PRIVATE SET-INTERSECTION WITH COMMON SET-UP

SANJIT CHATTERJEE*, CHETHAN KAMATH† AND VIKAS KUMAR

Department of Computer Science and Automation
Indian Institute of Science
Bangalore 560012, India

(Communicated by Neal Koblitz)

Abstract. The problem of private set-intersection (PSI) has been traditionally treated as an instance of the more general problem of multi-party computation (MPC). Consequently, in order to argue security, or compose these protocols one has to rely on the general theory that was developed for the purpose of MPC. The pursuit of efficient protocols, however, has resulted in designs that exploit properties pertaining to PSI. In almost all practical applications where a PSI protocol is deployed, it is expected to be executed multiple times, possibly on related inputs. In this work we initiate a dedicated study of PSI in the multi-interaction (MI) setting. In this model a server sets up the common system parameters and executes set-intersection multiple times with potentially different clients. We discuss a few attacks that arise when protocols are naively composed in this manner and, accordingly, craft security definitions for the MI setting and study their inter-relation. Finally, we suggest a set of protocols that are MI-secure, at the same time almost as efficient as their parent, stand-alone, protocols.

1. Introduction

The problem of private set-intersection (PSI) involves two parties, a client and a server, each having its private set. After the execution of the protocol the client learns the intersection of the sets, whereas, the server learns nothing. Both the parties, by default, learn the cardinality of the other party’s input set. In a variant called mutual-PSI, both parties learn the intersection [10]. The security requirements for protocols which solve the PSI problem can be broadly classified into two: security in presence of an honest-but-curious (or, semi-honest) adversary and, the stronger, malicious adversary [24]. An honest-but-curious (HbC) adversary aims at learning additional information without deviating from the protocol specification, whereas a malicious party can arbitrarily deviate from the protocol specification. Security in the HbC model suffices in many scenarios such as where it is difficult to modify the software or where some form of hardware or software attestation can be performed [30].

PSI protocols find application in many privacy-preserving scenarios. Consider the following motivating example from [20]: an airlines company acts as the server and the US Department of Homeland Security (DHS) as the client. The airlines company has its passengers’ list and the DHS has a terrorist watch list. The DHS
wants to find out whether any suspected terrorists are in the passengers’ list of the airlines. The solution is to use a PSI protocol. In a similar application PSI can be used by the banks of one country and the tax authority of another whereby the tax authority finds out whether any of its suspects is a customer of the concerned bank or not in a privacy preserving manner. In yet another interesting real-world application, PSI protocol is used in real-time to secure on-line gaming platforms from cheating attacks [9]. Many on-line strategy games (e.g., Warcraft, Age of Empires) utilise “fog of war” to hide the details of a particular player from another. Bursztein et al [9] developed a tool called Kartograph which enables a cheating user to tinker with the content of his system’s memory and undo the fog of war. This technique, which they call “map hacking”, gives the cheater an unfair advantage. To counter this, they developed a tool called OpenConflict which deploys the fog of war in a distributed fashion and, then, uses PSI protocol to reveal appropriate information. Other applications include privacy-preserving data-mining [35], human genome research [3], etc. One notable feature of most applications is that they require the set-intersection to be computed multiple times between a server and one or more clients, possibly on related inputs.

1.1. Related work. Given the interesting nature of the problem, it is imperative that the area witnessed a lot of research over the last decade. The problem was introduced by Freedman et al [23] who proposed a solution based on oblivious polynomial evaluation. Subsequent works along the lines of polynomial evaluation include [34, 18, 1, 16, 26]. In a different line, Hazay and Lindell [27] used an interesting primitive called oblivious pseudo-random function (OPRFs) to propose two protocols, one secure against a relaxed version of a malicious adversary and the other against the so-called covert adversaries. Jarecki and Liu [31, 32] improved upon this work by reducing both the computational and communication complexity to a linear factor assuming common reference string (CRS). In a further development, Hazay and Nissim [29] combined oblivious polynomial evaluation and OPRF to construct malicious model secure PSI protocols.

Another set of interesting results are due to De Cristofaro et al [20, 19] and Ateniese et al [2]. Among all the works on PSI, the results of De Cristofaro and Tsudik [20] are most relevant in our context. They proposed a set of extremely efficient protocols in the HbC model in the RSA and gap Diffie-Hellman (GDH) setting with linear computation and communication complexity. In particular the RSA-based protocol given in [20, Figure 4] — henceforth referred to as the “F4” protocol — has the additional benefit that one can pre-compute, off-line, the costly operations, leading to an efficient streamline protocol suitable for low-power devices like mobiles phones, PDAs, etc. De Cristofaro and Tsudik are also the first to consider more than one executions of the PSI protocol with common set-up. Though they didn’t formalise the notion, they do consider the (un)linkability of more than one executions and provide a heuristic argument of server and client security of their protocols in this setting. However, the formal security claims are in the stand-alone setting and are based on the non-standard one-more RSA or one-more GDH assumption.

The advent of “big data” naturally led to further emphasis on efficiency, resulting in solutions utilising alternative techniques like garbled circuits [30] and Bloom filters [22]. Also, see the results in [37] using garbled circuits and oblivious transfer.

The aforementioned work of Pinkas et al [37] provided a comprehensive comparison of performance (without any pre-computation) of PSI protocols based on
different techniques. They stress that the choice of a particular PSI protocol will depend on the application scenario taking into consideration the available communication and computational resources.

1.2. **OUR MOTIVATION.** Our interest in the problem of private set intersection stems from the realistic application scenarios where the PSI protocol is executed *multiple* times between a server and possibly different clients, likely on related inputs. In particular, whether the current definitions (both that of the protocol and its security) take this fact into account. For instance, recall the tool of OpenConflict: the underlying PSI protocol is executed multiple times on related inputs and at a very high frequency. It turns out, that most definitions (especially those concerning security) have evolved considering PSI to be a special case of the more general multi-party computation (MPC) where the convention is to design a secure protocol in the stand-alone setting and then use general composition theorems.

Although the composition of general MPC protocols has been well-studied, like any other problem, there is a gap between theory and practice: by treating a specific problem as an instance of a more *general* problem, using *general* security definitions for establishing its stand-alone security and then using general composition theorems to establish that it is secure when used multiple times, we might be, very well, missing out on properties that are *specific* to the problem. Even though researchers *have* strongly considered the first aspect and, as a consequence, crafted dedicated protocols for PSI, the latter two aspects — that of security definition and composition — remain relatively unexplored.

Our aim is to explore these two aspects more thoroughly to build efficient and practical PSI protocols that are secure under multiple interactions, simultaneously, bearing in mind the definitional facets. Some initial effort in this direction can be noticed in the well-known work of De Cristofaro and Tsudik [20]. The paper talks about the unlinkability property and hence implicitly assumes multiple interactions with a common set-up. However, as already mentioned, the paper didn’t attempt at a formal definition and deferred the issue of formal proof of unlinkability for the proposed protocols as a future work. In this context it is worth noting that the authors of [20] take advantage of off-line precomputation to make the protocol F4 extremely efficient. However, full benefit of pre-computation is accrued when the cost is amortized over multiple execution of the protocol with the same set-up. Unfortunately in such a scenario server unlinkability property of protocol F4 breaks down as noted in the paper itself.

1.3. **OUR CONTRIBUTION.** Taking a cue from [20], we formally investigate the problem of constructing efficient and secure PSI in the multiple interaction environment. Exploring the problem of “server linkability” further, we show that in protocol F4 an adversarial client can, in fact, compute intersection of sets across interactions. Suppose protocol F4 is executed twice: first with server set \( S_1 \) and client set \( C_1 \) and then respectively with \( S_2 \) and \( C_2 \). While the protocol allows the client to compute only \( C_1 \cap S_1 \) and \( C_2 \cap S_2 \); we observe that the client can actually also compute \( C_1 \cap S_2 \) and \( C_2 \cap S_1 \). This is a clear violation of server privacy.

We term the above attack as the “cross-intersection” (CI) attack which subsumes server-linkability attack. Both the attacks indicate limitations of the existing security definitions which do not capture the more realistic multi-interaction (MI) setting. That brings us to the question of stronger security for PSI protocols: the notion of “MI-security”. Our primary contribution lies in identifying a meaningful
defining for PSI in the multiple interaction environment. We also show that some of the efficient PSI protocols can be easily tweaked to achieve this stronger notion of security with almost no penalty in terms of efficiency. More details follow.

Keeping the practical applications in mind, we focus on the (natural) setting where a server interacts with multiple clients. The server uses a common set-up to establish the long-term public parameter which can be re-used each time the set-intersection is computed, possibly with different clients.

Most of the works mentioned earlier regard PSI to be a particular case of general MPC protocol, while some of the attempts at explicit definitions are quite informal. Our definition of PSI customised for the MI setting consists of two algorithms: i) a Set-up algorithm used by the server to generate the long-term, common parameter; and ii) an interactive Set-Intersection algorithm used to compute the actual intersection. A more formal definition follows.

**Definition 1.1 (Private Set-Intersection Protocol).** A PSI protocol consists of a PPT algorithm $G$ and a two-party interactive algorithm $\langle Is, Ic \rangle$.

- **Set-up**, $G(1^κ)$: The algorithm is used by the server for the common set-up: the long-term public parameter $pp$ and the server’s secret $sk$ (which could be null) are generated as a result.

- **Set-Intersection**, $\langle Is(S, sk), Ic(C) \rangle(pp)$: The algorithm, which is interactive, is run jointly by the server $S$ and the client $C$, each having its private (non-empty) sets $S$ and $C$ respectively. At the end of the interaction the client learns the intersection of the sets whereas the server learns nothing; i.e. $(\bot, C \cap S) \leftarrow Is(S, sk), Ic(C)(pp)$.

Note that, by default, both parties learn the cardinality of the other party’s set. Most of the PSI protocols either fit into, or can be modified appropriately to fit Definition 1.1: e.g., the protocol F4 which we alluded to earlier is actually using the above setting. However, some protocols do not: e.g., the schemes based on homomorphic encryption [23, 18] are in the “reverse” set-up (i.e., the client generates the public parameter).

### 1.3.1. Modelling MI-security.

The stand-alone security of PSI protocols in the malicious setting is argued using the simulation paradigm and follows the security definition for two-party computation (see [24, §7.2]). In the HbC setting, one can use simpler notions that are equivalent to the simulation paradigm (e.g., indistinguishability of views, existence of simulator etc.). We refer the reader to §2.1 for the basic definitions of client and server privacy in the HbC setting (IND-CP and SIM-SP, respectively). Arguing server privacy of even some of the basic PSI protocols using the simulation-based definition becomes quite involved and lengthy (e.g., see [20]) — things are bound to get further complex in the multi-interaction setting. To simplify the whole process, we first come up with simple indistinguishability-based (IND) notions for MI-security. However, for the sake of completeness (and intuition), we give the corresponding simulation-based (SIM) definition too. The protocol definition (Definition 1.1), in conjunction with the attacks that we described earlier, lead us to natural notions of security. We start by formalising the notion of server unlinkability through IND-SU and, then, lift this notion to get our central definition: server privacy in the multi-interaction setting (n-IND-SP and (n,m)-IND-SP).

---

1The only comprehensive definition that is somewhat related to our context is in [38]. However, it applies to a variant known as policy-enhanced PSI which uses PSI as a building block.
We come up with the corresponding client-side definitions (IND-CU and n-IND-CP) as well. In addition to formalising the notions, we also study the relationship between them. There is a strict hierarchy for the server-side notions (see Figure 2): the separation between the notions follows from natural examples. The client-side notions n-IND-CP and IND-CU, on the other hand, collapse to basic client privacy (see Figure 3). Finally, on putting together everything, we have:

**Definition 1.2.** A PSI protocol is said to be **unlinkable** when it is both IND-SU and IND-CU-secure. A PSI protocol is said to be **MI-secure** when it is both n-IND-SP and 1-IND-CP-secure.

**Remark 1.** The focus of the current work is to investigate the problem of constructing efficient PSI in the realistic multi-interaction setting. In this context it’s worth recalling the works on security definition for two-party key exchange where both SIM and IND based notions were used [6, 4, 12, 13]. Building on prior works, Canetti and Krawczyk in [12] formalized the security of key exchange with multiple parties each capable of establishing multiple sessions. Our definition of PSI with MI-security is similar in spirit to the approach of [12] as both model security when multiple sessions are executed. In a subsequent work [13], Canetti and Krawczyk looked at the UC notion of *single session* key-exchange protocols. The single session protocol may have access to a long term static key (much like the common set-up in PSI) and according to [13], the notion of UC with joint state [14] can be used to argue security of the multi-session case. Some of the PSI papers [27, 29] do consider the question of composition using the UC framework [11]. However, to the best of our knowledge, UC-security of PSI protocols with a common set-up has never been addressed in the literature and could be a potential future direction of research.

**1.3.2. MI-secure PSI protocols.** Finally, we investigate a set of existing protocols that are efficient but were originally designed for the stand-alone setting. Our aim here is to achieve MI-security with as little modification as possible. The protocols we derive involve simple tweaks with performance comparable to their parent protocols. We construct three protocols namely Σ (Figure 1), Π (Figure 4) and Ψ (Figure 5). Σ is an unlinkable protocol which is derived from the protocol F4 (hence, RSA-based) and retains all the advantages of the original protocol. Π is also derived from F4, but is MI-secure. Protocol Ψ, on the other hand, is obtained from the Jarecki-Liu protocol [32] and is MI-secure. We also observe that the “Figure 3” protocol given in [20] — henceforth referred to as the F3-protocol — achieves MI-security. Another feature worth mentioning is that our security reductions in the MI-security setting are based on standard assumptions like RSA and GDH. In contrast, the security arguments of the parent protocols in the stand-alone setting were based on non-standard one-more type assumptions. As summarised in Table 1, our results establish that it is possible to achieve the stronger MI-security based on standard computational assumptions without incurring any significant additional cost. An interesting open problem is to design more efficient MI-secure PSI protocol which allows pre-computation that can be reused across multiple executions.

**1.4. Notation and conventions.** We adopt the notation commonly used in the literature. In addition, we follow some conventions to facilitate easier reading of the paper. We use C and S, respectively, to denote the client and the server set. We use the letter i and j for indexing the client and the server set respectively. Thus, if v and w denote the upper bound on the number of elements in the client and server
| Protocol | MI-secure | Client Computation (Exp.) | Server Communication (bits) | Assumption |
|----------|-----------|--------------------------|-----------------------------|------------|
| F4 [20]  | No        | $2v + w$                 | $2v |N| + w \tau$                | OM-RSA     |
| $\Sigma$ | No        | $v$                      | $2v |N| + w \tau + l$           | RSA        |
| $\Pi$    | Yes       | $v$                      | $2v |N| + w \tau + l$           | RSA        |
| $\Psi$   | Yes       | $2v + w$                 | $2v |p| + w \tau$             | GDH        |
| F3 [20]  | Yes       | $2v + 2$                 | $v + w + 1$                 | GDH        |

Table 1. Comparison of protocols; cardinality of client (resp. server) set is $v$ (resp. $w$). In protocols F4, $\Sigma$ and $\Pi$ the server takes $v + w$ exponentiations where both the exponent and modulus are of size $|N|$ bits. Since the server knows the factorization of $N$ ($p$ and $q$), by using the Chinese remainder theorem, the computation cost for the server can be reduced to $2(v + w)$ exponentiations, where both the exponent and modulus are of size $|N|/2$ bits (refer to [36, Fact 14.75] and [21]). Note that we give an improved security analysis of protocol F3 (the original reduction is based on one-more GDH assumption). See §5 for further details.

set, then the sets are:

$$C := \{c_i\}_1^v$$ and $$S := \{s_j\}_1^w$$.

The letter $\ell$ is used in the superscript to denote the round/interaction number: e.g., $\mathcal{C}^{(\ell)} := \{c_1^{(\ell)}, \ldots, c_w^{(\ell)}\}$ denotes the client set used in the $\ell$th round; $n$ is used to denote the upper bound on the number of interactions. Unless explicitly mentioned, $\mathbb{R}$ denotes a set of random server elements (from appropriate domain).

Security for MPC protocols are argued using the notion of a “view” [24]. The view of a party $P$ (in the stand-alone setting) consists of: its input, its internal coin tosses and the message it exchanges with the other parties, i.e. $(x, r, m_1, \ldots, m_t)$. For a two-party protocol, we denote it by $\text{View}_p(x, y)$ where $y$ is the other party’s input. The generalisation of the view to include multiple interactions (of a two-party protocol) is straightforward: it consists of the collection of views (its input, its internal coin tosses and the message it exchanges with the other party) in each round of interaction, i.e.

$$(x^{(1)}, r^{(1)}, m_1^{(1)}, \ldots, m_t^{(1)}), \ldots, (x^{(n)}, r^{(n)}, m_1^{(n)}, \ldots, m_t^{(n)}).$$

We use the following notation.

$$\text{View}_p^n\left(\{x^{(\ell)}, y^{(\ell)}\}_{\ell=1}^n\right) \overset{\text{def}}{=} \{\text{View}_p(x^{(\ell)}, y^{(\ell)})\}_{\ell=1}^n.$$ 

It is to be interpreted as the view of the party $P$ when it engages in $n$ interactions, where its own input in round $\ell$ of interaction is $x^{(\ell)}$ and the other party’s input is $y^{(\ell)}$. Note that the other parties involved need not be the same.

1.4.1. Organisation. The basic security definitions for PSI along with the complexity assumptions that are relevant to the paper are given in §2. We elaborate and, then, formalise the notion of server unlinkability in §3. In §4 we develop the notion of MI-security through the cross-intersection attack and also study its relationship with the other existing notions. Subsequently, in §5, we construct a few MI-secure protocols. Finally, we end with the concluding remarks in §6. Security arguments for the protocols in §5 are given in full detail in Appendix A.
2. Preliminaries

2.1. Basic definitions. A function \( \mu : \mathbb{N} \to \mathbb{R} \) is negligible if for any \( n > 0 \), we have \( |\mu(x)| < 1/k^n \) for sufficiently large \( x \). Two random variables \( X \) and \( Y \) are computationally indistinguishable, denoted by \( X \equiv Y \), if for every PPT distinguisher \( D \), the value \( |P(D(X)) - P(D(Y))| \) is negligible. Computational distinguishability is denoted by \( \not\equiv \).

2.1.1. Basic privacy: security in the stand-alone setting. An honest-but-curious (HbC) party is assumed to follow the protocol specification. However, it might try to learn information on set elements of the other participating party from the transcript of the protocol executions. We provide a formal security definition for basic client and server privacy that are derived from \([24, \text{Definition 7.2.1}]\) (also see \([2]\)): client privacy is argued using indistinguishability of views whereas server privacy using the notion of a “simulator”.

**Definition 2.1** (IND-CP: basic client privacy). For every PPT algorithm \( A_s \) that plays the role of server \( S \), for every server set \( S \), and for any client input sets \((C^{(0)}, C^{(1)})\) such that \(|C^{(0)}| = |C^{(1)}|\), two views of \( A_s \) corresponding to client \( C \)'s inputs: \( C^{(0)} \) and \( C^{(1)} \), are computationally indistinguishable. i.e.,

\[
\text{View}_S (C^{(0)}, C^{(1)}) \equiv \text{View}_S (\overline{C^{(0)}}, \overline{C^{(1)}}).
\]

**Definition 2.2** (SIM-SP: basic server privacy). Let \( \text{View}_C (C, S) \) be a random variable representing \( C \)'s view during the execution of the protocol with inputs \( C, S \). There exists a PPT algorithm \( C^* \) such that

\[
\{ C^* (C, C \cap S) \} (C, S) \equiv \{ \text{View}_C (C, S) \} (C, S).
\]

In other words, there exists a “simulator” \( C^* \) which can generate a view for the client (by taking as input just \( C \) and \( C \cap S \)) that is computationally indistinguishable from the view of the client when it executes the actual protocol with the server. Hence, \( C \) can learn only that much of information as can be inferred from the client set and the intersection.

2.2. Complexity assumptions. In this subsection, we list the computational assumptions that are relevant to the paper.

**Assumption 1** (RSA: Standard RSA assumption). Let \( N \) be the product of two large distinct primes \( p, q \). Given \( (N, e, y) \), where \( e \leftarrow \mathbb{Z}_{\varphi(N)}^* \), \( y \leftarrow \mathbb{Z}_N^* \), and \( \varphi \) the Euler’s totient function, finding \( x \) such that \( x^e \equiv y \pmod{N} \) is \((t, \varepsilon)\)-hard. That is, for every PPT algorithm \( A \) that runs in polynomial time \( t \),

\[
P(x^e \equiv y \pmod{N} : x \leftarrow A(N, e, y)) \leq \varepsilon.
\]

**Assumption 2** (GDH: Gap Diffie-Hellman assumption). Let \( \mathbb{G} \) be a group of prime order \( p \), generated by \( g \). Given \( X, Y \leftarrow \mathbb{G} \) where \( X = g^\alpha, Y = g^\beta \) for some \( \alpha, \beta \in \mathbb{Z}_p^* \), and a decision Diffie-Hellman (DDH) oracle \( \mathcal{O}_{\text{DDH}} \), finding \( g^{\alpha \beta} \) is \((t, \varepsilon)\)-hard. DDH oracle takes input as \((X = g^\alpha, Y = g^\beta, Z) \in \mathbb{G}^3 \) and outputs 1 if \( Z = g^{\alpha \beta} \), otherwise outputs 0. That is, for every PPT algorithm \( A \) that runs in polynomial time \( t \),

\[
P(g^{\alpha \beta} \leftarrow A^{\mathcal{O}_{\text{DDH}}(\cdots)}(p, \mathbb{G}, g, X = g^\alpha, Y = g^\beta)) \leq \varepsilon.
\]
In other words, the computational Diffie-Hellman problem is hard even if the decision Diffie-Hellman problem is easy.

3. Server unlinkability

Our starting point is the notion of (server/client) unlinkability that was introduced in [20]. They informally described server unlinkability and client unlinkability as optional security requirements for PSI protocols. When it comes to concrete protocols, the paper claimed that protocol F3 achieves both server and client unlinkability while protocol F4 achieves only client unlinkability. The paper also briefly mentioned why server unlinkability is violated in F4 and discussed some fixes. However, a precise formulation of the notion of unlinkability is necessary to enable any sort of formal security claim. Here we briefly recall how server unlinkability fails in protocol F4 and then proceed to formal definition of the notion. Finally, we show how F4 can be modified to obtain a server-unlinkable protocol Σ, without incurring any significant efficiency loss.

3.1. F4-protocol and server unlinkability.

The F4-protocol is based on the idea of RSA blind signature [15]. On a high level, the server acts as the signer whereas the client acts as the verifier: thus, it is the server which has to generate the RSA parameters (keeping the secret key $d$ to itself). Hence the protocol does follow Definition 1.1. The client uses the blinding property of RSA to obtain signatures on its own elements. These signatures are then input to a hash function $H_2$ to obtain the tags. The server, along with the blinded signatures, sends tags on its own elements. The client, then, compares the two set of tags to compute the intersection. The deterministic nature of the tags (which, in turn, is due to the deterministic nature of RSA signature) is crucial for the correctness to hold.² We refer the reader to Figure 7 (in Appendix A.3) for the complete description of the protocol. The basic privacy of the protocol is proved under the One-More RSA (OM-RSA) assumption (Assumption 3 in Appendix A.4).

Linking two rounds. Simply put, server unlinkability requires an adversarial client not to be able to distinguish between two server sets involved in two rounds of interaction. The authors of F4-protocol note that, as the server-tags are deterministic, their protocol does not satisfy server unlinkability. The attack is quite simple: the adversarial client has to just compare the server tags obtained during two rounds of interaction. Any change in server elements results in change in the server tags (and vice-versa). This could have devastating consequences when the protocol is used in real-life applications. For instance, suppose that protocol F4 is used as the building block in OpenConflict. Even though a cheating player is unable to completely undo the effect of fog of war, he is still able to monitor any changes in the other player’s resources (e.g., troop movements). Thus, there is a need to formalise even a basic notion such as server unlinkability.

Remark 2. Note that when the authors [20] claim (lack of) server/client unlinkability, they are implicitly assuming more than one interactions with a common set-up. The RSA parameters $(N, e, d)$ are re-used across the two rounds. Otherwise, running the protocol twice won’t allow the client to learn changes in the server set: the RSA exponent $e$ changes and so do the tags. This in turn takes away

²In fact, one can substitute the RSA scheme with any other deterministic blind signature scheme (BSS) to obtain different PSI protocols. For instance, on substituting the GDH-based BSS due to Boldyreva [8], we get HbC version of the PSI scheme from [32].
most of the amortized efficiency advantages of F4-protocol in the form of off-line precomputation.

3.2. Formalising server unlinkability. Informally, server unlinkability requires that an adversarial client should not be able to “tell if any two instances of interactions are related, i.e., executed on the same inputs by the server” [20]. Let’s assume for simplicity that the client sets for both the rounds are same (\(C\)). Also, let \(S^{(0)}\) and \(S^{(1)}\) denote the server sets involved.\(^3\) The server sets, by definition, can be trivially linked if the intersections \(S^{(0)} \cap C\) and \(S^{(1)} \cap C\) are different. Thus, under the constraint that \(S^{(0)} \cap C = S^{(1)} \cap C\) what we intuitively require is:

\[
\text{View}_{C,pp}^2((C, S^{(0)}), (C, S^{(0)})) \equiv \text{View}_{C,pp}^2((C, S^{(0)}), (C, S^{(1)})).
\]

That is, an adversarial client should not be able to figure out whether it is interacting with the same server set or not.\(^4\) Thus we arrive at the following game-based definition.

**Definition 3.1 (IND-SU).** The notion of server unlinkability for a PSI scheme is argued in terms of the following game between a challenger \(S^*\) and an adversarial client \(A_c\).

- **Set-up:** \(S^*\) runs the Set-up algorithm to obtain public parameters \(pp\) and the secret key \(sk\). It passes \(pp\) to \(A_c\).
- **Challenge:** \(A_c\) commits to a client set \(C\) and two server sets \(S^{(0)}\) and \(S^{(1)}\) such that \(C \cap S^{(0)} = C \cap S^{(1)}\). \(S^*\) tosses a coin \(b\) and runs the Set-Intersection algorithm twice to generate \(\text{View}_{C,pp}^2((S^{(0)}, C), (S^{(b)}, C))\) and sends it to \(A_c\).
- \(A_c\) returns a bit \(b'\) and wins the game if \(b' = b\).

**Remark 3 (SIM-SU).** Definition 3.1 suffices for all technical purposes. However, for sake of completeness, we describe the simulation-based definition as well. Server unlinkability is ensured if there exists a PPT algorithm \(D^*\) such that

\[
\left\{D^*(pp, C, C \cap S^{(0)})\right\}_{C,S^{(0)}} \equiv \left\{\text{View}_{C,pp}^2((C, S^{(0)}), (C, S^{(1)}))\right\}_{C,S^{(0)},S^{(1)}}.
\]

Due to the constraint \(C \cap S^{(0)} = C \cap S^{(1)}\), it follows that the LHS is computationally indistinguishable from \(\text{View}_{C,pp}^2((C, S^{(0)}), (C, S^{(0)}))\) (for all client and server sets) as well.

3.3. Protocol \(\Sigma\). In order to fix the problem of server linkability with protocol F4, [20] came up with two suggestions: \(i\) select a new set of RSA parameters for each protocol instance; or \(ii\) change the hash function \(H_1\) for the server for each protocol instance. Note that both suggestions take one away from the common set-up originally used in F4-protocol and hence, some of the associated efficiency advantages. Here we suggest a protocol \(\Sigma\) by modifying the protocol F4 retaining the common set-up and associated efficiency advantages and formally show that the protocol is client/server unlinkable. The only modification is in tag computation: in each interaction of the protocol, the server randomises the computation of the

\(^3\)As we are not considering size-hiding PSI [2], we implicitly assume that both the server sets have the same cardinality: else, it is trivial to link two interactions.

\(^4\)We have used the same client set for both the rounds for the sake of simplicity. One may as well have the stronger condition

\[
\text{View}_{C,pp}^2((C^{(0)}, S^{(0)}), (C^{(1)}, S^{(0)})) \equiv \text{View}_{C,pp}^2((C^{(0)}, S^{(0)}), (C^{(1)}, S^{(1)})).
\]
tags. Computing different tags corresponding to an element in different interactions prevents the client from relating two interactions of the protocol. The complete description is given in Figure 1.

**Correctness.** The client computes \( k_{c,i} = y_i'/r_{c,i} = y_i^d/r_{c,i} = h_{c,i}^d \). If \( c_i \in C \cap S \), then there exists \( s_j \in S \) such that \( c_i = s_j, h_{c,i} = h_{s,j} \) and \( k_{c,i} = k_{s,j} \), hence, tag \( t'_i = H_2(c_i, k_{c,i}, r_s) = H_2(s_j, k_{s,j}, r_s) = t_j \). And if \( c_i \notin C \cap S \), then \( c_i \neq s_j, h_{c,i} \neq h_{s,j} \) and \( k_{c,i} \neq k_{s,j} \) for all \( j \), hence, tag \( t'_i = H_2(c_i, k_{c,i}, r_s) \) will be different from \( t_j = H_2(s_j, k_{s,j}, r_s) \) for all \( j \), except with a negligible probability of hash collision.

**Efficiency.** Our protocol is as efficient as the original protocol as it retains the on-line/off-line property. On the contrary, both the suggestions in [20] no longer are.

3.3.1. **Security.** We show that \( \Sigma \) is server unlinkable assuming that the RSA problem is hard (Assumption 1). Recall that the security of original F4-protocol (without server unlinkability property) is based on the non-standard OM-RSA assumption.

**Theorem 3.2.** The protocol \( \Sigma \) is unlinkable under the RSA assumption with the hash functions modelled as random oracles.

---

**Table 1. Protocol \( \Sigma \)**

| Set-up, G(\(1^*\)) | The server generates the public parameters \( \mathbf{pp} := (N, e, H_1, H_2) \) and a secret key \( \mathbf{sk} := d \), where \( d \equiv e^{-1} \mod \varphi(N) \). Here, \( H_1 \) and \( H_2 \) are two hash functions such that
| \( H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_N^\times \) \( \) and \( H_2 : \{0,1\}^* \times \mathbb{Z}_N^\times \times \{0,1\}^l \rightarrow \{0,1\}^r \). |
|---|---|
| **Set-Intersection, \( \langle l_s(\mathbb{S}), l_c(\mathbb{C})\rangle(\mathbf{pp}) \):** Let \( h_{c,i} \) and \( h_{s,j} \) denote \( H_1(c_i) \) and \( H_1(s_j) \) respectively. The steps involved are given below. |
| **Server** | **Client** |
|---|---|
| Off-line | |
| \( \{k_{s,j} := h_{s,j}^d \mod N\}_{j=1}^w \) | \( \{r_{c,i} \leftarrow \mathbb{Z}_N^*\}, y_i := h_{c,i} \cdot r_{c,i}^e \mod N\}_{i=1}^v \) |
| On-line | |
| \( r_s \leftarrow \{0,1\}^l \) | \( \mathbf{Y} := \{y_1, \ldots, y_v\} \) |
| \( \{t_j := H_2(s_j, k_{s,j}, r_s)\}_{j=1}^w \) | \( \{y'_1, \ldots, y'_w\}, \mathbf{T} := \{t_1, \ldots, t_w\}, r_s \) |
| \( \{y'_i := y_t^d \mod N\}_{i=1}^v \) | \( \mathbf{Y}' := \{y'_1, \ldots, y'_v\}, \mathbf{T} := \{t_1, \ldots, t_w\}, r_s \) |
| \( \{k_{c,i} = y'_i/r_{c,i}, t'_i := H_2(c_i, k_{c,i}, r_s)\}_{i=1}^v \) | \( \{k_{c,i} = y'_i/r_{c,i}, t'_i \in \mathbf{T}\} \) |

**Figure 1. Protocol \( \Sigma \)**
The detailed proof is given in Appendix A.4. The server unlinkability is based on the RSA assumption (Lemma A.1) whereas the client privacy is maintained unconditionally (Lemma A.3).

**Remark 4.** In the security argument for server unlinkability of $\Sigma$ we have relaxed the condition in challenge by requiring the server sets to differ by a *single* element. In other words, we set

$$S^{(0)} := S \cup \{s_0^*\}$$
$$S^{(1)} := S \cup \{s_1^*\}$$

with $s_0^*, s_1^* \notin S, C$.

We get an alternative definition with this relaxation. The only difference from Definition 3.1 is in the challenge phase: the adversary directly commits to $(C, S, s_0^*, s_1^*)$ instead of $(C, S^{(0)}, S^{(1)})$. It is not difficult to prove that these two formulations are equivalent: one can use a simple hybrid argument as in the proof of Theorem 4.4.

### 4. PSI in the multiple-interaction setting

We argue that in most scenarios there are more pressing and subtle security concerns beyond unlinkability. This is motivated by demonstrating the “cross-intersection” (CI) attack on the protocols $\Sigma$ (which we showed to be IND-SU-secure) and F4. Thus there is a need to model CI-attack for PSI. We then define the notion of MI-security in both IND and SIM settings and study their relationship.

#### 4.1. Cross-intersection attack

By participating in a PSI protocol, the client is supposed to learn the intersection only for that particular interaction. In a cross-intersection (CI) attack the client, although following the protocol steps, tries to gain information about server elements other than intersection in that particular interaction by utilizing the transcript of other interactions. For example let $C^{(i)}$ and $S^{(i)}$ be the client and server sets, respectively, in the $i^{th}$ round of interaction.

In a cross-intersection attack, the client attempts to learn $C^{(i)} \cap S^{(j)}$ where $i \neq j$.

#### 4.1.1. Computing cross-intersection

**Protocol F4.** The deterministic server-tags in protocol F4 causes far more devastation than just linking two server sets. To see this consider two separate rounds of interactions between the server $S$ and an (adversarial) client $A_c$. Let $S^{(0)}, S^{(1)}$ and $C^{(0)}, C^{(1)}$, respectively, be the server and client sets, where

$$S^{(b)} = \{s^{(b)}_j\}_{j=1}^w$$
$$C^{(b)} = \{c^{(b)}_i\}_{i=1}^v$$

with $b \in \{0, 1\}$.

At the end of first round of interaction, $A_c$ uses $k^{(0)}_{c,i}$ to compute the tag $t^{(0)}_i$ corresponding to the element $c_i \in C^{(0)}$. It also receives the set of server tags $T^{(0)}$ which enables it to determine $C^{(0)} \cap S^{(0)}$. The same happens for the second round of interaction. However, as the server tags are deterministic, $A_c$ can easily find the intersection of set $C^{(i)}$ with $S^{(j)}$ by matching the tags $t^{(0)}_i$ with the set $T^{(1)}$. In a similar fashion, it can find $C^{(i)} \cap S^{(0)}$. In general, if there are $n$ rounds of interactions, the client can learn $C^{(i)} \cap S^{(j)}$, for $i, j = 1, \ldots, n$, *in addition to* $C^{(\ell)} \cap S^{(\ell)}$ for $\ell = 1, \ldots, n$.

Thus the common set-up plays the role of double-edged sword in protocol F4. Allowing pre-computation it makes F4 one of the most efficient PSI protocols while simultaneously enabling an adversarial client to mount a realistic attack if the protocol is executed multiple times.
Protocol $\Sigma$. Even though we had shown that protocol $\Sigma$ is IND-SU-secure, we observe that the attack on F4 that we discussed works on $\Sigma$ as well. Since the randomiser $r_s$ is sent in the clear, the server tags in protocol $\Sigma$ are “effectively” deterministic. Hence, an adversarial client can compute any cross intersection it wants.

Clearly cross-intersection attack is an important security feature of the PSI functionality that goes beyond server unlinkability. The attack has serious repercussions when it comes to practice. Recall the example of airlines company and DHS: every time a new entry is made in its own watch-list, the DHS can go over the past protocol transcripts to find out whether that person was in the passengers’ list in the past. In the gaming example, computing cross-intersection enables a cheating client to locate the current position of the server troops rather than just detecting troop movement (even though the platform is secured using OpenConflict). This is a definite breach of server privacy. However, the existing security definitions, including unlinkability, do not capture such attacks.

4.2. Server privacy in the multiple-interaction setting. We now formulate an appropriate security model for PSI protocol in multiple interaction setting. We give two definitions for this purpose: a SIM-based notion (denoted by $n$-SIM-SP) and an IND-based notion (denoted by $n$-IND-SP) and study the relationship between these notions and then relate them with the notion of server unlinkability in §4.3.

4.2.1. Simulation-based definition.

Definition 4.1 ($n$-SIM-SP). Let the random variable representing $C$’s view during the execution of protocol (with inputs $C^{(\ell)}$ and $S^{(\ell)}$ in the $\ell$th interaction) be defined as in (1). Then, the PSI protocol is secure if there exists a PPT algorithm $D^*$ such that

$$
\left\{ D^* \left( \text{pp}, \{ C^{(\ell)}, C^{(\ell)} \cap S^{(\ell)} \}_{\ell=1}^n \right) \right\} \equiv
\left\{ \text{View}_C^{n \text{ pp}} \left( \{ C^{(\ell)}, S^{(\ell)} \}_{\ell=1}^n \right) \right\} \equiv\left\{ \text{View}_C^{n \text{ pp}} \left( \{ C^{(\ell)}, S^{(\ell)} \}_{\ell=1}^n \right) \right\}^{\text{ View}_C^{n \text{ pp}} \left( \{ C^{(\ell)}, S^{(\ell)} \}_{\ell=1}^n \right)}
$$

We, henceforth denote the distributions by $D_\text{ss}$ (the “simulated” distribution on the LHS) and $D_v$ (the distribution corresponding to the “view” of the adversary, in the RHS) respectively.

Definition 4.1 captures the realistic scenario where the server is engaged in the PSI protocol multiple times with (possibly different) adversarial client. The definition stipulates that the protocol maintains server privacy even when all the clients collude. The trivial information that the adversary can learn in such a scenario is the union of the individual set-intersections. Therefore, $D^*$ should be able to simulate the client-view by taking as input the client sets together with the intersections and the adversary should not be able to (computationally) distinguish this from the actual protocol view. By implication, an adversarial client can learn only that much of information which can be inferred from $C^{(\ell)}$ and $C^{(\ell)} \cap S^{(\ell)}$ (for $\ell = 1$ to $n$).

4.2.2. Indistinguishability-based definition. A major hurdle in giving an IND-based definition is that we have to somehow capture the interactive Set-Intersection algorithm. In the HbC setting the adversary follows the protocol specification, so it suffices to give the protocol view to the adversary. We model this by giving the
adversary access to a “view oracle” $O_v$ through which it can adaptively obtain views corresponding to the sets of its choice.

**Definition 4.2** $(n,m)$-IND-SP Game. The server privacy for a PSI scheme is argued in terms of the following game between a challenger $S^*$ and an adversarial client $A_c$.

**Set-up:** $S^*$ runs the **Set-up** algorithm to obtain public parameters $pp$ and the secret key $sk$. It passes $pp$ to $A_c$.

**View Query, Phase 1:** $O_v(C, S)$: $A_c$ can adaptively obtain views corresponding to arbitrary client and server sets through an oracle $O_v$. $S^*$ responds by returning a view $\text{View}_C(C, S)$, which is generated by running the **Set-Intersection** algorithm, to $A_c$. Note that $S^*$ has all the inputs required to generate the view.

**Challenge:** $A_c$ sends the challenge sets $(C^*, S^*)$ along with two sets of server elements $(S_0^*, S_1^*)$, with $|S_0^*| = |S_1^*| = m$, to $S^*$ provided that $S_0^* \cap C^* = S_1^* \cap C^*$ and $S_0^* \cap S^* = S_1^* \cap S^*$. $S^*$ picks a bit $b$ uniformly at random and returns $\text{View}_C(C^*, S^* \cup S_b^*)$, to $A_c$.

**View Query, Phase 2:** $A_c$ can make further view queries as in Phase 1.

**Guess:** $A_c$ returns a bit $b'$ and wins the game if $b' = b$.

One can further simplify the server privacy definition by setting $m = 1$, i.e. the two challenge server sets differ by a single element.

**Definition 4.3** $(n)$-IND-SP Game. The server privacy for a PSI scheme is argued in terms of the following game between a challenger $S^*$ and an adversarial client $A_c$.

**Set-up:** $S^*$ runs the **Set-up** algorithm to obtain public parameters $pp$ and the secret key $sk$. It passes $pp$ to $A_c$.

**View Query, Phase 1:** $O_v(C, S)$: $A_c$ can adaptively obtain views corresponding to arbitrary client and server sets through an oracle $O_v$. $S^*$ responds by returning a view $\text{View}_C(C, S)$, which is generated by running the **Set-Intersection** algorithm, to $A_c$. Note that $S^*$ has all the inputs required to generate the view.

**Challenge:** $A_c$ sends the challenge sets $(C^*, S^*)$ along with two server elements $(s_b^*, s_1^*)$ to $S^*$ provided that $s_0^*, s_1^* \notin C^*, S^*$ (else, they must be equal). $S^*$ picks a bit $b$ uniformly at random and returns $\text{View}_C(C^*, s_b^*)$, where $S_b^* := S^* \cup \{s_b^*\}$, to $A_c$.

**View Query, Phase 2:** $A_c$ can make further view queries as in Phase 1.

**Guess:** $A_c$ returns a bit $b'$ and wins the game if $b' = b$.

Using a hybrid argument, it can be shown that $n$-IND-SP implies $(n,m)$-IND-SP.

**Theorem 4.4** $n$-IND-SP $\Rightarrow$ $(n,m)$-IND-SP.

**Proof.** For $i = 0, \ldots, m$, consider the hybrid games $(n, m, i)$-IND-SP with the challenge in the $(n,m)$-IND-SP game modified as follows:

**Challenge:** $A_c$ sends the challenge sets $(C^*, S^*)$ along with two sets of server elements $(S_{0,i}^*, S_1^*)$ to $S^*$ provided that $S_{0,i}^* \cap C^* = S_1^* \cap C^* = S^*$ (say). Let $|S_{0,i}^* \setminus S^*| = |S_1^* \setminus S^*| = m$ and the corresponding sets are $S_b^* := \{s_{b,1,i}, \ldots, s_{b,m,i}\}$ for $b = 0, 1$. $S^*$ returns $\text{View}_C(C^*, S^* \cup S_b^* \cup \{s_{0,1,i}, s_{0,1,1,i+1}, \ldots, s_{1,m}\})$ to $A_c$. 


Note that the extreme games \((n, m, 0)\)-IND-SP and \((n, m, m)\)-IND-SP correspond to the \((n, m)\)-IND-SP game with \(b = 1\) and \(b = 0\) respectively. Now we proceed via contradiction. Let \(A_m\) denote an adversary that breaks security in the \((n, m)\)-IND-SP model with some success probability \(\epsilon\). Therefore, there exist two consecutive hybrid games \((n, m, i^*)\)-IND-SP and \((n, m, i^* + 1)\)-IND-SP that \(A_m\) can distinguish with probability at least \(\epsilon/m\). But this gives rise to an adversary \(A_1\) that can break \(n\)-IND-SP security — a contradiction. \(A_1\) works as follows.

The reduction. Let \(C\) denote the \(n\)-IND-SP challenger. \(A_1\) passes the public parameters \(pp\) that it obtains from \(C\) to \(A_m\); it also relays to \(C\) all the queries that \(A_m\) makes to the view oracle and returns the views. On challenge \((C^*, S^*, (S_0^*, S_1^*))\), \(A_1\) constructs the set \(S^{**} := S^* \cup \tilde{S}^* \cup \{s_{0,1}^*, \ldots, s_{0,i^*-1}, s_{1,i^*+1}^*, \ldots, s_{1,m}^*\}\), where \(0 \leq i^* \leq m\) is picked randomly. \(A_1\) then sends \((C^*, S^{**}, (s_{0,1}^*, s_{1,i^*}^*))\) as its challenge to \(C\). It then passes the resulting challenge view corresponding to \((C^*, S^{**} \cup \{s_{b,i}^*\})\) to \(A_m\) (where \(b\) is the bit flipped by \(C\) and is unknown to \(A_1\)). Further queries to the view oracle are handled as before the challenge. Finally, \(A_m\) returns to \(C\) the bit that \(A_1\) outputs. Note that if \(b = 0\), \(A_1\) effectively plays the game \((n, m, i^*)\)-IND-SP, whereas if \(b = 1\), it plays the game \((n, m, i^* + 1)\)-IND-SP.

We use the above definition of server privacy in our security arguments which basically requires the following two distributions to be computationally indistinguishable:

\[
D_0^c := \{\text{View}_C(C^{(i)}, S^{(i)}) \mid i \in \mathbb{N}\}, \ldots, \text{View}_C(C^{(n)}, S^{(n)})\}_{(C, \sigma) \in C^*, \mathbb{S}^*_0}
\]

(4)

\[
D_1^c := \{\text{View}_C(C^{(i)}, S^{(i)}) \mid i \in \mathbb{N}\}, \ldots, \text{View}_C(C^{(n)}, S^{(n)})\}_{(C, \sigma) \in C^*, \mathbb{S}^*_1}
\]

Note that if a protocol is prone to the CI attack, then an adversary will be trivially able to win the game.\(^5\)

4.3. Relationship between the definitions. Figure 2 sums up the relationship between the security definitions for server privacy. As in the case of encryption schemes, it turns out that the SIM-based definition \((n\text{-SIM-SP})\) is equivalent to the IND-based one \((n\text{-IND-SP})\). We argue this in Theorem 4.5. There is also a strict hierarchy among the notions of basic server privacy, server unlinkability and full server privacy: \(\text{IND-SP} \not\rightarrow \text{IND-SU} \not\rightarrow \text{n-IND-SP}\). The first separation is due to server-linkability (protocol F4: \text{IND-SP}-secure but not \text{IND-SU}-secure) whereas the second one is due to the cross-intersection attack (protocol \(\Sigma\): \text{IND-SU}-secure but not \text{n-IND-SP}-secure). We now proceed to our main theorem.

**Theorem 4.5.** \(n\text{-SIM-SP} \iff n\text{-IND-SP}\).

**Proof (Overview).** As in the case of encryption schemes [25], the forward direction (Claim 4.5.2) is relatively straightforward, whereas the reverse direction (Claim 4.5.1) is a bit involved. The latter proceeds via a hybrid argument in which the distinguishability of the intermediate distributions is used to break the indistinguishability of the distributions given in (4). A detailed proof follows.\(\square\)

\(^5\)The adversary arbitrarily chooses a server element \(s_0^*\) as one of the challenge elements, and then obtains a view for sets \((C, S)\) where \(C\) is chosen such that \(s_0^* \notin C\) but not \(s_1^*\). Let’s denote it by \(\text{View}_1\). Next, it arbitrarily chooses \(C^*, S^*\) as the challenge sets, an additional element \(s_1^*\) as the second challenge element and presents \((C^*, S^*, s_0^*, s_1^*)\) as the challenge to \(S^*\). Let \(\text{View}^*\) denote the challenge view thus obtained. If the cross-intersection obtained from the views \(\text{View}_1\) and \(\text{View}^*\) contains the element \(s_0^*\) then it outputs 0 to the \(S^*\). It is easy to verify that the adversary always wins the game.
n-SIM-SP $\leftrightarrow$ n-IND-SP

SIM-SU $\leftrightarrow$ IND-SU

1-SIM-SP $\leftrightarrow$ 1-IND-SP

Figure 2. Relationship between the security definitions for server privacy. $A \rightarrow B$ implies that if a protocol is secure according to definition $A$, then it is also secure according to definition $B$. $A \nrightarrow B$ indicates a separation.

Claim 4.5.1. n-IND-SP $\Rightarrow$ n-SIM-SP.

Proof. We proceed via contradiction: if the PSI protocol is not n-SIM-SP-secure, we come up with an n-IND-SP-adversary. But first, in order to simplify the proof, we introduce a “substitute” distribution $D_r$:

$$D_r \overset{\text{def}}{=} \{ \text{View}_C(\{C^{(1)}, C^{(1)} \cap S^{(1)} \cup R^{(1)}\}_{\ell=1}) \} \{C^{(t)}, S^{(t)}\}_{t=1}$$

where $R^{(t)}$ is a set of random elements (with $|R^{(t)}| = |S^{(t)} \setminus C^{(t)}|$). Now, recall the distributions $D_v$ (the distribution corresponding to the view of the client) and $D_s$ (the distribution corresponding to the output of the simulator $D^*$) from (3). It is not difficult to see that $D_v$ is computationally indistinguishable from $D_s$; another way of looking at server privacy is that the client’s view should not change when the server elements that are not in the intersection are replaced by random elements. As a consequence, we can rephrase server privacy: if $D_r$ and $D_v$ are indistinguishable then the scheme maintains server privacy.

The hybrid. If a PSI protocol is not secure according to the n-SIM-SP definition, then the distribution $D_s$ generated by any simulator $D^*$ is distinguishable from $D_v$. This, in turn, implies $D_r$ and $D_v$ are also distinguishable. The next step is to prove that

$$D_r \not\equiv D_v \Rightarrow D_1^* \not\equiv D_0^*$$

which yields an n-IND-SP adversary. This is achieved through a hybrid argument. Consider the series of intermediate distributions

$$D_j^{(t)} \overset{\text{def}}{=} \left\{ \text{View}_C(\{C^{(1)}, C^{(1)} \cap S^{(1)} \cup R^{(1)}\}, \ldots, \text{View}_C(\{C^{(t)}, C^{(t)} \cap S^{(t)} \cup \{r_1^{(t)}, \ldots, r_j^{(t)}, s_{j+1}^{(t)}, \ldots, s_{w-t}^{(t)}\}\}, \ldots) \right\} \{C^{(t)}, S^{(t)}\}_{t=1}$$

where $\ell$ varies from 1 to $n$, $j$ from 0 to $w$ and there are $t$ elements in the intersection ($n$ and $w$ denote the bound on the number of rounds and the size of server set, respectively). Also, $s_j^{(t)}$ denotes, for the $\ell$th round, the $j$th server element which is not in the intersection. $D_j^{(t)}$ represents the view of the client when

up to round ($\ell - 1$): all the server elements that are not in the intersection; and

---

6Henceforth, by indistinguishable we mean computationally indistinguishable.
Proof. Once again, we make use of the substitute distribution $D^{(1)}_i$ and $D^{(n)}_{i-1}$ are, by definition, the same as $D_a$ and $D_a$ respectively. Also, for a particular round (same $\ell$), consecutive distributions differ only at one position: $D^{(\ell)}_{j+1}$ is obtained from $D^{(\ell)}_j$ by replacing the $(j+1)^{th}$ server element that is not in the intersection by a random element as shown below.

$$\ldots, \text{View}_C(C^{(\ell)}, C^{(\ell)} \cap S^{(\ell)} \cup \{r^{(\ell)}_1, \ldots, r^{(\ell)}_j, s^{(\ell)}_{j+1}, s^{(\ell)}_{j+2}, \ldots, s^{(\ell)}_w\}), \ldots$$

$$\ldots, \text{View}_C(C^{(\ell)}, C^{(\ell)} \cap S^{(\ell)} \cup \{r^{(\ell)}_1, \ldots, r^{(\ell)}_j, s^{(\ell)}_{j+1}, s^{(\ell)}_{j+2}, \ldots, s^{(\ell)}_w\}), \ldots$$

This implies that two consecutive intermediate distributions are same as the distributions $D^*_0$ and $D^*_1$ obtained when the challenge (of the $n$-IND-SP game) is set at $j^{th}$ position in round $\ell$. This applies to the border distributions ($D^{(\ell)}_{w-1}$ and $D^{(\ell+1)}_1$) as well. As a consequence, if any two consecutive intermediate distributions are distinguishable then so are $D^*_0$ and $D^*_1$ and we win the $n$-IND-SP game.

The distinguisher. $A_{va}$ be an algorithm which $(\varepsilon, t)$-distinguishes $D_a$ from $D_r$. We establish that the same algorithm can be used to $(\varepsilon', t')$-distinguish $D^*_0$ from $D^*_1$. It follows from our definitions that:

$$\varepsilon \leq |P(1 \leftarrow A_{va}(D_v)) - P(1 \leftarrow A_{va}(D_r))|$$

$$= |P(1 \leftarrow A_{va}(D^{(1)}_1)) - P(1 \leftarrow A_{va}(D^{(n)}_v))|$$

$$= \sum_{\ell=1}^{n} \sum_{j=1}^{w-t} |P(1 \leftarrow A_{va}(D^{(\ell)}_{j-1})) - P(1 \leftarrow A_{va}(D^{(\ell)}_j))|$$

$$= \sum_{\ell=1}^{n} \sum_{j=1}^{w-t} \varepsilon' = n \cdot (w - t) \cdot \varepsilon'$$

which implies that $A_{va}$ also distinguishes $D^*_0$ from $D^*_1$. That concludes the proof. □

Claim 4.5.2. $n$-SIM-SP $\Rightarrow$ $n$-IND-SP.

Proof. Once again, we make use of the substitute distribution $D_a$ that is given in (5). Since the protocol is $n$-SIM-SP-secure, there exists a simulator $D^{*}$ such that $D_v \equiv D^*_a$. In particular, we have: i) $D^*_0 \equiv D_a$ and ii) $D^*_1 \equiv D_a$. By transitivity, we get $D^*_0 \equiv D^*_1$ which, in turn, implies that the protocol is $n$-IND-SP-secure. □

Remark 5. Note that the reverse direction (Claim 4.5.1) is not tight: the security degrades by a factor of $n \cdot w$.

4.4. Client-side definitions. The security definition for client privacy is fundamentally different from that for the server privacy due to the inherent asymmetry in the PSI protocol (the server learns nothing other than the size of client set). This simplifies the client-side picture by quite an extent (see Figure 3). We start by modifying the notion of basic client privacy (IND-CP) (reproduced in Definition 2.1, §2) to get the notion of 1-IND-CP. The definition is inspired by the security (blindness) of blind signature schemes originally given in [33]. In order to capture the situation where the key-pair is honestly generated, it is the challenger (although it acts as the client in this particular game) who generates the key-pair. The formal definition follows.
Definition 4.6 (1-IND-CP Game). The client privacy for a PSI scheme is argued in terms of the following game between a challenger $C^*$ and an adversarial server $A_s$.

**Set-up:** $C^*$ runs the *Set-up* algorithm to obtain public parameters $pp$ and secret $sk$. It passes $(pp, sk)$ to $A_s$.

**Challenge:** $A_s$ sends the challenge sets $(C^*, S^*)$ along with two client elements $(c^*_0, c^*_1)$ to $C^*$. $C^*$ picks a bit $b$ uniformly at random and returns $\text{View}_b(C^* \cup \{c^*_b\}, S^*)$ to $A_s$. $A_s$ returns a bit $b'$ and wins the game if $b' = b$.

**Full security.** We show that the above notion is sufficient for client privacy of PSI with common set-up. In particular, due to asymmetry in the PSI definition, the client privacy in the multiple interaction setting ($n$-IND-CP, which can be defined in an analogous way) collapses to stand-alone client privacy. As client unlinkability is a special case of $n$-IND-CP, it collapses to stand-alone privacy as well.

Claim 4.6.1. $1$-IND-CP $\iff$ $n$-IND-CP.

**Proof.** Since the adversary receives the secret key at the beginning of the game, we can do away with the view oracle: the adversary has all the information necessary to generate the views by itself. Thus, client privacy in the MI setting is reduced to the stand-alone case.

5. Protocols

We construct two protocols namely $\Pi$ (Figure 4) and $\Psi$ (Figure 5). $\Pi$ is derived from protocol F4 and hence is based in the RSA setting. $\Psi$, on the other hand, is obtained from the Jarecki-Liu protocol [32] and hence is described in the cyclic group setting. We also note that the “Figure 3” protocol in [20] is secure under multiple interaction with common set-up. The performance of these protocols is comparable to their parent protocols thus demonstrating that it is possible to achieve the stronger MI-security without incurring any significant additional cost (see Table 1). In addition, in both $\Pi$ and $\Psi$ server can precompute its tags offline, leading to further efficiency gains.

5.1. **Protocol $\Pi$.** The construction of protocol $\Pi$ is analogous to that of PSS from RSA-FDH [7, 6]. Recall that in PSS, the signature is computed on a randomised message and the randomiser used is sent as a part of the signature. In protocol $\Pi$, this randomiser is chosen by the server and communicated to the client at the beginning of *Set Interaction*. Thus, there is an additional round involved. The details of the construction is given in Figure 4.
Set-up, G(1\(^n\)): Let \(H_1\) and \(H_2\) be two hash functions defined as:
\[
H_1 : \{0, 1\}^* \times \{0, 1\}^l \rightarrow \mathbb{Z}_N^* \quad \text{and} \quad H_2 : \{0, 1\}^* \times \mathbb{Z}_N^* \rightarrow \{0, 1\}^r.
\]
Let \(N\) be an RSA modulus of appropriate size and \((e, d)\) be elements of \(\mathbb{Z}_{\varphi(N)}^*\) such that \(d \equiv e^{-1} \pmod{\varphi(N)}\). Return \(\text{pp} := (N, e, H_1, H_2)\) as the public parameter and \(\text{sk} := d\) as the secret key.

\textbf{Set-Intersection,} \((\ell_s(S, d), \ell_c(C))\)(\(\text{pp}\)): The steps involved are given below.

\begin{align*}
\text{Server} & \quad \text{Client} \\
& \quad \text{Client} \\
& \quad \text{Client} \\
& \quad \text{Client} \\
& \quad \text{Client} \\
& \quad \text{Client} \\
& \quad \text{Client} \\
\end{align*}

\textbf{Correctness.} The client computes \(k_{c,i} = y'_i/r_{c,i} = y_i^d/r_{c,i} = h_{c,i}^d \pmod{N}\). If \(c_i \in C \cap S\), then there exists \(s_j \in S\) such that \(c_i = s_j, h_{c,i} = h_{s,j}\) and \(k_{c,i} = k_{s,j}\), hence, tag \(t'_j = H_2(c_i, k_{c,i}) = H_2(s_j, k_{s,j}) = t_j\). And if \(c_i \notin C \cap S\), then \(c_i \neq s_j, h_{c,i} \neq h_{s,j}\) and \(k_{c,i} \neq k_{s,j}\) for all \(j\), hence, tag \(t'_j = H_2(c_i, k_{c,i})\) will be different from \(t_j = H_2(s_j, k_{s,j})\) for all \(j\), except with negligible probability of hash collision.

\textbf{Theorem 5.1.} The protocol II is MI-secure under the RSA assumption with the hash functions modelled as random oracles.

The detailed proof is given in Appendix A.5. The server privacy is based on the standard RSA assumption (Lemma A.4) whereas the client privacy is maintained unconditionally (Lemma A.5).

\textbf{Remark 6.} We get a tight reduction for server privacy based on the standard RSA assumption. However this comes at the cost of one additional round of interaction to communicate the randomiser \(r_s\). This additional round can be traded for a less tight reduction in the argument for server privacy. We exploit the fact that
the security argument only requires that \( r_s \) should be unique (like a nonce) across different protocol runs. To avoid the interaction, though, it has to be predictable for both the sides (e.g., one could use a loosely synchronised clock, or session-ids as in key-agreement protocols). However, choosing such an \( r_s \) would cause the security argument for server privacy (Lemma A.4) to fail\(^7\). We give an alternative proof for server privacy in the modified protocol (Lemma A.6) using the partitioning technique. But it comes at a price of polynomial degradation in tightness.

5.2. Protocol \( \Psi \). The protocol \( \Psi \) is derived from [32] and hence set in cyclic groups of prime order. We prove that it is \( MI \)-secure under the assumption that GDH problem is hard in these groups.

**Description.** Let \( G \) be a group of prime order \( p \) generated by \( g \) and \( H_1 \) and \( H_2 \) be two hash functions defined as

\[
H_1 : \{0,1\}^* \rightarrow G \text{ and } H_2 : \{0,1\}^* \times G \rightarrow \{0,1\}^\tau.
\]

Server runs setup algorithm to generate the public parameters. Client and server jointly interact with their input sets \( C \) and \( S \) respectively and at the end client computes intersection of both sets i.e. \( C \cap S \). Let \( h_{c,i} \) and \( h_{s,j} \) be the output of \( H_1 \) hash function on \( c_i \) and \( s_j \) respectively. The protocol follows in Figure 5.

### Set-up, \( G(1^\kappa) \):
Server generates public parameters \( \text{pp} := G, g, p, H_1, H_2 \).

### Set-Intersection, \( \langle I_s(S, \bot), I_c(C) \rangle(\text{pp}) \):

**Steps involved are given below.**

**Server**

\[
\{ r_{c,i} \leftarrow Z_p^*, y_i = (h_{c,i})^{r_{c,i}} \}_{i=1}^v
\]

\[
Y = \{ y_1, \ldots, y_v \}
\]

\[
r_s \leftarrow Z_p^*, \{ y'_i = y_i^{r_s} \}_{i=1}^v
\]

\[
\{ k_{s,j} = (h_{s,j})^{r_s}, t_i = H_2(s_j, k_{s,j}) \}_{j=1}^w
\]

\[
Y' = \{ y'_1, \ldots, y'_v \}, T = \{ t_1, \ldots, t_w \}
\]

\[
\{ k_{c,i} = y'_i^{r_{c,i}}, t'_i = H_2(c_i, k_{c,i}) \}_{i=1}^v
\]

**Output:** \( C \cap S = \{ c_i : t'_i \in T \} \)

**Client**

**Figure 5. Protocol \( \Psi \)**

**Theorem 5.2.** The protocol \( \Psi \) is \( MI \)-secure under the GDH assumption with the hash functions modelled as random oracles.

The detailed proof is given in Appendix A.6. The server privacy is based on the GDH assumption (Lemma A.7) whereas the client privacy in maintained unconditionally (Lemma A.8).

\(^7\)The adversary may ask query to \( H_1 \) oracle on \( (s_0^\ast, r_s) \) and force the reduction \( R \) to deterministically abort with failure.
5.3. Protocol F3. We refer the reader to Figure 6 (in Appendix A.2) for the description of the protocol. We note that the protocol achieves multiple interaction security with common set-up. However, the performance of protocol $\Psi$ is slightly better than protocol F3. Security is based on the GDH assumption with the hash functions modelled as random oracle. As the proof strategy is similar to that of protocol $\Psi$, we omit the details. Note that the original security argument [20] is based on the stronger one-more gap Diffie-Hellman (OM-GDH) assumption.

6. Conclusion

In this paper, we have placed the problem of constructing efficient PSI in the realistic multi-interaction environment on a formal theoretical foundation. Our framework facilitates better designs along with easier proofs, and we demonstrated its efficacy by designing several efficient protocols that are multi-interaction secure with common set-up. An interesting open problem is to construct protocols where it is possible to take advantage of pre-computation for further efficiency gains. A similar study for the malicious model would also be an interesting future direction. So would be coming up with a more inclusive definition for PSI, say like mutual PSI.

Acknowledgments

We thank the anonymous reviewer for several helpful suggestions and Amit Kumar Mandal for a careful reading of an earlier version of the paper.

References

[1] M. Abe, G. Fuchsbauer, J. Groth, K. Haralambiev and M. Ohkubo, Structure-preserving signatures and commitments to group elements, in Advances in Cryptology - CRYPTO 2010, Springer, 2010, 209–236.
[2] G. Ateniese, E. De Cristofaro and G. Tsudik, (If) Size matters: Size-hiding private set intersection, in Public Key Cryptography - PKC 2011, Springer, 2011, 156–173.
[3] P. Baldi, R. Baronio, E. De Cristofaro, P. Gasti and G. Tsudik, Countering gattaca: Efficient and secure testing of fully-sequenced human genomes, in Proc. 18th ACM Conf. Comp. Commun. Secur. - CCS’11, ACM, New York, 2011, 691–702.
[4] M. Bellare, R. Canetti and H. Krawczyk, A modular approach to the design and analysis of authentication and key exchange protocols (extended abstract), in Proc. 30th Ann. ACM Symp. Theory Comp. - STOC’98, ACM, New York, 1998, 419–428.
[5] M. Bellare, C. Namprempre, D. Pointcheval and M. Semanko, The one-more-RSA-inversion problems and the security of Chaum’s blind signature scheme, J. Cryptology, 16 (2003), 185–215.
[6] M. Bellare and P. Rogaway, Random oracles are practical: a paradigm for designing efficient protocols, in Proc. 1st ACM Conf. Comp. Commun. Secur. - CCS ’93, ACM, New York, 1993, 62–73.
[7] M. Bellare and P. Rogaway, The exact security of digital signatures – how to sign with RSA and Rabin, in Advances in Cryptology - EUROCRYPT 1996, Springer, 1996, 399–416.
[8] A. Boldyreva, Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman group signature scheme, in Public Key Cryptography - PKC 2003, Springer, 2002, 31–46.
[9] E. Bursztein, M. Hamburg, J. Lagarennze and D. Boneh, Openconflict: Preventing real time map hacks in online games, in Proc. 32nd IEEE Symp. Secur. Privacy, IEEE Comp. Soc., Berkeley, 2011, 506–520.
[10] J. Camenisch and G. M. Zaverucha, Private intersection of certified sets, in Financial Cryptography and Data Security, Springer, 2009, 108–127.
[11] R. Canetti, Security and composition of multiparty cryptographic protocols, J. Cryptology, 13 (2000), 143–202.
[12] R. Canetti and H. Krawczyk, Analysis of key-exchange protocols and their use for building secure channels, in *Advances in Cryptology - EUROCRYPT 2001*, Springer, 2001, 453–474.

[13] R. Canetti and H. Krawczyk, Universally composable notions of key exchange and secure channels, in *Advances in Cryptology - EUROCRYPT 2002*, Springer, 2002, 337–351.

[14] R. Canetti and T. Rabin, Universal composition with joint state, in *Advances in Cryptology - CRYPTO 2003*, Springer, 2003, 265–281.

[15] D. Chaum, Blind signatures for untraceable payments, in *Advances in Cryptology - Proc. CRYPTO ’82*, Plenum Press, New York, 1982, 199–203.

[16] J.-H. Cheon, S. Jarecki and J.-H. Seo, Multi-party privacy-preserving set intersection with quasi-linear complexity, *IEICE Trans.*, 95-A (2012), 1366–1378.

[17] J.-S. Coron, On the exact security of full domain hash, in *Advances in Cryptology - CRYPTO 2000*, Springer, 2000, 229–235.

[18] D. Dachman-Soled, T. Malkin, M. Raykova and M. Yung, Efficient robust private set intersection, in *Int. J. Appl. Crypt.*, 2 (2012), 289–303.

[19] E. De Cristofaro, J. Kim and G. Tsudik, Linear-complexity private set intersection protocols secure in malicious model, in *Advances in Cryptology - ASIACRYPT 2010*, Springer, 2010, 213–231.

[20] E. De Cristofaro and G. Tsudik, Practical private set intersection protocols with linear complexity, in *Financial Cryptography and Data Security*, Springer, 2010, 143–159.

[21] E. De Cristofaro and G. Tsudik, Experimenting with fast private set intersection, in *Trust and Trustworthy Computing*, Springer, 2012, 55–73.

[22] C. Dong, L. Chen and Z. Wen, When private set intersection meets big data: an efficient and scalable protocol, in *Proc. 2013 ACM SIGSAC Conf. Comp. Commun. Secur. - CCS ’13*, ACM, New York, 2013, 789–800.

[23] M. Freedman, K. Nissim and B. Pinkas, Efficient private matching and set intersection, in *Advances in Cryptology - EUROCRYPT 2004*, Springer, 2004, 1–19.

[24] O. Goldreich, *The Foundations of Cryptography - Volume 2, Basic Applications*, Cambridge Univ. Press, 2004.

[25] S. Goldwasser and S. Micali, Probabilistic encryption & how to play mental poker keeping secret all partial information, in *Proc. 14th Ann. ACM Symp. Theory Comp. STOC ’82*, ACM, New York, 1982, 365–377.

[26] C. Hazay, Oblivious polynomial evaluation and secure set-intersection from algebraic PRFs, in *Proc. 12th Theory Crypt. Conf. – TCC 2015*, Springer, 90–120.

[27] C. Hazay and Y. Lindell, Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries, in *Theory of Cryptography*, Springer, 2008, 155–175.

[28] C. Hazay and Y. Lindell, Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries, *J. Cryptology*, 23 (2010), 422–456.

[29] C. Hazay and K. Nissim, Efficient set operations in the presence of malicious adversaries, in *Public Key Cryptography - PKC 2010*, Springer, 2010, 312–331.

[30] Y. Huang, D. Evans and J. Katz, Private set intersection: Are garbled circuits better than custom protocols?, in *19th Ann. Network Distrib. System Secur. Symp. 2012*, San Diego, California, 2012.

[31] S. Jarecki and X. Liu, Efficient oblivious pseudorandom function with applications to adaptive OT and secure computation of set intersection, in *Theory of Cryptography*, Springer, 2009, 577–594.

[32] S. Jarecki and X. Liu, Fast secure computation of set intersection, in *Security and Cryptography for Networks*, Springer, 2010, 418–435.

[33] A. Juels, M. Luby and R. Ostrovsky, Security of blind digital signatures, in *Advances in Cryptology - CRYPTO ’97*, Springer, 1997, 150–164.

[34] L. Kissner and D. Song, Privacy-preserving set operations, in *Advances in Cryptology - CRYPTO 2005* (ed. V. Shoup), Springer, 2005, 241–257.

[35] Y. Lindell and B. Pinkas, Privacy preserving data mining, *J. Cryptology*, 15 (2002), 177–206.

[36] A. Menezes, P. van Oorschot and S. Vanstone, *Handbook of Applied Cryptography*, CRC, 1996.

[37] B. Pinkas, T. Schneider and M. Zohner, Faster private set intersection based on ot extension, in *Proc. 23rd USENIX Secur. Symp.*, 797–812, 2014.
E. Stefanov, E. Shi and D. Song, Policy-enhanced private set intersection: Sharing information while enforcing privacy policies, in Public Key Cryptography - PKC 2012 (M. Fischlin, J. Buchmann and M. Manulis), Springer, 2012, 413–430.

Received September 2015; revised June 2016.

E-mail address: sanjit@csa.iisc.ernet.in
E-mail address: chethan0510@gmail.com
E-mail address: vikaskumar@csa.iisc.ernet.in

Appendix A. Protocols, security arguments

A.1. Tools.

A.1.1. Simulating the random oracles. Most of the security arguments that are given here involve random-oracle programming. Therefore we set some common ground-rules regarding the simulation of these random oracles, primarily for ease of understanding. The queries are handled with the aid of tables. The table contains, on a high level, ordered tuples of the form \( \langle x, h, r \rangle \) where, \( x \) is the input to the hash function, \( h \) the response to that query. The field \( r \) is used to store auxiliary information concerning the query: e.g., it could store a status bit.

We say that a random oracle query \( H(x) \) is fresh, if it is the first query involving \( x \). Note that if there exists no tuple \( \langle x_i, h_i, r_i \rangle \) in the table with \( x_i = x \) then the query is fresh. The way fresh queries are handled varies from proof to proof. However, if a particular query is not fresh then, in order to maintain consistency, the random oracle has to respond with the same value it output during the previous query. In other words, if there exists a tuple \( \langle x_i, h_i, r_i \rangle \) with \( x_i = x \), it has to respond with \( h_i \). In our security arguments, we consider only fresh queries. Each table \( \mathcal{L} \) has two associated interfaces: \( \text{Add}(\cdot, \cdot, \cdot) \) and \( \text{Retrieve}(\cdot) \). \( \text{Add}(\cdot, \cdot, \cdot) \) is used to insert a new element in the table and by \( \text{Retrieve}(x) \), we mean “retrieve the tuple \( \langle x_i, h_i, r_i \rangle \) with \( x_i = x \)” (assuming such a tuple exists).

A.1.2. Coron’s partitioning technique. In some security arguments, we use the partitioning technique that was introduced by Coron in the security proof of RSA-FDH [17]. We describe it in the context of signature — the basic idea is to divide the message space \( M \) into two disjoint sets, \( M_1 \) and \( M_2 \), depending upon the outcome of a biased coin \( \delta \). The simulator, then, programs the random oracle in such a way that it is equipped to respond to signature queries on messages from \( M_1 \); however, it fails if the adversary signature queries on any message from \( M_2 \) (say, because the problem instance is embedded only in elements of \( M_2 \)). Finally, the simulator hopes that the adversary produces a forgery on a message from \( M_2 \). The optimal size of these sets is determined on analysis.

A.2. Protocol “F3”. In Figure 6 we describe a version of protocol F3 from [20] that has been crafted according to Definition 1.1 and set in a general cyclic group. Let \( G = \langle g \rangle \) be a group of prime order \( p \). \( H_1 \) and \( H_2 \) are two hash functions defined as

\[
H_1 : \{0, 1\}^* \rightarrow G \quad \text{and} \quad H_2 : G \rightarrow \{0, 1\}^\tau
\]

where \( \tau \) depends on the security parameter \( \kappa \). Also, \( h_{c, i} \) and \( h_{s, j} \) denote \( H_1(c_i) \) and \( H_1(s_j) \) respectively.
Set-up, $G(1^\kappa)$: Server generates the public parameters $pp := G, g, p, H_1, H_2$

Set-Intersection, $(I_S(\mathbb{S}, \bot), I_C(\mathbb{C}))$: The steps involved are given below.

| Server | Client |
|------------------|------------------|
| $r_c \xleftarrow{\$} Z_p^*$, $\{r_{c,1}, \ldots, r_{c,v}\} \xleftarrow{\$} Z_p^*$ | |
| $\pi := \prod_{i=1}^{v} h_{c,i}, X := \pi \cdot g^{r_c}$ | |
| $\{\pi_i := \pi / h_{c,i}, y_i := \pi_i \cdot g^{r_{c,i}}\}_{i=1}^{v}$ | $Y = \{y_1, \ldots, y_v\}$ |
| $r_s \xleftarrow{\$} Z_p^*, Z := g^{r_s}, \{y'_i = y_i^{r_s}\}_{i=1}^{v}$ | |
| $\{k_{s,j} = (X / h_{s,j})^{r_s}\}_{j=1}^{w}$ | $T = \{t_1, \ldots, t_w\}$ |
| $(t_j = H_2(k_{s,j}))_{j=1}^{w}$ | $Z, Y' = \{y'_1, \ldots, y'_v\}$ |
| $Z, Y' = \{y'_1, \ldots, y'_v\}$, $T = \{t_1, \ldots, t_w\}$ | |
| $\{k_{c,i} = y'_i \cdot Z^{r_c} \cdot Z^{-r_{c,i}}, t'_i = H_2(k_{c,i})\}_{i=1}^{v}$ | $\{k'_{c,i} = y'_i \cdot Z^{r_c} \cdot Z^{-r_{c,i}}, t'_i = H_2(k_{c,i})\}_{i=1}^{v}$ |
| Output : $\mathbb{C} \cap \mathbb{S} = \{c_i : t'_i \in T\}$ | |

**Figure 6.** F3-protocol in a general cyclic-group setting

### A.3. Protocol “F4”

Let $N$ be the product of two large primes $p, q$ and $e$ is part of common input chosen uniformly at random from $\mathbb{Z}^\times_{\varphi(N)}$. $H_1, H_2$ are two hash functions defined as

$$H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_N^*$$

$$H_2 : \mathbb{Z}_N^* \rightarrow \{0,1\}^\tau$$

where $\tau$ depends on the security parameter $\kappa$. The complete protocol is given in Figure 7 (here we follow our Definition 1.1).

### A.4. Protocol $\Sigma$: Security Argument

**Lemma A.1.** Let $A_c$ be an adversarial client that $(\epsilon, t, q_1, q_2)$-breaks protocol $\Sigma$ in the IND-SU model. If $H_1$ and $H_2$ are modelled as random oracles, we can construct an algorithm $R$ which $(\epsilon', t')$-breaks RSA problem, where

$$\epsilon' \geq \frac{\epsilon}{\sqrt{v}} \text{ and } t' \leq t + (q_1 + q_2)t_e.$$

Here $q_1$ and $q_2$ denote the upper bound on number of queries on $H_1$-oracle and $H_2$-oracle; $v$ denotes the upper bound on size of client-set; and $t_e$ denotes time required to compute an exponentiation in $\mathbb{Z}_N^*$. \[\]

**Proof.** Let $(N, e, y)$ be the instance of RSA problem that the reduction $R$ aims to solve. $R$ sets $(N, e)$ as the public parameter and passes it to $A_c$. It allows $A_c$ access to two random oracles $H_1$ and $H_2$. The queries to $H_2$-oracle are split in two: phase 1 queries are those made before the challenge and phase 2 queries are those made after. The random oracles are maintained with the help of two tables $\mathcal{L}_1$ and $\mathcal{L}_2$. $\mathcal{L}_1$ contains two auxiliary fields $\delta \in \{0,1\}$ and $r \in \mathbb{Z}_N^*$ which are
Set-up, $G(1^n)$: The server runs to generate public parameters $\text{pp} := (N, e, H_1, H_2)$ and a secret key $d$, where $d \equiv e^{-1} \pmod{\varphi(N)}$.

Set-Intersection, $(I_1(S, d), I_2(C))(\text{pp})$:

| Server          | Client          |
|-----------------|-----------------|
| Off-line        |                 |
| $\{k_{s,j} := h_{s,j}^d \pmod{N}\}_{j=1}^w$ | $\{r_{c,i} \leftarrow Z_N^*\}_{i=1}^v$ |
| $\{t_j := H_2(k_{s,j})\}_{j=1}^w$ | $\{y_i := h_{c,i} \cdot r_{c,i}^e \pmod{N}\}_{i=1}^v$ |
| On-line         |                 |
| $\mathcal{Y} := \{y_1, \ldots, y_v\}$ | $\mathcal{Y}' := \{y_1', \ldots, y_v'\},\mathcal{T} := \{t_1, \ldots, t_w\}$ |
| $\{y_i' := y_i^d \pmod{N}\}_{v=1}^v$ | $\{k_{c,i} := y_i'/r_{c,i}, t_i' := H_2(k_{c,i})\}_{i=1}^v$ |
|                 | Output : $C \cap S = \{c_i : t_i' \in \mathcal{T}\}$ |

**Figure 7.** Protocol F4

used to implement the partitioning technique. The details are given in Figure 8. Since $A_c$ gets random tag corresponding to $s_0^*$, $b \in_R \{0, 1\}$; to be successful (with a non-negligible probability), it must have queried the random oracle $H_2$ on one of $(s_0^*, k_{*0}^*, r_0^*)$, $(s_0^*, k_{*0}^*, r_0^*)$ or $(s_1^*, k_{*1}^*, r_1^*)$, where $r_0^*$ (resp. $r_1^*$) is the randomiser used in first (resp. second) interaction in the server unlinkability game. Hence $R$ gets the solution during phase 2 of $H_2$ oracle queries.

**Analysis.** $R$ aborts the game during the challenge phase if: i) $\delta_i = 0$ for any $c_i \in C$ (abort 1); or ii) $\delta^* = 1$ for $s^*_0$ or $s^*_1$ (abort 2). The probability that $R$ does not abort with failure in the security game is, thus, $\mathbb{P}(\text{not abort}) = (1 - \rho)^v \cdot \rho^2$. The expression is maximized at $\rho = 2/(v + 2)$ where we get $\mathbb{P}(\text{not abort}) = 4/(e^2(v + 2)^2)$. $R$ is successful when it does not abort and $A_c$ wins the game. Therefore

$$\epsilon' \geq \mathbb{P}(\text{not abort}) \cdot \epsilon \approx \frac{1}{v^2} \cdot \epsilon.$$

The time taken by $R$ is $t' \leq t + (q_1 + q_2)t_e$, where $t_e$ is the time to compute an exponentiation operation. It takes an exponentiation to answer each $H_1$ oracle queries and an exponentiation to answer each $H_2$ oracle queries in phase 2 if a query is of the form $(s_0^*, k, r)$ or $(s_1^*, k, r)$.

Next, we give an alternative tight reduction based on the following one-more assumption.

**Assumption 3** $(\epsilon, v)$-OM-RSA: One-More RSA assumption [5]. Let $N$ be the product of two large distinct primes $p, q$. The adversary is given $(N, \epsilon)$ — where $\epsilon \leftarrow \mathbb{Z}_{\varphi(N)}^*$ and $\varphi$ is the Euler’s totient function — as input, and access to two
procedure Reduction $R(N,e,y)$

Initialize global variables pk := $(N, e)$

return $s^*$

end procedure

1: procedure $H_2(a, k, r)$

2: if (challenge = 1) then

3: if $(a = s^*_0) \land (r = r^*_a) \land (k^e / r^e) = y$ then

4: solution := $k / r$

5: else if $(a = s^*_1) \land (r = r^*_a) \land (k^e / r^e) = y$ then

6: solution := $k / r$

7: end if

8: return $h$

9: $t \leftarrow \{0, 1\}^r$, $L_2$-Add((a, k, r), t)

10: return $t$

11: end procedure

procedure $O_c(\mathbb{Z}^*, \mathbb{S}^*, s^0, s^1)$

$b \leftarrow \{0, 1\}$, View$^2 := \emptyset$

for all $s^* \in \{s^0, s^1\}$ do

$r^*_{e, i}, r^*_c, \ldots, r^*_c, r^*_{e, i} \in \mathbb{Z}_N$

for all $c_i \in \mathbb{C}^*$ do

$H_2(c_i, (c_i, h_{c,i}, (\delta_i, r_i))) \leftarrow \Sigma_1$-Retrieve($c_i$)

if $(\delta_i = 1)$ then $y_i := r_i \cdot r_{e,i} \triangleright (r^*_i \cdot r^*_{e,i})^d = y_i^d$

else Halt $A_c$

end if

end for

for all $s_j \in \mathbb{S}^*$ do

$H_2(s_j, (s_j, h_{s,j}, (\delta_j, r_j))) \leftarrow \Sigma_1$-Retrieve($s_j$)

if $(\delta_j = 1)$ then $k_{s,j} := r_j, t_j := H_2(s_j, k_{s,j}, r_a) \triangleright r_j = (r^*_j)^d = h^d_{s,j}$

else $t_j \leftarrow \{0, 1\}^r$

end if

end for

$H_2(s^*, (s^*, h^*_s, (\delta^*, r^*))) \leftarrow \Sigma_1$-Retrieve($s^*$)

if $(\delta^* = 1)$ then Halt $A_c$ \triangleright abort$_2$

else $t^* \leftarrow \{0, 1\}^r$ \triangleright $H_2(s^*, k^*_s, r^*) = t^*$, cannot compute $h^*_{s,b} = (h^r_{s,b})^d = y^d \cdot r_b^d$

end if

View := \{r^*_c, \{r^*_{c,i}\}^r, \{y^*\}^r, \{t_j\}^r\} \triangleright Assemble the view

View$^2$ := View$^2 \cup$ View

end for

24: challenge := 1 \triangleright Update status bit

25: return View$^2$

26: end procedure

Figure 8. Protocol $\Sigma$: reduction for server unlinkability.

oracles RSA oracle $O_{RSA,e}$ and challenge oracle $O_c$. $O_{RSA,e}$ takes input as $y \in \mathbb{Z}_N^*$ and returns $x$ such that $x^e \equiv y \pmod N$; $O_c$ returns random $y_i \in \mathbb{Z}_N^*$. After making $\mathcal{O} := \{y_1, \ldots, y_q\}$ calls to $O_c$ and $v < q$, calls to $O_{RSA}$ finding \{(x_i, y_i)\} for any $(v + 1)$ $y_i$s (obtained from $O_c$) such that $x_i^e \equiv y_i$ is, $(\varepsilon, t)$-hard, where $i \in \{1, \ldots, q\}$. That is, for every PPT algorithm $A$ that runs in polynomial time $t$, we have

$\Pr(\land_{(x,y) \in \mathcal{O}}(x^e \equiv y \pmod N \land y \in \mathcal{O}) : \mathbb{L} \xleftarrow{\$} A^{O_{RSA}, O_c}(N, e), |\mathcal{L}| = v + 1) \leq \varepsilon$.

We sometimes drop the parameter $(q, v)$ for convenience.
Lemma A.2. Let $A_c$ be an adversarial client that $(\epsilon, t, q_1, q_2)$-breaks protocol $\Sigma$ in the IND-SU model. If $H_1$ and $H_2$ are modelled as random oracles, we can construct an algorithm $R$ that $(\epsilon', t')$-breaks the OM-RSA assumption, where

$$\epsilon' \geq \epsilon \text{ and } t' \leq t + q_2 t_c.$$

Proof (Sketch). The proof, on many levels, proceeds similar to that of Lemma A.1. Recall that we had used the partitioning technique to selectively plug in the problem instance, which in turn, led to the aborts abort$_1$ and abort$_2$. The major difference from the previous proof is that the stronger OM-RSA assumption helps us to avoid these aborts, leading to a tight reduction. The idea is to plug in the problem instance for all the queries to the $H_1$-oracle, and then, in the challenge phase, use the RSA oracle to obtain signatures on the client elements. The resulting security argument is much simpler than the previous one. The details are given below.

For all the queries made to the $H_1$-oracle, $R$ responds with the challenge $z_i$ obtained from the challenge oracle $O_c$. In the challenge phase, to compute $y'_i = h_{c,i}^d = z_i^d$, it calls RSA oracle $O_{RSA,c}$ on $z_i$. If $A_c$ wins the game, it must have queried $H_1$-oracle either on $(s_0^*, k_{0,i}^*)$ or $(s_1^*, k_{1,i}^*)$, along with the corresponding $r_i$. As soon as it makes the query, $R$ gets the one more solution required to win the OM-RSA game.

Analysis. Since no abort is involved, the reduction is tight: $R$ is successful whenever $A$ wins the game with non-negligible advantage, and thus $\epsilon' \geq \epsilon$. The time taken by $R$ is $t \geq t + q_2 t_c$, where $t_c$ is the time to compute an exponentiation. It takes an exponentiation to answer each $H_2$ oracle query in phase 2 if a query is in the form $(s_0^*, k, r)$ or $(s_1^*, k, r)$.

Lemma A.3. Protocol $\Sigma$ is information-theoretically secure in the IND-CU model.

Proof (Sketch). The server gets $y_i = h_{c,i} \cdot r_{c,i}^e$ corresponding to each client element. Since $r_{c,i}$ is chosen uniformly at random, each $y_i$ is also uniformly distributed in $\mathbb{Z}_N^*$. Therefore, linking two interactions is not possible for an adversary. $\square$

A.5. Protocol II: security argument.

Lemma A.4. Let $A_c$ be an adversarial client that $(\epsilon, t, q_1, q_2)$-breaks protocol $\Pi$ in the n-IND-SP model. If $H_1$ and $H_2$ are modelled as random oracles, we can construct an algorithm $R$ which $(\epsilon', t')$-breaks the RSA problem, where

$$\epsilon' \geq \epsilon \text{ and } t' \leq t + (q_1 + q_2) t_e.$$

Here $q_1$ and $q_2$ denote the upper bound on number of queries to $H_1$ and $H_2$ oracles respectively, and $t_e$ denotes the time taken for an exponentiation in $\mathbb{Z}_N^*$.

Proof. Let $(N, y, e)$ be the instance of the RSA problem that $R$ aims to solve. $R$ sets $(N, e)$ as the public parameter and passes it to the adversary $A_c$. It allows $A_c$ access to two random oracles $H_1$ and $H_2$ along with the view oracle $O_c$. As in the proof of Lemma A.1, the queries to the $H_2$-oracle are split into two phases and the random oracles are maintained with the help of two tables $\Sigma_1$ and $\Sigma_2$. $\Sigma_1$ contains an auxiliary field $r$ which is used to store the $e$-th root of the hash-response (if known). The details of the reduction is provided in Figure 9.

At the end of the game, $A_c$ returns its guess $b'$. In the case that it is successful (with a non-negligible probability), it will have queried the random oracle $H_2$ with either $(s_0^*, k_{0,i}^*)$ or $(s_1^*, k_{1,i}^*)$. $R$ uses the queried $k_{a,b}^*$ and the corresponding $r_i^*$ to output the solution of the RSA problem in phase 2 of $H_2$-oracle queries. $\square$
procedure REDUCTION R(N,e,y)
    solution := 1, challenge := 0  \* Initialize global variables
    pk := (N,e), b' = A_0^{O_0,H_0,H_1,H_2}(pk)
    return solution
end procedure

1: procedure O_c(C,S)
2:   r_s, r_{c,1}, \ldots, r_{c,e} \leftarrow \mathbb{Z}_N^*
3:   for all c_i \in C do
4:       H_1(c_i, r_s), \langle (c_i, r_s), r_i, h_{c,i} \rangle \leftarrow \mathcal{L}_1.Retrieve(c_i, r_s)
5:       y'_i := r_i \cdot r_{c,i} \* \mathcal{L}_1.Retrieve(c_i, r_s)
6:   end for
7:   for all s_j \in S do
8:       H_1(s_j, r_s), \langle (s_j, r_s), r_j, h_{s,j} \rangle \leftarrow \mathcal{L}_1.Retrieve(s_j, r_s)
9:       k_{s,j} := r_j, t_j := H_2(s_j, k_{s,j})  \* Generate server-tags
10:  end for
11:  return \{r_s, \{r_{c,1}\}_1, \{y'_i\}_1, \{t_j\}_1\}
end procedure

1: procedure H_2(a, k)
2:   if (challenge = 1) then
3:       if (a = s_0^k) \&\& (k^e/(r_0^e)^e = y) then
4:           solution := k/r_0^e
5:       else if (a = s_1^k) \&\& (k^e/(r_1^e)^e = y) then
6:           solution := k/r_1^e
7:   end if
8:  end if
9:  t \leftarrow \{0,1\}^r, \mathcal{L}_2.Add((a, k), t)
10: return t
end procedure

1: procedure O_c(C*, S*, s_0^k, s_1^k)
2:   b \leftarrow \{0,1\}, r_s^*, r_{c,1}^*, \ldots, r_{c,e}^* \leftarrow \mathbb{Z}_N^*
3:   for all c_i \in C do
4:       H_1(c_i, r_s^*), \langle (c_i, r_s^*), r_i, h_{c,i} \rangle \leftarrow \mathcal{L}_1.Retrieve(c_i, r_s^*)
5:       y'_i := r_i \cdot r_{c,i}^* \mathcal{L}_1.Retrieve(c_i, r_s^*)
6:   end for
7:   for all s_j \in S do
8:       H_1(s_j, r_s^*), \langle (s_j, r_s^*), r_j, h_{s,j} \rangle \leftarrow \mathcal{L}_1.Retrieve(s_j, r_s^*)
9:       k_{s,j} := r_j, t_j := H_2(s_j, k_{s,j})  \* Generate server-tags
10:  end for
11:  return \{r_s^*, \{r_{c,1}\}_1, \{y'_i\}_1, \{t_j\}_1\}
end procedure

Figure 9. Protocol II: security argument for server privacy

Analysis. R is successful whenever \(A_0\) wins the game with non-negligible advantage, thus \(\epsilon' = \epsilon\). The time taken by R is \(t' < t + (q_1 + q_2)t_e\), where \(t_e\) is the time to compute an exponentiation operation. It takes an exponentiation to answer each \(H_1\) oracle queries and an exponentiation to answer each \(H_2\) oracle queries in phase 2 if a query is in the form \((s_0^k, k)\) or \((s_1^k, k)\).

Lemma A.5. Protocol II is information-theoretically secure in IND-CP model.
Proof. Challenger \( C^* \) runs Set-up algorithm to generate public parameter \( pp = (N, e, H_1, H_2) \), and a secret key \( sk = d \), and gives it to \( A_s \).

**Challenge Phase:** \( A_s \) gives \((C^*, S^*)\) and \((c^*_0, c^*_1)\) to \( C^* \). \( C^* \) chooses \( b \overset{\$}{\leftarrow} \{0, 1\} \) and returns

\[
\text{View}_2(C^* \cup c^*_b, S^*) = \{ r_s, Y = \{ y_i \}_{1}^{t_1} \cup \{ y_b^* \} : y_i = h_{c,i} \cdot r_{c,i}^e ; r_s, r_{c,i} \overset{\$}{\leftarrow} \mathbb{Z}_N^* \}
\]

which is computed as follows:
1. Choose \( r_s, r_{c,1}, \ldots, r_{c,v} \overset{\$}{\leftarrow} \mathbb{Z}_N^* \).
2. For each \( c_i \in C^* \), compute \( h_{c,i} = H_1(c_i, r_s) \), \( y_i = h_{c,i} \cdot r_{c,i}^e \).
3. For \( c_b^* \) choose \( r_{c,b}^* \overset{\$}{\leftarrow} \mathbb{Z}_N^* \) and compute \( h_{c_b}^* = H_1(c_b^*, r_s) \), \( y_b^* = h_{c_b}^* \cdot r_{c,b}^e \).

At the end of the game \( A_s \) returns its guess \( b' \). Now we claim that it cannot win the game with a non-negligible advantage as \( y_b^* = y_b^* \) for some \( r_{c,b}^* \)

\[
y_b^* = h_{c,b}^* \cdot (r_{c,b}^*)^e = h_{c,b}^* \cdot (h_{c,b}^*)^{-1} \cdot h_{c,b}^* \cdot (r_{c,b}^*)^e = h_{c,b}^* \cdot (r_{c,b}^*)^e
\]

where \( r_{c,b}^* = (h_{c,b}^*)^{-1} \cdot (s_{c,b}^*)^d \cdot (s_{c,b}^*)^e \). Hence \( A_s \) cannot win with non-negligible advantage.

We now analyse server privacy of the modified protocol mentioned in Remark 6.

**Lemma A.6.** Let \( A_v \) be an \((\epsilon, t, q_1, q_2)\)-adversary against protocol II in n-IND-SP model. If \( H_1 \) and \( H_2 \) are modelled as random oracle, we can construct an algorithm \( R \) which \((\epsilon', t')\)-breaks RSA problem, where

\[
\epsilon' \geq \frac{1}{v^2 q_v^2} \cdot \epsilon \quad \text{and} \quad t' \leq t + (q_1 + q_2) t_e
\]

where \( q_1 \) and \( q_2 \) denote the upper bound on number of queries on \( H_1 \)-oracle and \( H_2 \)-oracle, \( v \) denotes upper bound of client’s set size, \( q_v \) denotes upper bound on number of view queries and \( t_e \) denotes time to compute an exponentiation.

**Proof.** The proof goes almost similar to the proof of Lemma A.4. Here \( H_1 \)-oracle queries are modelled using the Coron’s partitioning technique. On each \( H_1 \)-oracle query, \( R \) tosses a \( \rho \) biased coin to get \( \delta \) (\( \mathbb{P}(\delta = 0) = \rho \)). Based on the value of \( \delta \), query is responded. If \( \delta = 0 \), it plugs-in the problem instance as \( h = y \cdot r^e \), otherwise sets \( h = r^e \). \( H_2 \)-oracle queries of Phase 1 and Phase 2 are handled as in Lemma A.4.

In the view query phase and challenge phase, for any \( c_i \in C \), if corresponding tuple in \( \mathcal{L}_1 \) has \( \delta = 0 \), then it aborts with failure. Also in the challenge phase, if tuple corresponding to either \( s_{c_b}^* \) or \( s_{c_b}^* \) has \( \delta = 1 \), then it aborts with failure. These aborts with failure result in polynomial degradation in security reduction.

**Analysis.** \( R \) aborts with failure in view query phase and challenge phase if for any \( c_i \in C \) the corresponding tuple in \( \mathcal{L}_1 \) has \( \delta = 0 \). It also aborts with failure in challenge phase in the case when tuple corresponding to either \( s_{c_b}^* \) or \( s_{c_b}^* \) has \( \delta = 1 \) in \( \mathcal{L}_1 \). The probability that \( R \) does not abort with failure in the security game is

\[
\mathbb{P}(\neg \text{abort}) = (1 - \rho)^{v(q_v + 1)} \cdot \rho^2.
\]

This is maximized when \( \rho = 2/(v(q_v + 1) + 2) \). Putting the value of \( \rho \), we get

\[
\mathbb{P}(\neg \text{abort}) = \frac{1}{e^{2(v(q_v + 1) + 2)^2}}.
\]
So the advantage of R in solving the RSA challenge is

$$\epsilon' \geq \mathbb{P}(\text{abortion}) \cdot \epsilon \approx \frac{1}{v^2 q_v^2} \cdot \epsilon.$$ 

The time taken by R is

$$t' \leq t + (q_1 + q_2) t_e,$$

where $t_e$ is the time to compute an exponentiation operation. It takes an exponentiation to answer each $H_1$ oracle queries and an exponentiation to answer each $H_2$ oracle queries in phase 2 if a query is in the form $(s_0^*, k)$ or $(s_1^*, k)$.

### A.6. Protocol $\Psi$: Security Argument

**Lemma A.7.** Let $A_c$ be an adversarial client that $(\epsilon, t, q_1, q_2)$-breaks protocol $\Psi$ in the n-IND-SP model. If $H_1$ and $H_2$ are modelled as random oracles, we can construct an algorithm $R$ which $(\epsilon', t')$-breaks gap Diffie-Hellman (GDH) problem, where

$$\epsilon' \geq \frac{\epsilon}{v} \quad \text{and} \quad t' \leq t + (q_1 + q_2 + (q_v + 1)(v + w)) t_e.$$ 

Here $q_1$ and $q_2$ denote the upper bound on number of queries on $H_1$-oracle and $H_2$-oracle; $v$ and $w$ denote the upper bound on size of client and server input set, respectively; $t_e$ is the time to compute an exponentiation in $G$.

**Proof.** Let $(G, g, p, A = g^a, B = g^b, O_{\text{DDH}})$ be the instance of the GDH problem that $R$ aims to solve. $R$ gives public parameter $pp = (G, g, p)$ to $A_c$. It allows $A_c$ access to two random oracles $H_1$ and $H_2$ along with the view oracle $O_v$. The queries to $H_2$-oracle are split in two: The details of the reduction is given in Figure 10.

At the end of the game the $A_c$ returns $b'$ and wins the game if $b' = b$. Note that corresponding to $s_1^*$, $A_c$ gets only a tag chosen randomly which is set as output of $H_2(s_0^*(h_{s,b}^*)^\alpha)$. Hence to win the game $A_c$ must have to make query either $H_2(s_0^*(h_{s,0}^*)^\alpha)$ or $H_2(s_1^*(h_{s,1}^*)^\alpha)$. As $A_c$ makes this query $R$ gets the solution of GDH challenge from $H_2$-oracle queries in phase 2.

**Analysis.** $R$ aborts with failure in the above game if i) for any $c_i \in C^*$ the corresponding tuple $(c_i, r_i, h_i, \delta_i)$ in $G$ has $\delta_i = 0$ or ii) tuple corresponding to either $s_0^*$ has $\delta_0 = 1$ or $s_1^*$ has $\delta_1 = 1$. Probability that $R$ does not abort in the game is

$$\mathbb{P}(\text{abortion}) = (1 - \rho)^v \cdot \rho^2.$$ 

This is maximized when $\rho = 2/(v + 2)$. Putting the value of $\rho$, we get

$$\mathbb{P}(\text{abortion}) = \frac{4}{e^2(v + 2)^2}.$$ 

$R$ is successful when $A$ wins the game with non-negligible advantage and it does not abort with failure in the game, which gives the following bound

$$\epsilon' \geq \frac{4}{e^2(v + 2)^2} \cdot \epsilon \approx \frac{1}{v^2} \cdot \epsilon.$$ 

The time taken by $R$ is

$$t' \leq t + (q_1 + q_2 + (q_v + 1)(v + w)) t_e,$$

where $t_e$ is the time to compute an exponentiation operation. It takes an exponentiation to answer each $H_1$ oracle query and an exponentiation to answer each $H_2$ oracle query in phase 2 if a query is in the form $(s_0^*, b)$ or $(s_1^*, b)$. Further, $(v + w)$ exponentiations to answer each view query and challenge phase takes $(v + w)$ exponentiations.

**Lemma A.8.** The protocol $\Psi$ is information-theoretically secure in IND-CP model.

**Proof.** Challenger $C_1$ runs Set-up algorithm to generate public parameter $pp = (G, g, p, H_1, H_2)$ and gives it to $A_i$. 

---

**Private set-intersection with common set-up**

45
procedure Reduction $R^{O_{DDH}}(G, g, p, A, B)$  
  solution := 1, challenge := 0  
  $pk := (G, g, p), B \leftarrow A^O_{C \cdot O_{c \cdot H_1 \cdot H_2}}(pk)$  
  return solution  
end procedure  

1: procedure $O_e(C, S)$  
2: $r_s, r_{c,1}, \ldots, r_{c,v} \leftarrow Z_p^*$  
3: for all $c_i \in C$  
4: $H_1(c_i), (c_i, h_{c,i}, (\delta_i, r_i)) \leftarrow \Sigma_1.\text{Retrieve}(c_i), y'_i := (h_{c,i})^{r_c, r_s}$  
5: end for  
6: for all $s_j \in S$ do  
7: $H_1(s_j), (s_j, h_{s,j}, (\delta_j, r_j)) \leftarrow \Sigma_1.\text{Retrieve}(s_j)$  
8: $k_{s,j} := h_{s,j}^{\alpha}, t_j := H_2(s_j, k_{s,j})$  
9: end for  
10: return $\{r_s, \{r_{c,i}\}^v_1, \{y'_i\}^v_1, \{t_j\}^v_1\}$  

end procedure  

1: procedure $H_1(a)$  
2: if (challenge = 1) then  
3: $\delta \leftarrow \{0, 1\}$, $r \leftarrow Z_p^*$  
4: if (δ = 0) then  
5: else  
6: end if  
7: $\Sigma_1.\text{Add}(a, h, (\delta, r))$  
8: end if  
9: return $h$  
10: end procedure  

end procedure  

1: procedure $O_e^*(C^*, S^*, s_0^*, s_1^*)$  
2: $b \leftarrow \{0, 1\}, r_{c,1}, \ldots, r_{c,v} \leftarrow Z_p^*$  
3: for all $c_i \in C^*$ do  
4: $H_1(c_i), (c_i, h_{c,i}, (\delta_i, r_i)) \leftarrow \Sigma_1.\text{Retrieve}(c_i)$  
5: if $\delta_i = 0$ then  
6: else  
7: end if  
8: end for  
9: for all $s_j \in S^*$ do  
10: $H_1(s_j), (s_j, r_j, h_{s,j}, \delta_j) \leftarrow \Sigma_1.\text{Retrieve}(s_j)$  
11: if $\delta = 1$ then  
12: else  
13: end if  
14: $t_j \leftarrow \{0, 1\}^r$  
15: end if  
16: end for  
17: $H_1(s_0^*), (s_0^*, h_{s,k}^*, (\delta_0^*, r_0^*)) \leftarrow \Sigma_1.\text{Retrieve}(s_0^*)$  
18: if $\delta_0^* = 1$ then  
19: else  
20: end if  
21: challenge := 1  
22: return $\{r_s^*, \{r_{c,i}\}^v_1, \{y'_i\}^v_1, \{t_j\}^v_1\}$  
23: end procedure  

Figure 10. Protocol Ψ: security argument for server privacy
Challenge Phase: $A_s$ challenges $C^*$ with $(C^*, S^*)$ and $(c^*_0, c^*_1)$. $C^*$ chooses $b \leftarrow \{0, 1\}$ and responds with

$$
\text{View}_S(C^* \cup \{c^*_b\}, S^*) = \{r_s, Y = \{y_i\} \cup \{y^*_b\} : y_i = h_{c,i}^{r_{c,i}}, r_s, r_{c,i} \leftarrow Z_p^* \}
$$

with the values computed as follows:

1. Choose $r_s, r_{c,1}, \ldots, r_{c,v} \leftarrow Z_p^*$.
2. For each $c_i \in C^*$, compute $y_i = (H_1(c_i))^{r_{c,i}}$.
3. For $c^*_b$, choose $r^*_{c,b} \leftarrow Z_p^*$ and compute $y^*_b = (H_1(c^*_b))^{r^*_{c,b}}$.

At the end of the game $A_s$ returns its guess $b'$. Now we claim that it cannot win the game with a non-negligible advantage as $y^*_b = y^*_{\bar{b}}$ for some $r^*_{c,b}$. Let $H_1(c^*_b) = g^\alpha$ and $H_1(c^*_b) = g^\beta$,

$$
y^*_b = (H_1(c^*_b))^{r^*_{c,b}} = g^{\alpha r^*_{c,b}} = (g^\beta)^{(\alpha r^*_{c,b})^{\beta^{-1}}} = (H_1(c^*_b))^{r^*_{c,b}}
$$

where $r^*_{c,b} = \alpha r^*_{c,b} \beta^{-1}$.  

\[\square\]