PRIVATE SET INTERSECTION: PROBLEMS ON SAMPLING FROM THE INTERSECTION

TYLER BEAUREGARD AND JANABEL XIA

Advisor: Mike Rosulek
Oregon State University

Abstract. In this paper we explore the problem of Private Set Intersection (PSI), in which each participant of the protocol learns only the elements in the intersection of all of their sets, and nothing else. Though there exist protocols for several variations of this problem, including cardinality set-intersection and threshold set intersection, we look at functions on the intersection that have not yet been explored outside of general secure circuit computation: sampling an element from the intersection. In particular, we propose new secure 2-party protocols for the following problems: randomly sampling an element from the intersection, sampling an element from the intersection by one party's ranking, and sampling an element from the intersection by some joint scoring function of elements in the input sets. We show our protocols are secure in the presence of semi-honest adversaries.

1. Introduction

1.1. **Problem Statement and Motivation.** Private Set Intersection (PSI) allows two parties with private sets X and Y to compute the intersection of their sets $X \cap Y$ while hiding elements that are not in the intersection. Such protocols have a variety of applications, from private contact discovery to calculating statistics on ad efficacy rates. A more specific yet simple application of PSI is the concept of a "private Doodle poll", in which two parties wish to find times during which they are both available, but might not want to reveal their entire availability schedule to each other.

However, existing protocols do not consider the general problem of sampling from the intersection, which is important in cases where parties do not wish to reveal the entire intersection to each other. In our paper, we specifically look at the following three problems: i) sampling an element uniformly at random from the intersection, ii) finding the element in the intersection with the maximum combined score from both parties, and iii) finding the element in the intersection with the highest-rank according to one party. We provide formal descriptions of these problem statements below.

1.1.1. *Random Sample PSI.* Suppose party A has private input set X and party B has private input set Y. Then the problem of Random Sample PSI is to output an item uniformly at random from the intersection $X \cap Y$. In the Doodle poll example, this corresponds to

Date: August 12, 2021.

This work was done during the Summer 2021 REU program in Mathematics and Theoretical Computer Science at Oregon State University supported by NSF grant DMS-1757995.

choosing one meeting time that works for both parties, which is effective when parties do not wish to reveal more about their schedule than one availability they share in common.

1.1.2. *Combined Score PSI.* Suppose party A and party B have the respective private inputs

$$X = \{(x_1, s(x_1)), \ldots, (x_m, s(x_m))\}$$
$$Y = \{(y_1, t(y_1)), \ldots, (y_n, t(y_n))\}$$

where $s, t : \{0,1\}^* \to \mathbb{Z}$ are score functions chosen by A and B, respectively. Then the problem of Combined Score PSI is to output $\arg\max_{y_i \in X \cap Y}(f(s(y_i), t(y_i)))$ where f is a function of both parties scores, i.e. the item with maximal combined score according to some function f. In the Doodle poll example, this corresponds to choosing the best meeting time according to both party's preferences, where best is defined by some function of both party's preferences.

1.1.3. *One-sided Rank PSI.* Suppose party A and party B have the respective private inputs

$$X = \{x_1, \ldots, x_m\}$$
$$Y = \{(y_1, t(y_1)), \ldots, (y_n, t(y_n))\}$$

where $t : \{0,1\}^* \to [n]$ is an injective ranking function chosen by B. Then the problem of One-sided Rank PSI is to output $\arg\max_{y_i \in X \cap Y}(t(y_i))$, i.e. the item y_i such that y_i is of maximal rank according to B. In the Doodle poll example, this corresponds to choosing the best meeting time according to one party's preferences. This is useful when the other party's preferences are virtually nonexistent, as our proposed protocol for this problem is more straightforward than that for Combined Score PSI.

1.2. **Related Work.** Much of our work is based off of the first Diffie-Hellman-based PSI (DH-PSI) protocols discovered by [Mea86] [8] and [HFH99] [5] with linear communication complexity. However, state-of-the-art protocols for general semi-honest secure PSI, such as those proposed by [KKR$^+$16] [7], apply general oblivious evaluation of a pseudorandom function (OPRF) techniques in place of the specific DH-PSI framework. These OPRF techniques scale better to large sets and are independent of the input item bitlengths.

Although our protocols continue with the DH-PSI model, there exist various other approaches to PSI. The technique of Oblivious Polynomial Evaluation (OPE), for example, encodes elements as roots of an encrypted polynomial, which allows for computing combinatorial functions on sets by using algebraic properties of polynomials. Several PSI protocols use this approach, such as threshold set intersection by [KS04][6], where only intersection elements that appear over a certain number of times are revealed, and threshold cardinality by [ZC84] [11], where the cardinality of the intersection is only revealed if above a certain threshold.

The problem that we are looking at specifically, namely sampling a single element from the intersection according to various metrics, is just one instance of computing functions on the intersection in PSI. The best known approach for computing general functions on the intersection for large sets is efficient circuit-based PSI, proposed by [PST$^+$] [10], which has linear communication complexity.

1.3. **Our Contributions.** We provide protocols for all three problems defined above that are secure in the presence of semi-honest adversaries under the Decisional Diffie-Hellman assumption (DDH). For both the random sampling problem and the one-sided ranking problem, our protocol leaks only the cardinality of the intersection. For the combined score problem, our protocol leaks the cardinality of the intersection and a histogram of the sum of both parties' scores over elements in the intersection.

Though secure garbled circuit computation can compute any function on the intersection without leaking anything else in $O(n \log n)$ time (and in some cases $O(n)$ with precomputation), our protocols can also all be achieved in $O(n)$ time with the appropriate data structures and simple add-ons. Furthermore, our protocols are conceptually simple, as they are all based off of a Diffie-Hellman style of communication. Finally, our protocols consider the problem of sampling one element from the set intersection, a function that has not been widely considered in the world of PSI but nonetheless has a wide variety of applications.

2. Preliminaries

In this section, we introduce formal cryptographic concepts that we will use to describe and prove security of our protocols.

2.1. **Notation.** We make use of the following notation throughout this paper:

- $x \leftarrow S$ denotes the outcome of sampling x uniformly at random from the set S.
- $\{\ldots\}$ denotes an ordered set, so $\{e_1, \ldots, e_n\}$ is equivalent to the ordered tuple $(e_1, \ldots, e_n)$
- $[n]$ denotes the set $\{1, 2 \ldots, n\}$.
- $S[n]$ denotes the value at the n-th position in a set S (where n starts at 1).
- $A \approxeq B$ denotes that A and B are distributions that are computationally indistinguishable from each other.
- $\mathcal{F}_\pi$ denotes the ideal functionality of protocol π
- $\perp$ denotes an error (e.g. in a decryption of a ciphertext with the wrong key), or in some cases denotes no information
- $\mathsf{Enc}(K, m)$ denotes the resulting ciphertext from encrypting a message m with key K
- $\mathsf{Dec}(K, c)$ denotes the resulting plaintext from decrypting a ciphertext c with key K
- ORE(m) denotes the encryption of m under the Order Revealing Encryption scheme by [2]

2.2. **Decisional Diffie-Hellman Assumption.** First, to establish some notions used in the following sections, we have the following:

Definition 2.1. Let $\mathbb{G}$ be a cyclic group with generator g. The *Diffie-Hellman distribution* of triples, denoted $\mathsf{DH}_{1,\mathbb{G}}$, is the distribution of triples of the form:

$$(g^a, g^b, g^{ab})$$

where $g^a \leftarrow \mathbb{G}$ and $g^b \leftarrow \mathbb{G}$.

Definition 2.2. Let $\mathbb{G}$ be a cyclic group with generator g. The general *Diffie-Hellman distribution*, denoted $\mathsf{DH}_{n,\mathbb{G}}$, is the distribution of $(2n + 1)$-tuples of the form:

$$(\alpha_1, \ldots, \alpha_n, g^b, \alpha_1^b, \ldots, \alpha_n^b)$$

where $\alpha_i \leftarrow \mathbb{G}$.

We use $\mathsf{Rand}_{n,\mathbb{G}}$ to define a "random-looking" counterpart to the Diffie-Hellman distribution. It consists of $(2n+1)$-tuples of the form

$$(\alpha_1, \ldots, \alpha_n, g^b, \gamma_1, \ldots, \gamma_n)$$

where $\alpha_i \leftarrow \mathbb{G}$, $g^b \leftarrow \mathbb{G}$ and $\gamma_i \leftarrow \mathbb{G}$.

Now we can state the Decisional Diffie-Hellman assumption (DDH), which is the result we will primarily use throughout our paper:

Claim 2.3 ([Bon98][1])**.**

$$\mathsf{DH}_{1,\mathbb{G}} \approx \mathsf{Rand}_{1,\mathbb{G}}$$

and more generally,

$$\mathsf{DH}_{n,\mathbb{G}} \approx \mathsf{Rand}_{n,\mathbb{G}}$$

for all $n \geq 1$.

2.3. **Random Oracle.**

In the protocols described below, parties have access to a function H, called a random oracle. Formally, we can define it as the following:

Definition 2.4. A random oracle $H : \{0,1\}^* \to \mathbb{G}$ is a function chosen uniformly at random from the set of all functions from $\{0,1\}^*$ to the group $\mathbb{G}$.

Intuitively, we can think of the random oracle as a perfect black-box, where each party can query on any value, but for which it is impossible to "reverse-engineer" a value, or find its inverse.

Throughout this paper, we will be returning H in our reduction algorithms and simulators. Though it does not quite make sense to return H, as it is an exponentially large function $\{0,1\}^* \to G$, we can interpret this notation as giving the party polynomial-time access to H, i.e. any distinguisher algorithm a party runs on their view can access random oracle values a polynomial number of times. This allows us to treat the view returned by an algorithm as a distribution that includes the function H.

2.4. **Secure Two-Party Computation.**

We begin by defining the ideal functionality of a protocol.

Definition 2.5. An *ideal functionality* of a protocol π, denoted $\mathcal{F}_\pi$, is the following map:

$$(x, y) \mapsto (f_1(x, y), f_2(x, y))$$

Suppose we have a two-party protocol that wishes to reveal $F(x, y) = (f_1(x, y), f_2(x, y))$ to both parties and nothing else, where f_1, f_2 are specified functions on party inputs x and y. Then we say such a protocol implements ideal functionality $\mathcal{F}_\pi$.

To formalize our definition of security, we use the notion of an ideal world that relies on a trusted third party to compute information, which is secure by definition. Though our protocols do not live in the ideal world, it serves as an important security benchmark to which we can compare our real world protocols.

In the real world, two parties engage in a protocol to implement functionality $\mathcal{F}_\pi$. In the ideal world, two parties send their inputs x, y to a trusted third party, T, who securely computes $F(x, y) = (f_1(x, y), f_2(x, y))$ in a black box and returns it to both parties.

We define a party's *view* to be a list of their private inputs, any private randomness they generate, and any messages they receive during the protocol. Throughout this paper, we will be working with *semi-honest adversaries*, who follow the protocol honestly but may try to learn as much about the private inputs of others based on their own view.

Suppose parties A and B participate in protocol π with private inputs x, y respectively. Then $\mathsf{view}^{\pi}_{A,S}(x, y)$, denotes party A's *real view* while engaging in protocol π in the real world, consisting of private randomness and protocol messages received from B. Similarly, $\mathsf{view}^{\pi}_{B,S}(x, y)$ denotes party B's real view while engaging in protocol π.

To say our protocols are secure, it is sufficient to say that anything an adversary can do in the real world can also be done in the ideal world, as the adversary's view in the ideal world consists solely of its own inputs and $F(x, y)$. A *simulator* uses an adversary's inputs and the adversary's functionality output $f(x, y)$ to generate a simulated real-world view, while in the ideal-world itself.

Definition 2.6. Suppose parties A and B participate in protocol π with private inputs x, y respectively. Then $S^{\pi}_{A}(x, f_A(x, y))$ is party A's simulated view as generated by simulator S_A, and similarly $S^{\pi}_{A}(y, f_B(x, y))$ party B's simulated view as generated by simulator S_B.

Definition 2.7. We say that a two-party protocol π *securely computes* a function $F(x, y) = (f_A(x, y), f_A(x, y))$ on party inputs x and y in the presence of semi-honest adversaries if there exists simulators S_A, S_B such that

$$\mathsf{view}^{\pi}_{A}(x, y) \approxeq S^{\pi}_{A}(x, f_A(x, y))$$

and

$$\mathsf{view}^{\pi}_{B}(x, y) \approxeq S^{\pi}_{B}(y, f_B(x, y)).$$

Showing that such a simulator exists proves that there is nothing an adversary can compute in the real world that the adversary cannot compute in the ideal world, since an adversary can only compute functions of its view. Thus providing such a simulator proves that our protocol is secure.

2.5. **Hybrids.** Without loss of generality, to show that such a simulator for player A exists, we will provide a simulator algorithm $S^{\pi}_{A}(y, f_A(x, y))$ and show that it is indistinguishable from the real world view. Often we will use a reduction algorithm to obtain an intermediate protocol π' that is indistinguishable from the real protocol π, and so it will remain to show that $\mathsf{view}^{\pi'}_{A}(x, y) \approxeq S^{\pi}_{A}(y, f_A(x, y))$.

To show indistinguishability of views, we will provide multiple hybrid two-party protocols that live in both the real and ideal worlds, where we show indistinguishability between each hybrid. These hybrids rely on real world inputs from B, but also make calls to trusted third party T for elements in $f_A(x, y)$. The end goal is to have a protocol that does not rely on B's inputs, and instead relies entirely on calls to T for $f_A(x, y)$, at which point we have a working simulator.

Within our security proofs, we will often not write out the entire hybrid algorithm. The reader may assume that unless stated otherwise, each hybrid is equivalent to the previous hybrid (or reduction algorithm) except for the specified changes.

2.6. **Symmetric Key Encryption.** We will use a one-time symmetric encryption scheme, in which the same key that is used to encrypt a message can be used to decrypt the corresponding ciphertext.

Definition 2.8. A *one-time symmetric key encryption scheme* is an encryption scheme such that for any messages $m_0, m_1 \in \{0,1\}^*$ and $K \leftarrow \mathbb{G}$ a randomly generated encryption key,

$$\text{Enc}(K, m_0) \approx \text{Enc}(K, m_1).$$

We will be using this property in our security proofs.

2.7. **Order Revealing Encryption.** We will be using an Order Revealing Encryption scheme, or ORE for short, which is a symmetric encryption scheme that reveals no more than the ordering of the plaintexts. A scheme that is provably semantically secure has been discovered by [2]. More formally,

Definition 2.9. An Order Revealing Encryption scheme, denoted ORE, is an encryption scheme such that, for a secret key K and similarly ordered sets $P = \{p_1, \ldots, p_m\}$ and $Q = \{q_1, \ldots, q_m\}$, we have the distributions

$$(\mathsf{ORE}(K, p_1), \ldots, \mathsf{ORE}(K, p_m)) \approx (\mathsf{ORE}(K, q_1), \ldots, \mathsf{ORE}(K, q_m))$$

where $\mathsf{ORE}(m)$ denotes the ciphertext resulting from the encryption of message m.

We will use this indistinguishability in our security proofs for protocol $\mathsf{OneSidedRank}$. We will also often use the shorthand $\mathsf{ORE}(m)$ in place of $\mathsf{ORE}(K, m)$ when it is clear from the context what K is, e.g. when we are dealing with only one key K.

2.8. **A Note on Complexity.** In these protocols, we measure efficiency in terms of both computation and communication. Computational complexity is given asymptotically as an expression of the size of each party's secure input. For communication, we'll be measuring the number of rounds of communication necessary to complete the protocol. Here, a *round of communication* is any one-way transfer of data between parties, regardless of the volume of data shared. [1]

For each protocol, we will be presenting the computational and communication complexity without providing a rigorous justification. Since all of our protocols are close variations of Diffie-Hellman PSI (DH-PSI), it is usually clear that the complexity is also the same as DH-PSI. For the protocols where there may be some uncertainty, we make sure to provide some rationale.

3. Cardinality Protocol

In some cases, knowing the entirety of the intersection is too much. Instead, two parties may only want to know how many items they have in common. For instance, when measuring the conversion rate of an online advertisement, parties don't need to know which ad-viewers ultimately made a purchase, only how many.

We review the following protocol, due to [HFH99][5] to compute the cardinality of the intersection of Alice and Bob's sets.

[1] In other papers, the phrase "round of communication" could refer to the number of back-and-forth transfers of data or simply the number of transfers that must be performed sequentially.

```
cardinality

Parameters: Cyclic group $\mathbb{G}$ of order $q$
    Alice                                   Bob
    $X \subseteq \{0,1\}^*$                 $Y \subseteq \{0,1\}^*$

1. $a \leftarrow \mathbb{Z}_q$
2. $P := \{H(x)^a \mid x \in X\}$
                          $\xrightarrow{P}$
3.                                          $b \leftarrow \mathbb{Z}_q$
4.                                          $Q := \{p^b \mid p \in P\}$ (shuffled)
5.                                          $R := \{H(y)^b \mid y \in Y\}$
                          $\xleftarrow{Q, R}$
6. $S := \{r^a \mid r \in R\}$:
7. **output** $k := |Q \cap S|$
```

FIGURE 1. DH-PSI Cardinality Visualized

It is clear that this protocol is a close variant of the original DH-PSI protocol in [HFH99] [5]. The key difference is the shuffling Bob does before returning the values to Alice. Since Alice's inputs are raised to Bob's private exponent b, Alice cannot recognize her own inputs upon their return. In the original DH-PSI protocol, Alice could simply recall the order in which she sent them. Now that her elements are shuffled, Alice can only determine how many of her inputs match Bob's, not their identities. Thus, she only learns the cardinality of the intersection.

This protocol will serve as a useful reference point for our own proposed protocols, which will use some similar ideas. We will omit the proof of security here, as this is a well-known protocol and our detailed security proof for our protocol RandomSample in the next section is very similar.

4. Randomly Sampling from the Intersection

4.1. **Protocol Description.** We propose the following protocol to produce a random item in the intersection of Alice and Bob's sets, while also leaking the cardinality of those sets.

We see that protocol RandomSample is just an slight modification of the pre-established protocol cardinality. We start with standard DH-PSI and we blind Alice to the order of her elements via shuffling. Since Alice can recognize that two elements match, but is blind to the order of her own elements, she can choose an element that matches essentially at random and return it to Bob. Just as in Cardinality, Alice only learns the cardinality of the intersection. Upon receipt, Bob, who has maintained the order of his sets, can declassify the item (i.e. finding the corresponding $y \in Y$).

```
RandomSample

Parameters: Cyclic group $\mathbb{G}$ of order $q$
   Alice                                      Bob
   $X \subseteq \{0,1\}^*$                    $Y \subseteq \{0,1\}^*$

1. $a \leftarrow \mathbb{Z}_q$
2. $P := \{H(x)^a \mid x \in X\}$
                          $\xrightarrow{P}$
3.                                            $b \leftarrow \mathbb{Z}_q$
4.                                            $Q := \{p^b \mid p \in P\}$ (shuffled)
5.                                            $R := \{H(y)^b \mid y \in Y\}$
                          $\xleftarrow{Q, R}$
6. $S := \{r^a \mid r \in R\}$:
7. $k := |Q \cap S|$
   (if $k = 0$, terminate the protocol)
8. Choose uniformly an index $j$ s.t. $R[j]^a \in Q$
                          $\xrightarrow{j}$
9.                                            Bob calculates $y_j$ using $j$
                                              output $y_j \in Y$ (or None if $j = \perp$)
```

Figure 2. RandomSample Protocol Visualized

We claim our protocol above has the functionality that Alice only learns the cardinality of the intersection, and Bob only learns the desired element, a random sample from the intersection.

Theorem 4.1. *If the DDH assumption holds, then* CombinedScore$_p$ *securely computes*

$$F(X, Y) = ((|Y|, |X \cap Y|), (|X|, y^*))$$

where y^* is an element sampled uniformly at random from $X \cap Y$, or null if $X \cap Y = \emptyset$.

4.2. **Complexity.** As mentioned, RandomSample is a clear modification of the DH-PSI protocol. The only modifications made (shuffling Alice's input and selecting randomly from a set of values) bear no weight on the complexity. Because of this, we can see that this protocol runs in linear time $O(m)$ where m is the size of larger of the two party's secure input.

For the communication complexity, we can count three arrows in the protocol, corresponding to three rounds of communication. Since each round of communication is the product of computations dependent on the data given in the previous round (for example, the value r in the last round is dependent upon the values R and Q in the second), we can see that there are no redundant rounds of communication.

4.3. **Security Proof.** To prove Theorem 4.1, we will consider the views of Alice and Bob separately.

4.3.1. *RandomSample Security Proof: Alice.* We will now prove that the protocol RandomSample is secure in the view of Alice.

Theorem 4.2. *If the DDH-assumption holds, then protocol RandomSample securely computes $f_A(X, Y) = (|Y|, |X \cap Y|)$ in the presence of semi-honest adversary Alice.*

To prove Theorem 4.2, we make use of the following reduction algorithm and simulator, where $k = |X \cup Y|, l = |X \cap Y|$ and $m = |X|, n = |Y|$.

Reduction Algorithm:

Input: X – Alice's secure input
Y – Bob's secure input
$(\alpha_1, \ldots, \alpha_k,\ \beta_1, \ldots, \beta_k)$ – These inputs are part of the $(2k + 1)$-tuples from $\mathsf{DH}_{k,\mathbb{G}}$ or $\mathsf{RAND}_{k,\mathbb{G}}$ (so $\beta_i = \alpha_i^b$ or $\beta_i \leftarrow \mathbb{G}$)

```
1: procedure REDUCE($X, Y, \alpha_1, \ldots, \alpha_k, \beta_1, \ldots, \beta_k$)
2:     $a \leftarrow \mathbb{Z}_q$
3:     $\{v_1, \ldots, v_k\} := X \cup Y$
4:     program $H[v_i \to \alpha_i]$ a random oracle
5:     $M := \{\beta_i^a \mid v_i \in X\}$
6:     $W := \{\beta_i \mid v_i \in Y\}$
7:     return $(a, M \text{ shuffled}, W \text{shuffled}, H)$
8: end procedure
```

Simulator $S_A^{\mathsf{RandomSample}}$:

Input: $k = |X \cup Y|, l = |X \cap Y|$ and $n = |Y|$

```
1: procedure SIMULATE($k, l, m, n$)
2:     $H :=$ honest random oracle
3:     $a \leftarrow Z_q$
4:     $\beta_i \leftarrow \mathbb{G}$ for $i \in [k]$
5:     $\gamma_i = \beta_i^a$ for $i \in [l]$
6:     $\gamma_i \leftarrow \mathbb{G}$ for $i \in [n+1, \ldots, k]$
7:     $M := \{\gamma_i \mid i \in [l]\} \cup \{\gamma_i \mid i \in [n+1, k]\}$
8:     $W := \{\beta_i \mid i \in [n]\}$
9:     return $(a, M \text{ shuffled}, W \text{shuffled}, H)$
10: end procedure
```

Notice that our reduction algorithm takes in two distributions of inputs equivalent to $\mathsf{DH}_{k,\mathbb{G}}$ and $\mathsf{Rand}_{k,\mathbb{G}}$, which are indistinguishable under the DDH assumption. Let $\mathsf{Reduc}_{\mathsf{DH}}$ denote the distribution of $(a, M \text{ shuffled}, W \text{ shuffled}, H)$ for input $\mathsf{DH}_{n,\mathbb{G}}$, and let $\mathsf{Reduc}_{\mathsf{rand}}$

denote the distribution of $(a, M \text{ shuffled}, W \text{ shuffled}, H)$ for input $\mathsf{Rand}_{n,\mathbb{G}}$. Then it remains to show the following claim:

Claim 4.3. *The following distributions are indistinguishable:*

$$\mathsf{Reduc}_{\mathsf{DH}} \approxeq \mathsf{Real}_A^{\mathsf{RandomSample}}$$

$$\mathsf{Reduc}_{\mathsf{rand}} \approxeq S_A^{\mathsf{RandomSample}}(X, |Y|, |X \cap Y|).$$

Proof. First we show that $\mathsf{Reduc}_{\mathsf{DH}} \approxeq \mathsf{Real}_A^{\mathsf{RandomSample}}$. Our reduction algorithm uses randomly sampled group elements $\alpha_1, \ldots, \alpha_n$ to program the random oracle H by setting $H(v_i) = \alpha_i$ for all $v_i \in X \cup Y$. Note that since α_1 are random, this is equivalent to sampling H randomly from all random oracles, and evaluating it on $v_i \in X \cup Y$ in the real protocol.

Then $\beta_i = \alpha_i^b = H(v_i)^b$, so

$$M = \{\beta_v^a \mid v \in X\} = \{H(x_i)^{ab} \mid x_i \in X \}$$
$$W = \{\beta_v \mid v \in Y\} = \{H(y_i)^b \mid y_i \in Y\}$$

and now $\mathsf{Reduc}_{\mathsf{DH}}$ clearly matches the protocol.

Now we show that $\mathsf{Reduc}_{\mathsf{rand}} \approxeq S_A^{\mathsf{RandomSample}}(X, |X \cap Y|)$. Note that since all β_i are random, they are decoupled from specific elements in $X \cap Y$, so we can reorder which β_i correspond to which elements in $X \cup Y$. In particular, we can treat $\beta_1, \ldots, \beta_l$ as linked to elements $v \in X \cap Y$ (acting as $H(v)^b$ in the real protocol). Then the reduction algorithm is equivalent to the following hybrid:

$$M = \{\beta_i^a \mid i \in [l]\} \cup \{\beta_i^a \mid i \in [n+1, \ldots, k]\}$$
$$W = \{\beta_i \mid i \in [n]\}$$

where we can order the sets M, W as above for the sake of clarity, since they are going to be randomly shuffled once returned in the view $\mathsf{Reduc}_{\mathsf{rand}}$.

However, we see that β_i for $i \in [n+1, \ldots, |X \cup Y|]$ is disjoint from W. Furthermore, a and β_i were generated uniformly at random, so β_i^a remains random. Then we can sample new elements $\gamma_i \leftarrow \mathbb{G}$ in place of β_i^a, so that we have this hybrid:

$$M = \{\beta_i^a \mid i \in [l]\} \cup \{\gamma_i \leftarrow \mathbb{G} \mid i \in [n+1, \ldots, k]\}$$
$$W = \{\beta_i \mid i \in [n]\}$$

The hybrid above generates M, W using k, l, n only, without needing the actual private set Y. This is now clearly equivalent to the simulated view $S_A^{\mathsf{cardinality}}(X, |Y|, |X \cap Y|)$, so we are done.

□

4.3.2. *RandomSample Security Proof: Bob.* We will now show that the protocol RandomSample is secure in the view of Bob.

Theorem 4.4. *If the DDH-assumption holds, then protocol RandomSample securely computes $f_B(X,Y) = (|X|, y^*)$ in the presence of semi-honest adversary Bob, where $y^* \in X \cap Y$ is either the final element sampled at random from the intersection $X \cap Y$ or $y^* = \perp$ if $X \cap Y = \emptyset$.*

To prove Theorem 4.4, we make use of the following reduction algorithm and simulator, where $m = |X|$ and $n = |Y|$.

Reduction Algorithm:

Input: X – Alice's secure input
Y – Bob's secure input
$(\alpha_1, \ldots, \alpha_m, \beta_1, \ldots, \beta_m)$ – These inputs are part of the $(2m+1)$-tuples from $\mathsf{DH}_{m,\mathbb{G}}$ or $\mathsf{RAND}_{m,\mathbb{G}}$ ($\beta_i = \alpha_i^a$ or $\beta_i \leftarrow \mathbb{G}$)

```
1: procedure REDUCE($X, Y, \alpha_1, \ldots, \alpha_m, \beta_1, \ldots, \beta_m$)
2:     $b \leftarrow \mathbb{Z}_q$
3:     $X = \{x_1, \ldots, x_m\}$
4:     $Y = \{y_1, \ldots, y_n\}$
5:     program $H[x_i \to \alpha_i]$ a random oracle
6:     abort if $\exists z_1 \neq z_2$ s.t. $H(z_1) = H(z_2)$
7:     $M := \{\beta_i \mid i \in [m]\}$
8:     $W := \{H(y_i)^b \mid y_i \in Y\}$
9:     $J := \{j \mid y_j \in X\}$
10:    $j \leftarrow J$
11:    return $(b, M, j, H)$
12: end procedure
```

Simulator $S_B^{\mathsf{RandomSample}}$:

Input: $Y, m = |X|, y^*$

```
1: procedure SIMULATE($Y, m, y^*$)
2:     $H$ := honest random oracle
3:     $b \leftarrow Z_q$
4:     $\beta_i \leftarrow \mathbb{G}$ for $i \in [m]$
5:     $M := \{\beta_i \mid i \in [m]\}$
6:     $j \mid y_j = y^*$
7:     return $(b, M, j, H)$
8: end procedure
```

Notice that our reduction algorithm takes in two distributions of inputs equivalent to $\mathsf{DH}_{m,\mathbb{G}}$ and $\mathsf{Rand}_{m,\mathbb{G}}$, which are indistinguishable under the DDH assumption. Let $\mathsf{Reduc}_{\mathsf{DH}}$ be the distribution of (b, M, j, H) for input $\mathsf{DH}_{m,\mathbb{G}}$, and let $\mathsf{Reduc}_{\mathsf{rand}}$ be the distribution of (b, M, j, H) for input $\mathsf{Rand}_{m,\mathbb{G}}$. Then it remains to show the following claim:

Claim 4.5. *The following distributions are indistinguishable:*

$$\mathsf{Reduc}_{\mathsf{DH}} \approx \mathsf{Real}_B^{\mathsf{RandomSample}}$$

$$\mathsf{Reduc}_{\mathsf{rand}} \approx S_B^{\mathsf{RandomSample}}(Y, |X|, y^*).$$

Proof. First we show that $\mathsf{Reduc}_{\mathsf{DH}} \approx \mathsf{Real}_B^{\mathsf{RandomSample}}$. Our reduction algorithm uses randomly sampled group elements $\alpha_1, \ldots, \alpha_n$ to program the random oracle H by setting $H(v_i) = \alpha_i$ for all $v_i \in X \cup Y$. We also have input $\beta_i = \alpha_i^a$, so the reduction algorithm is equivalent to the following:

$$\begin{aligned} M &:= \{H(x_i)^a \mid i \in [m]\} \\ J &:= \{j \mid y_j \in X\} \end{aligned}$$

where M is clearly equivalent to the real protocol view. Now note that the algorithm aborts if there is a hash collision, only returning a view if there are none. Then $H(x)^{ab} = H(y)^{ab} \Leftrightarrow x = y$, so the following hybrid must be equivalent:

$$\begin{aligned} a &\leftarrow \mathbb{Z}_q \\ X^H &:= \{\beta^b \mid i \in [m]\} \\ J &:= \{j \mid (H(y_j)^b)^a \in X^H\} \\ j &\leftarrow J. \end{aligned}$$

Since the probability the algorithm aborts, i.e. that there is hash collision, is negligible, the hybrid above is indistinguishable to removing the abort functionality. Then our algorithm clearly matches the real world protocol.

Now we show that $\mathsf{Reduc}_{\mathsf{rand}} \approx S_B^{\mathsf{RandomSample}}(Y, |X|, y^*)$. The reduction algorithm takes in β_i where $\beta_i \leftarrow G$, which is equivalent to sampling these randomly in the algorithm itself. Also note that y^* is defined to be the element sampled randomly from the intersection by Alice, so that the following hybrid is equivalent:

```
progarm $H[x_i \to \alpha_i]$ a random oracle
    abort if $\exists z_1 \neq z_2$ s.t. $H(z_1) = H(z_2)$
    $\beta_i \leftarrow \mathbb{G} \mid i \in [m]$
    $M := \{\beta_i \mid i \in [m]\}$
    $j \mid y_j = y^*$.
```

Our reduction algorithm uses randomly sampled group elements $\alpha_1, \ldots, \alpha_n$ to program the random oracle H by setting $H(v_i) = \alpha_i$ for all $v_i \in X \cup Y$, but then does not use α_i anywhere else. Furthermore, since the probability the algorithm aborts is negligible, the hybrid above is indistinguishable to removing the programming H and abort functionality completely. Then our algorithm clearly matches the simulator, so we are done.

□

5. Combined Score Protocol - Prime Order

5.1. **Protocol Description.** Instead of computing all items shared between two parties, it may be useful to only compute the "best" item shared between two parties by some metric. In this scenario, we consider a natural example of this. Each party's input consists of two parts: an item and the score of that item. Their goal is to compute the item in the intersection such that the sum of that item's scores (the sum of the score assigned by each party) is maximized.

We break down the protocol and some tools utilized therein. The end goal is to find the item in the intersection with the best combined score. To meet this goal, Alice must compute the combined score $s_i + t_j$ for each $x_i = y_j$. A natural approach is to compute the sum as $g^{s_i} g^{t_j} = g^{s_i + t_j}$, so that exponentiation acts as a homomorphic encryption tool. Since the maximum value of the combined scores is $2N$, we can efficiently compute a discrete log. However, this also implies that we cannot reveal g^{s_i} or g^{t_j} separately. Instead, we mask these values before sending them over. Since we must keep items and their associated scores from being separated, we can also use the item itself to mask its score.

In particular, Alice converts her item-score pair (x, s) to the form $(H(x)^a, (g^s H(x))^c)$. Bob encodes his scores in a similar format, converting his item-score pairs (y, t) to the form $\left(g^t H(y)^{-1}\right)^d$. Now it is clear that neither party can learn information about the other party's individual scores of their elements. However, Bob's encoded scores complement Alice's encoded scores in that multiplying these encoded scores (after exponentiation within the protocol) over elements in the intersection $x_i = y_j$ gives us $(g^{cd})^{s(x_i)+t(y_j)}$. Having Bob encrypt his encoded scores with the key $H(y)^b$ also ensures that Alice can only learn these combined scores over the intersection, and does not learn "combined" scores for elements outside the intersection.

We claim that our protocol above has the functionality that Alice only learns combined scores over the intersection, and nothing about individual scores of any of Bob's elements. Since Bob simply receives an index at the end to return a specific element in his set, he only learns the desired element.

Theorem 5.1. *If the DDH assumption holds, then* $\mathsf{CombinedScore}_p$ *securely computes*

$$F(X, Y) = ((|Y|, \{s(v) + t(v) \mid v \in X \cap Y\}), (|X|, y^*))$$

where

$$y^* = \arg\max_{v \in X \cap Y} (s(v) + t(v))$$

or $y^* = \perp$ *if* $X \cap Y = \emptyset$.

5.2. **Complexity.** For computational complexity, we claim that the protocol runs in $O(m + N)$ operations, where m is the size of the larger of the two party's secure input and N is the value of the largest possible score. Because this protocol is a bit more complicated than regular DH-PSI, we provide an informal justification for this classification. Most of the steps here can clearly be calculated in linear (or sub-linear) time. Lines 8-9 are where linearity is perhaps most uncertain.

Line 8 of the protocol is a for-loop whose body contains an existential quantifier. At first glance, this could certainly seem to indicate that something is happening in quadratic

$\mathsf{CombinedScore}_p$

Parameters: Cyclic group $\mathbb{G}$ of order p prime

```
Alice                                                    Bob
$X := \{(x_1, s_1), \ldots, (x_m, s_m)\} \subseteq \{0,1\}^* \times [N]$    $Y := \{(y_1, t_1), \ldots, (y_m, t_m)\} \subseteq \{0,1\}^* \times [N]$

                Stage 1: Computing Shared Keys

1. $a \leftarrow \mathbb{Z}_p, c \leftarrow \mathbb{Z}_p,$
2. $P := \{(H(x)^a, g^{cs} H(x)^c) \mid (x, s) \in X\}$
                          $\xrightarrow{P}$
3.                                                       $b \leftarrow \mathbb{Z}_p, d \leftarrow \mathbb{Z}_p$
4.                                                       $Q := \{(p_1^b, p_2^d) \mid (p_1, p_2) \in P\}$ (shuffled)
                          $\xleftarrow{Q}$
5. $S := \{(q_1^{a^{-1}}, q_2^{c^{-1}}) \mid (q_1, q_2) \in Q\}$

                Stage 2: Encrypting and Decrypting

6.                                                       $E := \{\mathsf{Enc}(H(y)^b, g^{dt} H(y)^{-d}) \mid (y, t) \in Y\}$
                          $\xleftarrow{E, g^d}$
7. for $(K, s) \in S$:
       if $\exists\, e_i \in E$ s.t. $T := \mathsf{Dec}(K, e_i) \neq \perp$:
           $S_{\text{scores}}[i] := \big(\text{dlog}_{g^d}(T \cdot s)\big)$

                Stage 3: Declassifying the Max Score

if $S_{\text{scores}}$ empty:
    $j := \perp$
else:
8. $j := \arg\max_i \{S_{\text{scores}}[i]\}$
                          $\xrightarrow{j}$
9.                                                       output $y_j \in Y$ (or None if $j = \perp$)
```

time, not linear time. After all, without any extra mechanism, Alice would have to test each potential key with each ciphertext before seeing what works. However, if we mark each value in E with a tag (such as the first few bits of the key $H(y)^b$), then we can now match keys to ciphertexts in linear time without jeopardizing security.

The next part of the protocol that warrants closer inspection is the body of the if statement under line 8. Here we're calculating the discrete log $T \cdot s$, or equivalently, the discrete log base g^d of $g^{d(s+t_i)}$. Essentially, we compute this by naively multiplying g^d by itself until we reach the value $g^{d(s+t_i)}$. Now, this step can be computed in time $O(N)$ since the combined scores never exceed the value $2N$. Nonetheless, if we recall that this step is computed once for every item in the intersection, it may once again seem like the protocol is not linear.

This is not as problematic as it may seem. Since Alice is ultimately computing the discrete log base g^d for each combined score in the intersection, there is no reason she cannot compute the discrete log for each combined score simultaneously. This works by multiplying g^d by itself and recording each time we reach a value that matches any one of our combined scores. This makes it so that our entire for-loop runs in time $O(m+N)$.

The communication complexity is much more straightforward, but still requires some discussion. Counting arrows directly, one might assume that there are four rounds of communication in this protocol. However, since the values E, g^d can be sent along with Q, this brings the number of rounds down by one. Since the value I is dependent on Q, E, g^d and since Q is dependent on P, it is clear that three rounds is the minimum number of rounds of communication possible.

5.3. **Security Proof.** To prove Theorem 5.1, we will consider the views of Alice and Bob separately.

5.3.1. $\mathsf{CombinedScore}_p$ *Security of Alice's View.* First, we must show that $\mathsf{CombinedScore}$ is secure in the view of Alice.

Theorem 5.2. *If the DDH-assumption holds, then protocol* $\mathsf{CombinedScore}_p$ *securely computes* $f_A(X,Y) = (|Y|, \{s(v)+t(v) \mid v \in X \cap Y\}, |X \cap Y|)$ *in the presence of semi-honest adversary Alice.*

To prove Theorem 5.2, we make use of the following reduction algorithm and simulator, where we let $m = |X|$, $n = |Y|$, $j = |X \cap Y|$, $k = |X \cup Y|$:

Reduction Algorithm — Alice:

Input: X – Alice's secure input
Y – Bob's secure input
$(\alpha_1, \dots, \alpha_k, \beta_1, \dots, \beta_k, \delta, \gamma_1, \dots, \gamma_k)$ – Inputs where $\alpha_i \leftarrow \mathbb{G}$ and either $\delta = g^d$, $(\{\beta_i\}, \{\gamma_i\}) = (\{\alpha_i^b\}, \{\alpha_i^d\})$ or $\delta, \beta_i, \gamma_i \leftarrow \mathbb{G}$. Note these inputs are general forms of the $(2k+1)$-tuples from $\mathsf{DH}_{k,\mathbb{G}}$ or $\mathsf{RAND}_{k,\mathbb{G}}$.

```
1:  procedure REDUCE($X, Y, (\alpha_1, \dots, \alpha_k, \beta_1, \dots, \beta_k, \delta, \gamma_1, \dots, \gamma_k)$)
2:      $a, c \leftarrow \mathbb{Z}_p$
3:      $\{v_1, \dots, v_k\} := X \cup Y$
4:      program $H[v_i \mapsto \alpha_i]$ a random oracle for $v_i \in X \cup Y$
5:      $X = \{(x_1, s(x_1)), \dots, (x_m, s(x_m))\}$
6:      $Y = \{(y_1, t(y_1)), \dots, (y_n, t(y_n))\}$
7:      $M := \{(\beta_i^a, (\delta)^{cs(v_i)}\gamma_i^c) \mid v_i \in X\}$
8:      $E := \{\mathrm{Enc}(\beta_i, (\delta)^{t(v_i)}\gamma_i^{-1}) \mid v_i \in Y\}$
9:      return $(a, c, \delta, M,$ shuffled$, E, H)$
10: end procedure
```

Simulator $S_A^{\mathsf{CombinedScore}_p}$:

Input: X – Alice's secure input
m, n, j, k
$\text{scores}_{|X\cap Y|} = \{s(v) + t(v) \mid v \in X \cap Y\}$

```
1: procedure SIMULATE($X, m, n, j, k, \text{scores}_{|X\cap Y|}$)
2:     $a, c \leftarrow \mathbb{Z}_p$
3:     $\delta \leftarrow \mathbb{G}$
4:     $\alpha_i, \beta_i, \gamma_i \leftarrow \mathbb{G}$ for $i \in [k]$
5:     $m_i \leftarrow \mathbb{G}$ for $i \in [m+1, k]$ \\ generating dummy messages
6:     $r_i \leftarrow \mathbb{G}$ for $i \in [m-j]$
7:     $M = \{(\beta_i^a, r_i) \mid i \in [m-j]\} \cup \{(\beta_i^a, \delta^{c(s(v_i)+t(v_i))}\gamma_i^c) \mid i \in [m-j+1, m]\}$
8:     $E = \{\text{Enc}(\beta_i, \gamma_i^{-1}) \mid i \in [k-n+1, m]\} \cup \{\text{Enc}(\beta_i, m_i) \mid i \in [m+1, k]\}$
9:     return $(a, c, \delta, M \text{ shuffled}, E \text{ shuffled}, H)$
10: end procedure
```

Note that the input distributions to the reduction algorithm are indistinguishable by the DDH assumption. Let Reduc_{DH} denote the distribution of $(a, c, \delta, M,$ shuffled$, E, H)$ when the reduction algorithm takes input $(\{\alpha_i\}, \{\alpha_i^b\}, g^d, \{\alpha_i^d\})$, and let $\mathsf{Reduc}_{\text{rand}}$ denote the distribution of $(a, c, \delta, M,$ shuffled$, E, H)$ when the reduction algorithm takes input $(\{\alpha_i\}, \{\beta_i\}, \delta, \{\gamma_i\})$, where $\beta_i, \delta, \gamma_i \leftarrow \mathbb{G}$. Now it simply remains to show the following:

Claim 5.3. *The following distributions are indistinguishable:*

$$\mathsf{Reduc}_{DH} \approxeq \mathsf{Real}_A^{\mathsf{CombinedScore}_p}$$

$$\mathsf{Reduc}_{rand} \approxeq S_A^{\mathsf{CombinedScore}_p}(X, m, n, k, \{s(v) + t(v) \mid v \in X \cap Y\}).$$

Proof. First we show that $\mathsf{Reduc}_{DH} \approxeq \mathsf{Real}_A^{\mathsf{CombinedScore}_p}$. With the corresponding input distribution we have $\delta = g^d$, $\beta_i = \alpha_i^b$ and $\gamma_i = \alpha_i^d$. The reduction algorithm programs the random oracle $H(v_i) = \alpha_i$ for all $v_i \in X \cup Y$, so we have $\beta_i = \alpha_i^b = H(v_i)^b$ and $\gamma_i = \alpha_i^d = H(v_i)^d$. Then the algorithm returns

$$M = \{((H(v_i)^b)^a, (g^d)^{cs(v_i)}\gamma_i^c) \mid v_i \in X\}$$
$$E = \{\text{Enc}(H(v_i)^b, (g^d)^{t(v_i)}(H(v_i)^d)^{-1})) \mid v_i \in Y\}$$

which matches the real world protocol.

Now we show that $\mathsf{Reduc}_{\text{rand}} \approxeq S_A^{\mathsf{CombinedScore}_p}(X, m, n, k, \{s(v) + t(v) \mid v \in X \cap Y\})$. With the corresponding input distribution, we have the reduction algorithm programs the random oracle $H(x_i) = \alpha_i$, but doesn't use the α_i anywhere else.

Since all β_i are sampled uniformly at random from $\mathbb{G}$, which are acting as $H(v_i)^b$ for $v_i \in X \cup Y$, they are decoupled from the specific elements in $X \cap Y$. The same is true for γ_i, which are acting as $H(v_i)^d$. Since we are shuffling M, E before returning them (we can think of E as a shuffled set in Alice's perspective, since they are ordered only by Bob's input set), we can reorder the β_i and γ_i and . In particular, we let $v_1, \ldots, v_m$ correspond to X and $v_{k-n+1}, \ldots, v_k$ correspond to Y (so overlapping indices correspond to elements in the intersection). Then we have $\beta_1, \ldots, \beta_m, \gamma_1 \ldots, \gamma_m$ correspond to elements $v \in X$ and $\beta_{k-n+1}, \ldots, \beta_k, \gamma_{k-n+1}, \ldots, \gamma_k$ correspond to elements in $v \in Y$. Then the reduction algorithm is equivalent to the hybrid

$$\begin{aligned} M &:= \{(\beta_i^a, \delta^{cs(v_i)}\gamma_i^c) \mid i \in [m]\} \\ E &:= \{\mathrm{Enc}(\beta_i, \delta^{t(v_i)}\gamma_i^{-1})) \mid i \in [k-n+1, k]\} \\ &\text{return } (a, c, \delta, M, \text{ shuffled}, E \text{ shuffled}, H). \end{aligned}$$

Furthermore, Alice will not be able to decrypt anything in E besides elements in the intersection (for $i \in [k-n+1, m]$). Therefore she only sees the encrypted messages, and by definition of one-time symmetric encryption, this is indistinguishable from the hybrid

$$\begin{aligned} M &:= \{(\beta_i^a, \delta^{cs(v_i)}\gamma_i^c) \mid i \in [m]\} \\ E &:= \{\mathrm{Enc}(\beta_i, \delta^{t(v_i)}\gamma_i^{-1}) \mid i \in [k-n+1, m]\} \cup \{\mathrm{Enc}(\beta_i, m_i) \mid i \in [m+1, k]\} \end{aligned}$$

for any dummy messages m_i, which our simulator can generate randomly. Finally, we have that $\gamma_i \leftarrow \mathbb{G} \Rightarrow \delta^{cs(v_i)}\gamma_i^c$ and $\delta^{t(v_i)}\gamma_i^{-1}$ appear individually random for all i (since γ_i are not returned in Alice's view). Thus, the individual scores do not matter, and we only need the product of payloads and Alice's masked scores to be $\delta^{c(s(v)+t(v))}$ for $v \in X \cap Y$. Then we can use the following lemma, which is a direct result of the group definition:

Lemma 5.4. *Let $c \in \mathbb{G}$ be some fixed group element. If $\gamma_i' = \gamma_i \cdot c$, where $\gamma_i \leftarrow \mathbb{G}$, then*

$$\gamma_i' \equiv \gamma_i,$$

i.e. γ_i' is equivalent to the uniform random distribution over $\mathbb{G}$.

Specifically, we can set $\gamma_i' = \gamma_i \cdot \delta^{-t(v_i)}$ for $i \in [k-n+1, m]$, or over the intersection, which is equidistributed with $\gamma_i' \leftarrow \mathbb{G}$ for $\gamma_i \leftarrow \mathbb{G}$. Thus we have the following hybrid:

$$\begin{aligned} M &:= \{(\beta_i^a, r_i) \mid i \in [m-j]\} \cup \{(\beta_i^a, \delta^{c(s(v_i)+t(v_i))}\gamma_i'^c) \mid i \in [m-j+1, m]\} \\ E &:= \{\mathrm{Enc}(\beta_i, \gamma_i'^{-1}) \mid i \in [k-n+1, m]\} \cup \{\mathrm{Enc}(\beta_i, m_i) \mid i \in [m+1, k]\} \end{aligned}$$

where $r_i \leftarrow \mathbb{G}$, which now is equivalent to our simulated view, so we are done.

□

5.3.2. CombinedScore$_p$ *Security of Bob's View.* Now we show that CombinedScore$_p$ is secure in the view of Bob.

Theorem 5.5. *If the DDH-assumption holds, then protocol* CombinedScore$_p$ *securely computes $f_B(X, Y) = (|X|, y^*)$ in the presence of semi-honest adversary Bob, where either*

$$y^* = \underset{v \in X \cap Y}{\arg\max}(s(v) + t(v))$$

or $y^* = \perp$ if $X \cap Y = \emptyset$.

To prove Theorem 5.5, we use the following reduction algorithm and simulator, where $m = |X|$ and $n = |Y|$:

Reduction Algorithm — Bob:

Input: X – Alice's secure input
Y – Bob's secure input
$(\alpha_1, \ldots, \alpha_m, g^c, \beta_1, \ldots, \beta_m, \gamma_1, \ldots, \gamma_m)$ – Inputs where $\alpha_i \leftarrow \mathbb{G}$, and either $(\beta_i, \gamma_i) = (\alpha_i^a, \alpha_i^c)$ or $\beta_i, \gamma_i \leftarrow \mathbb{G}$. Note that these are just generalized forms of $\mathsf{DH}_{m,\mathbb{G}}$ or $\mathsf{RAND}_{m,\mathbb{G}}$.
H – random oracle access

```
1:  procedure REDUCE($X, Y, (\alpha_1, \ldots, \alpha_m, g^c, \beta_1, \ldots, \beta_m, \gamma_1, \ldots, \gamma_m), H$)
2:      $b, d \leftarrow \mathbb{Z}_p$
3:      program $H[x_i \mapsto \alpha_i]$ a random oracle for $x_i \in X$
4:      abort if $\exists z_1, z_2 \in X \cup Y$ s.t. $z_1 \neq z_2$ and $H(z_1) = H(z_2)$
5:      $X = \{(x_1, s(x_1)), \ldots, (x_m, s(x_m))\}$
6:      $S := \{s(x_1), \ldots, s(x_m)\}$
7:      $Y = \{(y_1, t(y_1)), \ldots, (y_n, t(y_n))\}$
8:      $T := \{t(y_1), \ldots, t(y_n)\}$
9:      $P := \{(\beta_i, (g^c)^{s(v_i)} \cdot \gamma_i) \mid i \in [m]\}$
10:     $j := \arg\max_{i | y_i \in X \cap Y}(s(y_i) + t(y_i))$
11:     return $(b, d, P, j, H)$
12: end procedure
```

Simulator $S_B^{\mathsf{CombinedScore}_p}$:

Input: $m = |X|$

```
1:  procedure SIMULATE($Y, m = |X|, y^*$)
2:      $b, d \leftarrow Z_p$
3:      $Y = \{y_1, \ldots, y_n\}$
4:      $\beta_i \leftarrow \mathbb{G}$ for $i \in [m]$
5:      $\gamma_i \leftarrow \mathbb{G}$ for $i \in [m]$
6:      $P := \{(\beta_i, \gamma_i) \mid i \in [m]\}$
7:      $j := j \in [m]$ s.t. $y_j = y^*$
8:      return $(b, d, P, j, H)$
9:  end procedure
```

Note that the input distributions to the reduction algorithm are indistinguishable under the DDH assumption. Let Reduc_{DH} denote the distribution of (b, d, P, j, H) when the reduction algorithm takes input $\beta_i = \alpha_i^a, \gamma_i = \alpha_i^c$, and let $\mathsf{Reduc}_{\text{rand}}$ denote the distribution of (b, d, P, j, H) when the reduction algorithm takes input where $\beta_i, \gamma_i \leftarrow \mathbb{G}$. Now it simply remains to show the following:

Claim 5.6. *The following distributions are indistinguishable:*

$$\mathsf{Reduc}_{DH} \approxeq \mathsf{Real}_B^{\mathsf{CombinedScore}_p}$$

$$\mathsf{Reduc}_{rand} \approxeq S_B^{\mathsf{CombinedScore}_p}(Y, |X|, y^*).$$

Proof. First we show that $\mathsf{Reduc}_{DH} \approx \mathsf{Real}_B^{\mathsf{CombinedScore}_p}$. With the input distribution $\mathsf{DH}_{m,\mathbb{G}}$, we have $(\alpha_1, \dots, \alpha_m, g^c, \beta_1, \dots, \beta_m) = (\alpha_1, \dots, \alpha_m, g^c, \alpha_1^c, \dots, \alpha_m^c)$. The reduction algorithm programs the random oracle $H(x_i) = \alpha_1$, so we have $\beta_i = \alpha_i^c = H(x_i)^c$ and the algorithm returns the distribution

$$P := \{H(x_i)^a, (g^c)^{s(x_i)} \cdot H(x_i)^c) \mid i \in [m]\}$$
$$j := \arg\max_{i \mid y_i \in X \cap Y}(s(y_i) + t(y_i))$$

Now since the algorithm aborts if there is a collision with the random oracle H on the elements in $X \cup Y$, returning a view at all ensures that there are no collisions, i.e. the protocol is correct. Therefore, the following hybrid is an equivalent way to generate j:

$$P' := \{(\alpha_i^b, f^d) \mid (e, f) \in P\}$$
$$W^d := \{(g^c)^{ds(x_i)} \cdot H(x_i)^{cd} \mid i \in [m]\}$$
$$E := \{\mathrm{Enc}(H(y_i)^b, g^{dt_i} H(y_i)^{-d}) \mid i \in [n]\}$$

```
$\mathbf{for}(K, s) \in P'$ :
$\mathbf{if}\ \exists i \in [n]$ s.t. $T = Dec(K, E[i]) \neq \perp$ :
 $\mathbf{store}(i, s \cdot T^c)$ in $J$
```

$$j := \arg\max_{i \mid (i, s \cdot T^c) \in J} (\mathrm{dlog}_{(g^c)^d}(sT)),$$

which is to say that adding the rest of the real world protocol for returning j is equivalent because of correctness. However, since collisions in random oracle H happen with negligible probability, this hybrid is indistinguishable from removing the **abort** functionality, which is now clearly equivalent to the view of real world protocol.

Now we show that $\mathsf{Reduc}_{\mathrm{rand}} \approx S_B^{\mathsf{CombinedScore}_p}(Y, |X|, y^*)$. With the input distribution $\mathsf{RAND}_{m,\mathbb{G}}$, we have the reduction algorithm programs the random oracle $H(x_i) = \alpha_i$, but indeed does not use α_i anywhere else. Since H collisions also occur with negligible probability, we can remove the programming and abort functionalities completely. Then the algorithm is equivalent to the following hybrid:

$$W := \{(g^c)^{s(x_i)} \cdot \beta_i \mid i \in [m]\}$$
$$j := \arg\max_{i \mid y_i \in X \cap Y}(s(y_i) + t(y_i))$$

where $\beta_i \leftarrow \mathbb{G}$ are random.

Note that $(g^c)^{s(x_i)} \cdot \beta_i$ and β_i' where $\beta_i' \leftarrow \mathbb{G}$ are equivalent distributions, since we are working inside group $\mathbb{G}$, so we have the following hybrid is equivalent:

$$W := \{\beta_i' \mid i \in [m]\}$$
$$j := \arg\max_{i \mid y_i \in X \cap Y}(s(y_i) + t(y_i))$$

where $\beta_i' \leftarrow \mathbb{G}$. Finally, we have that by definition of y_*, the hybrid above is equivalent to the following:

$$j \in [m] \mid y_j = y^*$$

This is now clearly equivalent to the simulated distribution, so we are done. □

6. Combined Score DH-PSI Protocol - General Cyclic Group

6.1. Protocol Description.

6.1. Protocol Description. The $\mathsf{CombinedScore}_p$ protocol only works effectively in prime-ordered cyclic groups of known order. The key reason behind this is the use of inverse powers (i.e. a^{-1}, d^{-1}) to blindly compute shared keys. In order to avoid this, we propose an alternative protocol which works in groups of any order, known or unknown. Though more general than $\mathsf{CombinedScore}_p$, we will see that the alternative protocol requires an extra round of communication. This is a slight difference, but important nonetheless.

Let Alice have secret exponents a_1, a_2 and Bob have secret exponents b_1, b_2. Further let (x_i, s_i) represent one of Alice's item-score pairs for all i, and similarly let (y_i, t_i) denote Bob's item-score pairs for all i. Then our proposed protocol is as follows:

```
CombinedScore
Parameters: Cyclic group $\mathbb{G}$ of order $q$ (known or unknown, not necessarily prime)
  Alice                                                   Bob
  $X := \{(x_1, s_1), \ldots, (x_m, s_m)\} \subseteq \{0,1\}^* \times [N]$    $Y := \{(y_1, t_1), \ldots, (y_n, t_n)\} \subseteq \{0,1\}^* \times [N]$

                    Stage 1: Computing shared Keys
1.                                                        $b_1, b_2, d \leftarrow \mathbb{Z}_n$
2.                                                        $U := \{H(y)^{b_1} \mid (y,t) \in Y\}$
                              $\xleftarrow{U}$
3. $a_1, a_2, c \leftarrow \mathbb{Z}_n$,
4. $V := \{u^{a_1 a_2} \mid u \in U\}$
5. $P := \{(H(x)^{a_1}, g^{cs} H(x)^c) \mid x \in X\}$
                              $\xrightarrow{V, P}$
6.                                                        $Q := \left\{(p_1^{b_1 b_2}, p_2^d) \mid (p_1, p_2) \in P\right\}$ (shuffled)
                              $\xleftarrow{Q}$
                    Stage 2: Encrypting and Decrypting
7.                                                        $E := \{\mathsf{Enc}({v_i}^{b_2}, g^{dt_i} H(y_i)^{-d}) \mid v_i \in V,$
                                                                                    $(y_i, t_i) \in Y\}$
                              $\xleftarrow{E, g^d}$
8. for $(k, s) \in Q$:
     if $\exists\, e_i \in E$ s.t. $T := \mathsf{Dec}(K, e_i) \neq \bot$:
           $S_{\text{scores}}[i] := \text{dlog}_{g^{cd}}(T^c \cdot s)$
                    Stage 3: Declassifying the Max Score
if $S_{\text{scores}}$ empty:
  $j := \bot$
else:
9. $j := \arg\max_i \{S_{\text{scores}}[i]\}$
                              $\xrightarrow{j}$
10.                                                       output $y_j \in Y$ (or $\mathsf{None}$ if $j = \bot$)
```

We can see that $\mathsf{CombinedScore}$ is very closely related to $\mathsf{CombinedScore}_p$. We have the same three stages, and the final two stages are identical in both protocols. The key difference lies in the way we compute the shared encryption key. Instead of having Alice raise values to a^{-1} to blindly compute $H(x)^b$, instead we use two extra secure keys: l for Alice and m for Bob. Also, when computing combined scores, we use a combination of Alice's key c and Bob's key d instead of using only d. This does not change the ultimate functionality of the protocol.

Theorem 6.1. *If the DDH assumption holds, then* CombinedScore *securely computes*

$$F(X, Y) = ((|Y|, \{s(v) + t(v) \mid v \in X \cap Y\}), (|X|, y^*))$$

where

$$y^* = \underset{v \in X \cap Y}{\arg\max}(s(v) + t(v))$$

or $y^* = \perp$ *if* $X \cap Y = \emptyset$.

6.2. **Complexity.** In terms of computational complexity, there is no difference between CombinedScore and CombinedScore$_p$. Both run in time $O(m+N)$, where m is the size of the larger of the two party's data sets, and N is the upper bound for the scores s_i and t_i. The for-loop in line 8 seems to contradict this, but using the same argument used to verify the complexity of CombinedScore$_p$, we can ascertain that the complexity is as stated.

Communication complexity is where we see the difference between CombinedScore$_p$ and CombinedScore. While the former ultimately had three rounds of communication, we can see that the latter has four rounds of communication. By counting the number of arrows, it may seem that there instead five rounds. However, counting arrows can be misleading. Just like in CombinedScore$_p$, we can combine the arrows sending Q and E, g^d simultaneously. To get an accurate count of the rounds of communication, we should consider a chain of dependencies. Note that j is dependent upon E, which is dependent upon V, which is dependent on U. This confirms that CombinedScore needs a minimum of four rounds of communication.

6.3. **Security Proof.** To prove Theorem 6.1, we will consider the views of Alice and Bob separately.

6.3.1. CombinedScore *Security of Alice's View.* First, we must show that CombinedScore is secure in the view of Alice.

Theorem 6.2. *If the DDH-assumption holds, then protocol* CombinedScore *securely computes* $f_A(X, Y) = (|Y|, \{s(v)+t(v) \mid v \in X \cap Y\}, |X \cap Y|)$ *in the presence of semi-honest adversary Alice.*

To prove Theorem 6.2, we make use of the following reduction algorithm and simulator, where we let $m = |X|$, $n = |Y|$, $j = |X \cap Y|$, $k = |X \cup Y|$:

Reduction Algorithm — Alice:

Input: X – Alice's secure input
Y – Bob's secure input
$(\alpha_1, \dots, \alpha_k, \beta'_1, \dots, \beta'_k, \delta, \gamma_1, \dots, \gamma_k)$ – Inputs where $\alpha_i \leftarrow \mathbb{G}$ and either $\delta = g^d$, $(\{\beta'_i\}, \{\gamma_i\}) = (\{\alpha_i^{b_1 b_2}\}, \{\alpha_i^d\})$, or $\dot{e}lta, \beta'_i \gamma_i \leftarrow \mathbb{G}$. Note these inputs are general forms of the $(2k+1)$-tuples from $\mathsf{DH}_{k,\mathbb{G}}$ or $\mathsf{RAND}_{k,\mathbb{G}}$.

```
1: procedure REDUCE($X, Y, (\alpha_1, \dots, \alpha_k, \beta_1, \dots, \beta_k, \beta'_1, \dots, \beta'_k, \delta, \gamma_1, \dots, \gamma_k)$)
2:     $a_1, a_2, c \leftarrow \mathbb{Z}_p$
3:     $\{v_1, \dots, v_k\} := X \cup Y$
4:     program $H[v_i \mapsto \alpha_i]$ a random oracle for $v_i \in X \cup Y$
5:     $X = \{(x_1, s(x_1)), \dots, (x_m, s(x_m))\}$
6:     $Y = \{(y_1, t(y_1)), \dots, (y_n, t(y_n))\}$
7:     $M := \{(\beta_i'^{a_1}, (\delta)^{cs(v_i)} \gamma_i{}^c) \mid v_i \in X\}$
8:     $E := \{\mathrm{Enc}(\beta_i'^{a_1 a_2}, (\delta)^{t(v_i)} \gamma_i^{-1}) \mid v_i \in Y\}$
9:     return $(a_1, a_2, c, \delta, M$ shuffled$, E, H)$
10: end procedure
```

Simulator $S_A^{\mathsf{CombinedScore}_p}$:

Input: X – Alice's secure input
m, n, j, k
$\mathrm{scores}_{|X \cap Y|} = \{s(v) + t(v) \mid v \in X \cap Y\}$

```
1: procedure SIMULATE($X, m, n, j, k, \mathrm{scores}_{|X \cap Y|}$)
2:     $H :=$ honest random oracle
3:     $a_1, a_2, c \leftarrow \mathbb{Z}_p$
4:     $\delta \leftarrow \mathbb{G}$
5:     $\alpha_i, \beta'_i, \gamma_i \leftarrow \mathbb{G}$ for $i \in [k]$
6:     $m_i \leftarrow \mathbb{G}$ for $i \in [m+1, k]$
7:     $r_i \leftarrow \mathbb{G}$ for $i \in [m-j]$
8:     $M = \{(\beta'^{a_1}_i, r_i) \mid i \in [m-j]\} \cup \{(\beta'^{a}_i, \delta^{c(s(v_i)+t(v_i))} \gamma_i^c) \mid i \in [m-j+1, m]\}$
9:     $E = \{\mathrm{Enc}(\beta_i'^{a_1 a_2}, \gamma_i^{-1}) \mid i \in [k-n+1, m]\} \cup \{\mathrm{Enc}(\beta'_i, m_i) \mid i \in [m+1, k]\}$
10:    return $(a_1, a_2, c, \delta, M$ shuffled$, E$ shuffled$, H)$
11: end procedure
```

Note that the input distributions to the reduction algorithm are indistinguishable under the DDH assumption. Let Reduc_{DH} denote the distribution of $(a_1, a_2, c, \delta, M$ shuffled$, E, H)$ when the reduction algorithm takes input $\beta'_i = \alpha_i^{b_1 b_2}$, $\gamma_i = \alpha_i^c$. and let $\mathsf{Reduc}_{\mathrm{rand}}$ denote the distribution of $(a_1, a_2, c, \delta, M$ shuffled$, E, H)$ when the reduction algorithm takes input $\beta'_i, \gamma_i \leftarrow \mathbb{G}$. Now it simply remains to show the following:

Claim 6.3. *The following distributions are indistinguishable:*

$$\mathsf{Reduc}_{DH} \approx \mathsf{Real}_A^{\mathsf{CombinedScore}}$$

$$\mathsf{Reduc}_{rand} \approx S_A^{\mathsf{CombinedScore}}(X, m, n, k, \{s(v) + t(v) \mid v \in X \cap Y\}).$$

Note that this is almost identical to the prime proof reduction algorithm and simulators; the only differences lie in the exact exponents in the keys. However, we can see that β'_i is analogous to β_i in the security proof of $\mathsf{CombinedScore}_p$. The same arguments about decoupling β'_i from the elements $v_i \in X \cup Y$ apply here as well. Since the remainder of the proof is identical, we will omit it for the sake of brevity.

6.3.2. $\mathsf{CombinedScore}$ *Security of Bob's View.* Now we show that $\mathsf{CombinedScore}$ is secure in the view of Bob.

Theorem 6.4. *If the DDH-assumption holds, then protocol* $\mathsf{CombinedScore}$ *securely computes* $f_B(X, Y) = (|X|, y^*)$ *in the presence of semi-honest adversary Bob, where either*

$$y^* = \arg\max_{v \in X \cap Y}(s(v) + t(v))$$

or $y^* = \perp$ *if* $X \cap Y = \emptyset$.

To prove Theorem 6.4, we use the following reduction algorithm and simulator, where $m = |X|$ and $n = |Y|$:

Reduction Algorithm — Bob:

Input: X – Alice's secure input
Y – Bob's secure input
$(\alpha_1, \dots, \alpha_m, g^c, \beta'_1, \dots, \beta'_m, \gamma_1, \dots, \gamma_m)$ – Inputs where $\alpha_i \leftarrow \mathbb{G}$, and either $(\beta'_i, \gamma_i) = (\alpha_i^{a_1}, \alpha_i^c)$ or $\beta'_i, \gamma_i \leftarrow \mathbb{G}$. Note that these are just generalized forms of $\mathsf{DH}_{m,\mathbb{G}}$ or $\mathsf{RAND}_{m,\mathbb{G}}$.
H – random oracle access

```
1: procedure REDUCE($X, Y, (\alpha_1, \dots, \alpha_m, g^c, \beta'_1, \dots, \beta'_m, \gamma_1, \dots, \gamma_m), H$)
2:     $b_1, b_2, d \leftarrow \mathbb{Z}_p$
3:     program $H[x_i \mapsto \alpha_i]$ a random oracle for $x_i \in X$
4:     abort if $\exists z_1, z_2 \in X \cup Y$ s.t. $z_1 \neq z_2$ and $H(z_1) = H(z_2)$
5:     $X = \{(x_1, s(x_1)), \dots, (x_m, s(x_m))\}$
6:     $S := \{s(x_1), \dots, s(x_m)\}$
7:     $Y = \{(y_1, t(y_1)), \dots, (y_n, t(y_n))\}$
8:     $T := \{t(y_1), \dots, t(y_n)\}$
9:     $P := \{(\beta'_i, (g^c)^{s(v_i)} \cdot \gamma_i) \mid i \in [m]\}$
10:    $j := \arg\max_{i | y_i \in X \cap Y}(s(y_i) + t(y_i))$
11:    return $B^H(b_1, b_2, d, P, j, H)$
12: end procedure
```

Simulator $S_B^{\mathsf{CombinedScore}}$:

Input: $m = |X|$

```
1: procedure SIMULATE($Y, m = |X|, y^*$)
2:     $H :=$ honest random oracle
3:     $b_1, b_2, d \leftarrow Z_p$
4:     $Y = \{y_1, \ldots, y_n\}$
5:     $\beta'_i \leftarrow \mathbb{G}$ for $i \in [m]$
6:     $\gamma_i \leftarrow \mathbb{G}$ for $i \in [m]$
7:     $P := \{(\beta'_i, \gamma_i) \mid i \in [m]\}$
8:     $j := j \in [m]$ s.t. $y_j = y^*$
9:     return $(b_1, b_2, d, P, j, H)$
10: end procedure
```

Note that the input distributions to the reduction algorithm are indistinguishable under the DDH assumption. Let Reduc_{DH} denote the distribution of (b_1, b_2, d, P, j, H) when the reduction algorithm takes input $\beta'_i = \alpha_i^{a_1}$, $\gamma_i = \alpha_i^c$. and let $\mathsf{Reduc}_{\text{rand}}$ denote the distribution of (b_1, b_2, d, P, j, H) when the reduction algorithm takes input $\beta'_i, \gamma_i \leftarrow \mathbb{G}$. Now it simply remains to show the following:

Claim 6.5. *The following distributions are indistinguishable:*

$$\mathsf{Reduc}_{DH} \approxeq \mathsf{Real}_B^{\mathsf{CombinedScore}}$$

$$\mathsf{Reduc}_{rand} \approxeq S_B^{\mathsf{CombinedScore}}(Y, |X|, y^*).$$

However, note that this is now analogous to the $\mathsf{CombinedScore}_p$ protocol, where β'_i now replace the β_i. While prove $\mathsf{Reduc}_{DH} \approxeq \mathsf{Real}_B^{\mathsf{CombinedScore}_p}$ for Bob's view by adding in the rest of the protocol to calculate the index j, now we will simply add in the corresponding decryption section of the protocol $\mathsf{CombinedScore}$ to prove that $\mathsf{Reduc}_{DH} \approxeq \mathsf{Real}_B^{\mathsf{CombinedScore}}$. The proof for indistinguishability to the simulated view is equivalent. Thus, will omit the detailed proofs for the sake of brevity.

7. Sampling with One Sided Ranking

7.1. **Protocol Description.** We present another protocol based off of $\mathsf{Cardinality}$ PSI, this time for the problem of sampling according to one party's ranking of their elements. Without loss of generality, let this party by Bob. Our protocol uses the ORE scheme from [BLW14][2], which has the property that returning ciphertexts reveals only the relative ordering of plaintexts and nothing else. If Bob sends an encrypted payload, where the payload is his an encrypted ranking of his elements, then Alice can returns Bob's highest ranked item in the intersection without learning what his actual ranking of the element was. Note that this is not possible if Bob simply sends over his encrypted items in the order of his preference, as Alice can keep track positions of the elements in Bob's set that are in the intersection and consequently all of their ranks.

```
OneSidedRank
Parameters: Cyclic group $\mathbb{G}$ of order $q$
    Alice                                   Bob
    $X \subseteq \{0,1\}^*$                 $Y = \{(y_1, t(y_1)), \ldots, (y_n, t(y_n))\}$
                                                $\subseteq \{0,1\}^* \times [n]$

                    Stage 1: Computing Shared Keys

1. $a \leftarrow \mathbb{Z}_q$
2. $P := \{H(x)^a \mid x \in X\}$
                         $\xrightarrow{P}$
3.                                          $b \leftarrow \mathbb{Z}_q$
4.                                          $Q := \{p^b \mid p \in P\}$ (shuffled)
                         $\xleftarrow{Q}$

                    Stage 2: Encrypting and Decrypting

5.                                          $E := \{\mathsf{Enc}(H(y_i)^b, \mathsf{ORE}(t(y_i))) \mid y_i \in Y\}$
                         $\xleftarrow{E}$
6. for $q \in Q$:
        if $\exists\ e_i \in E$ s.t. $T_i := \mathsf{Dec}(q, e_i) \neq \perp$:
            $T[i] := T_i$

          Stage 3: Declassifying the Max-Ranked Common Element

7. if $T$ empty:
        $j := \perp$
   else:
        $j := \arg\max_i \{T[i]\}$
                         $\xrightarrow{j}$
8.                                          Bob calculates $y_j$ using $j$
9.                                          output $y_j \in Y$ (or None if $j = \perp$)
```

FIGURE 3. DH-PSI OneSidedRank Visualized

This protocol builds off of our protocol CombinedScore by using symmetric key encryption to encrypt Bob's payloads. The payloads in this case are the order revealing encryptions of his scores. In this sense, ORE replaces our encryption through exponentiating the generator g, and the algorithm to determine relative orders in ORE replaces our discrete log. Note that we cannot immediately apply ORE in the $\mathsf{CombinedScore}$, as the ORE scheme we use is not homomorphic.

After the key exchange protocol, we can see that Alice can only decrypt Bob's ORE values for elements in the intersection. Even after decrypting Bob's ORE values, she cannot learn Bob's exact ranking of his elements by definition of ORE, but she can still determine the relative ordering of the ranks. Thus, she can choose the element with the highest rank and send that index j to Bob, where the index j identifies an element with respect to Bob's set E. Upon receipt, Bob, who has maintained the order of his sets, can declassify the item by finding the corresponding $y_j \in Y$.

We claim that our protocol above has the functionality that Alice only learns the cardinality of the intersection, and nothing about Bob's ranks over elements in the intersection. Since Bob simply receives an index at the end to return a specific element in his set, he only learns the desired element.

Theorem 7.1. *If the DDH assumption holds, then* $\mathsf{CombinedScore}_p$ *securely computes*

$$F(X,Y) = ((|Y|, |X \cap Y|\}), (|X|, y^*))$$

where

$$y^* = \arg\max_{v \in X \cap Y}(t(v))$$

or $y^* = \perp$ *if* $X \cap Y = \emptyset$.

7.2. **General Cyclic Groups.** Similar to $\mathsf{CombinedScore}_p$, the protocol above uses inverse powers, which we can only guarantee when working with $\mathbb{G} = \langle g \rangle^p$ a group of known prime order p. However, we note that we can use the same key sharing sub-protocol as in the generalized protocol $\mathsf{CombinedScore}_p$ here as well. Thus we also have a $\mathsf{OneSidedRank}$ protocol for groups of known or unknown order, whose description we will omit. The security proof, which we will also omit, is very similar and borrows from the security proof of $\mathsf{CombinedScore}$ for the key sharing stage.

7.3. **Complexity.** This protocol is almost identical in structure to $\mathsf{CombinedScore}$ and $\mathsf{CombinedScore}_p$. It would make sense that the computational complexity of $\mathsf{OneSidedRank}$ could be similar to the other two. However, before we make any assertions about the complexity, we need to consider how the ORE computation plays into the calculation. Counting each ORE encryption and each comparison under ORE as one operation, we can see that the computational complexity of this protocol is $O(m)$, where m is the size of the larger of the two party's data sets Note that whether we are using the protocol specialized for prime-order groups or the general protocol, the computational complexity remains the same.

For the communication complexity, it is important to note that there are two protocols being described: one exclusively for prime-order groups and one for groups of any order (known or unknown). Just like how using $\mathsf{CombinedScore}$ cost one more round of communication than $\mathsf{CombinedScore}_p$, the general version of this protocol uses four rounds of communication while the prime version uses three. The justification for this is identical to those of the previous two protocols.

7.4. **Security Proof.** To prove Theorem 7.1, we will consider the views of Alice and Bob separately.

7.4.1. *OneSidedRank Security of Alice's View.* First, we must show that OneSidedRank is secure in the view of Alice.

Theorem 7.2. *If the DDH-assumption holds, then protocol OneSidedRank securely computes $f_A(X, Y) = (|X|, |X \cap Y|)$ in the presence of semi-honest adversary Alice.*

To prove Theorem 7.2, we let $m = |X|$, $n = |Y|$, and $k = |X \cup Y|$. Also let $t'_1, \ldots, t'_n$ be the sequence that has the same relative ordering as the sequence $t(y_1), \ldots, t(y_n)$ and is also a permutation of $[n]$ (so we can think of collapsing the ranks $t_1, \ldots, t_n$). We will call these t'_i *condensed scores.* Now we can make use of the following reduction algorithm and simulator:

Reduction Algorithm — Alice:

Input: X – Alice's secure input
Y – Bob's secure input
$(\alpha_1, \ldots, \alpha_k, \beta_1, \ldots, \beta_k, \gamma_1, \ldots, \gamma_k)$ – Inputs where $\alpha_i \leftarrow \mathbb{G}$, either $\beta_i = \alpha_i^b$, $\gamma_i = \mathsf{ORE}(t_i)$ or $\beta_i \leftarrow \mathbb{G}$, $\gamma_i = \mathsf{ORE}(t'_i)$.

```
1: procedure Reduce($X, Y, (\alpha_1, \ldots, \alpha_k, \beta_1, \ldots, \beta_k, \gamma_1, \ldots, \gamma_n)$)
2:     $a \leftarrow \mathbb{Z}_p$
3:     $\{v_1, \ldots, v_k\} := X \cup Y$
4:     program $H[v_i] \mapsto \alpha_i$ a random oracle for $v_i \in X \cup Y$
5:     $X = \{x_1, \ldots, x_m\}$
6:     $Y = \{(y_1, t(y_1)), \ldots, (y_n, t(y_n))\}$
7:     $Q := \{\beta_i^a \mid v_i \in X\}$
8:     $E := \{\mathsf{Enc}(\beta_i, \gamma_i) \mid v_i \in Y\}$
9:     return $(a, Q$ shuffled$, E, H)$
10: end procedure
```

Simulator $S_A^{\mathsf{OneSidedRank}}$:

Input: X – Alice's secure input
$m = |X|, n = |Y|, j = |X \cap Y|, k = |X \cup Y|$

```
1: procedure Simulate($X, m, n, k$)
2:     $H :=$ honest random oracle
3:     $a \leftarrow \mathbb{Z}_p$
4:     $\gamma_i := \mathsf{ORE}(i)$
5:     $\alpha_i \leftarrow \mathbb{G}$ for $i \in [k]$
6:     $\beta_i \leftarrow \mathbb{G}$ for $i \in [k]$
7:     $Q = \{(\beta_i^a \mid i \in [m]\}$
8:     $E = \{\mathsf{Enc}(\beta_i, \gamma_i) \mid i \in [k - n + 1, k]\}$
9:     return $(a, Q$ shuffled$, E$ shuffled$, H)$
10: end procedure
```

Note that the input distributions to the reduction algorithm are indistinguishable by the properties of ORE and the DDH assumption. Let Reduc_{DH} denote the distribution of $(a, Q \text{ shuffled}, E, H)$ when the reduction algorithm takes input $(\{\alpha_i\}, \{\alpha_i^b\}, \{\mathsf{ORE}(t(y_i))\})$, and let $\mathsf{Reduc}_{\text{rand}}$ denote the distribution of $(a, Q \text{ shuffled}, E, H)$ when the reduction algorithm takes input $(\{\alpha_i\}, \{\beta_i\}, \{\mathsf{ORE}(t'_i)\})$, where $\beta_i \leftarrow \mathbb{G}$. Now it simply remains to show the following:

Claim 7.3. *The following distributions are indistinguishable:*

$$\mathsf{Reduc}_{DH} \approxeq \mathsf{Real}_A^{\mathsf{OneSidedRank}}$$
$$\mathsf{Reduc}_{rand} \approxeq S_A^{\mathsf{OneSidedRank}}(X, m, n, j, k).$$

Proof. First we show that $\mathsf{Reduc}_{DH} \approxeq \mathsf{Real}_A^{\mathsf{OneSidedRank}}$. With the corresponding input distribution we have $\beta_i = \alpha_i^b$ and $\gamma_i = \mathsf{ORE}(t(y_i))$. The reduction algorithm programs the random oracle $H(v_i) = \alpha_i$ for all $v_i \in X \cup Y$, so we have $\beta_i = \alpha_i^b = H(v_i)^b$. Then the algorithm returns

$$Q = \{(H(v_i)^b)^a \mid v_i \in X\}$$
$$E = \{\mathsf{Enc}(H(v_i)^b, \mathsf{ORE}(t(v_i))) \mid v_i \in Y\}$$

which matches the real world protocol.

Now we show that $\mathsf{Reduc}_{\text{rand}} \approxeq S_A^{\mathsf{OneSidedRank}}(X, m, n, j, k)$. With the corresponding input distribution, we have the reduction algorithm programs the random oracle $H(x_i) = \alpha_i$, but doesn't use the α_i anywhere else. Note that by definition of symmetric key encryption, $\mathsf{Enc}(\beta_i, \gamma_i) \approxeq \mathsf{Enc}(\beta_i, m_i)$ for random messages $m_i \leftarrow 0, 1^*$ for any γ_i not in the intersection, since Alice does not have they key. Then our reduction algorithm is equivalent to the hybrid

```
$m_i \leftarrow 0, 1^* \mid i \in [k]$
$Q := \{\beta_i^a \mid v_i \in X\}$
$E := \{\mathsf{Enc}(\beta_i, \gamma_i) \mid v_i \in X \cap Y\} \cup \{\mathsf{Enc}(\beta_i, m_i) \mid v_i \notin X \cap Y\}$
**return** $(a, Q \text{ shuffled}, E\text{shuffled}, H)$,
```

where we can shuffle E since Bob's elements are already "shuffled" to Alice.

Since all β_i, which are acting as $H(v_i)^b$ for $v_i \in X \cup Y$, are sampled uniformly at random from $\mathbb{G}$, they are decoupled from the specific elements in $X \cap Y$, so we can reorder them. In particular, we can think that $\beta_1, \ldots, \beta_n$ corresponds to elements $v \in Y$ and $\beta_{k-m+1}, \ldots, \beta_k$ corresponds to elements $v \in X$ (so overlapping indices correspond to elements in the intersection).

Furthermore, these β_i are also decoupled from associated scores $t(y_i)$, so we can order β_i more specifically; we let β_1 correspond to the element $y_i \in X \cap Y$ such that $t(y_i)$ is the minimum score, β_2 correspond to the element with the next smallest score, and so on up to β_j (recall $j = |X \cap Y|$). For notation's sake, let T' be a sorted list of the encryptions $\mathsf{ORE}(t'_i)$ associated with elements $v_i \in X \cap Y$ in the intersection. Let $T'[i]$ denote the i^{th} element in the list. Now the previous hybrid is equivalent to the following hybrid:

$$Q := \{\beta_i^a \mid i \in [k - m + 1, k]\}$$
$$E := \{\mathsf{Enc}(\beta_i, T'[i]) \mid i \in [j]\} \cup \{\mathsf{Enc}(\beta_i, m_i) \mid i \in [n - j]\}.$$

Finally, by definition of ORE, we have $(T'[1], \dots T'[j]) \approx (\mathsf{ORE}(1), \dots, \mathsf{ORE}(j))$. Thus, this hybrid is equivalent to

$$\begin{aligned} Q &:= \{\beta_i^a \mid i \in [k-m+1, k]\} \\ E &:= \{\mathsf{Enc}(\beta_i, \mathsf{ORE}(i)) \mid i \in [j]\} \cup \{\mathsf{Enc}(\beta_i, m_i) \mid i \in [n-j]\}. \end{aligned}$$

which now is equivalent to our simulated view, so we are done. □

7.4.2. *OneSidedRank Security of Bob's View.* Now we show that OneSidedRank is secure in the view of Bob.

Theorem 7.4. *If the DDH-assumption holds, then protocol* CombinedScore *securely computes* $f_B(X, Y) = (|X|, y^*)$ *in the presence of semi-honest adversary Bob, where either*

$$y^* = \arg\max_{v \in X \cap Y} (t(v))$$

or $y^* = \perp$ *if* $X \cap Y = \emptyset$.

To prove Theorem 7.4, we make use of the following reduction algorithm and simulator, where we let $m = |X|$, $n = |Y|$:

Reduction Algorithm — Bob:

Input: X – Alice's secure input
Y – Bob's secure input
$(\alpha_1, \dots, \alpha_m, \beta_1, \dots, \beta_m)$ – Inputs where $\alpha_i \leftarrow \mathbb{G}$, and either $\beta_i = \alpha_i^a$ or $\beta_i \leftarrow \mathbb{G}$. Note that these are just generalized forms of $\mathsf{DH}_{m,\mathbb{G}}$ or $\mathsf{RAND}_{m,\mathbb{G}}$.
H – random oracle access

```
1: procedure REDUCE($X, Y, (\alpha_1, \dots, \alpha_m, \beta_1, \dots, \beta_m, H)$)
2:     $b, d \leftarrow \mathbb{Z}_p$
3:     program $H[x_i \mapsto \alpha_i]$ a random oracle for $x_i \in X$
4:     abort if $\exists z_1, z_2 \in X \cup Y$ s.t. $z_1 \neq z_2$ and $H(z_1) = H(z_2)$
5:     $P := \{(\beta_i \mid i \in [m]\}$
6:     $j := \arg\max_{i | y_i \in X \cap Y}(t(y_i))$
7:     return $(b, P, j, H)$
8: end procedure
```

Simulator $S_B^{\mathsf{OneSidedRank}}$:

Input: $m = |X|$

```
1: procedure SIMULATE($Y, m = |X|, y^*$)
2:     $H :=$ honest random oracle
3:     $b \leftarrow Z_p$
4:     $Y = \{y_1, \dots, y_n\}$
5:     $\beta_i \leftarrow \mathbb{G}$ for $i \in [m]$
6:     $P := \{\beta_i \mid i \in [m]\}$
7:     $j := j \in [m]$ s.t. $y_j = y^*$
8:     return $(b, P, j, H)$
9: end procedure
```

Note that the input distributions to the reduction algorithm are indistinguishable under the DDH assumption. Let Reduc_{DH} denote the distribution of (b, P, j, H) when the reduction algorithm takes input $(\alpha_1, \dots, \alpha_m, \alpha_1^a, \dots, \alpha_m^a)$ and let $\mathsf{Reduc}_{\text{rand}}$ denote the distribution of (b, P, j, H) when the reduction algorithm takes input $(\alpha_1, \dots, \alpha_m, \beta_1, \dots, \beta_m)$ where $\beta_i \leftarrow \mathbb{G}$. Now it simply remains to show the following:

Claim 7.5. *The following distributions are indistinguishable:*

$$\mathsf{Reduc}_{DH} \approxeq \mathsf{Real}_B^{\mathsf{OneSidedRank}}$$
$$\mathsf{Reduc}_{rand} \approxeq S_B^{\mathsf{OneSidedRank}}(Y, m = |X|, y^*).$$

Proof. First we show that $\mathsf{Reduc}_{DH} \approxeq \mathsf{Real}_B^{\mathsf{OneSidedRank}}$. With the input distribution $\mathsf{DH}_{m,\mathbb{G}}$, we have $(\alpha_1, \dots, \alpha_m, \beta_1, \dots, \beta_m) = (\alpha_1, \dots, \alpha_m, \alpha_1^a, \dots, \alpha_m^a)$. The reduction algorithm programs the random oracle $H(x_i) = \alpha_1$, so we have $\beta_i = \alpha_i^a = H(x_i)^a$ and the algorithm returns the distribution

$$P := \{H(x_i)^a \mid i \in [m]\}$$
$$j := \arg\max_{i \mid y_i \in X \cap Y}(t(y_i))$$

Now since the algorithm aborts if there is a collision with the random oracle H on the elements in $X \cup Y$, returning a view at all ensures that there are no collisions, i.e. the protocol is correct. Therefore, the following hybrid is an equivalent way to generate j:

```
$Q := \{\beta_i^b \mid i \in [m]\}$
$E := \{\mathsf{Enc}(H(y_i)^b, \mathsf{ORE}(t(y_i))) \mid i \in [n]\}$
for $K \in P'$:
    if $\exists e_i \in E$ s.t. $T_i = Dec(K, e_i) \neq \perp$:
        store $(i, T_i)$ in $J$
$j := \arg\max_{i \mid (i, T_i) \in J}(T_i)$,
```

which is to say that adding the rest of the real world protocol for returning j is equivalent because of correctness. However, since collisions in random oracle H happen with negligible probability, this hybrid is indistinguishable from removing the **abort** functionality, which is now clearly equivalent to the view of real world protocol.

Now we show that $\mathsf{Reduc}_{\text{rand}} \approx S_B^{\mathsf{OneSidedRank}}(Y, m = |X|, y^*)$. With the input distribution $\mathsf{RAND}_{m,\mathbb{G}}$, we have the reduction algorithm programs the random oracle $H(x_i) = \alpha_i$, but indeed does not use α_i anywhere else. Since we also have that H collisions occur with negligible probability, we can remove the programming and abort functionality altogether. Then the algorithm is equivalent to the following hybrid:

$$\begin{aligned} \beta_i &\leftarrow \mathbb{G} \mid i \in [m] \\ P &:= \{\beta_i \mid i \in [m]\} \\ j &:= \underset{i \mid y_i \in X \cap Y}{\arg\max}(t(y_i)) \end{aligned}$$

Then, by definition of y_*, the hybrid above is equivalent to the following:

$$j \in [m] \mid y_j = y^*$$

This is now clearly equivalent to the simulated distribution, so we are done.

□

8. Conclusions

In this paper, we present four new protocols for variations on a PSI problem that has not yet been considered outside of general multi-party computation, namely sampling from the intersection in the two-party case. Specifically, our protocols deal with: random sampling from the intersection, sampling according to one party's preference, and sampling according to both party's preferences. This general problem is an important one to consider in the field of PSI, as parties engaging in PSI may only require one element in their intersection by some metric (e.g. preference) for their use. These protocols allow parties to do so without revealing the entire intersection, which is especially important when parties suspect their intersection is large and/or don't want to reveal that much information about their input sets. Our protocols are all secure against semi-honest adversaries.

Furthermore, our protocols all have time complexity that is linear in the size of the input sets, and communication complexity of at most three or four rounds of communication. Besides being efficient, these protocols have an advantage over general garbled circuit computation in that they only rely on a simple Diffie-Hellman style of communication. Thus, they require much less machinery and are conceptually much easier to understand.

Several open problems remain from our research; though we have protocols for all three of our problems, it is unknown whether there are protocols that leak less than we do. For example, one could ask if a protocol that samples randomly from the intersection without leaking its cardinality to either party exists. We can also pose new open problems to extend the functionality of our protocols. An interesting question is whether we can generalize $\mathsf{CombinedScore}$ to only reveal elements in the intersection with a combined score above a certain threshold t. Again, we can ask if we can achieve this without leaking the cardinality. We suspect that many of these open problems will require fundamentally different encryption techniques, such as Oblivious Polynomial Evaluation (OPE), as Diffie-Hellman style communication is prone to leaking cardinality. These are important questions that will potentially guide our future work.

9. Acknowledgements

We would like to thank our wonderful mentor, Dr. Mike Rosulek, for guiding us through this research. We would also like to thank Dr. Holly Swisher for organizing the REU program. Finally, we would like to thank the National Science Foundation (NSF) for funding our research through grant DMS-1757995.

References

[1] Dan Boneh. The decisional diffie-hellman problem. pages 48–63, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[2] Dan Boneh, Kevin Lewi, Mariana Raykova, Amit Sahai, Mark Zhandry, and Joe Zimmerman. Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation. Cryptology ePrint Archive, Report 2014/834, 2014. `https://ia.cr/2014/834`.

[3] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Technical report, Boston University and Tel Aviv University, 2013.

[4] David Evans, Vladimir Kolesnikov, and Mike Rosulek. A pragmatic introduction to secure multi-party computation. *Foundations and Trends in Privacy and Security*, 2(2-3):70–246, 2018.

[5] Bernardo A. Huberman, Matt Franklin, and Tad Hogg. Enhancing privacy and trust in electronic communities. In *In Proc. of the 1st ACM Conference on Electronic Commerce*, pages 78–86. ACM Press, 1999.

[6] Lea Kissner and Dawn Song. Privacy-preserving set operations. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, pages 241–257, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[7] Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. Efficient batched oblivious prf with applications to private set intersection. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 818829, New York, NY, USA, 2016. Association for Computing Machinery.

[8] C. Meadows. A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. page 134, Los Alamitos, CA, USA, apr 1986. IEEE Computer Society.

[9] Goldreich Oded. *Foundations of Cryptography: Volume 2, Basic Applications.* Cambridge University Press, USA, 1st edition, 2009.

[10] Benny Pinkas, Thomas Schneider, Oleksandr Tkachenko, and Avishay Yanai. Efficient circuit-based psi with linear communication. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019*, pages 122–153, Cham, 2019. Springer International Publishing.

[11] Yongjun Zhao and Sherman S. M. Chow. Can you find the one for me? privacy-preserving matchmaking via threshold psi. Cryptology ePrint Archive, Report 2018/184, 2018. `https://ia.cr/2018/184`.

Truman State University
E-mail address: trb4137@truman.edu

MIT
E-mail address: janabel@mit.edu