r RSA.Setup(\tau), y \leftarrow_r \mathbb{Z}_N^* : \mathcal{A}(N, e, y) = \beta \text{ s.t. } \beta^e = y \bmod N] \leq \mathsf{negl}(\tau).$$

We later assume that y is chosen uniformly at random from QR_N (the set of quadratic residues in $\mathbb{Z}_N^*$). Thus, the order of y is $p'q'$. In this case, we let e be a random integer (chosen independently of y) such that $\gcd(e, p'q') = 1$ with overwhelming probability. If $e = 2^t u$ for an odd integer u, then, if $t \geq 1$, $\mathcal{A}$ would compute square root of y, which is infeasible if the factoring assumption holds. If $t = 0$, then e is odd and $\mathcal{A}$ would solve an instance of the standard RSA problem.

4 SHI-PSI Construction

We now present our SHI-PSI protocol. Its two main building blocks are: (1) tools similar to RSA accumulators [4], and (2) *unpredictable* function $f_{X,p,q}(y) = (X^{1/y}) \bmod N$ (under the RSA assumption on safe moduli).[2]

Specifically, client computes a global witness for its input $\mathcal{C} = \{c_1, \cdots, c_v\}$ in the form of an RSA accumulator: $(g^{\prod_{i=1}^{v} H(c_i)}) \bmod N$, where g is a generator of QR_N and $H(\cdot)$ is a full-domain hash function [3]. Then, client securely blinds the accumulator with a random exponent and sends the result (denoted as X) to server. The latter learns no information about client input, not even its size. For its part, for each item $s_j \in \mathcal{S}$, server computes unpredictable function f over client message X. Server then applies a one-way function (in practice, a suitable cryptographic hash function) to each output of f. The results are a set of *tags*, one for each s_j. These tags are then returned to client for matching (details below). We note that the outer hash is crucial, since server privacy is based on the fact that, in ROM, a hash of an unpredictable function is a PRF.

In the above, $H(\cdot)$ is a standard random oracle that does not have to output large primes. Also, we obviate the technical issue of computing the inverse of $H(s_j)$ "in the exponent" by selecting the RSA modulus N as a product of safe primes to ensure that the order of X is itself a product of large and unknown primes (see proof for details).

Client learns the set intersection as well as $|\mathcal{S}|$) since it can only match tags corresponding to the items in the intersection. The intuition is that client computation of $g^{(\prod_{l \neq i} H(c_l))}$ leads to it finding a matching tag only if $c_i \in \mathcal{S} \cap \mathcal{C}$.

Before presenting the actual protocol, we introduce the notation in Table 1.

$a \leftarrow_r \mathcal{A}$	variable a is chosen uniformly at random from set $\mathcal{A}$
τ	security parameter
τ_1, τ_2	security parameters that depend on τ
p, q	two τ-bit safe primes, i.e., $p = 2p' + 1, q = 2q' + 1$
$N = pq, e, d$	RSA modulus, public and private exponents
g	generator of QR_N
$H(\cdot)$	random oracle $H : \{0,1\}^* \rightarrow \{0,1\}^{\tau_1}$
$F(\cdot)$	random oracle $F : \{0,1\}^* \rightarrow \{0,1\}^{\tau_2}$
$\mathcal{C}, \mathcal{S}$	client and server sets, respectively
v, w	sizes of $\mathcal{C}$ and $\mathcal{S}$, respectively
$i \in [1, v]$	indices of elements of $\mathcal{C}$
$j \in [1, w]$	indices of elements of $\mathcal{S}$
c_i, s_j	i-th and j-th elements of $\mathcal{C}$ and $\mathcal{S}$, respectively
hc_i, hs_j	$H(c_i)$ and $H(s_j)$, respectively
π	random permutation

Table 1. Notation.

[2] A function (family) $f_k(\cdot)$ is an $(t, q_f, \epsilon)-$unpredictable if, for any t-time algorithm $\mathcal{A}$ and any auxiliary information z, it holds that: $Pr[(x^*, f_k(x^*) \leftarrow_r \mathcal{A}^{f_k(\cdot)}(z)) \wedge x^* \notin \mathcal{Q}] \leq \epsilon$ where $\mathcal{A}$ makes at most q_f queries to $f_k(\cdot)$, and $\mathcal{Q}$ is the set of queries.

4.1 Protocol Description

Fig. 1 shows our SHI-PSI protocol. Common input is extracted from the output of $RSA.Setup(\tau)$. Primes p' and q' are provided exclusively to server. Client must treat its exponents as large integers. We emphasize that arithmetic operations in the exponent are performed in $\mathbb{Z}^*_{p'q'}$. (In particular, squaring is a permutation of QR_N, in our setting.)

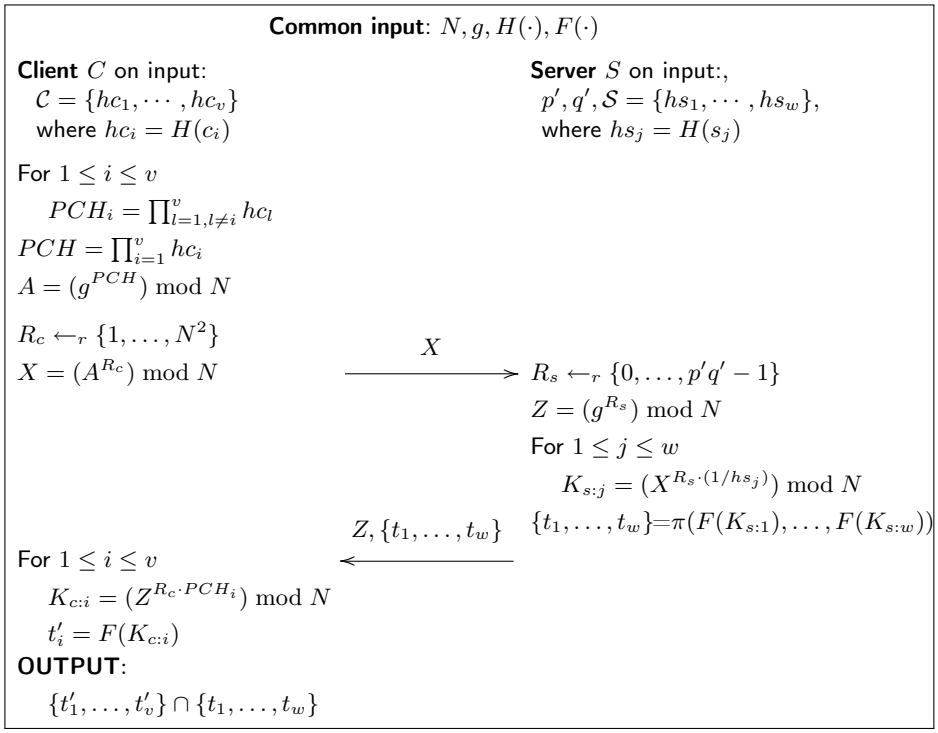

Fig. 1. SHIPSI protocol secure under RSA assumption in ROM (notation from Table 1).

Theorem 1. *Under the RSA assumption on safe moduli, the protocol in Fig. 1 is a server- and client-private SHI-PSI, satisfying Definitions 1-4. in the Random Oracle Model (ROM).*

Proof. We show that the protocol satisfies *correctness as well as client and server privacy*, defined in Section 3. We assume that all server elements are distinct.[3] Hash functions $H(\cdot)$ and $F(\cdot)$ are modeled as random oracles.

Correctness. We note that: $\forall c_i \in \mathcal{S} \cap \mathcal{C}, \exists\ j\ s.t. c_i = s_j$. Hence, $hc_i = hs_j$ and:

[3] However, we can remove this assumption by adding a counter to the input of $H(\cdot)$.

$$K_{c:i} = Z^{R_c \cdot PCH_i} = g^{R_c R_s PCH(1/hs_j)}$$
$$K_{s:j} = X^{R_s \cdot (1/hs_j)} = g^{R_c R_s PCH(1/hs_j)}$$

Consequently, $t'_i = F(K_{c:i}) = F(K_{s:j}) = t_j$; thus, client learns: $c_i \in \mathcal{S} \cap \mathcal{C}$ as well as $|\mathcal{S}| = |\{t_1, \ldots, t_w\}|$.

Client Privacy. Since client's only message to server is $X = g^{(PCH \cdot R_c)} \bmod N$, we claim that the distribution of X is essentially equivalent to that of random elements in QR_N, which is a cyclic group of order $p'q'$. Since PCH and $p'q'$ are relatively prime (with overwhelming probability), we assume that $A = g^{PCH} \bmod N$ is a generator of QR_N. Moreover, R_c is chosen uniformly at random from $\{1, \ldots, N^2\}$. Thus, if $R_c = r_1 p'q' + r_2$ with $r_2 \in \{0, \ldots, p'q' - 1\}$, we have that the distribution of r_2 is *statistically* indistinguishable from the uniform distribution on $\{0, \ldots, p'q' - 1\}$ and r_1 and r_2 are essentially independent (see, e.g., [6]). Therefore, $X = A^{R_c} \bmod N$ is essentially distributed as a random quadratic residue independent of PCH *even if factorization of* N *is known.*

Server Privacy. To show that client's view can be efficiently simulated by a PPT algorithm, we follow a hybrid argument: The entire client's view is gradually transformed by replacing values (received by client) that are **outside** the set intersection, with elements chosen uniformly and independently at random. It then suffices to show that this progressive substitution cannot be detected by any efficient algorithm.

Let $I = \mathcal{C} \cap \mathcal{S}$, and $|I| = t$. For any $(\mathcal{C}, \mathcal{S})$, we show that two distributions:

$$\mathcal{D}_0 = \left\{ (R_c, T) \ : \ R_c \leftarrow_r \{1, \ldots, N^2\}, T = \pi \left(F(X^{R_s(1/hs_{j1})}), \cdots, F(X^{R_s(1/hs_{jw})}) \right) \right\}$$

and

$$\mathcal{D}_{w-t} = \left\{ (R_c, T) \ : \ R_c \leftarrow_r \{1, \ldots, N^2\}, T = \pi \left(F(X^{R_s(1/hs_{j1})}), \cdots, r_{t+1}, \cdots, r_w \right) \right\},$$

are computationally indistinguishable, where $(hs_{j1}, \cdots, hs_{jt}) \in I$ and values in $(r_{t+1}, \cdots, r_w)$ are chosen uniformly and independently at random from $\{0,1\}^{\tau_2}$ (i.e., the co-domain of the random oracle $F(\cdot)$).

Our proof follows the standard hybrid argument: Let $z = w - t$. We define a series of intermediate distributions $\mathcal{D}_i$, for $0 < i < z$, where T is constructed by replacing the first i outputs of items NOT in I with random values in the co-domain of $F(\cdot)$.

After fixing index i and probabilistic polynomial-time distinguisher D, we define:

$$\epsilon(\tau) = |\Pr[D = 1 | \mathcal{D}_{i+1}] - \Pr[D = 1 | \mathcal{D}_i]|$$

Our claim is that $\epsilon(\tau)$ is negligible in τ. Let us assume that this claim is false. The only difference between $\mathcal{D}_i$ and $\mathcal{D}_{i+1}$ is the way T is defined. Specifically, $(i+1)$-st item of T not in I is $F(X^{R_s(1/hs_l)})$ for $\mathcal{D}_i$ and a random value for $\mathcal{D}_{i+1}$.

Since $F(\cdot)$ is a random oracle, distinguisher D must compute $X^{R_s(1/hs_l)} = g^{R_s R_c PCH/hs_l}$ for $hs_l \notin I$. Then, we can build an efficient algorithm $\mathcal{A}$ that, given a challenge (N, e, y), returns $y^{1/e} \bmod N$. (We assume that y is chosen uniformly

at random from QR_N. Thus, the order of y is $p'q'$.) The simulation proceeds as follows: First, $\mathcal{A}$ sets $g = y$ and, by programming the random oracle $H(\cdot)$, $\mathcal{A}$ assigns random values to outputs of $H(\cdot)$ and computes $d = \gcd(R_s R_c PCH, hs_l)$, for some integers e and b with $hs_l = ed$ and $R_s R_c PCH = bd$. Since $F(\cdot)$ is a random oracle, $\mathcal{A}$ sees $g^{R_s R_c PCH/hs_l} = g^{b/e}$. Given that $(g^{b/e})^e = g^b$ and $\gcd(e, b) = 1$, $\mathcal{A}$ can use the extended Euclidean algorithm to compute $g^{1/e} = y^{1/e}$ via the well-known *Shamir's trick*[4]. Thus, under the RSA assumption on safe moduli, formulated for a random exponent, $\epsilon(\tau)$ is negligible in τ.

Remarks. We stress that exponents in our scheme (i.e., outputs of $H(\cdot)$) do not have to be prime, unlike related reductions, such as [34,4,29,15]. This is because client cannot compute $g^{R_s R_c PCH/hs_l}$, for $l \in \{1, \ldots, w\}$, unless $R_c PCH/hs_l$ is an integer. (Recall that R_c is generated honestly). Clearly, if $hs_l \notin I$, $R_c PCH/hs_l$ is, – with negligible probability – an integer as long as hs_l is sufficiently large: random oracles are indeed *division intractable*, as shown in [15,5] (in particular, [5] presents an algorithm for finding division collisions sub-exponential in τ_1, the digest size).
We readily acknowledge that our construction assumes both semi-honest players and the Random Oracle Model. Nevertheless, on a positive side, it is interesting to observe that generic 2PC techniques, following traditional definitions that also apply to malicious adversaries, do not achieve size-hiding of client input. Goldreich [17] remarks that the program of each party (in a protocol for computing the desired functionality) must either depend only on the length of that party's input or obtain information on the counterpart's input length. One intuitive argument against the feasibility of input size-hiding protocols secure in the malicious model is that proving well-formed-ness of client input is only possible by considering each client input set element separately (e.g., via some ZK proofs). Thus, combined proofs would have to reveal at least the upper bound on client input size.

Computational and Communication Complexity. The protocol in Fig. 1 incurs the following computational complexity (in terms of modular exponentiations). In each interaction, server needs to compute $O(w)$ exponentiations, one for each of its items. Whereas, client operations are divided into *off-line* and *on-line* categories. Client *off-line* work amounts to $O(v)$ exponentiations for the computation of $A = g^{PCH} \bmod N$, since PCH is the non-modular product of v values. Additionally, client computes $K_{c:i} = Z^{Rc \cdot PCH_i} \bmod N$ for each item. As each of these operations requires $O(v)$ exponentiations, *on-line* client complexity amounts to $O(v^2)$ exponentiations.[5] Communication complexity in each

[4] Note that this is similar to the reduction in [15]. However, in contrast to Theorem 5 in [15], our reduction is not based on the strong RSA assumption, but on the standard RSA assumption in ROM. This is because e is generated independently of base y and, thus, e is effectively provided as input to the adversary. Indeed, the signature scheme in [15] is actually secure under the standard RSA assumption in ROM; this was confirmed via private communication [14].

[5] Note that, if client knew the factorization of N, it could compute PCH and PCH_i's using multiplication $\bmod \phi(N)$, thus significantly reducing complexity of each expo-

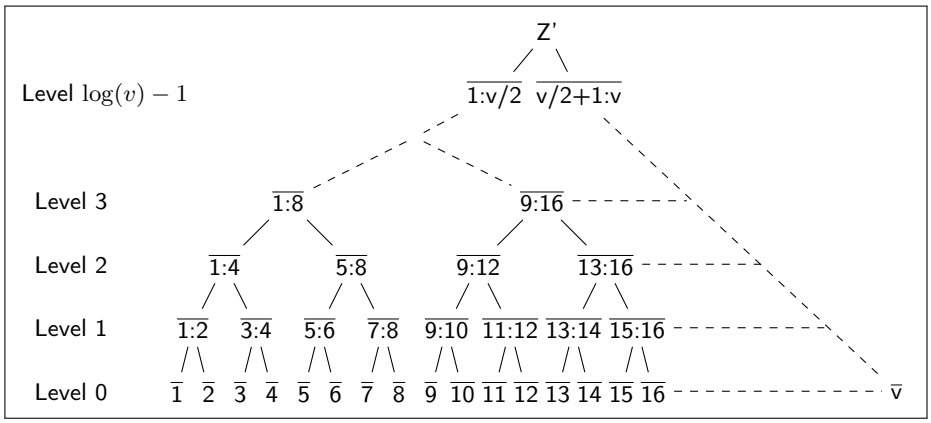

Fig. 2. Tree-based strategy to reduce client computation.

interaction is dominated by $O(w)$ outputs of $F(\cdot)$ sent from S to C in the second message. (The first message involves the transmission of a single $\log(N)$-bit value).

4.2 Reducing Client Complexity

We now discuss a simple technique to reduce client computation. Note that the naïve computation of $K_{c:i}$ leads to $O(v^2)$ exponentiations. However, this can be reduced to $O(v \log(v))$ via dynamic programming. Our intuition is as follows: For any (i, j), $K_{c:i}$ and $K_{c:j}$ only differ by one exponent, since $PCH_i = \prod_{l=1, l\neq i}^{v} hc_l$, whereas, $PCH_j = \prod_{l=1, l\neq j}^{v} hc_l$.

We define $Z' = Z^{Rc}$, and $\overline{\mathsf{i{:}j}} = Z'^{[\prod_{l\notin[i,j]} hc_l]} \bmod N$. We illustrate this technique using a tree in Fig. 2. The leaves contain values $K_{c:i}$, for $1 \leq i \leq v$, e.g., $\overline{i} = Z'^{[\prod_{l\neq i} hc_l]} \bmod N = Z^{Rc \cdot PCH_i} \bmod N = K_{c:i}$.

We now sum up the total number of exponentiations needed to compute all these values. Note that, from a node with value $\overline{\mathsf{i{:}j}}$, one can obtain the children, $\overline{\mathsf{i{:}h}}$ and $\overline{\mathsf{h{+}1{:}j}}$, as follows:

$$\overline{\mathsf{i{:}h}} = \left(\overline{\mathsf{i{:}j}}\right)^{\left(\prod_{l=h+1}^{j} hc_l\right)} \pmod N$$

$$\overline{\mathsf{h{+}1{:}j}} = \left(\overline{\mathsf{i{:}j}}\right)^{\left(\prod_{l=i}^{h} hc_l\right)} \pmod N$$

For $h = i + (j - i + 1)/2$, each of these operations involves exactly $(j - i + 1)/2$ exponentiations.

At level 0, there are v values, each obtained with a single exponentiation from the parents at level 1. At level 1, there are $v/2$ values, each obtained with

nentiation. However, as discussed earlier, the fact that client does not know $\phi(N)$ is crucial to server privacy.

2 exponentiations from nodes at level 2. In general, at level i, there are $v/2^i$ values, each obtained with 2^i exponentiations from nodes at level $i+1$. Thus, client overhead can be estimated as:

$$\#\text{ exponentiations} = \sum_{i=0}^{\log(v)-1} \left(2^i \frac{v}{2^i}\right) = v\log(v).$$

5 Extensions

In this section, we discuss possible extensions to our SHI-PSI construction presented above.

5.1 Linear-Complexity SHI-PSI

In many scenarios, parties engage in multiple interactions, and it is important to hide (from client) any changes in server input. This feature is sometimes referred to as *unlinkability*: client cannot determine whether any two server interactions are related, i.e., executed on the same input (e.g., see unlinkability definitions in [9]).

The SHI-PSI construct in Fig. 1 clearly guarantees unlinkability: server tags are unlinkable across multiple interactions, since server computes a new random R_s and a new $Z \in QR_N$, in each execution. However, although we clearly value unlinkability, it is worth considering scenarios where it might be reasonable to trade off unlinkability for better efficiency.

To this end, we sketch a modified SHI-PSI protocol that reduces the number of client on-line exponentiations to linear. The main intuition is that removing R_s allows client to pre-compute the exponentiations involving (long) PCH_i values. We illustrate the resulting protocol in Fig. 3. Note that security arguments in Theorem 1 also apply to this protocol variant. Indeed, given assuming semi-honest client, $X = g^{PCH \cdot R_c}$, similar to X^{R_s} in protocol of Fig. 1, is also uniformly distributed in QR_N, for $R_c \leftarrow_r \{1, \ldots, N^2\}$. A more formal treatment of the problem as well as complete formal proofs are deferred to the full version of this paper.

Correctness. Observe that $\forall c_i \in \mathcal{S} \cap \mathcal{C}, \exists\, j\, s.t. c_i = s_j$. Hence, $hc_i = hs_j$ and:

$$K_{s:j} = X^{1/hs_j} = g^{R_c PCH(1/hs_j)} = g^{R_c \cdot PCH_i} = K_{c:i}$$

Consequently, $t'_i = F(K_{c:i}) = F(K_{s:j}) = t_j$, and client learns $c_i \in \mathcal{S} \cap \mathcal{C}$. Also, note that client learns $|\mathcal{S}| = |\{t_1, \ldots, t_w\}|$.

Computational and Communication Complexity. The amended SHI-PSI construct in Fig. 3 incurs the following computational complexity: Server overhead is unaltered from Fig. 1, i.e., $O(w)$ exponentiations. However, client performs $O(v\log(v))$ exponentiations *off-line*, and only $O(v)$ exponentiations *on-line*. Communication overhead is the same as in the protocol of Fig. 1.

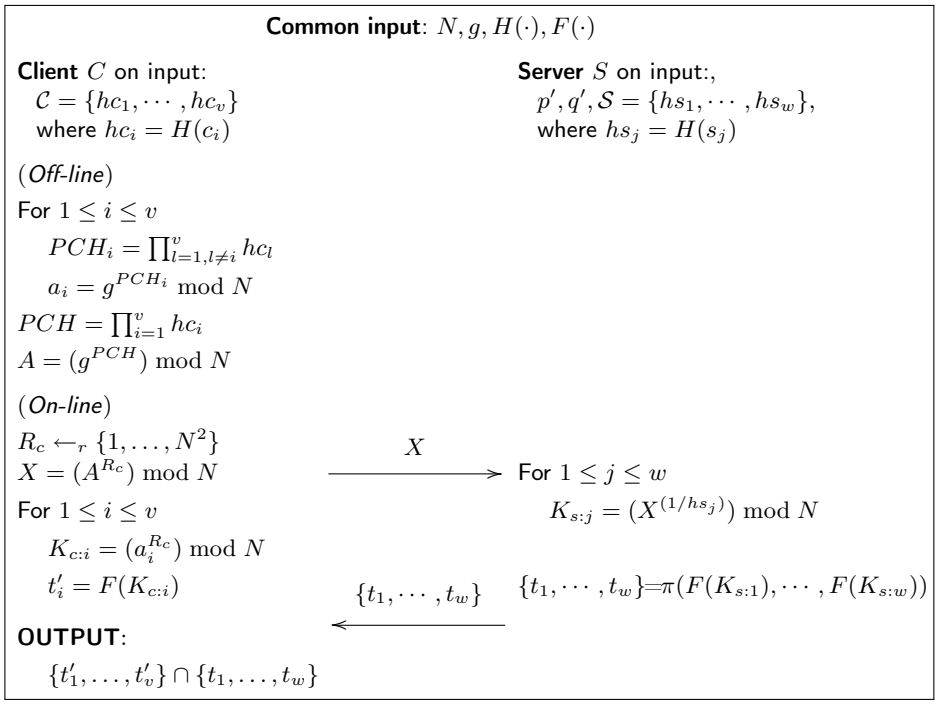

Fig. 3. Modified construction of the SHI-PSI protocol to achieve linear client complexity. (Notation is from Table 1).

5.2 SHI-PSI with Data Transfer

We now show how to extend proposed SHI-PSI constructs to support data transfer. Informally, in *SHI-PSI with Data Transfer*, client additionally obtains data records associated with the items in the intersection. The main idea is to let server encrypt records using a symmetric key (using a secure symmetric cipher, such as AES [8], used with a proper mode of operation to guarantee CPA security) derived from the output of the unpredictable function. For example, keys can be derived by computing a one-way function (e.g., a cryptographic hash) over the unpredictable function output. Correctness and server privacy of SHI-PSI guarantee that client can derive the decryption keys only for items with matching tags, i.e., those in the intersection.

Let $F_{enc}(\cdot)$ be a secure cryptographic hash function (modeled as a random oracle): $F_{enc} : \{0,1\}^* \rightarrow \{0,1\}^{\tau_2}$, chosen at setup. For every j, server computes $k_{s:j} = F_{enc}(K_{s:j})$ and encrypts associated data using $k_{s:j}$. For its part, client, for every i, computes $k_{c:i} = F_{enc}(K_{c:i})$ and decrypts only ciphertexts corresponding to matching tags. (Note that $k_{s:j} = k_{c:i}$ iff $s_j = c_i$ and $t_j = t'_i$). As long as the underlying encryption scheme is CPA-secure, this extension does not affect security or privacy arguments for any protocol discussed thus far. Finally, note that this extension leaves the complexity of both protocols unaltered.

6 Cost of Hiding Size

Although prior work produced a number of PSI protocols with different security assumptions and complexities, we presented the first PSI protocol that hides client input size. Therefore, it seems somewhat counterintuitive to compare our SHI-PSI constructs with prior PSI protocols. Nonetheless, we provide an estimate of the slow-down incurred by client input size-hiding.

We consider prior PSI techniques and evaluate their asymptotic computation and communication complexities in the RAM computational model (i.e., using a single-processor machine). We estimate computation overhead as the number of *on-line* modular exponentiations performed by server and client. Note that, in order to make the comparison as fair as possible, all protocols are instantiated to provide similar degrees of security in the same model, i.e., semi-honest players and ROM. For instance, we do not count zero-knowledge proofs of protocol compliance in protocols secure against malicious adversaries. Results, reflected in Table 2, summarize: security model (standard or ROM), adversaries (honest-but-curious or malicious), availability of client input size-hiding, communication overhead, number of modular exponentiations by server and client, size of random exponents (i.e., whether they can be selected from subgroups).

Protocol	**Model**	**Adv**	**Size Hiding**	**Comm. Overhead**	**Server Exp-s**	**Client Exp-s**	**Exp-s Length**
[13]	Std	HbC	No	$O(w+v)$	$O(w(\log\log v))$	$O(w+v)$	Short
[20]	Std	Mal	No	$O(w+v)$	$O(w(\log\log v))$	$O(w+v)$	Short
[27]	Std	HbC	No	$O(w+v)$	$O(w \cdot v)$	$O(w+v)$	Long
[24]	Std	Mal	No	$O(w+v)$	$O(w+v)$	$O(v)$	Long
[25]	ROM	Mal	No	$O(w+v)$	$O(w+v)$	$O(v)$	Short
[11] (Fig.3)	ROM	HbC	No	$O(w+v)$	$O(w+v)$	$O(v)$	Short
[11] (Fig.4)	ROM	HbC(*)	No	$O(w+v)$	$O(w+v)$	$O(v)$ *mults*	Long
[10]	ROM	Mal	No	$O(w+v)$	$O(w+v)$	$O(v)$	Short
Our Fig.1	ROM	HbC	**Yes**	$O(w)$	$O(w)$	$O(v \log v)$	Long
Our Fig.3	ROM	HbC	**Yes**	$O(w)$	$O(w)$	$O(v)$	Long

(*) This construct actually achieves malicious security with one-sided simulatability.

Table 2. Performance Comparison of PSI and SHI-PSI constructions.

Observe that the most efficient PSI-s secure in the random oracle model incur linear computational complexities, i.e., $O(w+v)$ for server and $O(v)$ for client, and involve short exponents (e.g., 160-bit) in prime order groups. Whereas, our SHI-PSI protocol (Fig. 1) uses exponents with length close to the RSA modulus (e.g., 1024-bit) and incurs $O(v \log v)$ client complexity. However, such a drawback is experienced by one player – client – that benefits from additional privacy, as its set size is hidden from server. Also, note that our second SHI-PSI construct (Fig. 3) reduces client complexity to $O(v)$.

Finally, we remark that our SHI-PSI constructs achieve better server complexity than PSI-s, in settings where v (the size of client's set) is not negligible.

In fact, *server's computational load in SHI-PSI is independent of client's input size*. Also, *protocols not hiding sizes incur higher communication overhead*: client sends a number of values proportional to its set size (as opposed to a single value in SHI-PSI).

7 Conclusions and Future Work

This paper motivated the importance, and introduced the concept, of Size-Hiding Private Set Intersection (SHI-PSI). It also presented two secure and efficient SHI-PSI constructs, Since this work represents an initial foray into SHI-PSI protocols, much remains to be done. Items for future work, include (but are not limited to):

1. Amending proposed SHI-PSI constructs to eliminate the random oracle.
2. Exploring SHI-PSI secure against fully malicious (rather than semi-honest) participants.
3. Investigating SHI-PSI variants that provide authorization of client input, i.e., requiring each item in client set to be pre-authorized by some trusted authority.
4. Extending SHI-PSI to support multiple clients to obtain private computation of size-hiding a *multi-party* set intersection.

Acknowledgements. We are grateful to Stanislaw Jarecki and Adi Shamir for some useful hints in the early stage of SHI-PSI protocol design, as well as the anonymous reviewers for providing valuable feedback. This research was supported by the US Intelligence Advanced Research Projects Activity (IARPA) under grant number FA8750-09-2-0071.

References

1. Y. Aumann and Y. Lindell. Security Against Covert Adversaries: Efficient Protocols for Realistic Adversaries. In *TCC*, pages 137–156, 2007.
2. M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko. The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. *Journal of Cryptology*, 16(3):185–215, 2008.
3. M. Bellare and P. Rogaway. The exact security of digital signatures-How to sign with RSA and Rabin. In *Eurocrypt*, pages 399–416, 1996.
4. J. Benaloh and M. De Mare. One-way accumulators: A decentralized alternative to digital signatures. In *Eurocrypt*, pages 274–285, 1994.
5. J. Coron and D. Naccache. Security analysis of the Gennaro-Halevi-Rabin signature scheme. In *Eurocrypt*, pages 91–101, 2000.
6. R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. *ACM Transactions on Information and System Security*, 3(3):185, 2000.
7. D. Dachman-Soled, T. Malkin, M. Raykova, and M. Yung. Efficient Robust Private Set Intersection. In *ACNS*, pages 125–142, 2009.
8. J. Daeman and V. Rijmen. AES proposal: Rijndael. 1999.

9. E. De Cristofaro, S. Jarecki, J. Kim, and G. Tsudik. Privacy-Preserving Policy-Based Information Transfer. In *PETS*, pages 164–183, 2009.
10. E. De Cristofaro, J. Kim, and G. Tsudik. Linear-Complexity Private Set Intersection Protocols Secure in Malicious Model. In *Asiacrypt*, pages 213–231, 2010.
11. E. De Cristofaro and G. Tsudik. Practical Private Set Intersection with Linear Complexity. In *Financial Cryptography and Data Security*, pages 143–159, 2010.
12. M. Freedman, Y. Ishai, B. Pinkas, and O. Reingold. Keyword search and oblivious pseudorandom functions. In *TCC*, pages 303–324, 2005.
13. M. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Eurocrypt*, pages 1–19, 2004.
14. R. Gennaro. Private Communication, 2010.
15. R. Gennaro, S. Halevi, and T. Rabin. Secure hash-and-sign signatures without the random oracle. In *Eurocrypt*, pages 123–139, 1999.
16. R. Gennaro, C. Hazay, and J. Sorensen. Text Search Protocols with Simulation Based Security. In *PKC*, pages 332–350, 2010.
17. O. Goldreich. *Foundations of Cryptography. Vol 2.* Cambridge Univ. Press, 2004.
18. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *STOC*, pages 218–229, 1987.
19. C. Hazay and Y. Lindell. Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. In *TCC*, pages 155–175, 2008.
20. C. Hazay and K. Nissim. Efficient Set Operations in the Presence of Malicious Adversaries. In *PKC*, pages 312–331, 2010.
21. C. Hazay and T. Toft. Computationally secure pattern matching in the presence of malicious adversaries. In *Asiacrypt*, pages 195–212, 2010.
22. W. Henecka, S. Koegl, A.-R. Sadeghi, T. Schneider, and I. Wehrenberg. TASTY: Tool for Automating Secure Two-partY computations. In *ACM CCS*, 2010.
23. Y. Ishai and A. Paskin. Evaluating branching programs on encrypted data. In *TCC*, pages 575–594, 2007.
24. S. Jarecki and X. Liu. Efficient Oblivious Pseudorandom Function with Applications to Adaptive OT and Secure Computation of Set Intersection. In *TCC*, pages 577–594, 2009.
25. S. Jarecki and X. Liu. Fast Secure Computation of Set Intersection. In *SCN*, 2010.
26. J. Katz and L. Malka. Secure text processing with applications to private dna matching. In *ACM CCS*, pages 485–492, 2010.
27. L. Kissner and D. Song. Privacy-preserving set operations. In *Crypto*, pages 241–257, 2005.
28. Y. Lindell and B. Pinkas. Privacy Preserving Data Mining. In *Crypto*, pages 36–54, 2000.
29. S. Micali, M. Rabin, and S. Vadhan. Verifiable random functions. In *FOCS*, pages 120–130, 1999.
30. S. Micali, M. O. Rabin, and J. Kilian. Zero-knowledge sets. In *FOCS*, 2003.
31. M. Naor and B. Pinkas. Oblivious polynomial evaluation. *SIAM Journal on Computing*, 1–35(5):1254, 2006.
32. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Eurocrypt*, pages 223–238, 1999.
33. B. Pinkas, T. Schneider, N. Smart, and S. Williams. Secure two-party computation is practical. In *Asiacrypt*, pages 250–267, 2009.
34. A. Shamir. On the generation of cryptographically strong pseudorandom sequences. *ACM Transactions on Computer Systems*, 1(1):38–44, 1983.
35. A. Yao. Protocols for secure computations. In *FOCS*, pages 160–164, 1982.