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

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

[1] Johns Hopkins University, *ateniese@cs.jhu.edu*

[2] University of California, Irvine, *{edecrist, gts}@ics.uci.edu*

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 (if not always efficient) solutions have been proposed. Among them, Private Set Intersection 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 stronger privacy properties that include hiding of the set size held by one of the two entities (Client). This new and important privacy feature turns out to be attainable at relative low additional cost. We illustrate a pair of concrete SHI-PSI (Size-Hiding Private Set Intersection) protocols that offer a trade-off between stronger privacy and better efficiency. Both protocols are provably secure under very standard cryptographic assumptions. We demonstrate their practicality via experimental results obtained from a prototype implementation. We also consider size-hiding in a group PSI setting and construct a Group SHI-PSI extension that incurs surprisingly low overhead.

1 Introduction

Operations that involve sharing sensitive or private information are increasingly often encountered in everyday life. A typical scenario involves two entities: one that seeks certain information and the other that possible 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 operation. To better 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 (and/or per county), afflicted by a contagious disease, 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?

The three examples above exhibit some similar features: Neither party can reveal its information wholesale. What they are willing to reveal is limited to common information, i.e., items appearing on each party's 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 somewhat subtle, feature common to our three examples is the need to keep input sizes secret. Specifically, DHS does not reveal the number of names on its 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 (e.g., number of entries in a list) represents sensitive information. Another related reason is that, given multiple interactions between the same two parties, fluctuations in input size are equally (or even more) sensitive. There is an additional factor motivating secrecy of one party's input size, related to the amount of computation imposed to the other party; we discuss it later, in Section 7.

We note from the outset that there are limits to input size hiding. For instance, both parties in the interaction cannot hide their respective input sizes from each other. 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 is itself a list (or a set) the size of which leaks information about overall input size. This is particularly the case whenever there are multiple interactions between the same two parties.

In this paper, we focus on privacy-preserving interactions characterized by the examples above, where one party (the one learning the intersection of the two inputs) aims to keep the size of its input secret. Next, we argue that no current cryptographic primitive supports the desired size-hiding feature. We also discuss the inadequacy of some trivial approaches.

1.2 Input Size Hiding with Current Tools?

Private Set Intersection (PSI) is an appealing cryptographic primitive, introduced in [15], that allows two parties: server S and client C, to interact on their respective private input sets $\mathcal{S}$ and $\mathcal{C}$, such that, at the end of the interaction, C learns $\mathcal{S} \cap \mathcal{C}$ and S learns nothing. Since only C obtains the set intersection, we refer to this operation as *one-way*. In *mutual* PSI, both parties learn the intersection of their respective sets. This is easily obtained by running one-way PSI twice, assuming that neither party aborts prematurely.

Over the last few years, the research community has devised a number of PSI techniques that vary in features, such as computational & bandwidth complexities, underlying security assumptions and the adversarial model. We discuss prior work on PSI in Section 4. One common feature of all current PSI protocols is that client input size (# of elements in client's set) is revealed to the server. It is not at all clear how – or whether it is even possible – to extend any current PSI technique to support client input size hiding.

One trivial approach is for the client to employ fixed-size input, i.e., fill up its input with random padding up to a certain fixed size. However, this has several drawbacks. First and foremost, this approach always leaks the upper bound of client 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 bandwidth.

1.3 Roadmap and Contributions

In this paper, we motivate the need for, and introduce the concept of, *Size-Hiding Private Set Intersection* (SHI-PSI).[1] We then present three concrete SHI-PSI schemes that offer provable security and efficient operation. Two are geared for the more common 2-party scenario, and the third – for a group setting where a set of n clients interact with one server with the goal of computing a private group set intersection, such that the server does not learn client input or its size. To confirm the practicality of proposed schemes, we report on a full-blown implementation of proposed schemes and experimental performance results.

Key features of our work include:

- Proposed SHI-PSI schemes offer a superset of privacy properties of prior PSI schemes: they additionally hide client input size from the server. We stress that client input is hidden unconditionally, without relying on any computational assumption, which is, a contribution in its own right.
- Proposed 2-party SHI-PSI schemes are very efficient: (1) bandwidth overhead is linear in server input size, (2) server computation complexity is linear in the size of its input *only*, (3) client computation complexity is sub-quadratic – $O(v \cdot \log v)$ – in the size of its input, i.e., v. We note that this complexity is remarkably low since the most efficient PSI schemes offer linear bandwidth and computation complexities; hence, the only "penalty" for attaining the size-hiding property is a small increase in client computation: $O(v \cdot \log v)$ vs $O(v)$.
- In contrast to (most) prior results, proposed schemes are shown to be secure under very standard cryptographic assumptions: RSA assumption in the Random Oracle Model (ROM). (However, this is only in the semi-honest participant/adversary model; some prior results are secure against malicious participants.)
- Our schemes are particularly attractive for scenarios where the server is mandated (e.g., by law) to take part in the interaction with the client(s). In such scenarios, it makes sense to minimize the burden on the server, especially, when client input contains many elements. Current "efficient" PSI schemes force the amount of server computation to depend on client input size. In contrast, our schemes offer a nice side-effect of server computation being dependent on its own input size only.

Organization: after reviewing cryptographic assumptions in Section 2, we define Size-Hiding Private Set Intersection as a concept and specify its desired security/privacy properties in Section 3. Related work is overviewed in Section 4. Then, Section 5 constructs two SHI-PSI protocols and argues their security. Group SHI-PSI is addressed in Section 6 and performance characteristics of proposed protocols are discussed in Section 7. The paper concludes with a laundry-list of future work in Section 8.

2 Preliminaries

This section recaps standard cryptographic assumptions used in the rest of the paper.

The RSA Assumption. Let $RSASetup(\tau)$ be an algorithm that outputs so-called safe RSA instances, i.e., pairs (N, e) where $N = pq$, e is relatively prime to $\phi(N)$, and p, q are randomly generated τ-bit primes subject to the constraint that $p = 2p' + 1$, $q = 2q' + 1$ for prime $p', q', p' \neq q'$. The RSA problem is (τ, t)-hard on 2τ-bit safe RSA moduli, if for every algorithm $\mathcal{A}$ that runs in time t, it holds:
$Pr[(N, e) \leftarrow_r RSASetup(\tau), \alpha \leftarrow_r \mathbb{Z}_N^* : \mathcal{A}(n, e, \alpha) = \beta \text{ s.t. } \beta^e = \alpha \pmod N] \leq \tau$.

[1]To the best of our knowledge, this is the first attempt to do so.

Unpredictable Function. We also rely on the standard concept of an *unpredictable function*, sometimes also referred to as *unique signature*. A function (family) $f_k(\cdot)$ is an $(t, q_f, \epsilon)-$unpredictable function if for any t-time algorithm $\mathcal{A}$, any auxiliary information z, it holds

$$Pr[(x^*, f_k(x^*) \leftarrow_r \mathcal{A}^{f_k(\cdot)}(z)) \wedge x^* \notin \mathcal{Q}] \leq \epsilon$$

where $\mathcal{A}$ makes at most q_f queries to $f_k(\cdot)$, and $\mathcal{Q}$ is the set of queries.

3 Size-Hiding PSI

Traditional private set intersection protocols allow two parties: a server and a client, to interact on their respective input sets, such that the client learns the intersection, and the server learns nothing *beyond the client set size*. Since only one party obtains the intersection, we call this protocol *one-way* and denote it as PSI.

Next, we introduce and formalize the concept of *Size-Hiding* Private Set Intersection (SHI-PSI). Informally, SHI-PSI extends PSI by requiring that client input size is not revealed to the server. Clearly, SHI-PSI implies PSI. For ease of presentation, instead of first defining PSI and then adding the size-hiding requirement, we immediately provide a formal and concise SHI-PSI definition. Desired properties of a SHI-PSI scheme are: correctness, client privacy, and server privacy. Additional requirements are: client unlinkability and server unlinkability. We define SHI-PSI as a protocol that satisfies these definitions.

Definition 1. (SHI-PSI.) *A two-party protocol:*

- *Consisting of two algorithms, i.e., {*Setup,Interaction*}.* Setup *is a process wherein all global parameters are selected.* Interaction *is a protocol between a server S on input $\mathcal{S} = \{s_1, \cdots, s_w\}$, and a client C on input $\mathcal{C} = \{c_1, \cdots, c_v\}$, and at the end of the protocol the client learns $\mathcal{S} \cap \mathcal{C}$.*
- *Satisfying correctness, server privacy, client privacy, in Definitions 2, 3, 4.*

Definition 2. (**Correctness.**) *If both parties are honest, then, at the end of* Interaction*, run on inputs $(\mathcal{S}, \mathcal{C})$, the server outputs $\perp$, and the client outputs $\mathcal{S} \cap \mathcal{C}$ or $\perp$ if no intersection exists.*

Definition 3. (**Client Privacy.**) *For any* PPT *S^*that plays the server role on input $\mathcal{S}$', and for any client input sets $(\mathcal{C}^{(0)}, \mathcal{C}^{(1)})$, the views of S^*if client inputs $\mathcal{C}^{(0)}$ and if client inputs $\mathcal{C}^{(1)}$ are computationally indistinguishable.*

Client privacy is defined by means of indistinguishability, i.e., no information is leaked about client input. In particular, S^*cannot distinguish between $\mathcal{C}^{(0)}$ and $\mathcal{C}^{(1)}$, not even if $|\mathcal{C}^{(0)}| \neq |\mathcal{C}^{(1)}|$. In fact, Definition 4 is strictly stronger than the traditional client privacy definition for standard PSI protocols that reveal client set size. In this case, the indistinguishability would be relaxed by the constraint $|\mathcal{C}^{(0)}| = |\mathcal{C}^{(1)}|$.

Definition 4. (**Server Privacy.**) *For every semi-honest client C^*, let $\mathcal{D}$ be the joint distribution of the outputs of S (on input $\mathcal{S}$) and C^* (on input $\mathcal{C}$). There exists a* PPT *machine C' that plays the client role in the ideal-world protocol Ideal-SHIPSI (which is given a black-box access to $\mathcal{C}$and $\mathcal{S}$), such that $\mathcal{D}$ is computationally indistinguishable from the output of C'.*

Definition 5. (**Client Unlinkability.**) *For any* PPT *S^* that plays the server role on input $\mathcal{S}$', and for any client input sets $(\mathcal{C}^{(0)}, \mathcal{C}^{(1)})$, after S^* interacts with client on input $\mathcal{C}^{(0)}$, the views of S^*in a successive interaction if client inputs $\mathcal{C}^{(0)}$ and if client inputs $\mathcal{C}^{(1)}$ are computationally indistinguishable.*

Client unlinkability strengthens the concept of client privacy: Observing an interaction does not give the server any advantage in violating client privacy or in determining whether two interactions are run on the same client input.

Definition 6. **(Server Unlinkability.)** *For every semi-honest client C^*, let $\mathcal{D}^{(0)}, \mathcal{D}^{(1)}$ be the joint distributions of the outputs of C^* (on input $\mathcal{C}^{(0)}$ and $\mathcal{C}^{(1)}$, resp.) and S (on input $\mathcal{S}^{(0)}$ and $\mathcal{S}^{(1)}$, resp.). Given that $\mathcal{C}^{(0)} \cap \mathcal{S}^{(0)} = \mathcal{C}^{(1)} \cap \mathcal{S}^{(1)}$, the distributions $\mathcal{D}^{(0)}$ and $\mathcal{D}^{(1)}$ are computationally indistinguishable.*

While we define SHI-PSI to satisfy Definition 2, 3 and 4, we describe client and server unlinkability (Definitions 5 and 6) as optional properties.

Note that, beyond preventing one party from noticing changes in the other party's input, unlinkability minimizes the possibility of privacy leaks. Without it, if inputs of one interaction are leaked, then all linked inputs would be leaked. Moreover, note that unlinkability is needed to guarantee security when the protocol is composed over multiple runs.

Adversarial Model. All proposed schemes work in the random oracle model (ROM) and assume semi-honest (or honest-but-curious) parties. In particular, we assume that parties always faithfully follow all protocol specifications and do not misrepresent any information related to their inputs, e.g., size and content. However, during or after protocol execution, any party might (passively) attempt to infer additional information about the other party's input. This models precisely the class of adversaries considered in our applications. For instance, in our Example 1 of 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 place a significant value into any information about the content (or the size) of the Terror Watch List. If stronger privacy and security is desired, our protocols can be hardened against more active adversaries via standard zero-knowledge proofs and other tools. However, this would entail a heavy performance penalty. Moreover, we emphasize that our protocols represent the initial means of hiding client input size and much remains to be explored. We argue that this step is similar to CPA- and CCA-secure encryption: Ideally, CCA-secure encryption is desired, but one first needs to introduce and study CPA – generally more efficient and sufficient for certain applications.

We assume that the communication between parties are performed over secure (private and authentic) channels. Finally, note that we do not address active attacks by third parties, including compromise of either the client or the server, which we leave as part of future work.

4 Related Work

The problem of Private Set Intersection (either size-hiding or not) resembles that addressed by several well-known cryptographic constructs. We first review related work and then prior work on PSI.

4.1 Related primitives

Secure Multiparty Computation. Given two parties with inputs x and y respectively, the goal of Secure Multiparty Computation (SMC) is to compute a function $f(x, y)$, such that the two parties learn $f(x, y)$, and nothing else [19]. Yao [35] was the first to give a general procedure for two-party secure computation of any function expressed as a Boolean circuit. Although one could implement PSI with SMC, this would incur high computation and communication overhead, and would thus be impractical for the applications we consider.

Oblivious Transfer (OT). Oblivious Transfer [32] involves a sender with n secret messages and a receiver with one index i. The receiver wants to retrieve the i-th among the sender's messages. The sender does not learn which message has been retrieved, and the receiver does not learn any other message, obtaining so-called 1-out-n OT [28]. Next, k-out-n OT generalizes the previous mechanism with the receiver retrieving k messages rather than a single one. Although OT's privacy requirements resemble those of PSI, it is not clear how to efficiently adapt OT to implement PSI. In fact, whereas, in PSI the client and the server input items (e.g., keywords), in OT the client needs to know and input an index.

Private Information Retrieval (PIR). PIR allows a user to retrieve an item from a server in possession of a database, without revealing which item it is retrieving [6]. One trivial but inefficient PIR would require the server to transfer the entire database to the user. The challenge is to achieve a communication complexity smaller than the size of the database. Note that, in PIR, the server database is public, i.e., its privacy is not protected. Symmetric PIR (SPIR) [18] adds such a requirement on top of PIR, and thus achieves an Oblivious Transfer with reduced communication complexity. Similarly to OT, in PIR the client inputs the index of the item (in server database) it wants to retrieve, as opposed to a set item. Extension to support keywords have been presented in [5, 4]. However, as they focus on reducing communication complexity, they incur a significant increase in the computational overhead.

Branching Programs (BP). Recent work in [23] presents a generic construction to the following problem. Given a branching program P (held by a server) and an encryption c of an input x (held by a client), it is possible to compute a ciphertext c' from which $P(x)$ can be decoded using the secret key, and the size of c' depends polynomially on the size of x and the length of P. Thus, client work and the communication overhead do not depend on the size of the server input P, which remains hidden to the client. Although one can implement evaluation of set intersection with a branching program and hide the *server* set size (whereas, we aim at hiding the client set size), we argue that such a generic construction would yield a solution with an unbearable computation overhead polynomial in the size of the inputs. Also, it would require v parallel executions (where v is the size of the client set). As this work focuses on reducing communication rather than computational overhead and would hide the server rather than the client set size, we defer to the extended version of the paper detailed overhead comparison between this generic construction and our dedicated efficient SHI-PSI constructions.

Oblivious Keyword Search (OKS). Similar to OT protocols, OKS (also called *Private Keyword Search* or *Keyword OT*) entails a receiver that obliviously retrieves messages from a sender [30, 14]. However, while in OT the receiver chooses the index of the messages it wants to retrieve, in OKS it specifies keywords (the sender is assumed to associate one or more keywords to its private data contents). As highlighted in [14, 11], OKS and PSI provide similar functionalities. Hence, for ease of presentation, we only focus on PSI.

4.2 Prior Work on PSI-s

We now review several PSI constructions. We assume that the server set contains w items, while the client set – v.

Oblivious Polynomial Evaluations. The work in [15] (FNP) introduces the concept of PSI and presents solutions based on Oblivious Polynomial Evaluation (OPE) [29]. In FNP, the idea is to represent a set as a polynomial, and the elements of the set as its roots. Specifically, a client C represents elements in its private set, $\mathcal{C} = (c_1, \cdots, c_v)$, as the roots of a v-degree polynomial over a ring R, i.e., $f = \prod_{i=1}^{v}(t - c_i) = \sum_{i=0}^{k} \alpha_i t^i$. Then, assuming pk_C to be C's public key of any additively homomorphic cryptosystem (such as Paillier [31]), C encrypts the coefficients with pk_C, and sends them to server S. S's private set is denoted with $\mathcal{S} = (s_1, \cdots, s_w)$. S evaluates f at each $s_j \in \mathcal{S}$ homomorphically. Note that $f(s_j) = 0$ if and only if $s_j \in \mathcal{C} \cap \mathcal{S}$. Hence S, for each $s_j \in \mathcal{S} = (s_1, \cdots, s_w)$ returns $u_j = E(r_j f(s_j) + s_j)$ to C (where r_j is chosen at random). If $s_j \in \mathcal{C} \cap \mathcal{S}$ then C learns s_j upon decrypting. If $s_j \notin \mathcal{C} \cap \mathcal{S}$ then u_j decrypts to a random value. Therefore, the number of server operations is related to the evaluation of client's encrypted polynomial, with v coefficients, on w points in $\mathcal{S}$. Using Horner's rule and balanced bucket allocation, the client needs to compute to $O(v + w)$ exponentiations, whereas, the server needs to compute $O(w \ln \ln v)$ exponentiations. This protocol is proven secure against semi-honest adversaries in the standard model, and can be extended for malicious adversaries with half-simulatability in the Random Oracle Model (ROM), with an increased cost.

Subsequently, [26] has proposed additional OPE-based protocols that apply to several set operations (e.g., union, intersection, etc.) and may involve more than two players. Protocols are secure in the standard model against semi-honest (with similar complexity to FNP) and also malicious adversaries. The latter incurs quadratic computation overhead, i.e., $O(w \cdot v)$, and involves expensive zero-knowledge proofs, whereas, a more efficient construction has been recently proposed by [9], specifically $O(wk^2 \log^2(v))$ communication and $O(wvk \log(v) + wk^2 \log^2(v))$ computation complexity – where k is the security parameter.

Oblivious PRF-s. Other constructs rely on so-called Oblivious Pseudo-Random Functions (OPRF-s). An OPRF is a two-party protocol (between a sender and a receiver) that securely computes a pseudorandom function $f_k(\cdot)$ on key k contributed by the sender and input x contributed by the receiver, such that the former learns nothing from the interaction, and the latter learns only the value $f_k(x)$. OPRF-based PSI-s work as follows: Server S holds a secret random key k. Then, for each $s_j \in \mathcal{S}$, S pre-computes $u_j = f_k(s_j)$, and publishes (or sends the client) the set $\mathcal{U} = \{u_1, \cdots, u_w\}$. Then, C and S engage in an OPRF computation of $f_k(c_i)$ for each $c_i \in \mathcal{C}$ (of size v), such that S learns nothing about $\mathcal{C}$ (except the size) and C learns $f_k(c_i)$. Finally, C learns that $c_i \in \mathcal{C} \cap \mathcal{S}$ if and only if $f_k(c_i) \in \mathcal{U}$. The idea of using OPRFs for PSI protocols is due to Hazay and Lindell [20]. Their protocol is secure in the standard model in the presence of a malicious server and a semi-honest client. It has been since improved by Jarecki and Liu [24], who proposed a protocol secure in the standard model against both malicious parties, based on the Decisional q-Diffie-Hellman Inversion assumption, in the Common Reference String (CRS) model, where a safe RSA modulus must be pre-generated by a trusted party. In the case of [24], assuming m to be the number of bits needed to represent each set item, the server performs at least $O(w)$ PRF evaluations, i.e., m-bit exponentiations, plus $O(v)$ group exponentiations, whereas, the client at least $O(v)$ m-bit exponentiations plus $O(v)$ group exponentiations. However, [12] shows that these protocols, though secure in the standard model, are quite inefficient in practice and do not scale to large sets.

A very recent result by Hazay and Nissim [21] presents an improved construction of OPE-based PSI based on [15], but without ROM. Specifically, it introduces zero-knowledge proofs that allow client to demonstrate that encrypted polynomials are correctly produced. Also, it uses a technique based on a perfectly hiding commitment scheme with an OPRF evaluation protocol to prevent the server from deviating from the protocol. The PSI protocol in [21] incurs $O(v + w(\log\log v + m))$ computational and $O(v + w \cdot m)$ communication complexity, where m is the number of bits needed to represent a set item. Note that execution of the underlying OPRF in [21] requires m oblivious transfer invocations, and hence $O(m)$ modular exponentiations, for each set item. However, such overhead can be avoided by instantiating the protocol in ROM.

PSI-s based on One-More assumptions. The very recent work in [25] leverages an idea similar to the OPRF, namely the newly-introduced *Parallel Oblivious Unpredictable Function* (POUF), $f_k(x) = (H(x)^k \bmod p)$, in the Random Oracle Model. It constructs a two-party computation of this function, with a server S contributing the key k and a client C the argument x: C picks a random exponent α and sends $y = H(x)^\alpha$ to S, that replies with $z = y^k$, so that C recovers $f_k(x) = z^{1/\alpha}$. Note that random exponents, given that the hash functions are carefully chosen, could be taken from a subgroup (e.g., they can be 160-bits long). Similarly to OPRF-based solutions, the POUF can then be used to implement PSI, under the *One-More-Gap-DH* assumption in ROM [1]. We remark that in practice this solution resembles those previously proposed by [22] and [13], although the security of these protocols was only superficially analyzed. The computational complexity of the POUF-based PSI in presence of semi-honest adversaries amounts to $O(w+v)$ exponentiations for the server and $O(v)$ for the client. Similarly, the protocol in Figure 4 in [12] presents a similar PSI secure under the One-More-RSA assumption [1], based on blind-RSA signatures rather than the POUF. It achieves the same computational complexity, but (1) the server computes RSA signatures, i.e., exponentiations with longer exponents (e.g., 1024-bit), and (2) the client workload is reduced to only multiplications if the RSA public key, e, is chosen short enough (e.g., $e = 3$). Note that server's computation of the POUF (or RSA signatures) over its items can be done off-line (and once for all interactions), since they are independent from client input. Thus, only $O(v)$ exponentiations (or RSA signatures) must be computed by the server on-line. On the other hand, it is unclear how to make these two protocols server-unlinkable, as discussed in [12].

Also, [12] (Figure 3) provides another PSI protocol secure under the One-More-Gap-DH assumption [1], that achieves full unlinkability. The server needs to perform $O(w+v)$ exponentiations, and the client – $O(v)$. Again, the exponents can be taken from subgroups (e.g., they can be 160 bits).

Finally, a simple modification of the latter protocol introduces the concept of client authorization[2], is given in [12] (Figure 2). It preserves full unlinkability and achieves the same asymptotic overhead (but using longer exponents, as they are taken from RSA groups), and security is based on the RSA assumption.

5 Efficient SHI-PSI constructions

In this section, we construct two SHI-PSI protocols using similar tools to RSA one-way accumulators [3] and the unpredictable function $f_{X,p,q}(y) = (X^{1/y}) \bmod N$ under the RSA assumption. The basic idea is the following. The client computes a global witness of its set $\mathcal{C} = \{c_1, \cdots, c_v\}$ as an RSA accumulator, i.e., $(g^{\prod_{i=1}^{v} H(c_i)}) \bmod N$, where g is a generator of QR_N and $H(\cdot)$ a full-domain hash function [2]. Then, the client securely blinds this value (with a random exponent) and sends it to the server, who gains no information about client set or its size. On the other hand, for each of its items s_j, the server computes the unpredictable function f over client input X. The server presents the hash of the unpredictable function's output as a *tag* of each item in its set. Note that this outer layer hash is crucial for the security of the protocol, which is based on the fact that, in ROM, the hash of an unpredictable function is a PRF. In addition, we point out that $H(\cdot)$ 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 modulus N as the product of safe primes to ensure that the order of X is itself a product of large and unknown primes (details are in the proof). We show that the client learns the intersection (and nothing more) since it can only match the tags of the items in the intersection. The intuition is that client computation of $g^{(\prod_{l \neq i} H(c_l))}$ leads to find a matching tag, only if $c_i \in \mathcal{S} \cap \mathcal{C}$.

We propose two constructions with different properties and computational complexity. The first provides both server and client unlinkability while incurring a higher number of exponentiations at the client side. The second trades off server unlinkability for a linear number of client exponentiations.

Before going into further details, 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] [12] defines Authorized-PSI (APSI) to extend PSI as follows: each element in the client set must be authorized (signed) by some recognized and mutually trusted authority. Nonetheless, the server learns nothing about client inputs nor authorizations.

5.1 Fully Unlinkable SHI-PSI

Figure 1 shows a SHI-PSI protocol with server and client unlinkability. Common input is extracted from the output of $RSASetup(\tau)$. The primes p' and q' are provided exclusively to the server. The client, instead, must treat the exponents as large integers. We emphasize that arithmetic in the exponent is performed in $\mathbb{Z}^*_{p'q'}$. (In particular, squaring is a permutation of QR_N, in our setting.)

Common input: $N, g, H(\cdot), F(\cdot)$

Client C on input: $\mathcal{C} = \{hc_1, \cdots, hc_v\}$ where $hc_i = H(c_i)$		**Server** S on input: $p', q', \mathcal{S} = \{hs_1, \cdots, hs_w\}$, where $hs_j = H(s_j)$
(*Off-line*)		
For $1 \leq i \leq v$, $PCH_i = \prod_{l=1, l \neq i}^{v} hc_l$		
$PCH = \prod_{i=1}^{v} hc_i,\ A = (g^{PCH}) \bmod N$		
(*On-line*)		
$R_c \leftarrow_r \{1, \ldots, N^2\},\ X = (A^{R_c}) \bmod N$	$\xrightarrow{X}$	$R_s \leftarrow_r \{0, \ldots, p'q' - 1\}$ and $Z = (g^{R_s}) \bmod N$
		For $1 \leq j \leq w$
		$K_{s:j} = (X^{R_s \cdot (1/hs_j)}) \bmod N$
		$\{t_1, \ldots, t_w\} = \pi(F(K_{s:1}), \ldots, F(K_{s:w}))$
For $1 \leq i \leq v$	$\xleftarrow{Z, \{t_1, \ldots, t_w\}}$	$T := \{t_1, \ldots, t_w\}$
$K_{c:i} = (Z^{R_c \cdot PCH_i}) \bmod N$		
$t'_i = F(K_{c:i})$		
OUTPUT: $\{t'_1, \ldots, t'_v\} \cap \{t_1, \ldots, t_w\}$		

Figure 1: Fully-Unlinkable SHI-PSI Protocol

Theorem 1. *Under the RSA assumption, the protocol in Figure 1 is a server and client unlinkable SHI-PSI, i.e., it satisfies Definitions 1-6, in the random oracle model (ROM).*

Proof. We show that the protocol satisfies *correctness, client privacy and unlinkability*, as well as *server privacy and unlinkability*, defined in Section 3. We assume that all server elements are distinct, i.e., there are no repetitions.[3] Hash functions $H(\cdot)$ and $F(\cdot)$ are modeled as random oracles.

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:

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

Consequently, $t'_i = F(K_{c:i}) = F(K_{s:j}) = t_j$, and the client learns $c_i \in \mathcal{S} \cap \mathcal{C}$.

Client Privacy and Unlinkability. Observe that the client's only message to the server, i.e., $X = g^{(PCH \cdot R_c)} \bmod N$, is (essentially) distributed as a random element in QR_N, which is a cyclic group of order $p'q'$.

Since PCH and $p'q'$ are relatively prime (with overwhelming probability), we can 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 we write $R_c = r_1 p'q' + r_2$ 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., [8]). Therefore, the element $X = A^{R_c} \bmod N$ is essentially distributed as a random quadratic residue and independent of PCH *even if factorization of N is known.*

Finally, client unlinkability simply follows from the fact that the client generates a new random value R_c per protocol instance.

Server Privacy and Unlinkability. The intuition is that a semi-honest client, C^*, does not get more (or different) information than the intersection. We consider an ideal protocol, Ideal-SHIPSI, that gets the inputs of the two parties and outputs the intersection. We show that in the real implementation

[3]However, it is possible to handle repeated values by adding a counter in the argument of the hash function $H(\cdot)$.

of the protocol, i.e., the one where C^* interacts with real-world S, the client does not learn different information than in the ideal protocol. Specifically, C^* cannot distinguish an interaction in the real implementation from one in the ideal protocol.

Recall that C^*'s input is $\mathcal{C} = \{hc_1, \cdots, hc_v\}$ and S's input is $\mathcal{S} = \{hs_1, \cdots, hs_w\}$. Let $I = \mathcal{C} \cap \mathcal{S}$, and $|I| = t$. Let Ideal-SHIPSI be the ideal protocol and let $\overline{T}$ be its output, built as follows:

$$\overline{T} = \pi \left(F(X^{R_s(1/hs_{j1})}), \cdots, F(X^{R_s(1/hs_{jt})}), r_{t+1}, \cdots, r_w \right)$$

where $(hs_{j1}, \cdots, hs_{jt}) \in I$, while values in $(r_{t+1}, \cdots, r_w)$ are chosen uniformly and independently at random from the codomain of the random oracle $F(\cdot)$. Correctness of Ideal-SHIPSI is straightforward (C^* recovers all elements in I with overwhelming probability), as well as the server privacy: as the rest of the elements in $\overline{T}$ are random, thus independent of those in $\mathcal{S} \backslash I$.

On the other hand, let T be S's output to C^*in the real implementation (presented in Figure 1):

$$T = \pi \left(F(X^{R_s(1/hs_{j1})}), \cdots, F(X^{R_s(1/hs_{jw})}) \right)$$

We claim that C^* cannot distinguish an interaction in the ideal protocol ($\overline{T}$) from one with the real-world server S (T) with a probability non-negligibly greater than $1/2$. Our proof follows the standard hybrid argument.

Let z=$w - t$, we define a series of games Game_i for $0 \leq i \leq z$. Each game is between a challenger and C^*. In Game_i, the challenger's output to C^* is the same as the interaction with real-world S except that: (1) the first i outputs of the items NOT in I are replaced with random values $r_1, \cdots, r_i$ in the codomain of $F(\cdot)$, and (2) the remaining $(w - i)$ outputs are generated as in the real game.

C^* must output either 0 or 1. We must show that C^* cannot detect whether it is playing in Game_i or in Game_{i+1} for $i \in \{0, \ldots, z - 1\}$. Formally, we fix an index i and some probabilistic polynomial-time distinguisher D and define

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

The claim is that $\epsilon(\tau)$ is negligible in τ.

The only difference between Game_i and Game_{i+1} is that in the tuple returned by the challenger to C^* in Game_i the $(i + 1)$st item not in I is $F(X^{R_s(1/hs_{jl})})$ for some $l \in \{1, \ldots, w\}$, while in Game_{i+1} that value is replaced with a random element in the codomain of $F(\cdot)$.

Since for any unpredictable function $f_k(x)$, $F(f_k(x))$ is a pseudo-random function if F is a random oracle, it is enough to show that $f_{Z,p,q}(y) = Z^{1/H(y)} \bmod N$ is indeed an unpredictable function. Given that $Z = X^{R_s}$ is uniformly distributed in QR_N and $H(\cdot)$ is modeled as random oracle, we can borrow the reduction argument from Verifiable Random Functions [27] (based, in turn, on Shamir's RSA-based unpredictable number generator [33]) and thus state that, under the RSA assumption, $\epsilon(\tau)$ is negligible in τ.

Finally, server unlinkability simply follows from the fact that the server computes a new random $Z \in QR_N$ at each protocol execution.

Remark. We note that we could simplify our reduction to RSA without relying on the arguments in [27] and [33]. Indeed, the exponents in our scheme (i.e., the outputs of $H(\cdot)$) do not have to be primes as in [27] and [33]. Intuitively, this is because the 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 is not in the intersection, then $R_c PCH/hs_l$ will very unlikely be an integer as long as hs_l is sufficiently large: random oracles are indeed *division intractable*, as shown in [17, 7] (in particular, [7] describes an algorithm for finding division collisions that is sub-exponential in τ_1, the digest size).

Formally, suppose there exists an adversary A that violates server privacy, that is, A computes $F(g^{R_s R_c PCH/hs_l})$ for some $l \in \{1, \ldots, w\}$ s.t. hs_l is not in the intersection. Then, one can build an efficient algorithm A' that is given a challenge (y, e, N) and returns $(y^{1/e} \bmod N)$. Here we assume that y is chosen uniformly at random from QR_N, thus the order of y is $p'q'$. Note also that e is a random integer that is chosen independently of y and that $\gcd(e, p'q') = 1$ with overwhelming probability. If $e = 2^t u$, for an odd integer u, then when $t \geq 1$ A' would compute a square root of y which is infeasible if the factoring assumption holds. If $t = 0$ then e is odd and A' would solve an instance of the standard RSA problem (formulated for a random exponent).

Since R_c is generated honestly, we assume that it is known to A' via standard techniques (for instance, R_c is the output of a random oracle which A' simulates[4]). First, A' sets $g = y$ and guesses the expected index l for which A computes $g^{R_s R_c Pch/hs_l}$. Then, by programming the random oracle $H(\cdot)$, A' assigns random values to the outputs of $H(\cdot)$ and computes $d = \gcd(R_s R_c PCH, hs_l)$, for some integer e with $hs_l = ed$ and $R_s R_c PCH = bd$. Since $F(\cdot)$ is a random oracle, A' sees $g^{R_s R_c PCH/hs_l} = g^{b/e}$. Given that $(g^{b/e})^e = g^b$ and $\gcd(e, b) = 1$, A' can use the extended Euclidean algorithm to compute $g^{1/e} = y^{1/e}$ (via *Shamir's trick*)[5].

5.1.1 Computational and Bandwidth Complexity

The fully-unlinkable protocol in Figure 1 incurs the following computational complexity (estimated in terms of modular exponentiations). Recall that $|\mathcal{S}| = w$, and $|\mathcal{C}| = v$. In each interaction, the server needs to compute $O(w)$ exponentiations, one for each of its items. On the other hand, operations by the client are divided into *off-line* (pre-computed once for all interactions) and *on-line* (done during each interaction). Client's *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, the client computes $K_{c:i} = Z^{Rc \cdot PCH_i} \bmod N$ for each item. As each of these operations amounts to $O(v)$ exponentiations, the *on-line* client complexity amounts to $O(v^2)$ exponentiations.

Note that, if the client knew the factorization of N, it could compute PCH and PCH_i's using multiplications modulo $\phi(N)$, thus significantly reducing complexity of each exponentiations. However, as discussed earlier, the fact that client does not know $\phi(N)$ is a crucial factor in proving server privacy.

In Section 5.1.2, we discuss a simple tree-based technique to reduce client *on-line* complexity from $O(v^2)$ to $O(v \log(v))$. This technique benefits from the following observation: For any (i, j), $K_{c:i}$ and $K_{c:j}$ only differ by one exponent, as $PCH_i = \prod_{l=1, l \neq i}^{v} hc_l$, whereas, $PCH_j = \prod_{l=1, l \neq j}^{v} hc_l$.

Finally, bandwidth complexity in each interaction amounts to $O(w)$ outputs of the hash function $F(\cdot)$ from the server to the client, whereas, the client only transmits a single value.

5.1.2 Reducing Client Complexity for Fully-Unlinkable SHIPSI

We now discuss a simple technique aimed at reducing client overhead in the protocol described in Figure 1, i.e., our server- and client-unlinkable SHI-PSI. In fact, the naïve computation of $K_{c:i}$ leads to $O(v^2)$ exponentiations (where v is client set size), while we show that it can be reduced to $O(v \log(v))$.

We define $Z' = Z^{Rc}$, and $\overline{\mathsf{i{:}j}} = Z'^{\left(\prod_{l \notin [i,j]} hc_l\right)} \bmod N$. Observe the tree in Figure 2. The leaves contain the values $K_{c:i}$, for $1 \leq i \leq v$. For instance,

$$\bar{i} = Z'^{\left(\prod_{l \neq i} hc_l\right)} \bmod N = Z^{Rc \cdot PCH_i} = K_{c:i}$$

We now count 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 $\overline{\mathsf{h{+}1{:}j}}$:

$$\begin{aligned} \overline{\mathsf{i{:}h}} &= (\overline{\mathsf{i{:}j}})^{\left(\prod_{l=h+1}^{j} hc_l\right)} (\bmod N) \\ \overline{\mathsf{h{+}1{:}j}} &= (\overline{\mathsf{i{:}j}})^{\left(\prod_{l=i}^{h} hc_l\right)} (\bmod N) \end{aligned}$$

[4]That is, $R_c = H_c(s)$ for a secret seed s and a random oracle $H_c(\cdot)$.

[5]This is similar to the reduction in [17]. However, contrary to what is stated in Theorem 5 of [17], it is not based on the strong RSA assumption but rather on the standard RSA in ROM. This is because the exponent e is generated independently of the base y, thus e is effectively provided as input to the adversary. Indeed, the signature in [17] is actually secure under the standard RSA assumption in ROM (this was confirmed via private communication [16]).

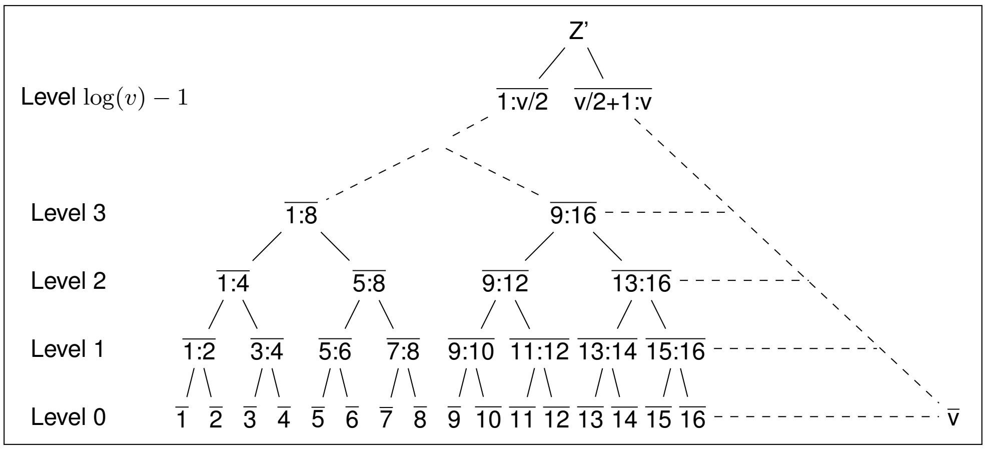


Figure 2: Tree strategy to reduce client computation overhead

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

Notice that 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 2 exponentiations from nodes at level 2. Generalizing, we observe that, at level i, there are $v/2^i$ values, each obtained with 2^i exponentiations from nodes at level i+1. Thus, the client overhead can be estimated as:

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

5.2 Client-Unlinkable SHI-PSI

We now present a modified protocol that trades server unlinkability for improved on-line computation overhead. Specifically, the number of client exponentiations is reduced to $O(v)$. The main intuition behind this modification is the following. In the protocol of Figure 1, the server contributes the value R_s—at the exponent—in the computation of its items' tags. Note that this has no effect on the security of the protocol (specifically, server privacy). This random value is needed to achieve server unlinkability if it is refreshed for each protocol execution. Removing R_s leads to a faster protocol by means of pre-computation. The resulting protocol is illustrated in Figure 3.

Common input: $N, g, H(\cdot), F(\cdot)$

Client C on input: $\mathcal{C} = \{hc_1, \cdots, hc_v\}$
where $hc_i = H(c_i)$

Server S on input: $p', q', \mathcal{S} = \{hs_1, \cdots, hs_w\}$,
where $hs_j = H(s_j)$

(*Off-line*)

For $1 \leq i \leq v$

$PCH_i = \prod_{l=1, l\neq i}^{v} hc_l$

$a_i = g^{PCH_i} \bmod N$

$PCH = \prod_{i=1}^{v} hc_i, \; A = (g^{PCH}) \bmod N$

(*On-line*)

$R_c \leftarrow_r \{1, \ldots, N^2\}, \; X = (A^{R_c}) \bmod N$ $\xrightarrow{X}$

For $1 \leq j \leq w$

$K_{s:j} = (X^{(1/hs_j)}) \bmod N$

$\{t_1, \cdots, t_w\} = \pi(F(K_{s:1}), \cdots, F(K_{s:w}))$

$T := \{t_1, \cdots, t_w\}$

$\xleftarrow{\{t_1, \cdots, t_w\}}$

For $1 \leq i \leq v$

$K_{c:i} = (a_i^{R_c}) \bmod N, \; t'_i = F(K_{c:i})$

OUTPUT: $\{t'_1, \cdots, t'_v\} \cap \{t_1, \cdots, t_w\}$

Figure 3: Client-Unlinkable SHI-PSI Protocol

Theorem 2. *Under the RSA assumption, the protocol illustrated in Figure 3 is a client-unlinkable SHI-PSI, i.e., it satisfies Definitions 1-5, in the random oracle model (ROM).*

Proof (Sketch).
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:

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

Consequently, $t'_i = F(K_{c:i}) = F(K_{s:j}) = t_j$, and the client learns $c_i \in \mathcal{S} \cap \mathcal{C}$.

Client Privacy and Unlinkability. The arguments for client privacy and unlinkability are the same of those discussed in Section 5.1 and we do not repeat them here.

Server Privacy. Similarly, the proof for the server privacy mirrors that for the fully-unlinkable protocol. The difference is in the base of the exponentiation yielding the unpredictable function: rather than X^{Rs}, it is now just X. However, as we assume the client to be semi-honest (i.e., it does not deviate arbitrarily from the protocol), the base $X = g^{PCH \cdot R_c}$ is also uniformly distributed in QR_N, for $R_c \leftarrow_r \{1, \ldots, N^2\}$. Hence, the security argument is unaltered.

We defer the complete proof and the extension to malicious clients to the extended version of the paper.

5.2.1 Computational and Bandwidth Complexity

Our client-unlinkable SHI-PSI illustrated in Figure 3 incurs the following computational complexity (estimated in terms of modular exponentiations). In each interaction, the server needs to compute $O(w)$ exponentiations, one for each of its items. Off-line, the client performs $O(v \log(v))$ exponentiations, using the technique discussed in Section 5.1.2, whereas, on-line, it performs $O(v)$ exponentiations. Bandwidth overhead is the same as in the fully-unlinkable protocol, i.e., $O(w)$ outputs of $F(\cdot)$.

5.3 SHI-PSI with Data Transfer

We now show how to extend our SHI-PSI protocols (illustrated in Figure 1 and 3) to support data transfer. Informally, in a SHI-PSI with *Data Transfer*, the client additionally gets data records associated with the items in the intersection. The main idea is to let the server encrypt records using a symmetric key (for a secure symmetric cipher, such as AES [10] used with a proper mode of operation to guarantee CPA security) derived from the output of the unpredictable function. For instance, keys can be derived computing a one-way function (e.g., a cryptographic hash function) over the unpredictable function's output. Correctness and server privacy of SHI-PSI guarantee that the 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 during setup. For every j, the server computes $k_{s:j} = F_{enc}(K_{s:j})$ and encrypts associated data using $k_{s:j}$. For its part, the 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 the protocols unaltered.

6 Extension to Group SHI-PSI

We begin this section with yet another example, that motivates a group version of SHI-PSI.

Example 4: Suppose that the U.S. Internal Revenue Service (IRS), together with its German and Italian counterparts (tax authorities), is investigating notorious international tax evaders. The countries have

Common input: $N, g, H(\cdot), F(\cdot), F'(\cdot)$

Client C_u on input: $\mathcal{C}_u = \{hc_{u:1}, \cdots, hc_{u:v}\}$ where $hc_{u:i} = H(c_{u:i})$		**Server** S on input: $p'q', \mathcal{S} = \{hs_1, \cdots, hs_w\}$, where $hs_j = H(s_j)$
(*Off-line*)		
For $1 \leq i \leq v$		
$PCH_{u:i} = \prod_{l=1, l\neq i}^{v} hc_{u:l}$		
$a_{u:i} = g^{PCH_{u:i}} \bmod N$		
$PCH_u = \prod_{i=1}^{v} hc_{u:i}$, $A_u = (g^{PCH_u}) \bmod N$		
(*On-line*)		
$Rc_u \leftarrow_r \{1, \ldots, N^2\}$, $X_u = (A^{Rc_u}) \bmod N$	$\xrightarrow{X_u}$	For $1 \leq u \leq n$, for $1 \leq j \leq w$
For $1 \leq i \leq v$		$Ks_{u:j} = (X_u^{(1/hs_j)}) \bmod N$
$Kc_{u:i} = (a_{u:i})^{Rc_u}$, $t'_{u:i} = F(Kc_{u:i})$	$\xleftarrow{\{t_1, \cdots, t_w\}}$	$t_j = F'\left[F(Ks_{1:j}), \cdots, F(Ks_{u:j})\right]$
		$T := \{t_1, \cdots, t_w\}$
Broadcast: $\{t'_{u:1}, \cdots, t'_{u:v}\}$ to all clients		
OUTPUT: $\{t''_1, \cdots, t''_v\} \cap \{t_1, \cdots, t_w\}$ where $t''_i = F'[t'_{1i_1}, t'_{2i_2}, \cdots, t'_{ni_n}]$ for all possible combinations of $(i_1, i_2, \cdots, i_n)$		

Figure 4: Group SHI-PSI Protocol executed between the server, S, and clients C_u $(1 \leq u \leq n)$.

a trilateral agreement stipulating cooperation **only** if they are investigating the same entity (person or corporation). A large Swiss bank is suspected to hold accounts of potential tax evaders. Although this bank is somehow forced to cooperate, it balks at revealing the entire list of account-holders. The three tax authorities do not trust each other (or the bank) to expose their own lists of suspects. Moreover, they do not want the bank to learn the number of suspects on their respective lists since that knowledge might be of value to other entities (e.g., the underworld or other banks).

We present a simple SHI-PSI extension that realizes multiple-client set intersection, which we refer to as Group SHI-PSI. We introduce additional notation in Table 2. Then, after discussing the functionality and the requirements for Group SHI-PSI, we outline a concrete construction.

n	number of clients in Group SHI-PSI
$u \in [1, n]$	index of the u-th client C_u in Group SHI-PSI
$\mathcal{C}_u$	C_u's set
$u{:}i \in ([1, n], [1, v])$	i-th element of $\mathcal{C}_u$
$hc_{u:i}$	$H(c_{u:i})$
$F'(\cdot)$	cryptographic hash function: $F' : \{0,1\}^* \rightarrow \{0,1\}^\tau$
Rc_u	random value generated by the client C_u

Table 2: Additional Notation used for Group SHI-PSI

In Group SHI-PSI, a server S on input $\mathcal{S}$ interacts with n clients, $C_1, \cdots C_n$, on input $\mathcal{C}_1, \cdots \mathcal{C}_n$, respectively. At the end of the interaction, each client learns

$$I = \mathcal{S} \cap \mathcal{C}_1 \cap \mathcal{C}_2 \cap \cdots \cap \mathcal{C}_n.$$

Similarly to the single-client SHI-PSI, we define Group SHI-PSI to enforce the following informal privacy requirements:

- **Server privacy:** No client learns any information about server items that are NOT in the intersection.
- **Client privacy against the server:** The server learns no information about any item of any client, or the size of the client's set.

- **Client privacy against other clients:** No client C_u learns any information about any item of any other client $C_{u'}$, except those that are in the intersection, but learns the size of the other client set.

In Figure 4, we present an efficient Group SHI-PSI construction.

Security arguments for server privacy and client privacy against the server mirror those of the single-client protocols in Section 5. Note that, even if any number of clients collude, server privacy is still guaranteed. Also, clients only exchange outputs of hash functions (modeled as a random oracle), where inputs are random values (if RC_u is taken appropriately). Thus, one client's privacy against other clients is straightforward.

The protocol in Figure 4 has the following computational complexity: The server incurs $O(nw)$ modular exponentiations, whereas, each client performs $O(v \log(v))$ off-line and $O(v)$ on-line exponentiations. Plus, in the final step, each client performs $O(v^n)$ hash computations. Optimizations aimed to reduce the number of client hash computations are deferred to future work.

Finally, bandwidth overhead amounts to $O(w)$ between the server and each client, and $O(nv)$ among the clients.

7 Performance Analysis

In this section, we analyze the complexity of proposed SHI-PSI protocols and discuss their performance in a real implementation. Our goal is three-fold: (1) evaluate overhead with respect to non size-hiding PSI protocols (instantiated in similar settings), (2) identify possible obstacles and optimizations in real deployment of our protocols, and (3) assess whether our protocols are fast enough for their intended use.

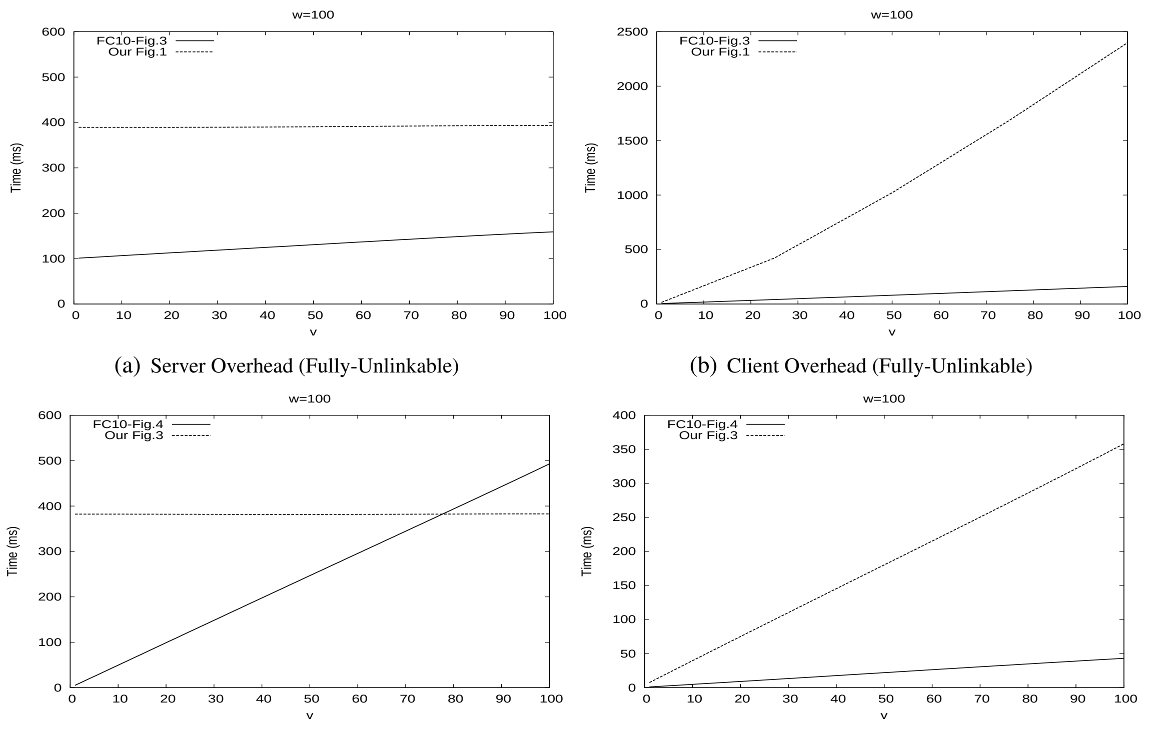
Cost of hiding size. Several years of research in the context of PSI have produced a number of protocols with different security assumptions and settings, as well as improvements in the related overhead. On the other hand, we introduce the first protocol to hide the size of the client set. Therefore, it is somehow inaccurate to compare our protocols with available PSI-s. Nonetheless, we attempt to provide an estimate of the slow-down introduced by the size hiding property. To this end, we compare the efficiency of our size-hiding protocols to the state-of-the-art PSI-s that reveal set sizes. To the best of our knowledge, the most efficient PSI-s secure in the random oracle model against semi-honest adversaries are in [12]. In particular, we choose the protocol described in Figure 3 in [12] for fully-unlinkable PSI and that in Figure 4 in [12] for client-only unlinkable PSI.

We evaluate asymptotic computation complexity in the RAM computational model, i.e., using a single-processor machine. Computation overhead is estimated as the number of *on-line* modular exponentiations performed by the server and the client. Finally, we analyze the communication overhead in terms of values exchanged from the client to the server and vice-versa. Results are reflected in Table 3.

Protocol	**Size Hiding**	**Unlinkability**	**Server**	**Client**	**Exp-s**	**Bandwidth**
Fig.3 [12]	✗	C,S	$O(w$+$v)$	$O(v)$	short	$O(w$+$2v)$
Our Fig.1	✓	C,S	$O(w)$	$O\,(v\log(v))$	long	$O(w)$
Fig.4 [12]	✗	C	$O(v)$	$O(v)$ mult	long	$O(w$+$2v)$
Our Fig.3	✓	C	$O(w)$	$O(v)$	long	$O(w)$

Table 3: Performance Comparison

First, consider fully-unlinkable PSI and SHI-PSI. The former can benefit from the use of short exponents (e.g., 160-bit) in prime order groups, whereas, the latter uses exponents with length close to the RSA modulus (e.g., 1024-bit). Then, note that our protocol achieves better server complexity in settings where v, the size of the client's set, is not negligible. In fact, *server computational load in* SHI-PSI *is independent of client input size.*



(a) Server Overhead (Fully-Unlinkable)

(b) Client Overhead (Fully-Unlinkable)

(c) Server Overhead (Client-Unlinkable)

(d) Client Overhead (Client-Unlinkable)

Figure 5: Protocols' Computation overhead (in ms) for fixed server set size (w=100) and increasing client set size

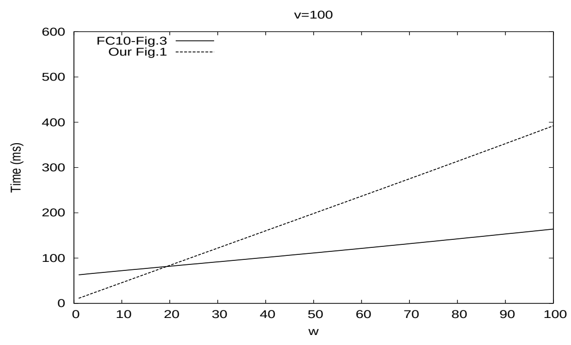
Not surprisingly, client overhead is higher, i.e., $O(v \log(v))$. However, such a drawback is experienced by one player—the client—that benefits from additional privacy, as its set size is hidden from the server.

Next, consider the client-only unlinkable protocols. Recall that in the PSI of Figure 4 in [12], server cryptographic operations performed on *its* private set items (i.e., RSA signatures) can be pre-computed once, at setup time, and are valid for all interactions. Hence, we do not include them in our evaluation. In other words, server on-line computational load in this PSI only depends on the number of client inputs. Whereas, in our SHI-PSI, the server computes a number of exponentiations linear in its set size. Next, note that the client experiences a relatively increased overhead, since multiplications of the non size-hiding solution are replaced with exponentiations.

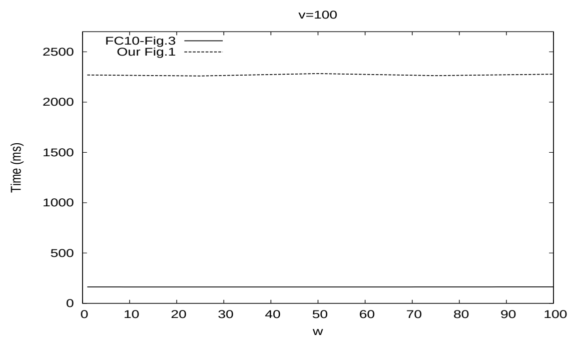
Finally, *non size-hiding protocols incur higher bandwidth overhead*, since the client sends a number of values proportional to its set size (as opposed to a single value in SHI-PSI). In [12], these values have fixed length, i.e., they are the output of a hash function (e.g., 160-bit) or group elements (e.g., 1024-bit). Thus, if the protocols are executed on high-speed networks, such an overhead is negligible. On the other hand, if they are run over the Internet, increasing values of v may incur significant slow-downs.

Additionally, we stress that fast PSI protocols such as those in [12] are secure under assumptions of the *One-More* type—i.e., One-More-RSA and One-More-DH [1]—whereas, protocols proposed in this paper are secure under the RSA assumption. Although a complete experimental evaluation of all PSI-s is beyond the scope of this paper, we remark that both SHI-PSI constructs are more efficient, in practice, than prior work on PSI-s based on standard assumptions (e.g., [15, 24]) that reveal client set size.

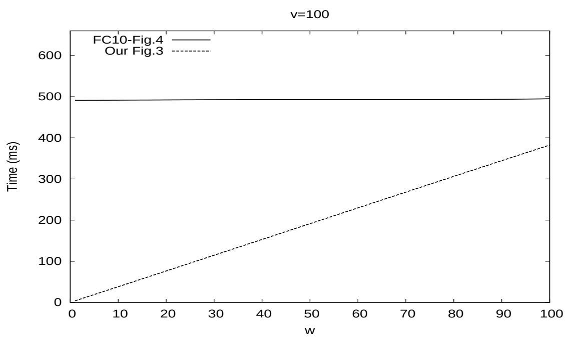
Experimental evaluation. We developed a prototype implementation of protocols in Table 3. (Its source code will be published along with the final version of the paper). We tested numerous protocol executions on a single machine: a Dell PC with two quad-core CPUs Intel Xeon at 1.60GHz with 8GB RAM. All computation was performed on a single core. Source code was developed in ANSI C, using the OpenSSL library [36]. Client and server store their sets in a text file, where each row contains a set item (e.g., an arbitrarily-long keyword). Additionally, the server stores one or more files (e.g., images, text, or binary files) associated with set items. We used the OpenSSL standard implementation of RSA



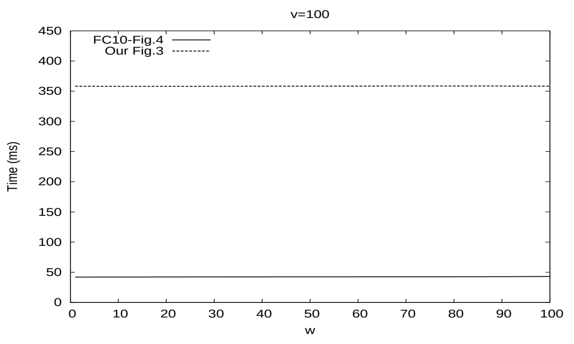
(a) Server Overhead (Fully-Unlinkable)



(b) Client Overhead (Fully-Unlinkable)



(c) Server Overhead (Client-Unlinkable)



(d) Client Overhead (Client-Unlinkable)

Figure 6: Protocols' Computation overhead (in ms) for fixed client set size (v=100) and increasing server set size

key generation for our protocols and for protocol in Figure 4 in [12] (with 1024-bit moduli), and of the DSA counterpart for DH-like parameters for protocol in Figure 3 in [12] (with 160-bit exponents and 1024-bit moduli). Random exponents R_c and R_s, described in Figure 1 and Figure 3 as taken in $\{1, \cdots, N^2\}$, were instead taken from $Z_{N/4}$ to reduce overhead in exponentiations. Hash functions described as *full-domain* were implemented according to Bellare-Rogaway solution [2]. We selected SHA-1 [34] as standard hash function. Finally, we used AES [10] (with 128-bit keys) to encrypt data associated with server items.

We focused on computation time for both the server and the client. However, we did not measure off-line operations, setup, symmetric-key en-/de-cryption operations, and transmission delay for transferring data between the server and the client.

We conducted two experiments. First, we set the number of server inputs, $w = 100$, and executed the protocols with the number of client inputs (v) ranging from 1 to 100. Results (in ms) are shown in Figure 5. Then, we fixed the number of client inputs, $v = 100$ and ran the protocols with the number of server inputs (w) ranging from 1 to 100. Computation times are reflected in Figure 6. Not surprisingly, experiments confirmed the asymptotic analysis in Table 3.

8 Conclusions and Future Work

This paper motivated the importance, and introduced the concept, of Size-Hiding Private Set Intersection (SHI-PSI). It also presented several concrete examples of SHI-PSI protocols that are both secure (in the semi-honest player model) and efficient, as confirmed by both analysis and experiments.

Since this work represents only an initial foray into SHI-PSI protocols, we expect that much remains to be done. Items for future work, include (but are not limited to):

- Exploring SHI-PSI protocols secure in the fully malicious adversary model (both client and server).
- Investigating SHI-PSI protocols that provide authorization on client input, i.e., the requirement that each item in the client set must be pre-authorized (e.g., signed) by some trusted authority.

- Considering different flavors of Group SHI-PSI protocols, e.g., a model where one client (that wants to keep secret its input size) interacts with multiple servers.

9 Acknowledgements

We are grateful to S. Jarecki and A. Shamir for some useful hints in the early stage of SHI-PSI protocol design.

References

[1] 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.

[2] M. Bellare and P. Rogaway. The exact security of digital signatures-How to sign with RSA and Rabin. In *Eurocrypt'96*, pages 399–416, 1996.

[3] J. Benaloh and M. De Mare. One-way accumulators: A decentralized alternative to digital signatures. In *Eurocrypt'93*, pages 274–285, 1994.

[4] D. Boneh, E. Kushilevitz, R. Ostrovsky, and W. Skeith. Public key encryption that allows PIR queries. In *CRYPTO'07*, pages 50–67, 2007.

[5] B. Chor, N. Gilboa, and M. Naor. Private information retrieval by keywords. *Manuscript*, 1998.

[6] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan. Private information retrieval. *Journal of the ACM*, 45(6):965–981, 1998.

[7] J. Coron and D. Naccache. Security analysis of the Gennaro-Halevi-Rabin signature scheme. In *EUROCRYPT'00*, pages 91–101. Springer, 2000.

[8] R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. *ACM Transactions on Information and System Security (TISSEC)*, 3(3):185, 2000.

[9] D. Dachman-Soled, T. Malkin, M. Raykova, and M. Yung. Efficient Robust Private Set Intersection. In *ACNS'09*, pages 125–142. Springer, 2009.

[10] J. Daeman and V. Rijmen. AES proposal: Rijndael. 1999.

[11] E. De Cristofaro and G. Tsudik. Practical Private Set Intersection Protocols with Linear Computational and Bandwidth Complexity. Cryptology ePrint Archive, `http://eprint.iacr.org/2009/491.pdf`, 2009.

[12] E. De Cristofaro and G. Tsudik. Practical Private Set Intersection with Linear Complexity. In *Financial Cryptography'10*, 2010.

[13] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *PODS'03*, pages 211–222, 2003.

[14] M. Freedman, Y. Ishai, B. Pinkas, and O. Reingold. Keyword search and oblivious pseudorandom functions. In *TCC'05*, pages 303–324, 2005.

[15] M. J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Eurocrypt'04*, pages 1–19, 2004.

[16] R. Gennaro. Private Communication, 2010.

[17] R. Gennaro, S. Halevi, and T. Rabin. Secure hash-and-sign signatures without the random oracle. In *EUROCRYPT'99*, pages 123–139. Springer, 1999.

[18] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting data privacy in private information retrieval schemes. In *STOC'98*, 1998.

[19] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *STOC'87*, pages 218–229, 1987.

[20] C. Hazay and Y. Lindell. Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. In *TCC'08*, pages 155–175, 2008.

[21] C. Hazay and K. Nissim. Efficient Set Operations in the Presence of Malicious Adversaries. *PKC'10*, pages 312–331, 2010.

[22] B. Huberman, M. Franklin, and T. Hogg. Enhancing privacy and trust in electronic communities. In *ACM Conference on Electronic Commerce*, pages 78–86, 1999.

[23] Y. Ishai and A. Paskin. Evaluating branching programs on encrypted data. *Theory of Cryptography*, pages 575–594.

[24] S. Jarecki and X. Liu. Efficient Oblivious Pseudorandom Function with Applications to Adaptive OT and Secure Computation of Set Intersection. In *TCC'09*, pages 577–594, 2009.

[25] S. Jarecki and X. Liu. Fast Secure Computation of Set Intersection. Manuscript available from the authors. To appear in SCN'10, 2010.

[26] L. Kissner and D. Song. Privacy-preserving set operations. In *CRYPTO'05*, pages 241–257, 2005.

[27] S. Micali, M. Rabin, and S. Vadhan. Verifiable random functions. In *FOCS'99*, volume 40, pages 120–130, 1999.

[28] M. Naor and B. Pinkas. Oblivious Transfer and Polynomial Evaluation. In *STOC'99*, pages 245–254, 1999.

[29] M. Naor and B. Pinkas. Oblivious polynomial evaluation. *SIAM Journal on Computing*, 1–35(5):1254, 2006.

[30] W. Ogata and K. Kurosawa. Oblivious keyword search. *Journal of Complexity*, 20(2-3):356–371, 2004.

[31] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Eurocrypt'99*, pages 223–238, 1999.

[32] M. Rabin. How to exchange secrets by oblivious transfer. Technical report, TR-81, Harvard Aiken Computation Laboratory, 1981.

[33] A. Shamir. On the generation of cryptographically strong pseudorandom sequences. *ACM Transactions on Computer Systems*, 1(1):38–44, 1983.

[34] F. I. P. Standards. Secure Hash Standard. `http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenot%ice.pdf`, 2002.

[35] A. Yao. Protocols for secure computations. In *FOCS'82*, pages 160–164, 1982.

[36] E. Young and T. Hudson. OpenSSL: The Open Source toolkit for SSL/TLS. `http://www.openssl.org`.